6

ИИКТ - БАН
eISSN: 2367-8666
Лекции по компютърни науки и технологии

Програмиране на Android

Практическо ръководство

Тодор Балабанов

eISBN: 978-619-7320-02-2

Поредицата „**Лекции по компютърни науки и технологии на Института по информационни и комуникационни технологии при Българската академия на науките**“ публикува в електронен вид учебници и учебни помагала, предназначени за студенти и докторанти по различни програми по информатика, изчислителна математика, математическо моделиране, комуникационни технологии, и др., както и за всички читатели, интересуващи се от тези научни области. Учебниците се базират върху курсове лекции, водени от учени на Института по информационни и комуникационни технологии – БАН в различни български университети и в Центъра за обучение на докторанти в БАН. Публикуваните материали са с отворен достъп - те са свободно достъпни без заплащане.

Редактори

Геннадий Агре (Главен редактор) – ИИКТ-БАН
e-mail: agre@iinf.bas.bg

Вера Ангелова – ИИКТ-БАН
e-mail: vangelova@iit.bas.bg

Пенчо Маринов – ИИКТ-БАН
e-mail: pencho@bas.bg

eISSN: 2367-8666

Съдържание

1. Въведение.....1
2. Логическата игра за дъска Ithaka.....2
3. Управление на проект с отворен код.....4
4. Генериране на печалба от проекти с отворен код.....11
5. Визуализиране на помощна информация.....23
6. Потребителски интерфейс.....31
7. Обектноориентиран модел.....36
8. Извършване на пресмятания.....40
9. Разпространение на продукта.....66
10. Заключение.....78
Литература.....79

1. Въведение

През първото десетилетие на 21 век, в ежедневния бит на хората масово навлезе използването на мобилни изчислителни устройства, основно под формата на умни телефони и таблети. Наличието на глобални свободни пазари се оказа ключов фактор за развитието на интензивна конкурентна среда. Първите масово разпространени „умни“ устройства бяха предложени от компанията Apple, с операционна система iOS [1] и програмен език Objective-C. В кратки срокове след това бе предложена отворена платформа за развитие на мобилни устройства – Android [2], за развитието на която основна роля изигра компанията Google, залагайки като програмен език Java [3]. Компанията Microsoft също предложи свое решение, базирано на операционната система Windows Phone, с основен програмен език C#. Трите водещи производителя се различават основно в степента на отвореност, която предлагат за своите системи. В случая на Apple платформата е напълно затворена. В случая на Android стремежът е платформата да бъде напълно отворена. Докато Microsoft заемат относително междинна позиция.

В настоящото практическо ръководство ще бъде представен процесът за създаване на програмни продукти за платформата Android. Изборът на тази платформа е основан на идеята за свободен достъп до технологии, а също така и на факта, че производствените разходи за платформата Android са най-ниски. Практическото ръководство е предназначено за читатели със средно или напреднало ниво в областта на програмирането и софтуерното инженерство. Ръководството не е подходящо за читатели, които тепърва навлизат в областта на компютърните науки.

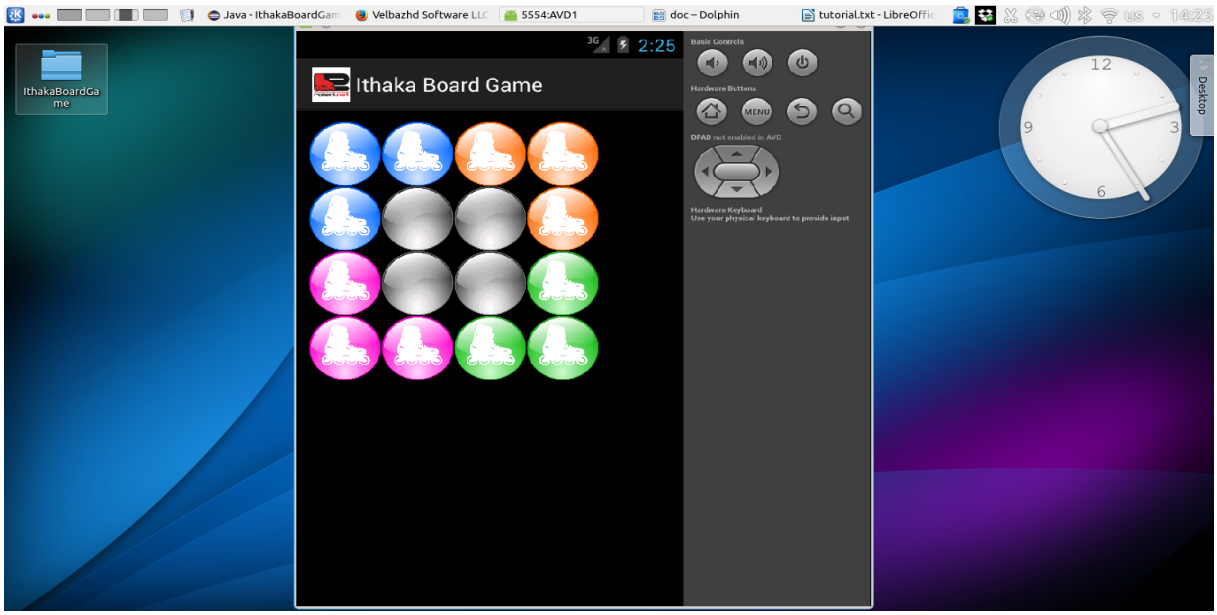
Поради своя практически характер настоящото ръководство излага минимални теоретични знания и акцентира върху практическите стъпки за създаване и поддръжка на софтуер. Обект на разработката представлява логическа игра, която се играе на дъска. По-сложни игри, разчитащи на библиотеки като OpenGL [4,5], се разглеждат в книги за читатели с повече знания и опит в областта. Софтуерният проект е под GPL3 отворен лиценз и може да се намери свободно в публичното пространство.

2. Логическата игра за дъска Ithaka

Логическата игра Ithaka е публикувана официално през 2002 година от L. Lynn Smith.

Фиг. 1 Официална страница на логическата игра Ithaka в уеб сайта BoardGameGeek

Играта се играе от двама играчи на дъска, която е оформена в 4x4 клетки. Върху дъската са разположени пулове в четири различни цвята. Целта на играчите, за да победят, е да формират линия от три едноцветни пула (вертикално, хоризонтално или диагонално).



Фиг. 2 Начална конфигурация на игралното табло

Всеки от играчите може да мести пулове от четирите различни цвята. На всеки ход играчът мести само един пул по права линия (хоризонтално, вертикално и диагонално)

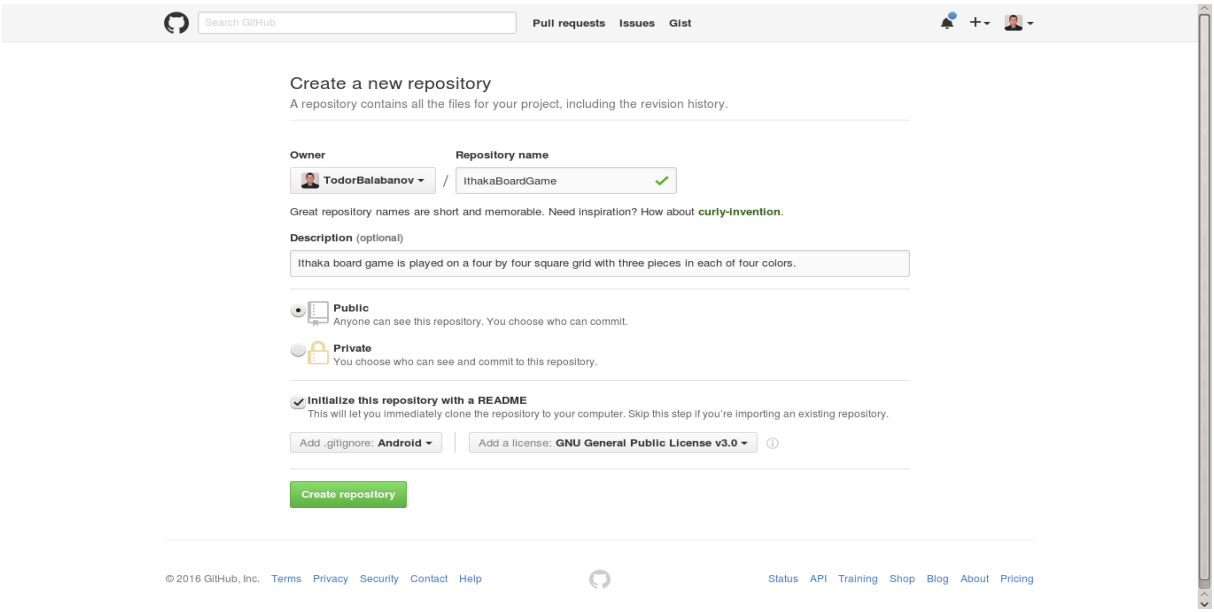
през произволен брой празни клетки. Първото ограничение при местене на пул е, че не може да се играе с последно преместения пул. Второто ограничение е, че може да се местят само пулове, които имат в съседство пул от същия цвят. Основното условие за победа е да бъде формирана права линия от три пула от един и същи цвят. Играчът побеждава опонента си и ако постигне конфигурация, в която опонентът няма позволен за местене ход. Третото условие за победа е, ако противникът извърши един и същи ход три последователни пъти.

Играта Ithaka има достатъчно опростени правила и поради тази причина е много добър кандидат за реализиране под формата на приложение за мобилни устройства. Размерът на игралното табло в тази игра е 16 клетки. Тя е със сложност, сходна на популярната игра Connect Four. При логическите игри с развлекателен характер от съществено значение е сложността на създаването, на компютърно управляван опонент (изкуствен интелект). За игри с относително малко пространство на състоянията (както е играта Tic-Tac-Toe) дори алгоритми с пълно изчерпване дават достатъчно добри резултати в приемливо време за изчисление.

3. Управление на проект с отворен код

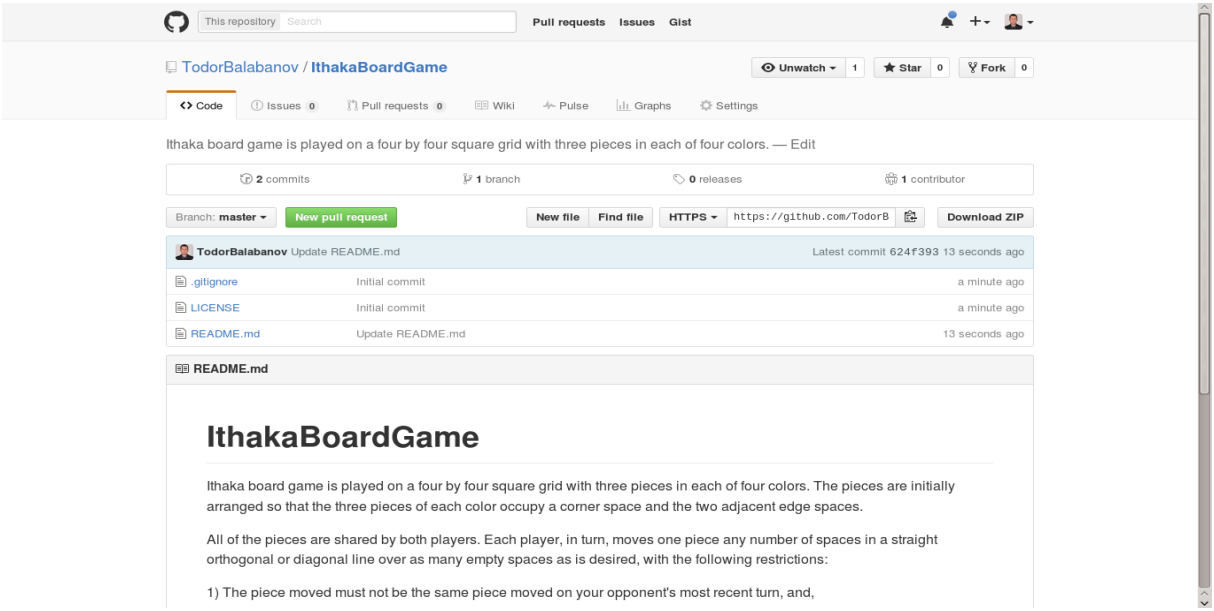
Управлението на софтуерни проекти се осъществява със софтуерни системи, създадени специално за тази цел. За нуждите на настоящото изложение разработваният софтуерен проект е с GPL3 лиценз за софтуер с отворен код. Една от най-подходящите алтернативи да се управлява такъв проект е хранилището GitHub. GitHub представлява облачна услуга, с която се публикуват софтуерни проекти. Включва различни системи, които позволяват работата на програмисти в екип. Достъпни са система за проследяване на дефекти и система за контрол на версиите (базирана на софтуерния инструмент Git).

Създаването на един софтуерен продукт [6,7] започва със заделяне на хранилище в което ще се разполага програмният код.



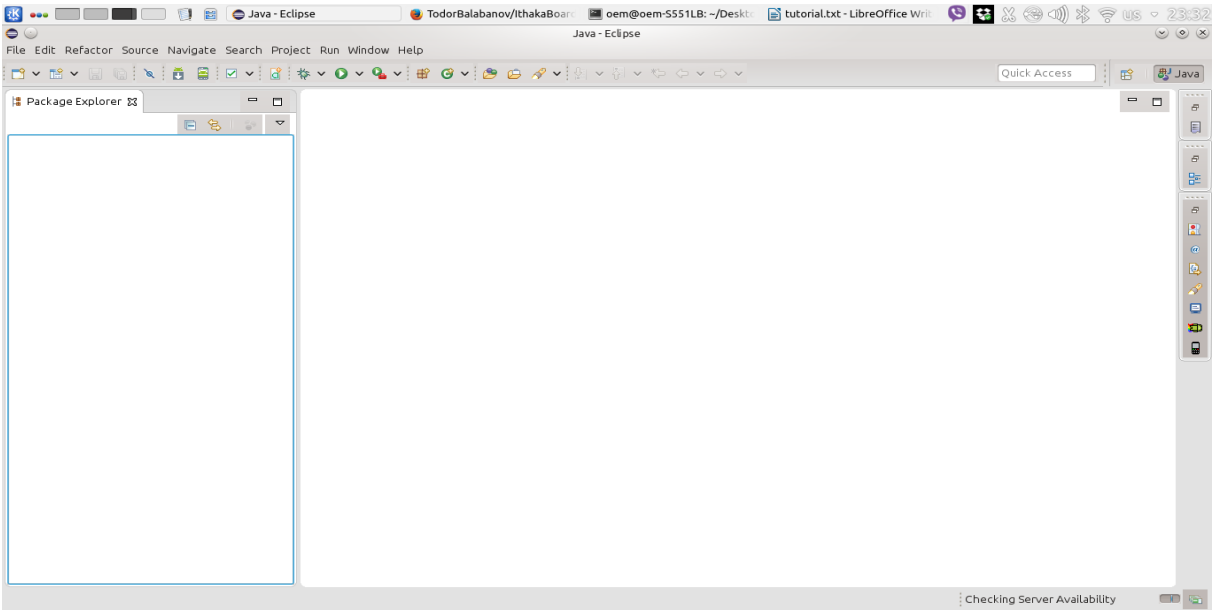
Фиг. 3 Създаване на хранилище в GitHub

При първоначалното конфигуриране на отдалеченото хранилище в GitHub се въвежда име на проекта, определя се съдържанието на .gitignore файла (в конкретния случай Android конфигурация) и се избира подходящ лиценз (в конкретния случай GPL3).



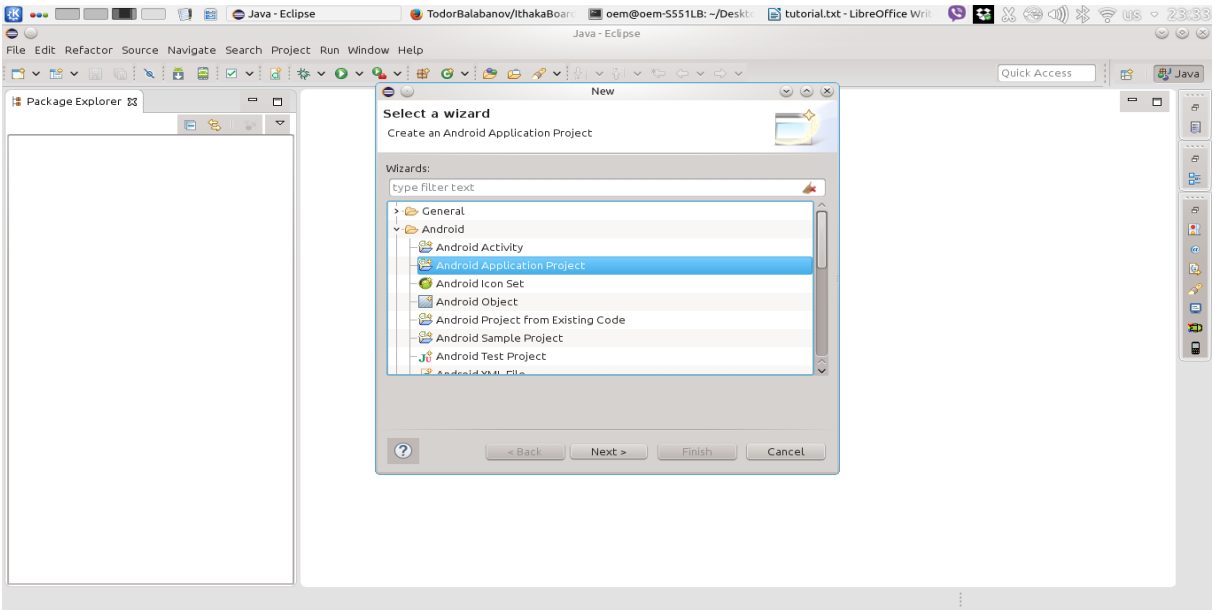
Фиг. 4 Състояние на новосъздаденото хранилище в GitHub

При използването на езика за програмиране Java са възможни различни развойни инструменти. Един от най-популярните и най-широко използвани е Eclipse [8]. Eclipse представлява интегрирана среда за разработка (IDE), която основно включва текстов редактор и механизъм за извикване на развойните инструменти, предоставени от компанията Oracle. За нуждите на Android програмирането, Eclipse поддържа допълнителна приставка, наречена Android Developer Tools – Plugin. За разработка на Android приложения, Eclipse се използва в комбинация с Android Software Development Kit, който включва и емулатор на различни виртуални устройства.



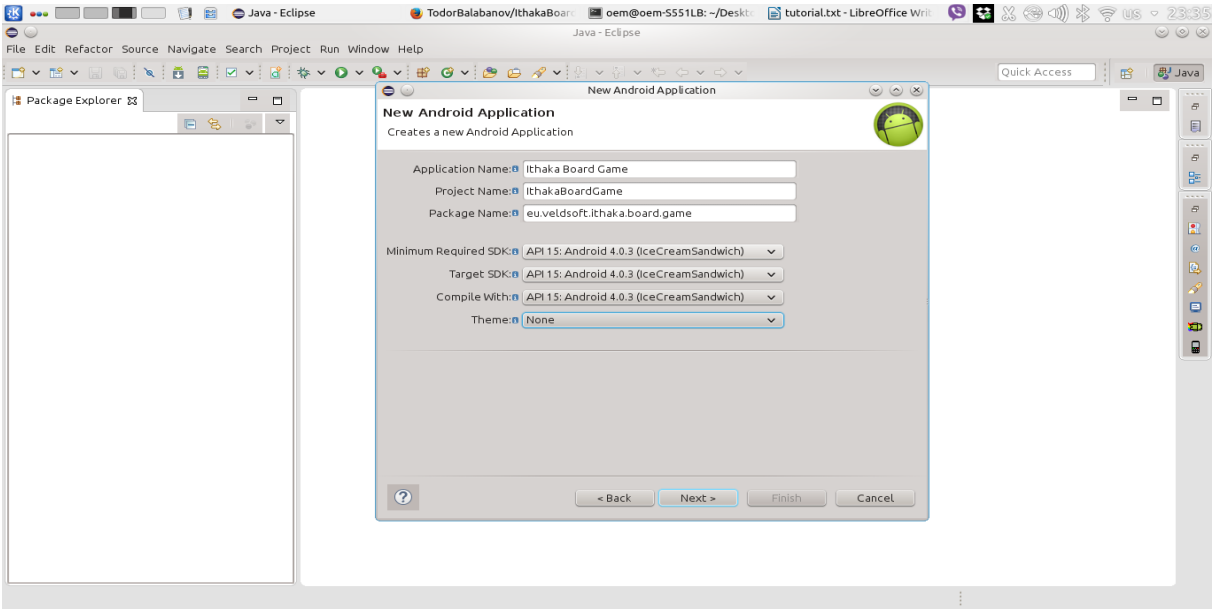
Фиг. 5 Начален работен екран в Eclipse

Eclipse е развойна среда, която организира работата на проектен принцип. За тази цел всяка разработка се оформя като Eclipse Project.



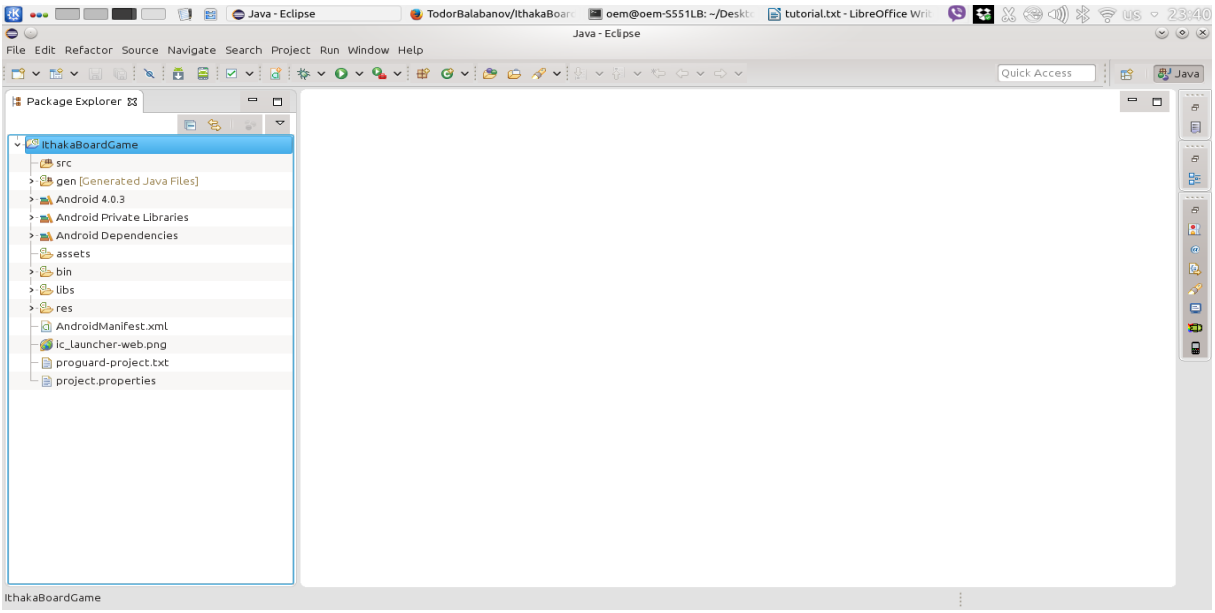
Фиг. 6 Създаване на нов Eclipse проект

Развойната среда позволява работа с множество различни проекти, но в случая на Android разработка най-удачният вариант е генерирането на Android Application Project.



Фиг. 7 Основни параметри на новосъздадения Eclipse проект

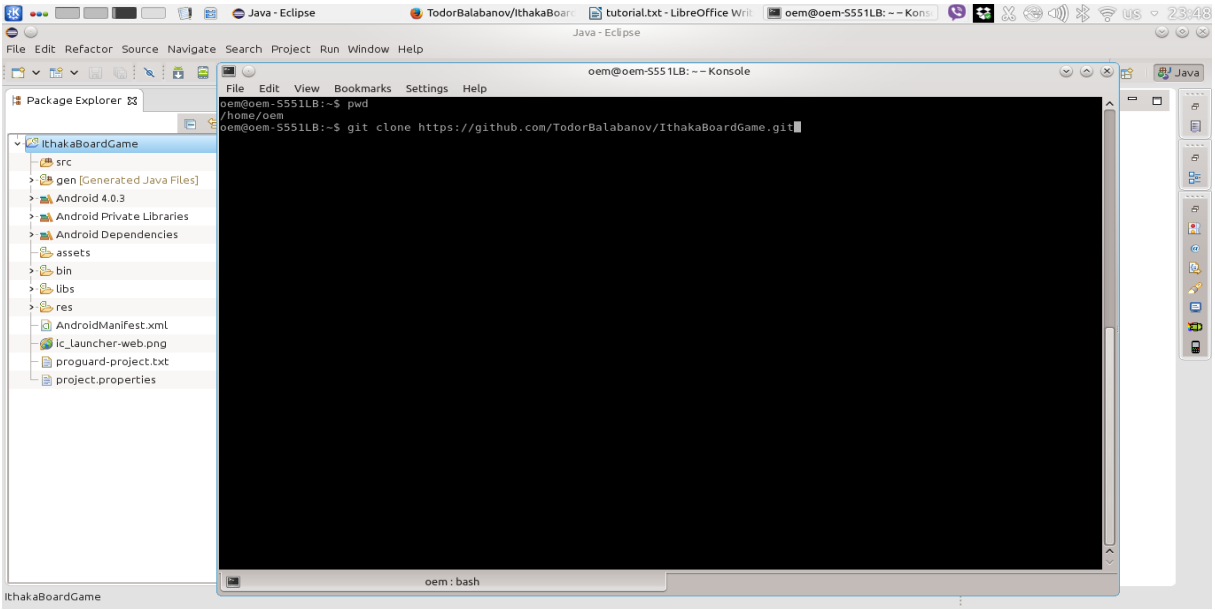
В работното пространство на Eclipse всеки проект има уникално наименование. Също така всеки проект представлява софтуерен продукт, ето защо се въвежда и наименование на продукта. В езика Java софтуерът се организира под формата на пакети и поради тази причина е необходимо избирането на подходящо название за пакет. Названието на пакета трябва да е уникално и да не води до колизия с наименованията на други пакети, дори когато става въпрос за софтуери с глобален мащаб на разпространение. Названието на пакета има и друга ключова отговорност, а именно уникалност при публикуването на готовия продукт в Google Play Store. Точно на тази фаза от разработката се определят и Android версиите, които ще бъдат поддържани.



Фиг. 8 Структура на новосъздаден проект в Eclipse

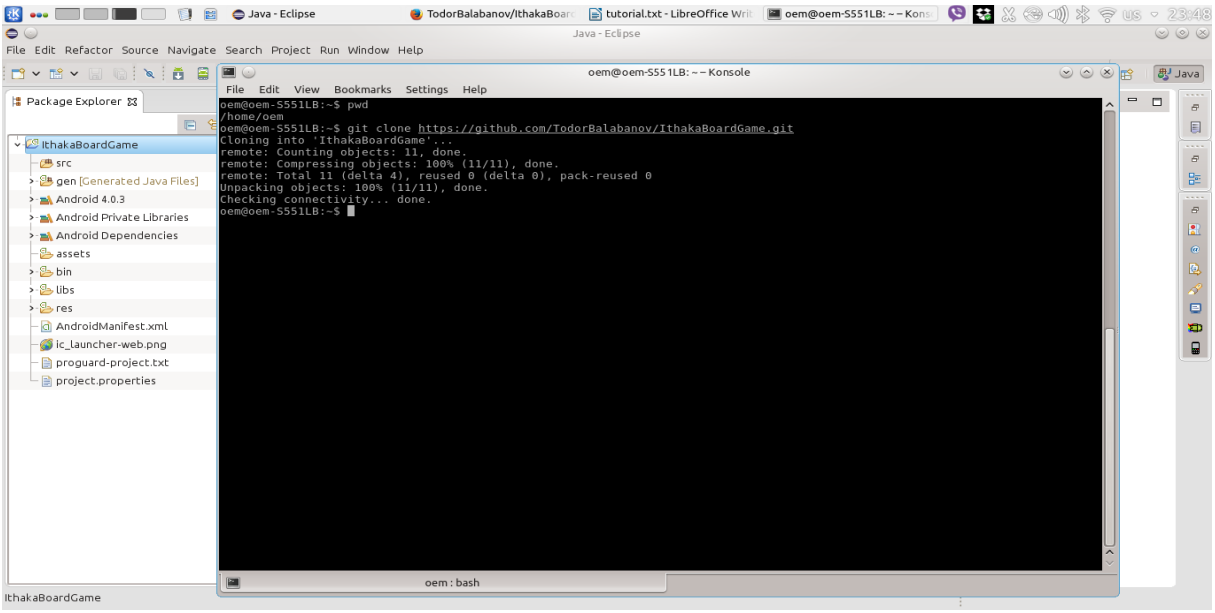
Като икона за предложението се запазват подразбиращите се характеристики и в етапа за създаване на проект не се добавя първоначален прозорец (Activity).

За да бъде добавен, новосъзданият Eclipse проект в отдалеченото хранилище на GitHub е необходимо клониране на отдалеченото хранилище в директория на локалния компютър.



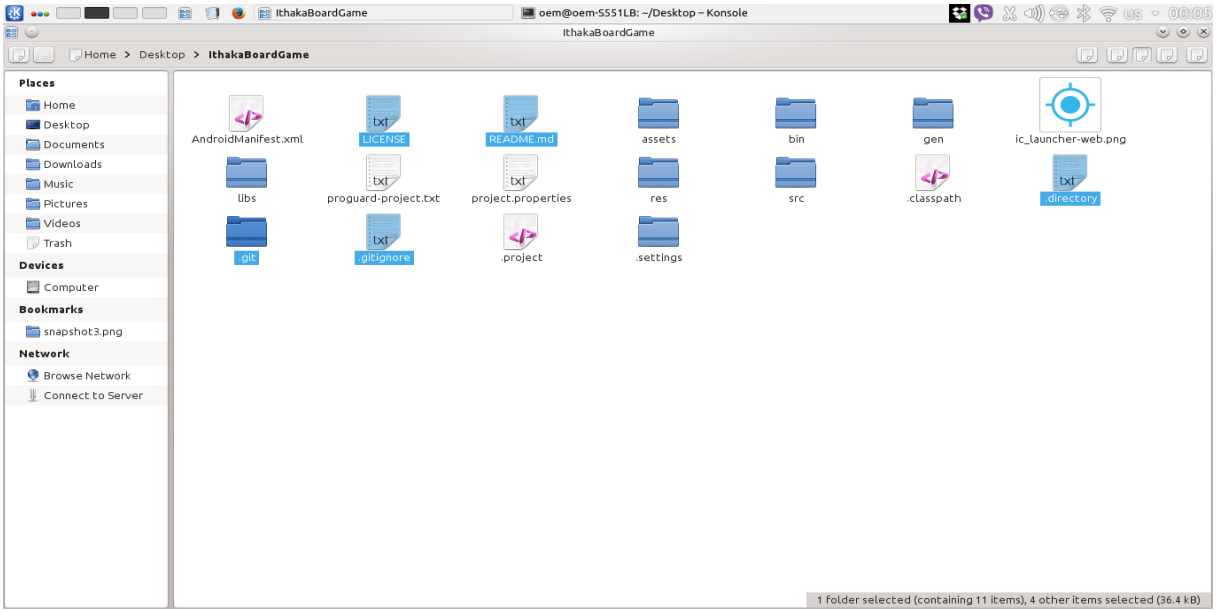
Фиг. 9 Команда за клониране от отдалеченото хранилище на GitHub

В резултат от командата за клониране на работния плот се появява папка с локално копие на GitHub хранилището.



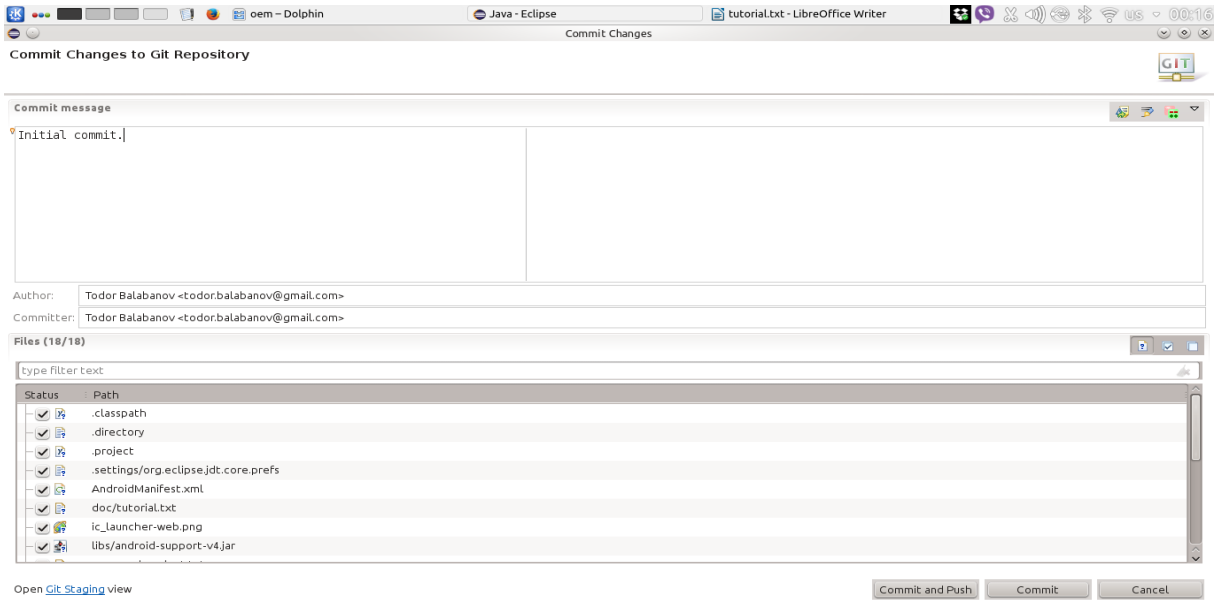
Фиг. 10 Резултат от командата за клониране

Файловете от двете папки (папката, създадена от Eclipse и папката, получена в следствие на процеса за клониране) се обединяват в една обща папка.



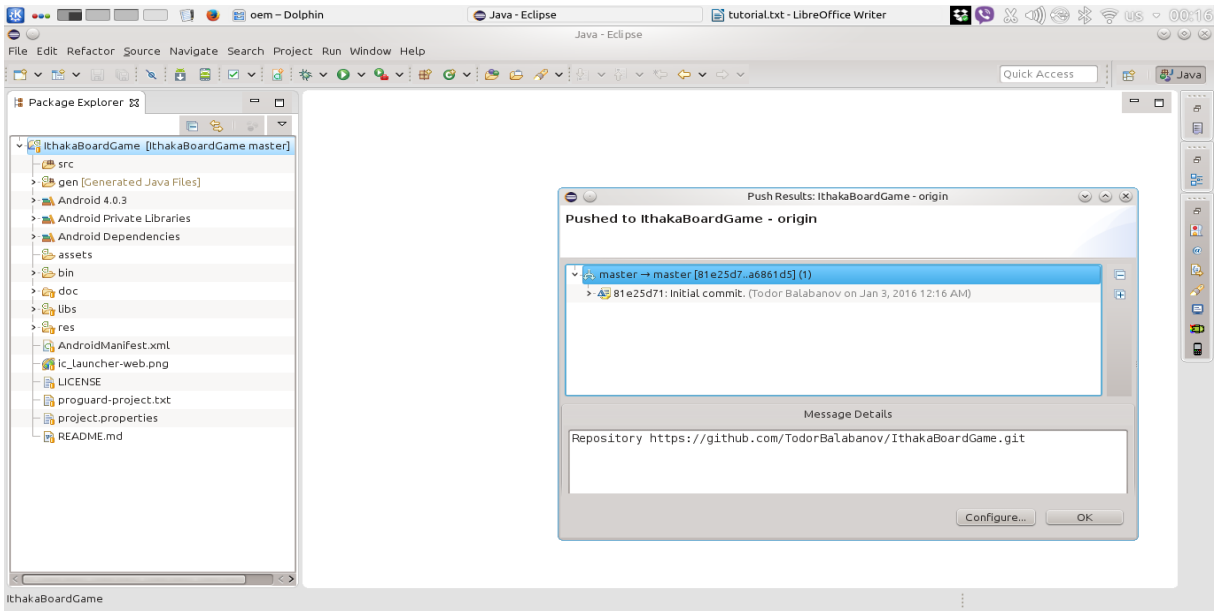
Фиг. 11 Обединяване на файловете за проекта

След обединяването на двете папки не е нужно повече да се използват командните инструменти на Git, а може да се използват вградените Git инструменти в средата Eclipse.



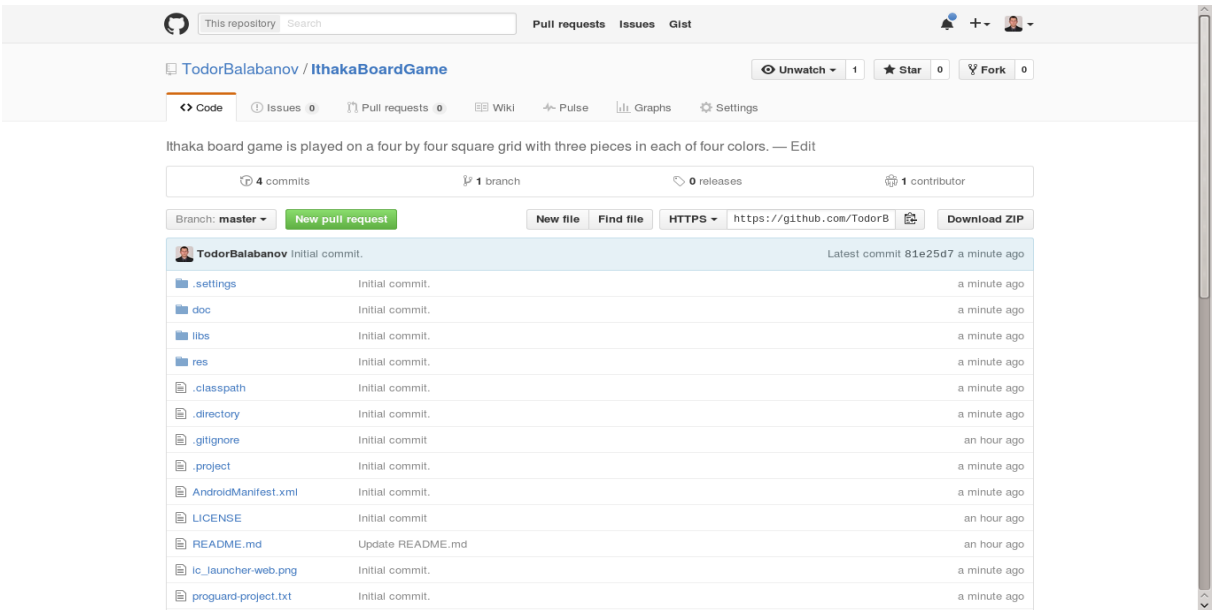
Фиг. 12 Публикуване на промените в локалното Git хранилище

Git е система за контрол на версиите, която работи на две степени (локално и глобално хранилище). Локалното хранилище се ползва при липса на връзка към Глобалната мрежа, докато публикуването в глобалното хранилище прави файловете публично достъпни в Глобалната мрежа.



Фиг. 13 Публикуване на промените в отдалеченото (глобално) хранилище на GitHub

Започването на нов проект с отворен код приключва с публикуването на първоначалната версия в публичното пространство.

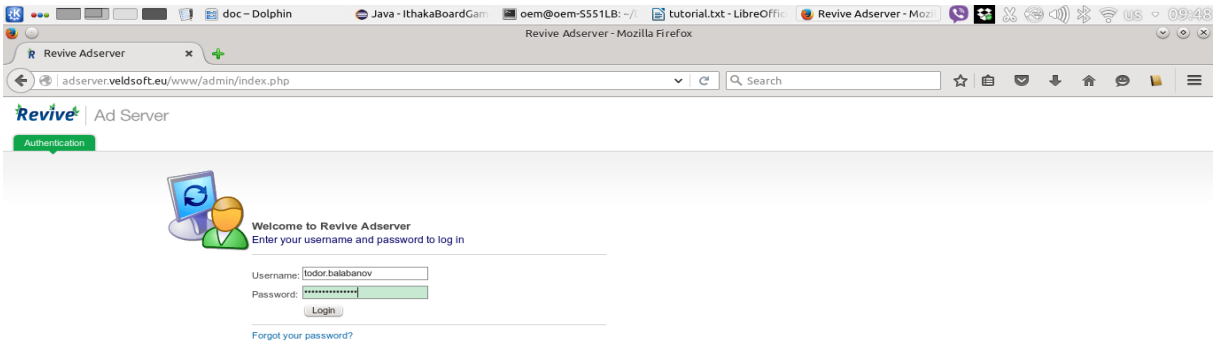


Фиг. 14 Първоначален изглед на новосъздадения Eclipse проект в GitHub

4. Генериране на печалба от проекти с отворен код

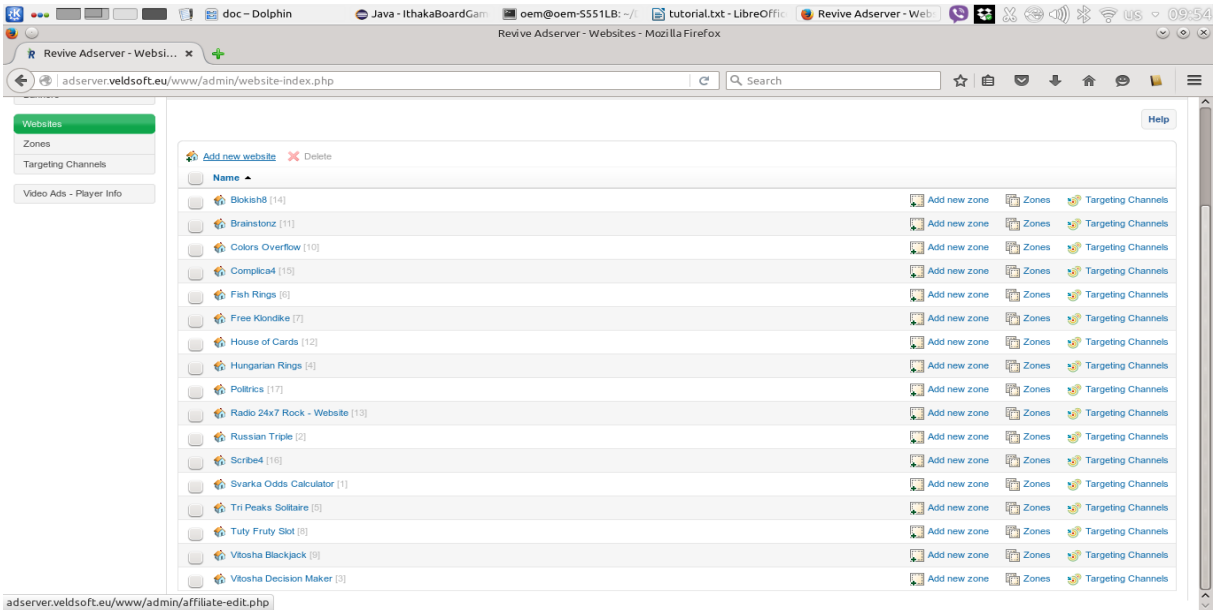
Софтуерът като продукт, дава множество възможности за извършването на търговска дейност. Класическият вариант е свързан с продажба на лицензи за употребата на конкретен софтуер. Съществено е да се отбележи, че обект на търговската сделка е лицензът (правото за ползване), а не самият софтуер. Тъй като по своята природа, софтуерът е малко по-различен продукт от класически продаваните продукти, то той позволява и серия други модели за генериране на финансови приходи. Един значителен дял от софтуерните продукти се разпространява под свободен лиценз, което изключва заплащането, от страна на потребителя, за това че ползва софтуера. При продуктите с отворен код е заложена дълбоката философия за това, че достъпът до знание в съвременния свят, трябва да бъде безплатен. Въпреки това, създателите на софтуер с отворен код е необходимо да генерират по някакъв начин финансови приходи, защото в противен случай не биха могли да поддържат дейността по създаването и поддръжката на софтуера. Най-често срещаният модел за генериране на приходи е на база дарения. Зад едни от най-мащабните софтуерни проекти с отворен код често стоят нетърговски организации, даренията, към които позволяват дейността да бъде финансирана. Вторият много популярен модел е предлагането на поддръжка в замяна на абонамент. При този модел целта на производителите е да създадат софтуерен продукт, толкова масово наложен и да решават определени бизнес потребности, че потребителите да са склонни, да си заплащат услугите по поддръжката. В света на мобилните приложения и двата модела са трудно постижими, особено, когато става дума за малък проект, с малък търговски потенциал. Точно в тази ситуация най-подходящ се оказва третият модел, който залага на реклами, визуализирани пред потребителите. Приходи от реклама биха били най-подходящи за продукта, който е представен в тази разработка.

Съществуват различни възможности за вграждане на рекламни материали в мобилните приложения, под формата на външни услуги. При тези външни услуги производителят на софтуера добавя конкретен модул в приложението си, който осъществява интеграцията със системата за обмен на реклами. Негативният ефект при този вариант е, че производителят става зависим от доставчика на услугата за обмен на реклами. Поради тази причина много по-удачен вариант е използването на собствена система за обмен на визуални реклами.



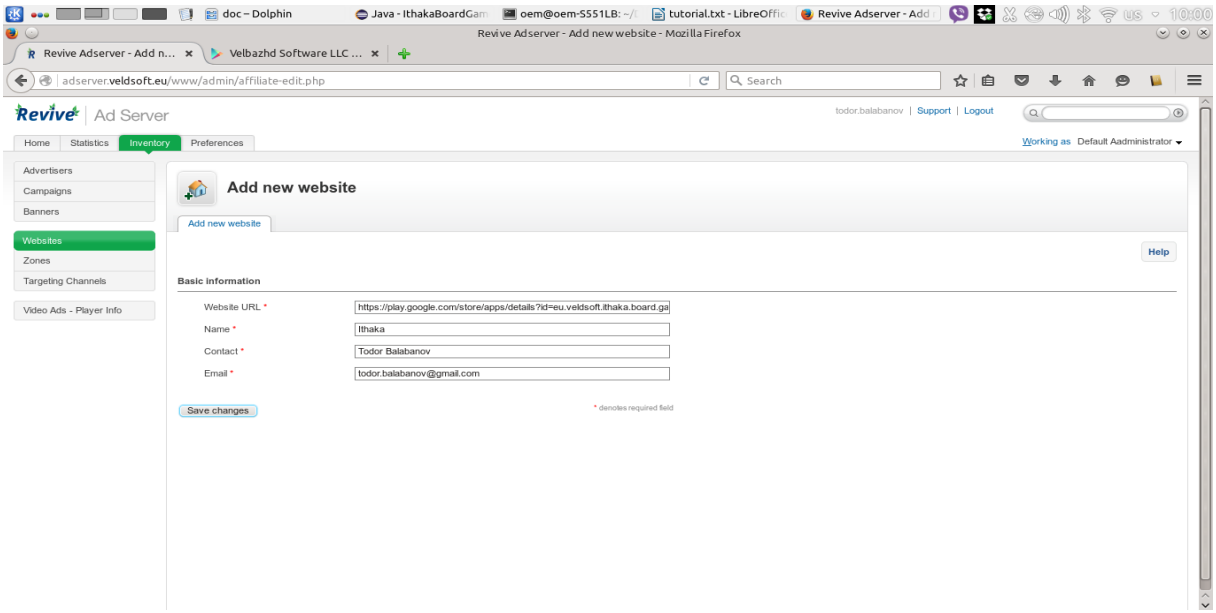
Фиг. 15 Revive Ad Server система за обмен на реклами

Една от най-разпространените платформи за обмен на реклами е Revive Ad Server. Представлява PHP/MySQL сървър скрипт приложение, което се разпространява под свободен лиценз и може да се използва без да се заплаща. Revive Ad Server е предназначен за визуализация на реклами в уеб страници. Съществува и приложение за доставка на реклами в мобилни приложения, но то се разпространява срещу лицензна такса и не би било икономически изгодно при разработването на платформа от софтуер с отворен код. За нуждите на настоящата разработка ще бъдат използвани възможностите на Android платформата за визуализация на цялостни уеб страници.



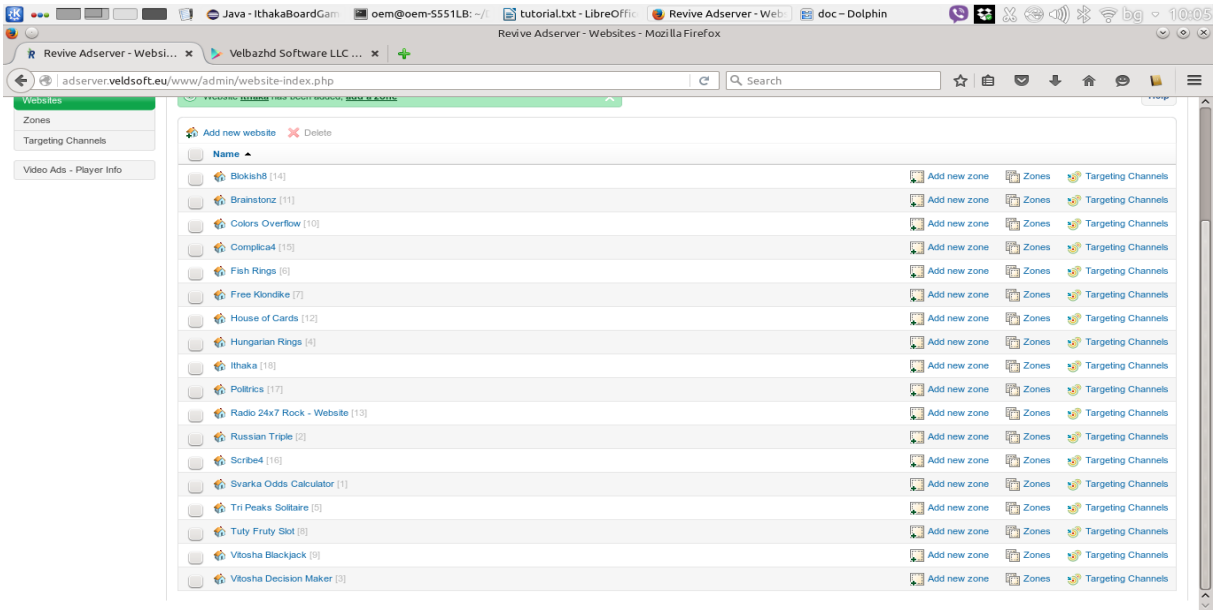
Фиг. 16 Revive Ad Server организация на уеб сайтове

Организацията на рекламния сървър е основана на поддръжката на множество уеб сайтове. В случая на мобилните приложения за всяко мобилно приложение съществува профил в рекламния сървър.



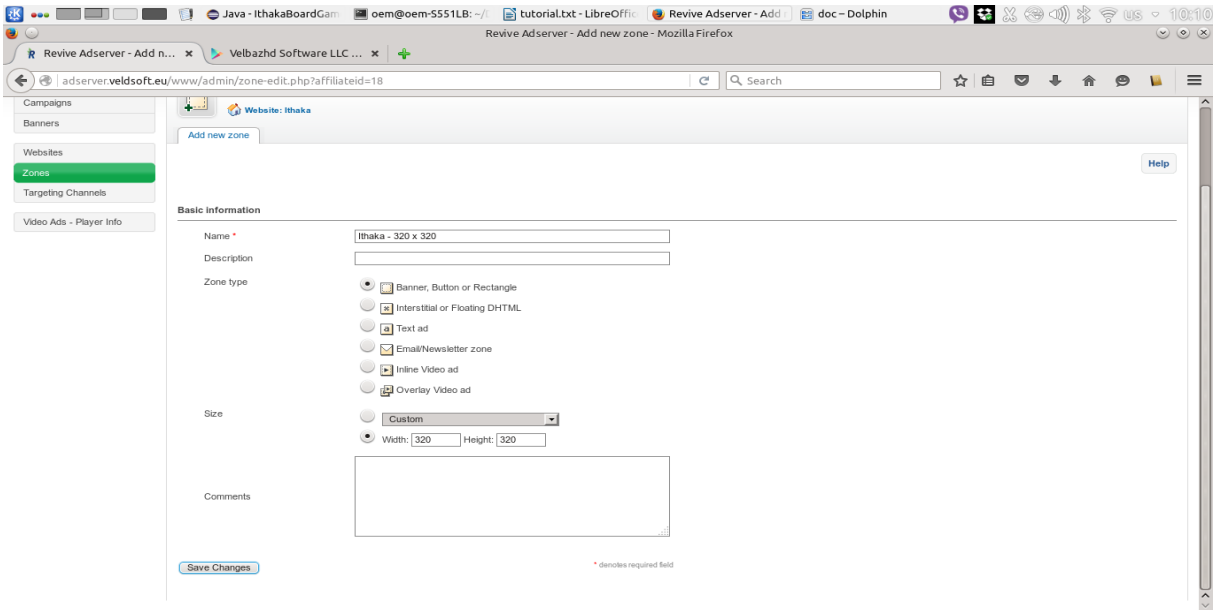
Фиг. 17 Revive Ad Server профил на мобилното приложение Ithaka

В профила за мобилното приложение се въвежда името (в случая Ithaka), лицето за контакти, което е отговорно за изпълнението на рекламни кампании в това приложение и URL адрес. При използването на безплатната версия, на Revive Ad Server могат да се обработват само уеб сайтове, които от своя страна се характеризират с URL адрес. Тъй като мобилните приложения не притежават URL адрес, то най-логично е да се въведе URL адрес, характеризиращ по някакъв начин мобилното приложение. Такъв URL адрес е точно адресът, на който ще бъде публикувано приложението в Google Play Store.



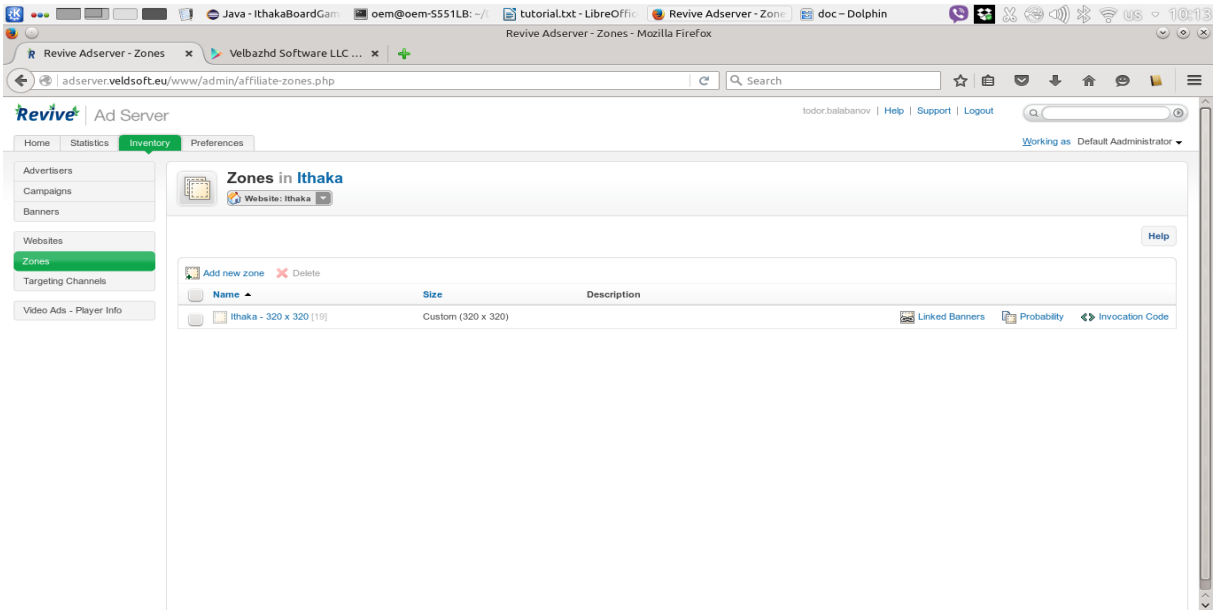
Фиг. 18 Профил на мобилното приложение Ithaka след създаването му

Разделянето на уеб сайтове е на първично ниво в рамките на рекламния сървър. На вторично ниво всеки уеб сайт се характеризира с конкретни зони за визуализация на реклами.



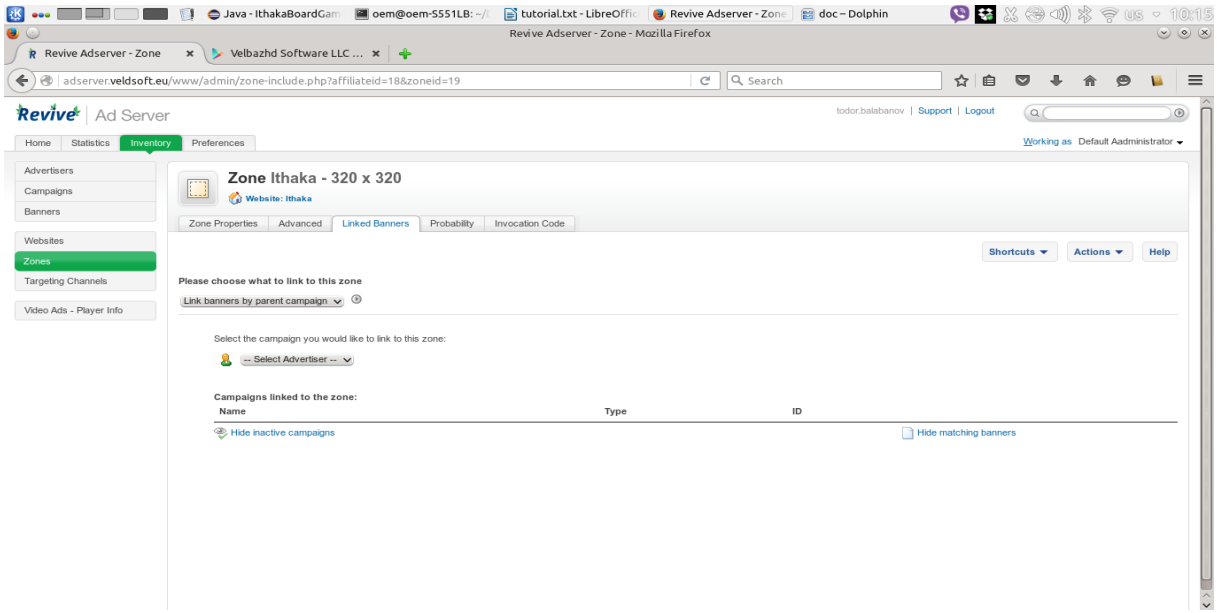
Фиг. 19 Основна зона за реклама в стартовия прозорец на играта Ithaka

Различните уеб сайтове могат да имат различни петна, в които да се показват изображенията на рекламата. Това важи и за мобилните приложения. В различни екрани на мобилното приложение могат да се показват различни по размер и вид рекламни изображения.



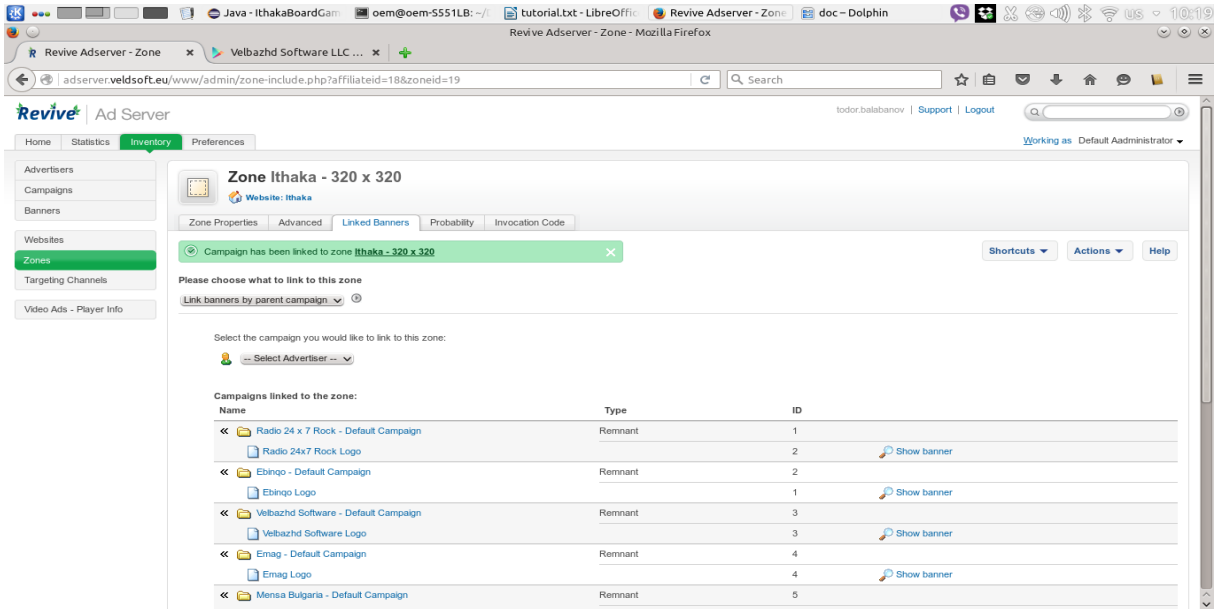
Фиг. 20 Единствена зона за реклами в играта Ithaka

В играта Ithaka ще има само една зона за реклама и това ще е първоначалният екран при зареждане на приложението. Поради наличието на различни по размер и вид екрани на мобилни устройства един от най-удачните размери за реклами е 320x320.



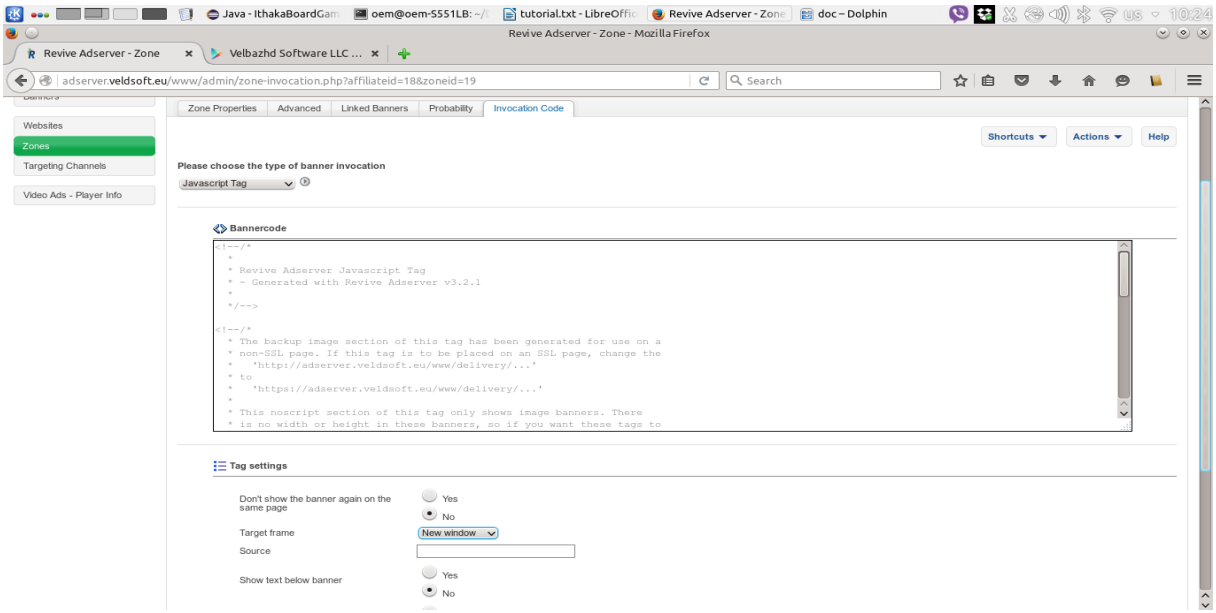
Фиг. 21 Празна зона, без закачени рекламни банери

Към всяка обособена зона трябва да се асоциират определени визуални реклами (банери).



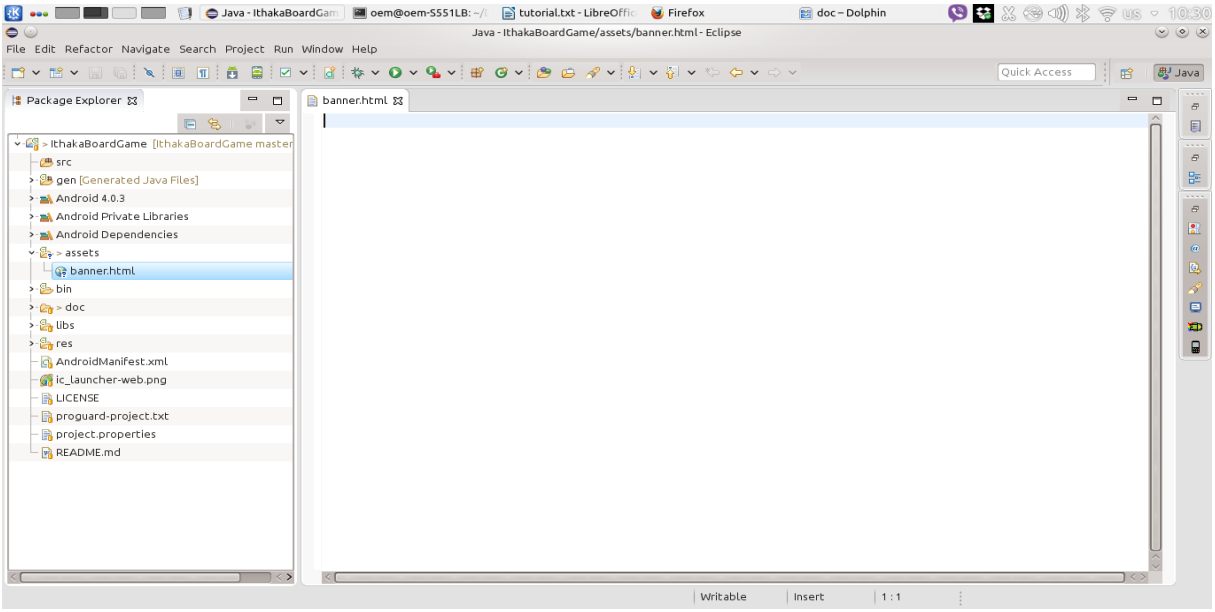
Фиг. 22 Множество от асоциирани банери

Към обособената зона могат да се асоциират конкретни банери, което дава възможност за контрол на съдържанието и подбиране на тематична реклама според вида на аудиторията, която използва конкретното приложение.



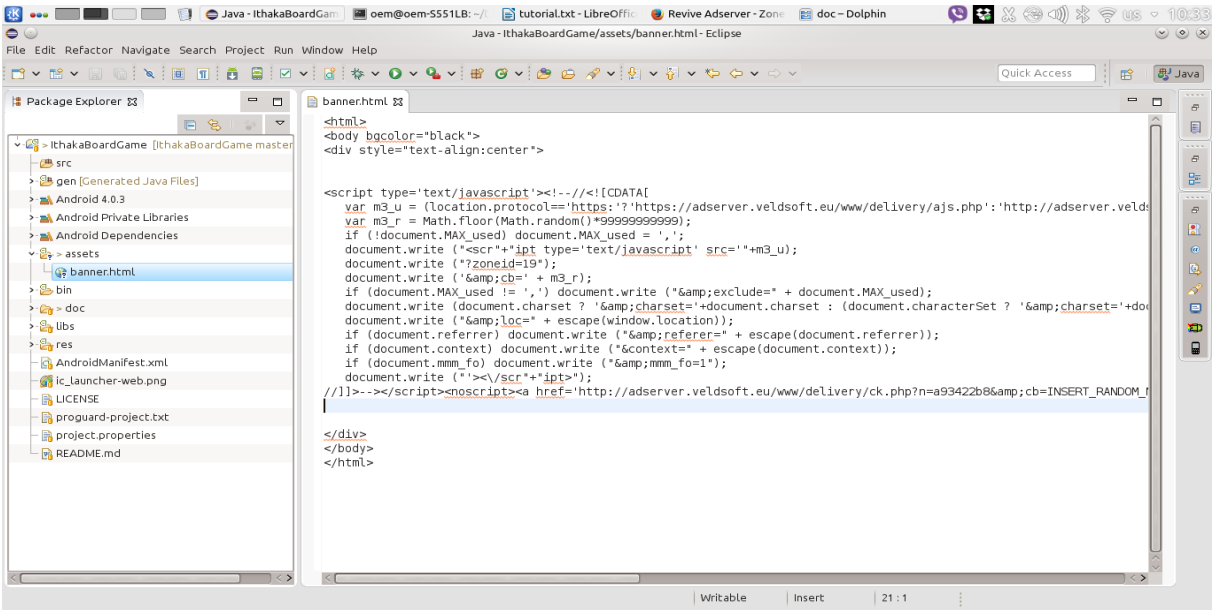
Фиг. 23 HTML код за вграждане в уеб страница

Конфигурирането на рекламния сървър завършва с генериране на HTML код, който трябва да се разположи в уеб страницата, която ще визуализира рекламата.



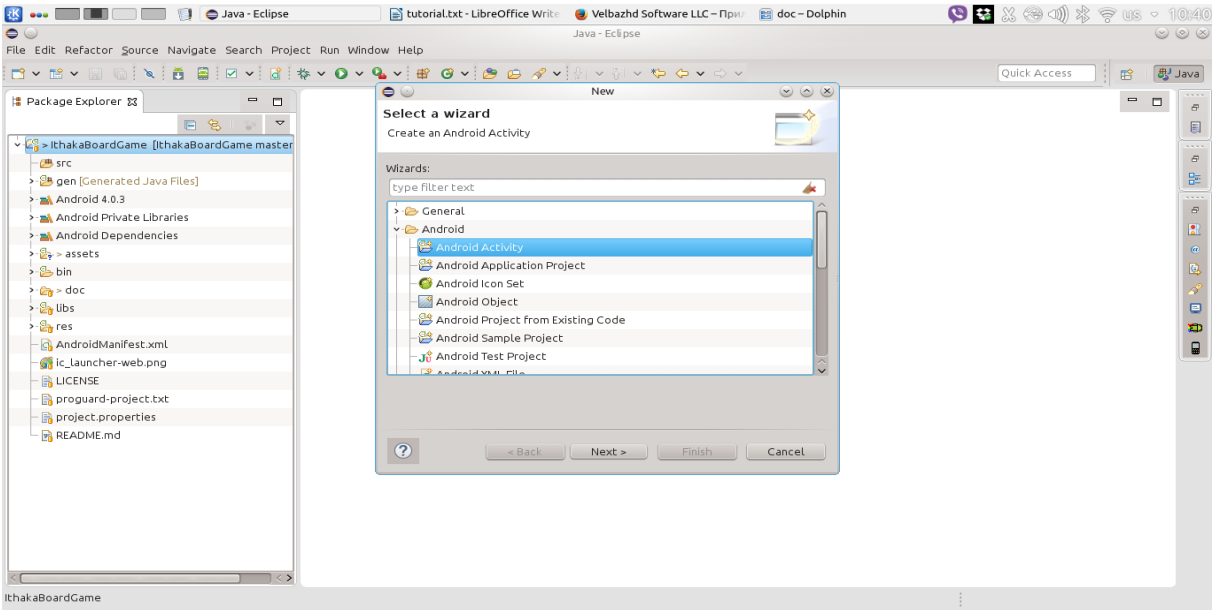
Фиг. 24 Локална HTML страница в пакета на Android приложението

Android платформата позволява да се съхраняват локални ресурси (в това число и локални HTML страници) в специално предназначена за този случай директория – assets. За име на локалния файл е избрано banner.html.



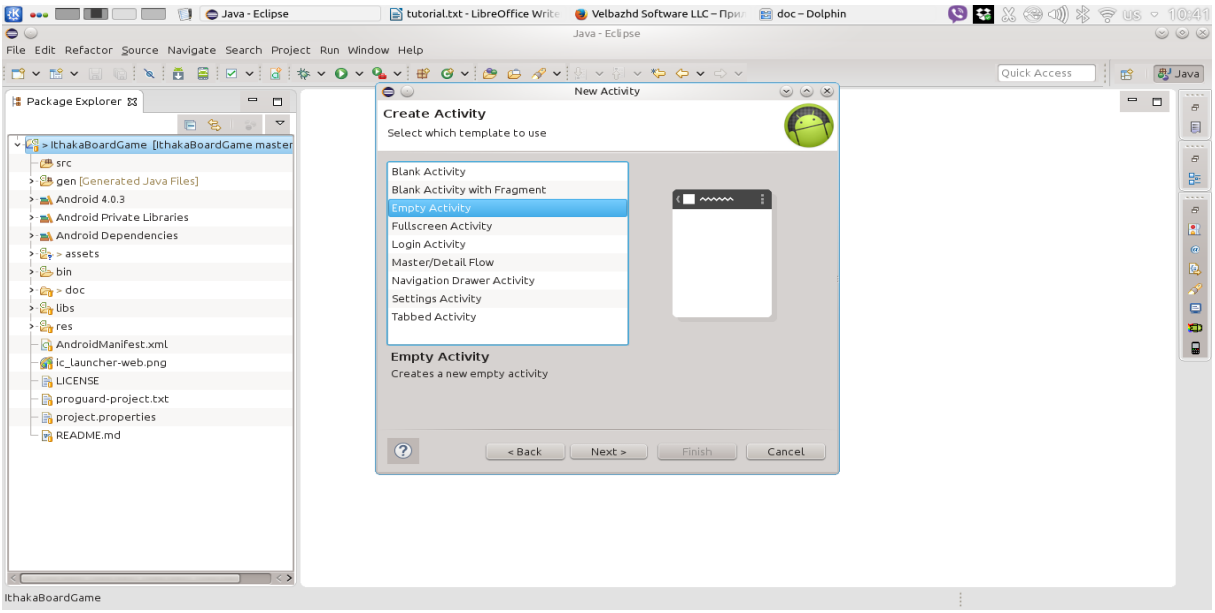
Фиг. 25 Програмен текст на локалната HTML страница

Съдържанието на локалната уеб страница е изключително тривиално и се състои от шаблон на HTML5 документ, в тялото на който е разположен генерираният от Revive Ad Server програмен код.



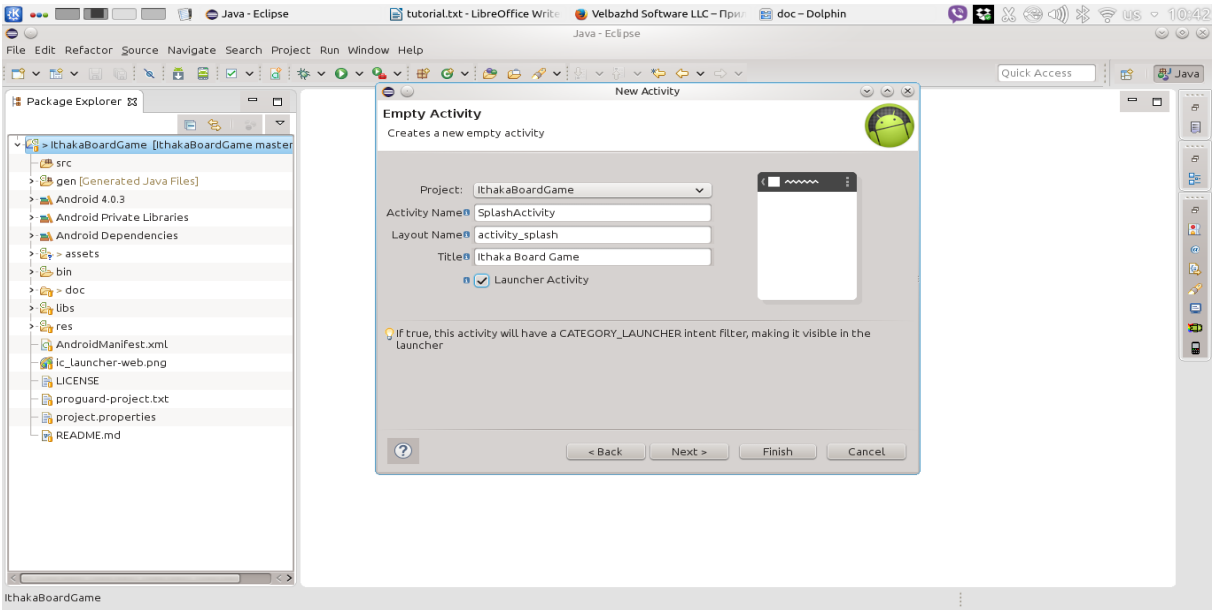
Фиг. 26 Създаване на начален екран за визуализиране на рекламни изображения

Android приложенията са организирани на работни екрани, наречени Activity. Идеологията е такава, че всеки работен екран може да се активира по всяко време, без да се използва една-единствена точка за стартиране (single entry point). Операционната система активира избран прозорец при начално стартиране, като тази задача се изпълнява от процес, наречен Launcher.



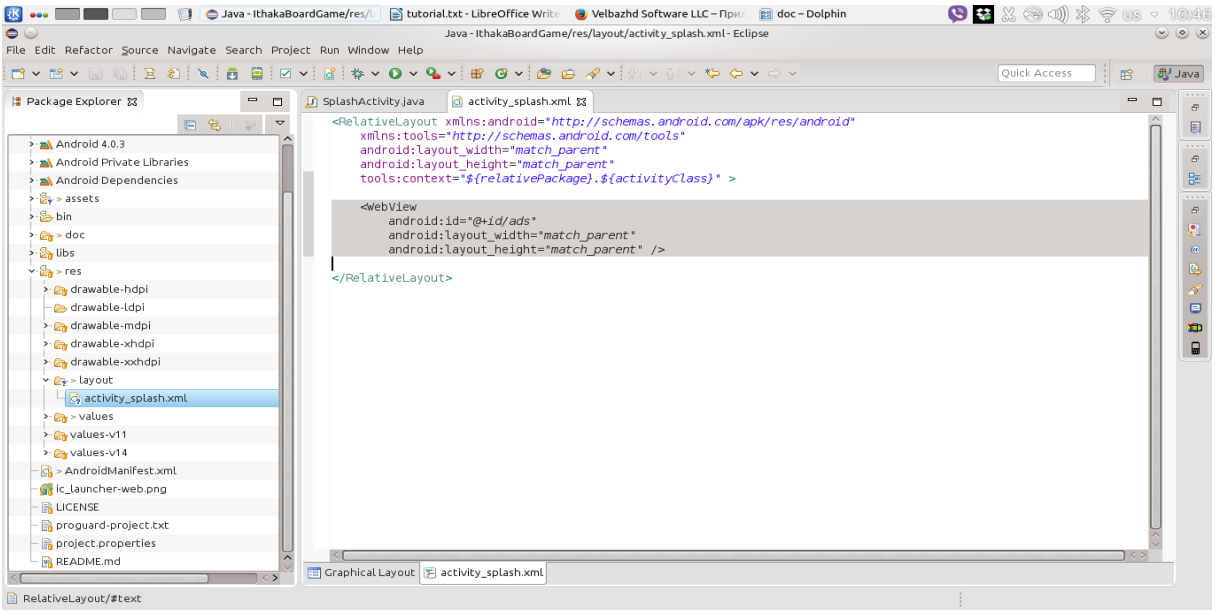
Фиг. 27 Задаване на празен екран

Най-подходящият шаблон, от списъка с възможни екрани, е празният. В този празен екран ще бъде поместен само един визуален компонент, изпълняващ ролята на уеб браузър.



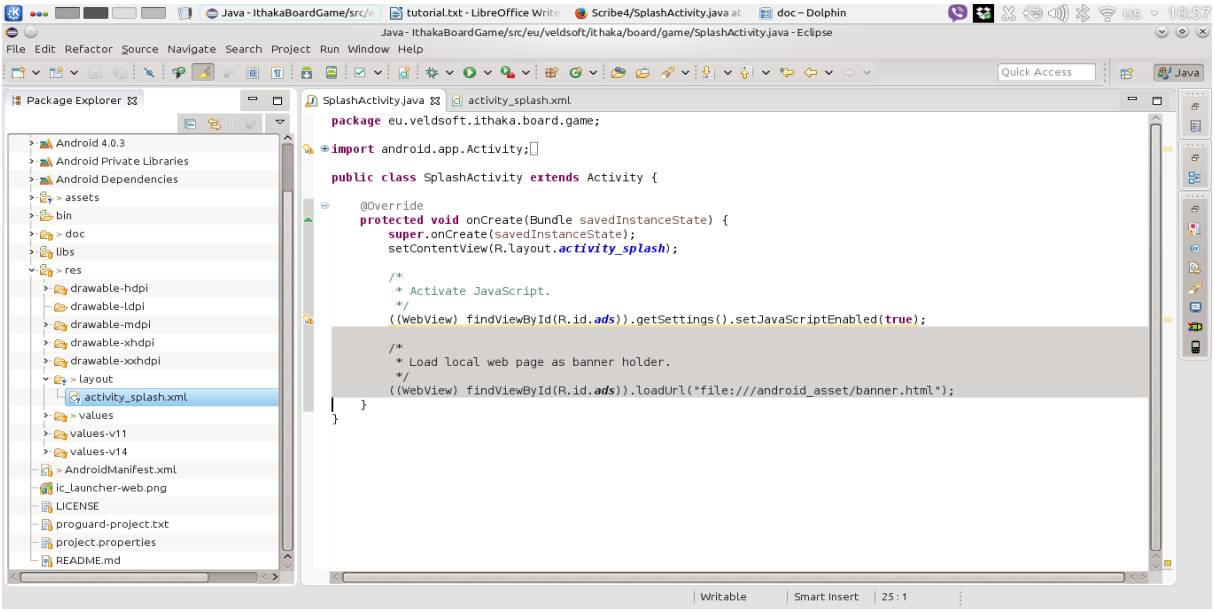
Фиг. 28 Маркиране на прозореца като стартов

Всяко Activity в Android се описва от файл за графичния интерфейс (XML формат) и Java файл с програмен код.



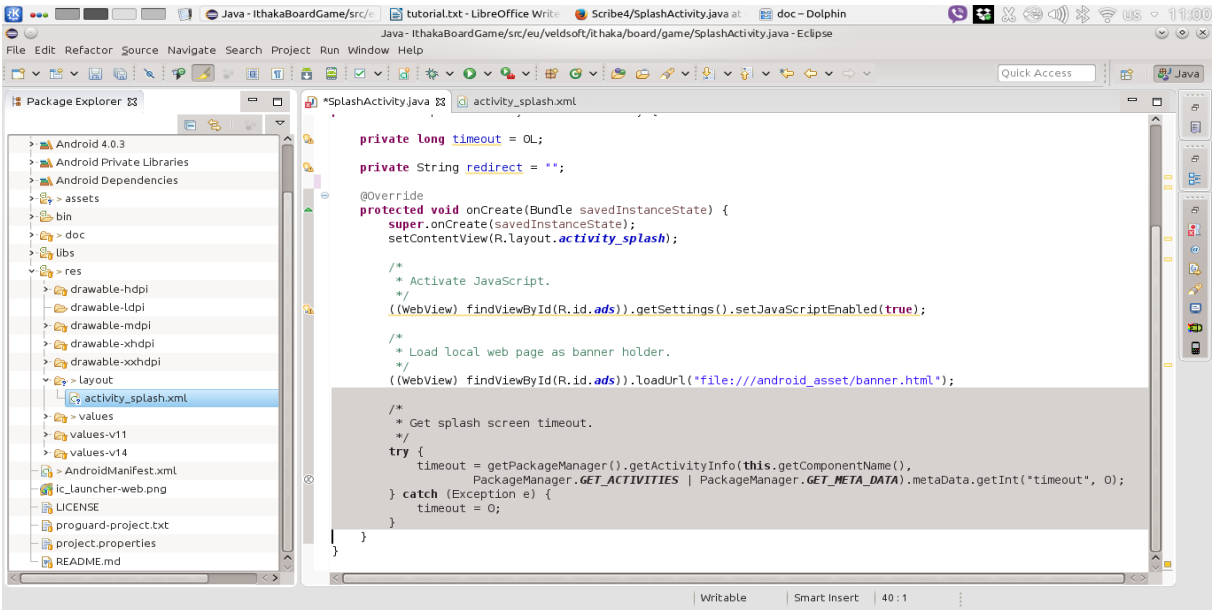
Фиг. 29 Вграждане на уеб браузър компонент в началния прозорец

WebView е стандартен визуален компонент от палитрата с компоненти за изграждането на графичен потребителски интерфейс в Android.



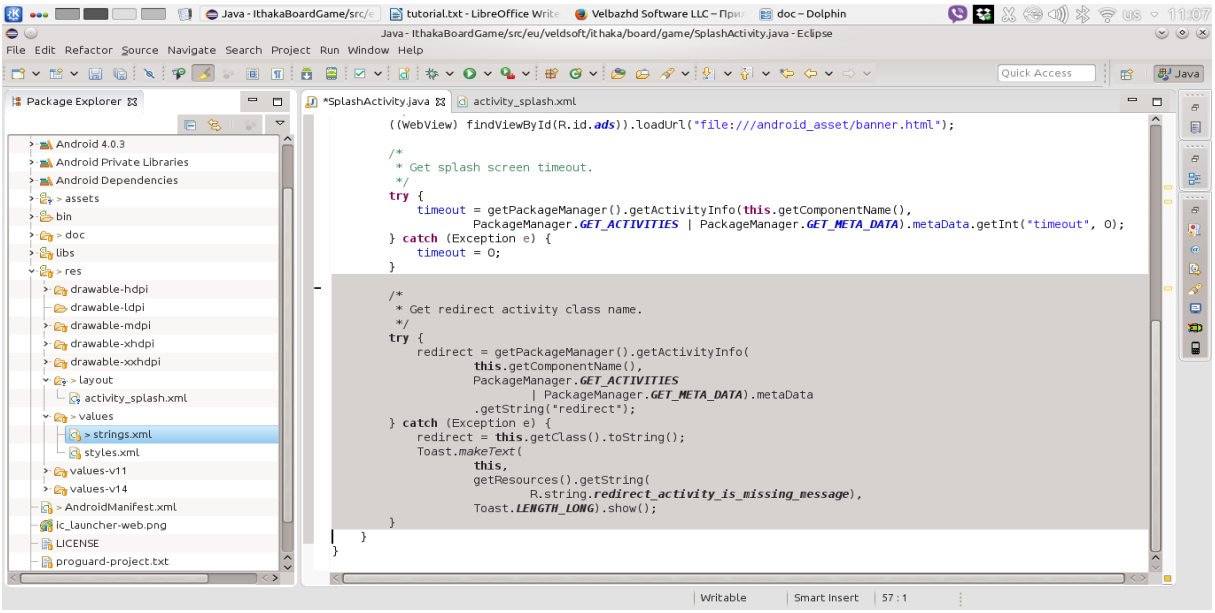
Фиг. 30 Връзка за програмно управление на вградения браузър

За да се използва уеб браузър компонента в програмния файл на Java е необходимо да се получи референция към обекта, който операционната система ще създаде в процеса на стартиране на приложението. Още при зареждането се указва използване на JavaScript, тъй като рекламното съдържание се изтегля от сървъра с помощта на JavaScript код. Също така се подава и URL адресът на локално съхранената уеб страница.



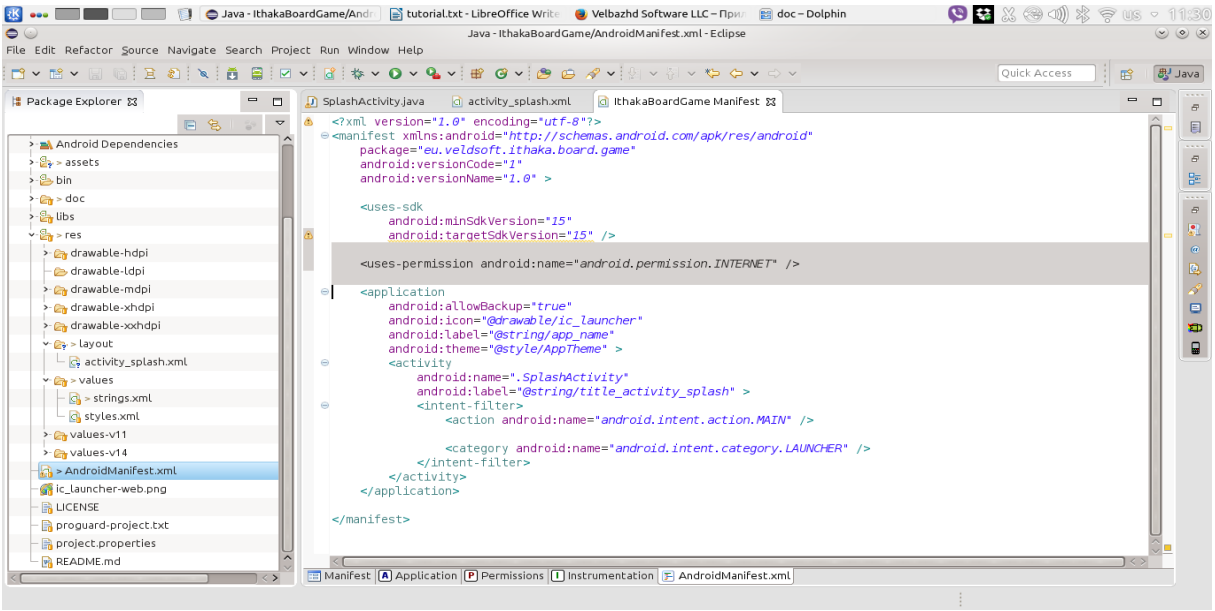
Фиг. 31 Времетраене на визуализацията

Екранът, показващ рекламните изображения, получава две характеристики от външната среда (в случая манифест файла). Първата характеристика определя колко милисекунди да се визуализира рекламата, а втората характеристика определя кой екран да бъде визуализиран в последствие.



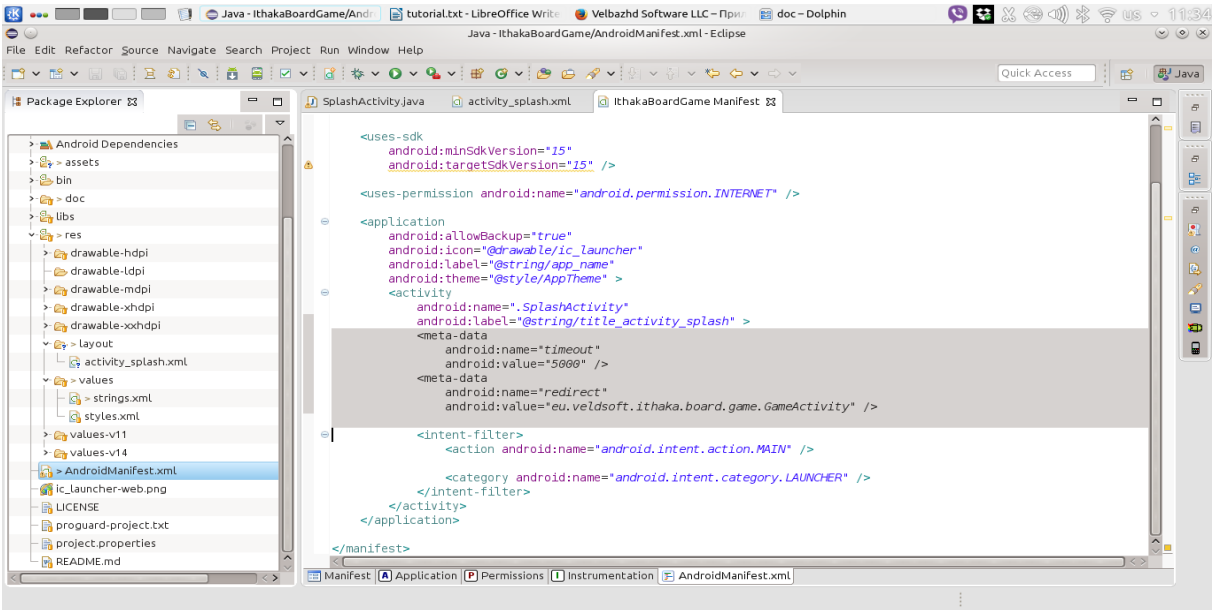
Фиг. 32 Пренасочване към следващ прозорец

След изтичането на определеното време за визуализация се изпълнява код за пренасочване към основния екран на играта.



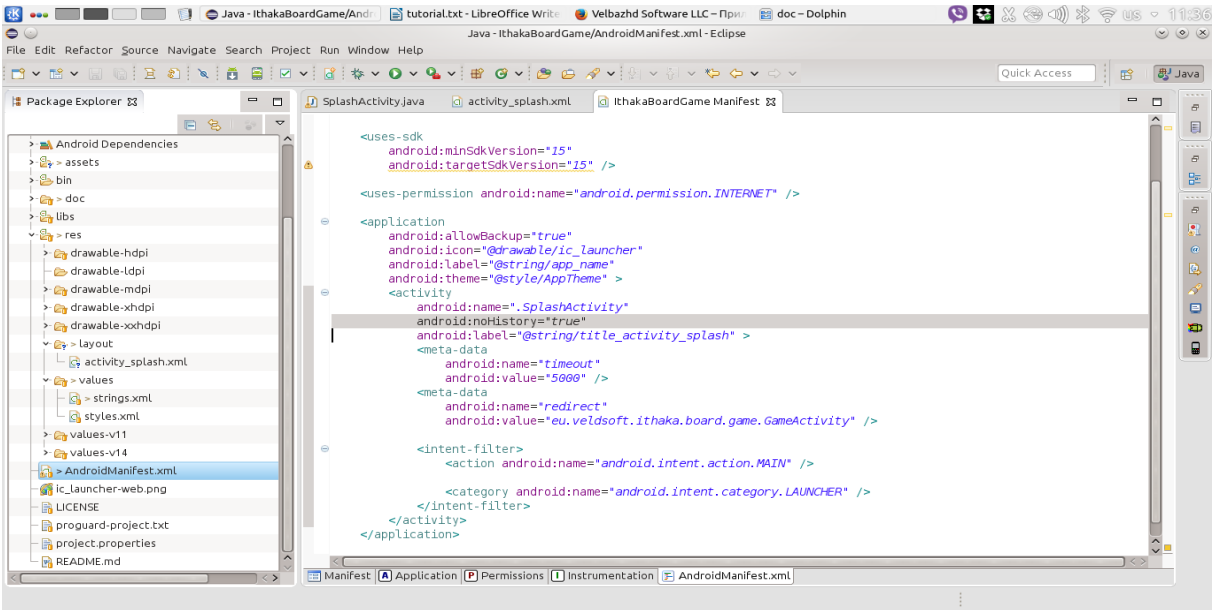
Фиг. 33 Разрешение за използване на Интернет връзка

Моделът за сигурност на Android изисква потребителят да се съгласи с всички правила за достъп до устройството при първоначално инсталиране на приложението. За да се получават и визуализират рекламните изображения е необходимо приложението да използва Интернет свързаност.



Фиг. 34 Параметри при извикване на прозорец

Прозорците в Android могат да получават определени параметри (без да се налага прекомпилиране на Java кода) - чрез въвеждане на параметрите в описателната част за прозореца в манифест файла на приложението.

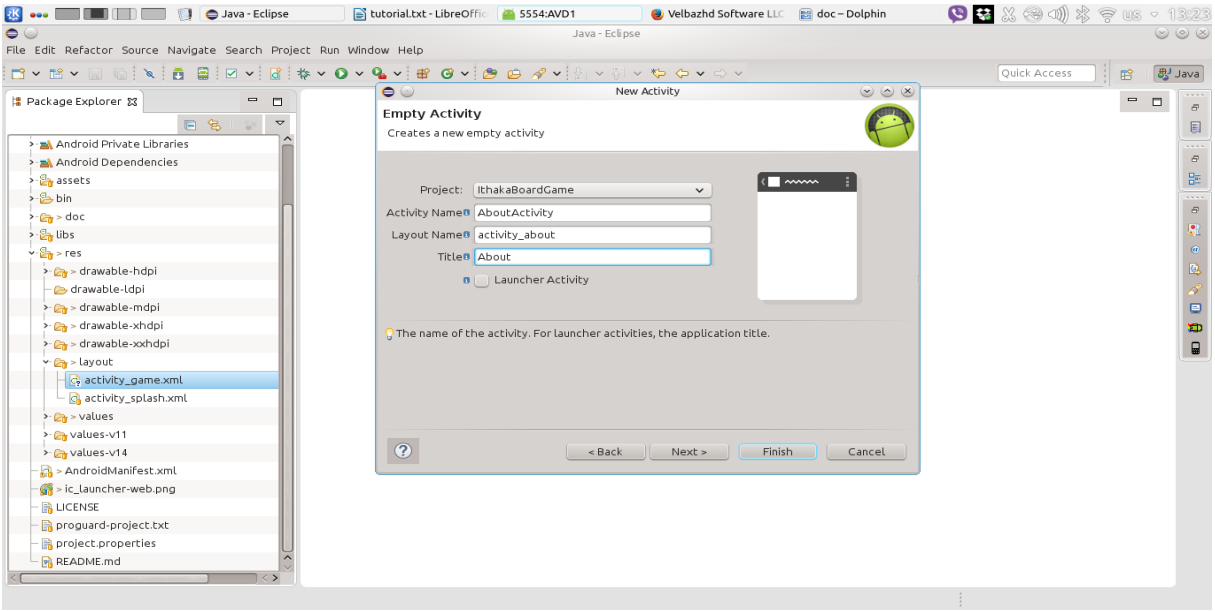


Фиг. 35 Поведение при затваряне на прозореца

Последната стъпка от създаването на рекламния прозорец е да се укаже, че той няма да попада в стека от активирани прозорци. Операционната система поддържа стек с всички активирани прозорци. Когато потребителят се връща (често има хардуерен бутон за връщане назад), предходните прозорци се вадят от стека. Целта при показването на реклами в настоящия проект е рекламата да се визуализира само един път и да не се активира, когато потребителят прелиства прозорците назад.

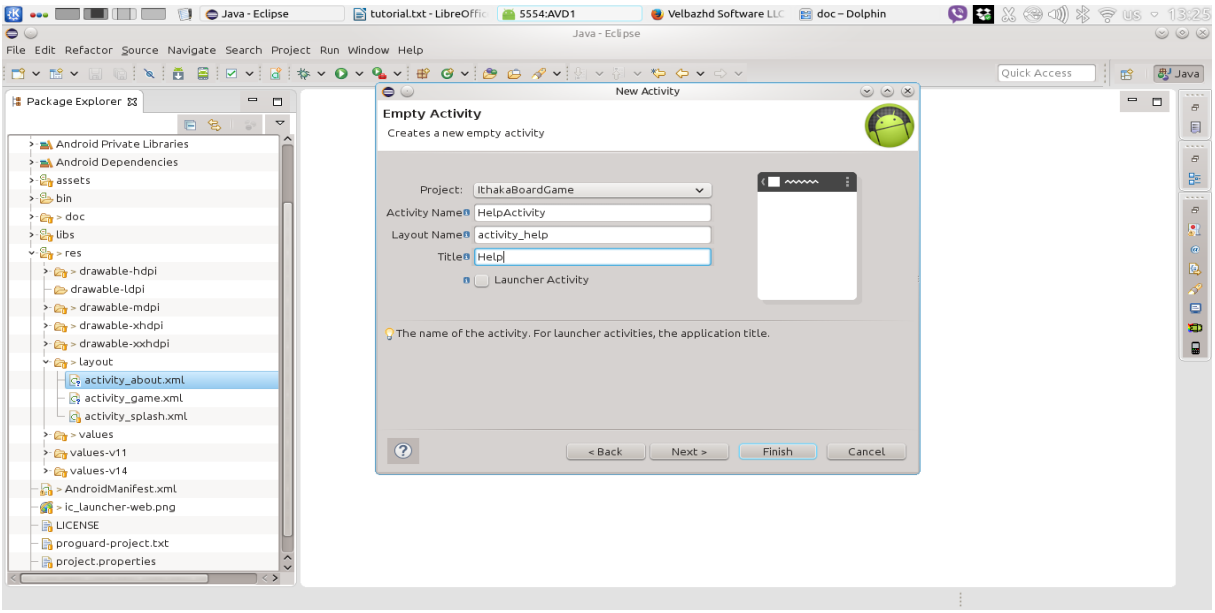
5. Визуализиране на помощна информация

По отношение на информацията, едни от най-важните компоненти в софтуерните продукти са названието на продукта, носителите на авторските права и разпространителите. Почти всеки софтуерен продукт притежава под някаква форма (диалогов прозорец или текстово съобщение) тази информация. Също така от голямо значение е предоставянето на някаква помощна информация. При софтуерни продукти, които се разпространяват с отворен код, наличието на изчерпателна помощна информация далеч не е толкова важно, тъй като целият код на приложението е публично достъпен и всеки възникнал въпрос може да се разследва директно в оригиналните програмни кодове. При софтуерните продукти със затворен (комерсиален) лиценз, това далеч не е така и там се налага предоставянето на изчерпателна потребителска документация за експлоатацията на продукта.



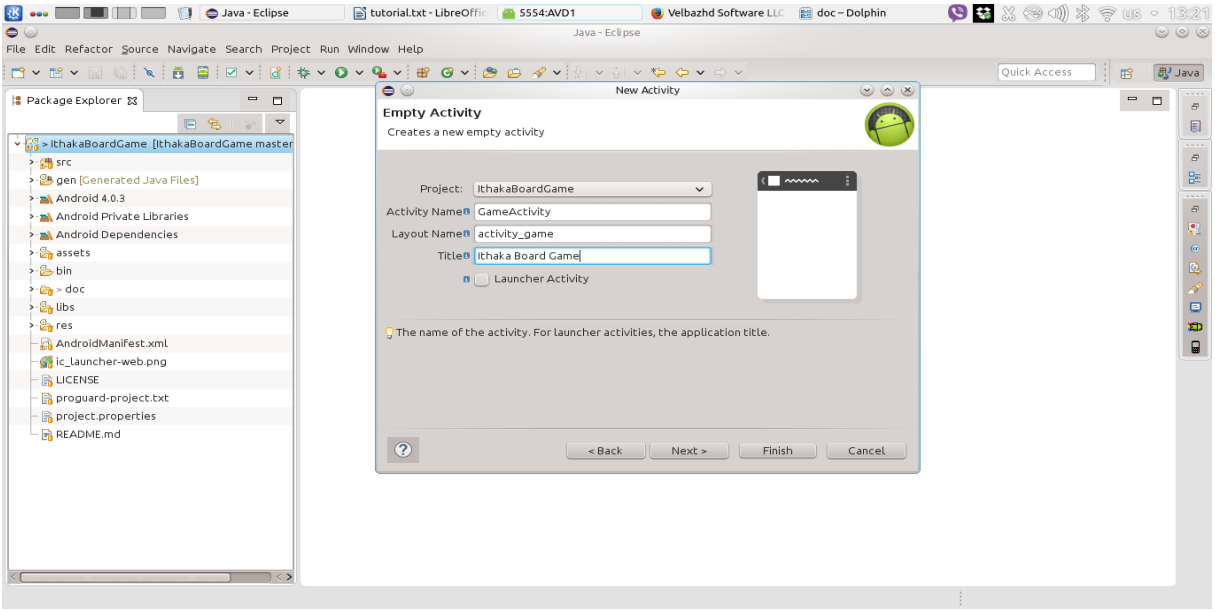
Фиг. 36 Създаване на информационен прозорец за продукта

В настоящия програмен продукт ще бъдат създадени два отделни прозореца (Activity), като единият ще съдържа информацията за продукта, а другият - минимално количество помощна информация (правилата на играта).



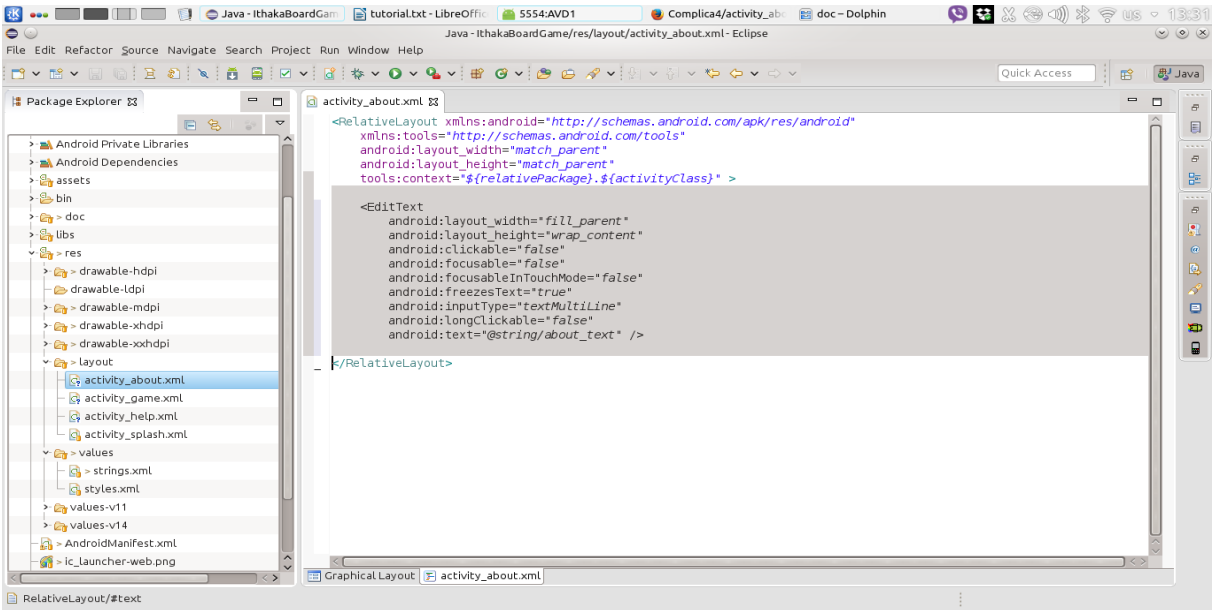
Фиг. 37 Създаване на прозорец с помощна информация

Двата помощни прозореца се извикват от основния екран на играта, за който също се създава отделно Activity.



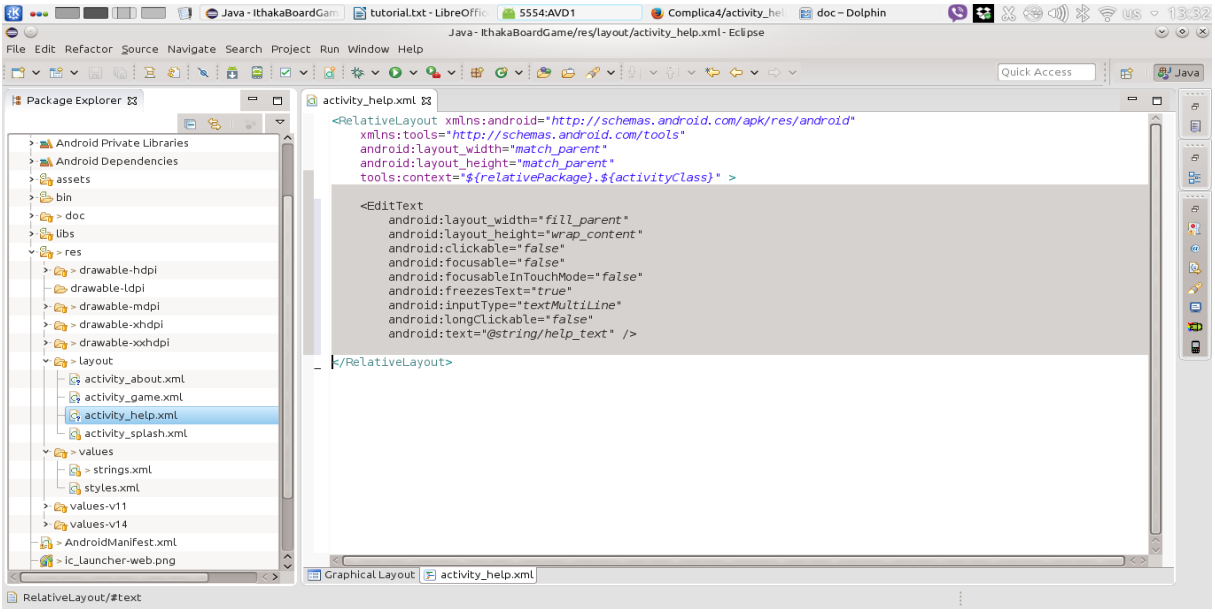
Фиг. 38 Създаване на основен прозорец за играта

Най-бързият и лесен начин да се създадат информационните прозорци е като в ресурсния им XML файл, се добави EditText визуален контрол, който е забранен за писане.



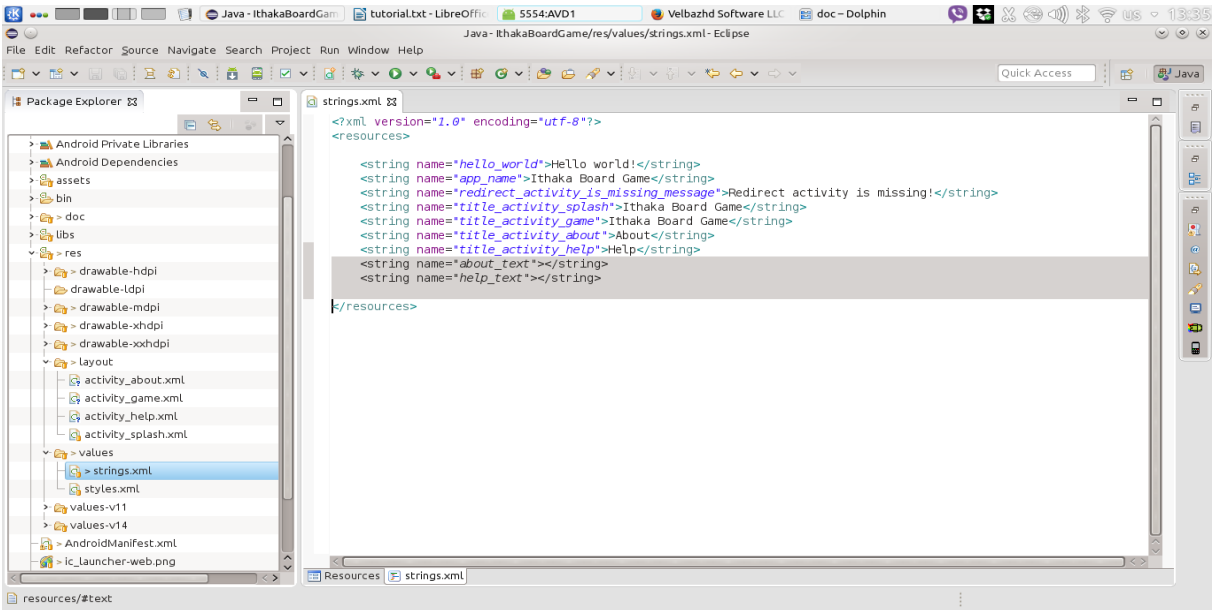
Фиг. 39 Текстов компонент, който визуализира информацията за продукта.

Единствената разлика между двата текстови контрола е референцията към предварително зададен текст в XML файла за текстови ресурси.



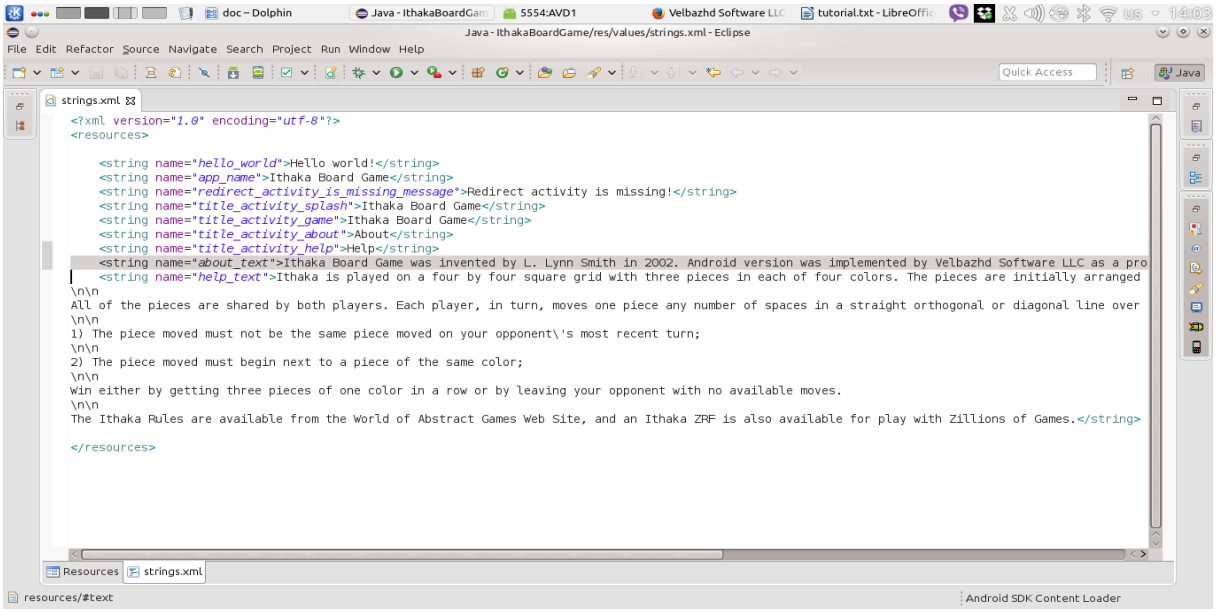
Фиг. 40 Текстов компонент, който визуализира помощната информация.

Платформата Android организира текстовите ресурси в специално отреден за тази цел XML файл. Тази организация е въведена с цел максимално лесно превеждане на текстовете в приложението към множество други говорими езици.



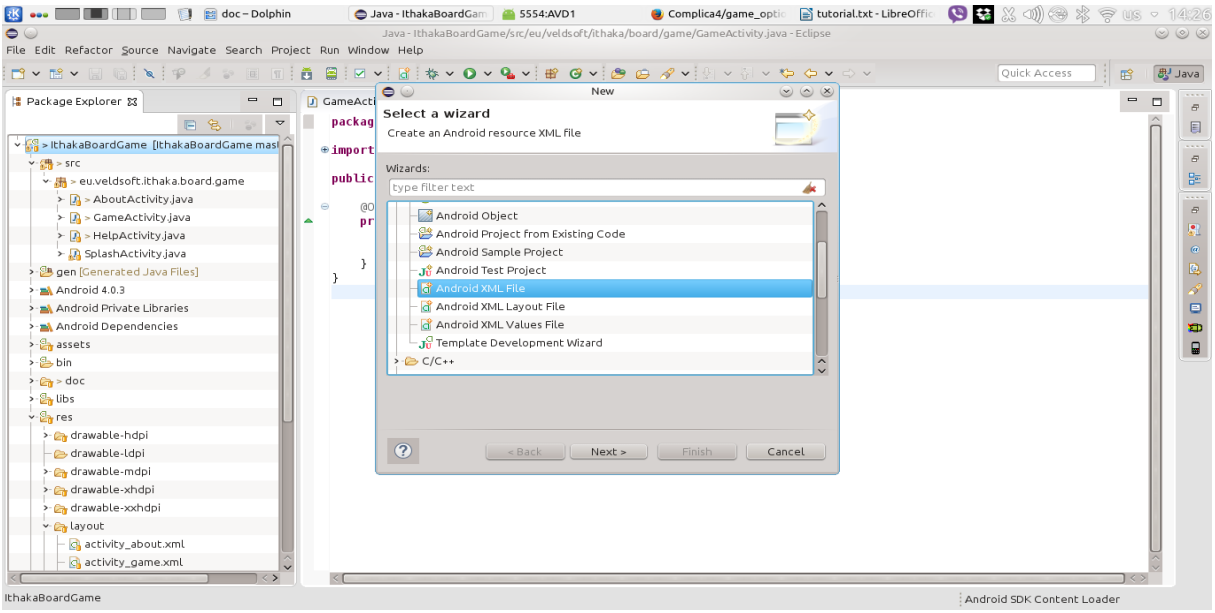
Фиг. 41 Текстови ресурси за продукта и помощна информация

Тъй като разработваният софтуерен продукт е изключително елементарен, то е достатъчно в помощната информация да се публикуват правилата на играта.



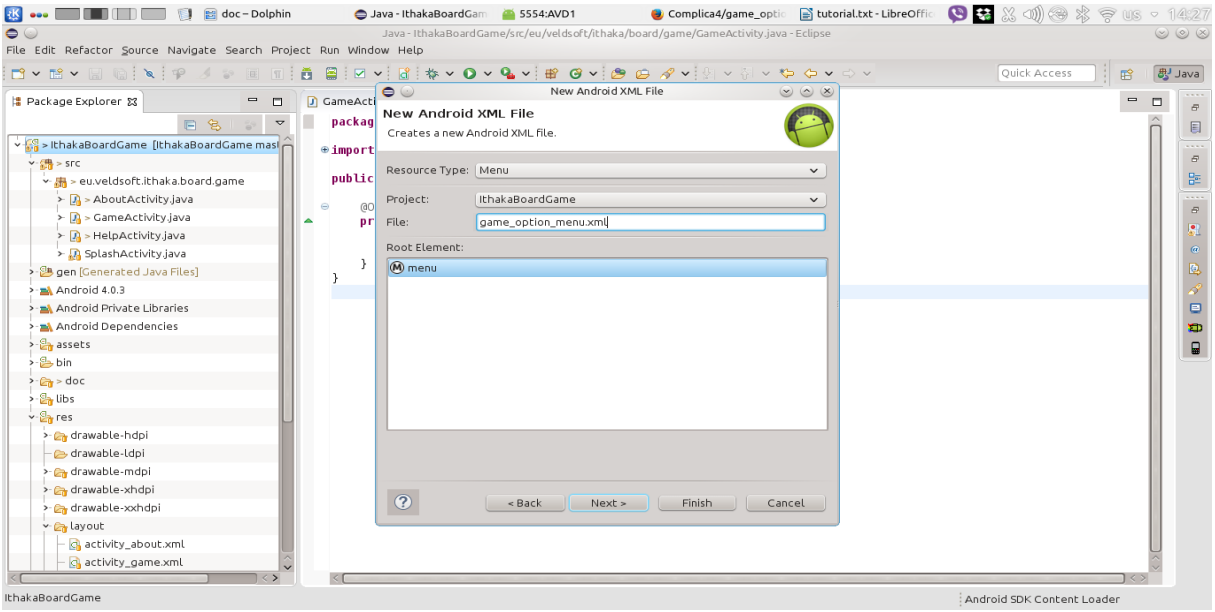
Фиг. 42 Правилата на играта и информация за продукта, под формата на текстови ресурси

От съществено значение е да се отбележи, че представянето на текстовата информация в ресурсен файл води до някои особености, а именно използване на специални символи за нов ред (\n – както е в програмния език Java) и символа апостроф (\' - както е в програмния език Java). Използването на специалните символи за изписване в Java се налага, тъй като XML ресурсите биват компилирани до Java низове и там от значение са правилата за представяне на текстова информация в Java символни низове.



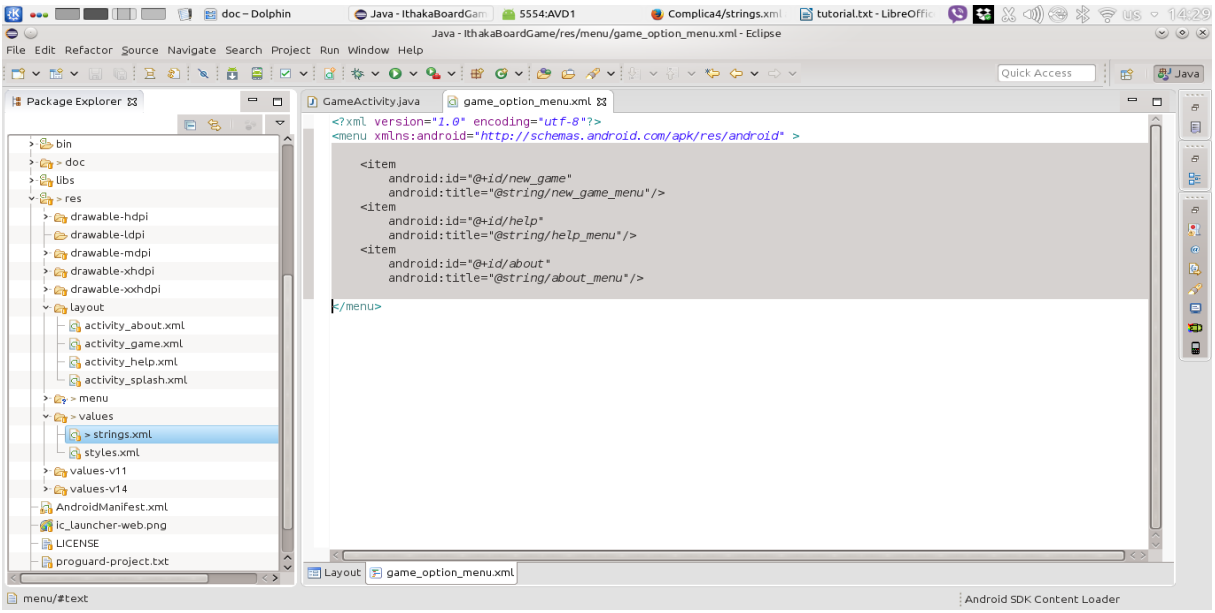
Фиг. 43 Създаване на XML ресурс за менюта

За да бъде възможно извикването на прозорците с помощна информация е необходимо да се създаде меню XML ресурс и това меню да бъде заредено в основния екран на играта.



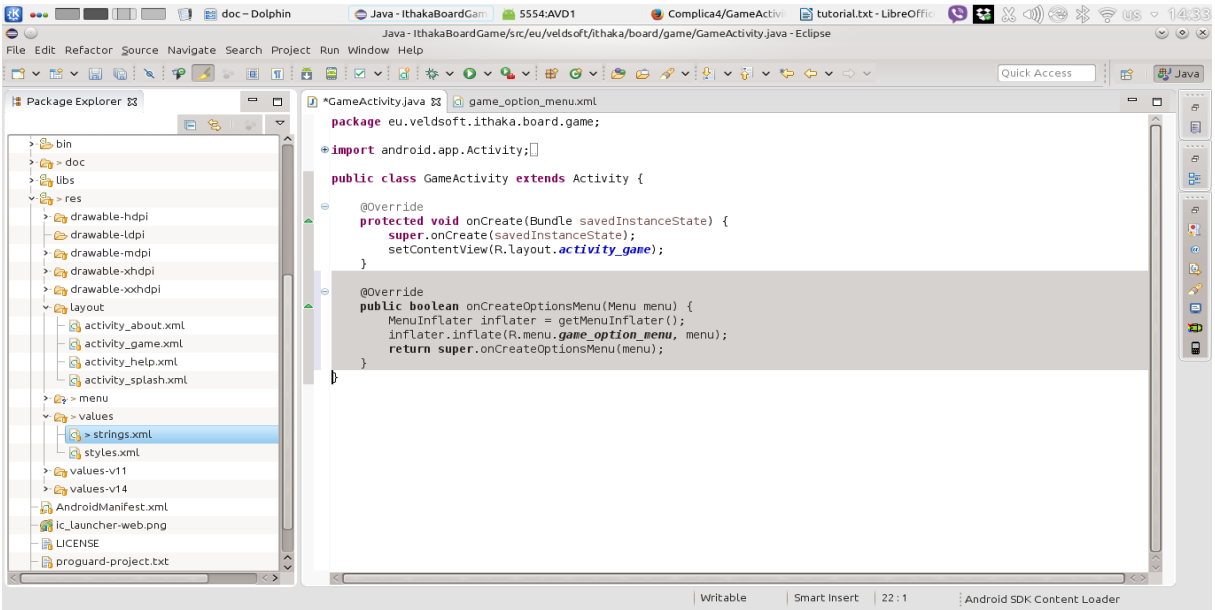
Фиг. 44 Избор на име за меню ресурса

Тъй като играта е изключително елементарна, то е достатъчно да има три опции в менюто – нова игра, помощ и за продукта. При Android платформата не е прието потребителят да затваря прозорците, които е отворил и поради тази причина в менюто не е добавена опция за изход от играта.



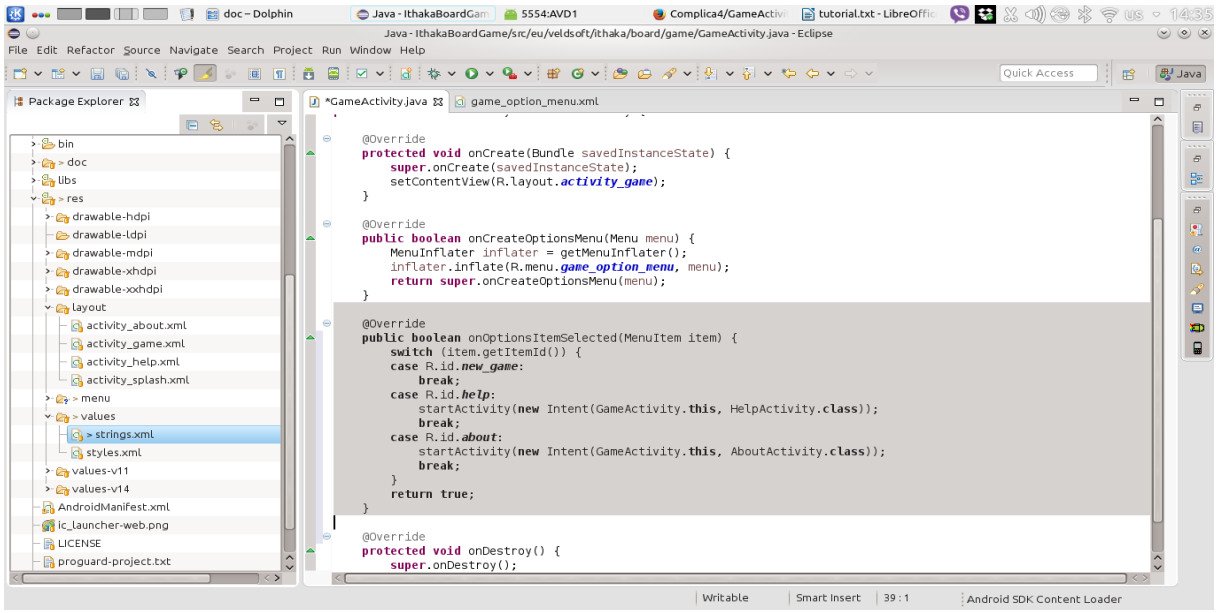
Фиг. 45 Описание на опциите в менюто

За да бъде достъпно менюто в основния екран на играта описателният файл трябва да бъде зареден при създаването на прозореца и към менюто трябва да се получат програмни референции.



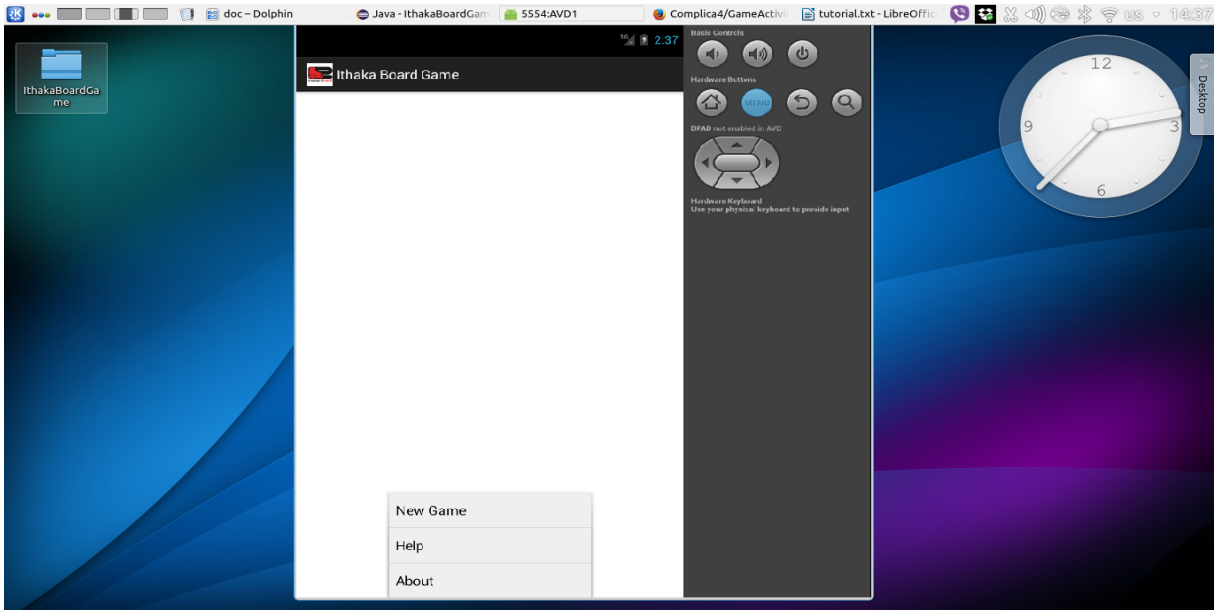
Фиг. 46 Програмно конструиране на менюто на база описателния файл

Освен конструиране на менюто е необходимо да се пренапише събитието, което отговаря за избиране на опции в менюто. Коя от всичките опции е избрана се определя от идентификаторите (id-тата), вписани за всяка опция в XML файла.



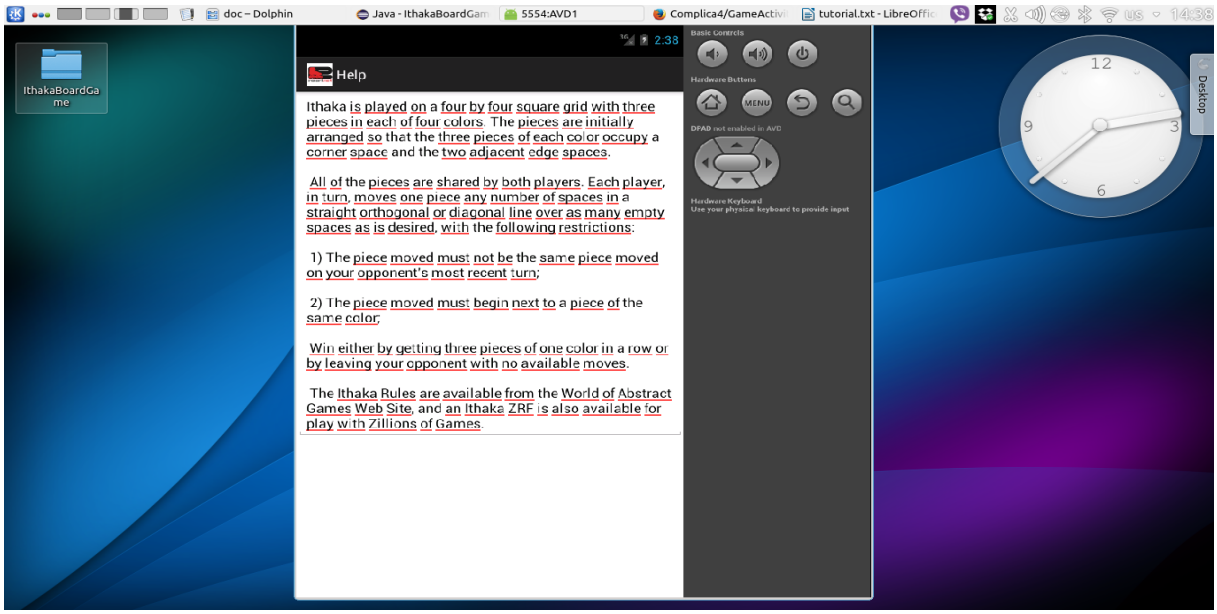
Фиг. 47 Определяне на опция от менюто, която да бъде изпълнена

В резултат на направените модификации основният прозорец на играта разполага с меню, което се активира от хардуерен бутон на устройството (или специално предназначен софтуерен бутон, когато устройството е без бутони).



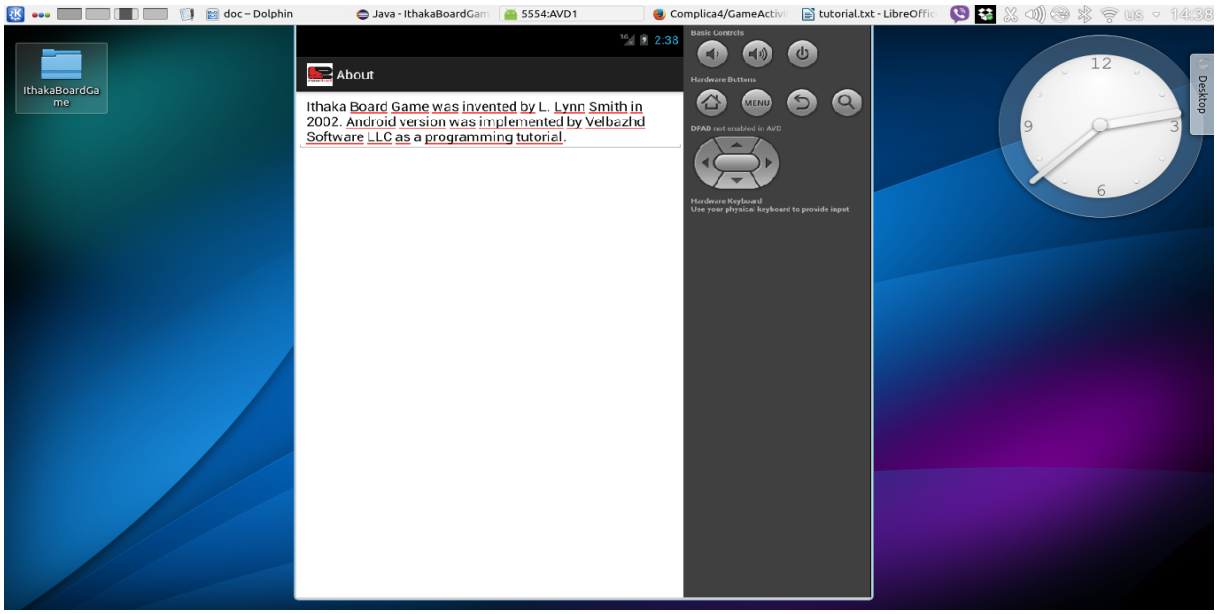
Фиг. 48 Изглед на менюто

Помощната информация се появява под формата на текст в текстово поле.



Фиг. 49 Екран с помощна информация

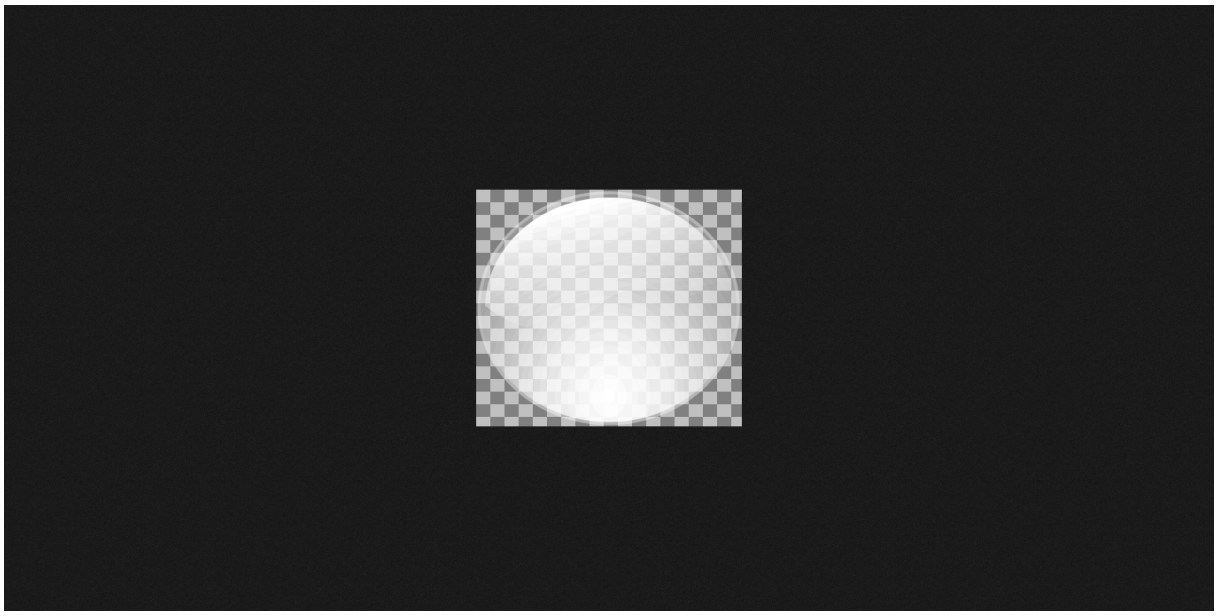
По аналогичен начин изглежда и информацията за продукта.



Фиг. 50 Екран с информация за продукта

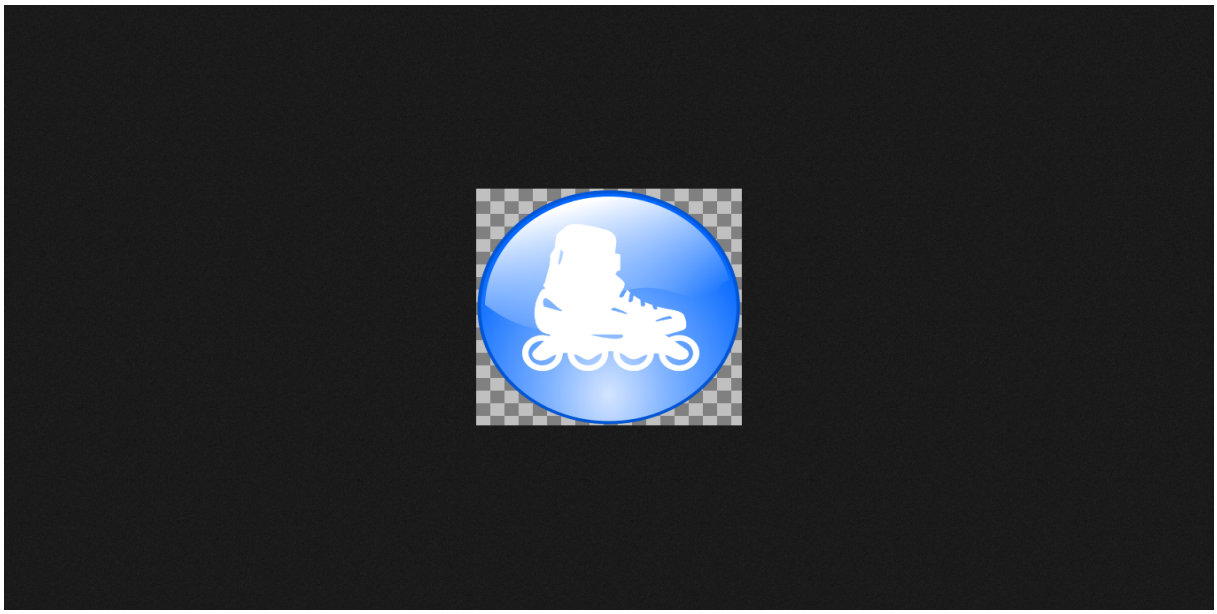
6. Потребителски интерфейс

Разработваното приложение е достатъчно просто, за да бъде поместено в един програмен екран. Правилото при разработка на Android приложения е информацията да се разбива в повече на брой, но с по-малко визуални компоненти, екрани. Изключително популярен подход е софтуерните приложения да се разделят в три слоя – визуализация, обектен модел и съхранение. Визуализацията в Android се описва под формата на XML файлове за графичния интерфейс. Обектният модел се създава под формата на Java обекти с програмен код в методите на класовете. Слоят за съхранението се реализира под формата на SQLite релационна база данни. Според това от кой слой ще започне разработката, има два основни подхода – top-down и bottom-up. При top-down, първо се създава графичният интерфейс с всичките работни екрани, които той ще съдържа, след това се описват обектите в междинния слой и най-накрая се изработва релационната база данни. Между обектния модел и релационния модел се извърша операция, наречена обектно релационен мапинг. При bottom-up подхода, първо се анализират данните и се създава релационен модел, след това, от релациите се оформя обектен модел и накрая се изработва графичен интерфейс, който да отразява моментното състояние на обектния модел. Изборът между двата подхода зависи от обема на проекта и от наличната документация за решавания проблем. Тъй като играта Ithaka е изключително елементарна и действието й ще се развива само в един екран, то може да се приложи подход top-down. В първоначалната версия няма да бъде представена релационна база данни, но в последствие е възможно да се добави и този слой, като в него се натрупва статистика за изиграните игри и най-добрите постигнати резултати.

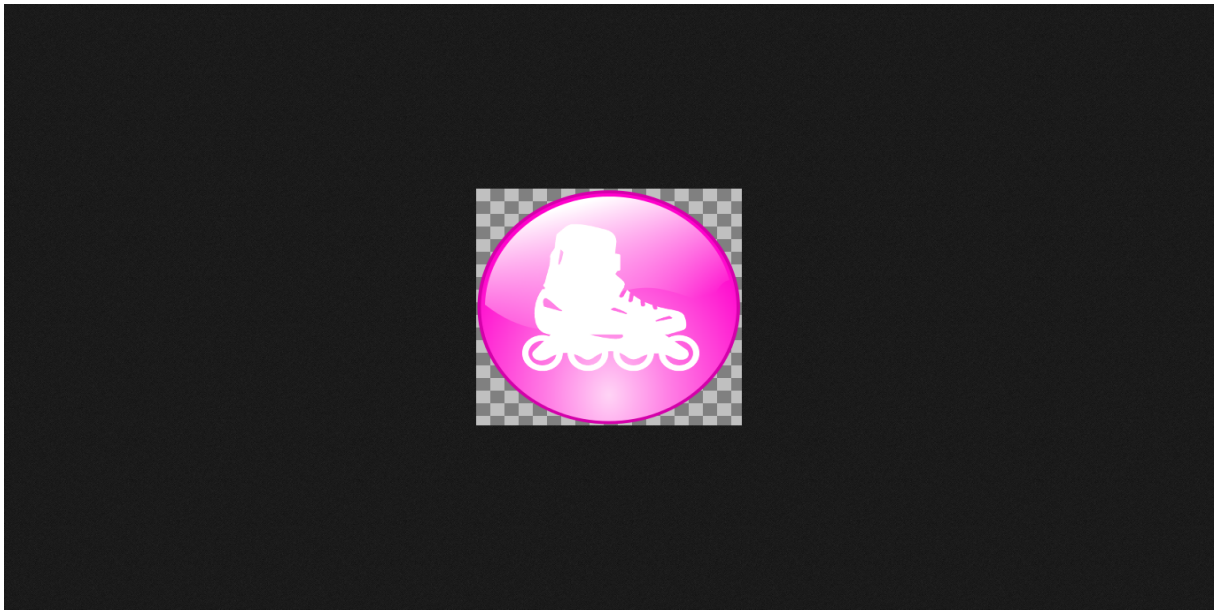


Фиг. 51 Изображение използвано за празен пул

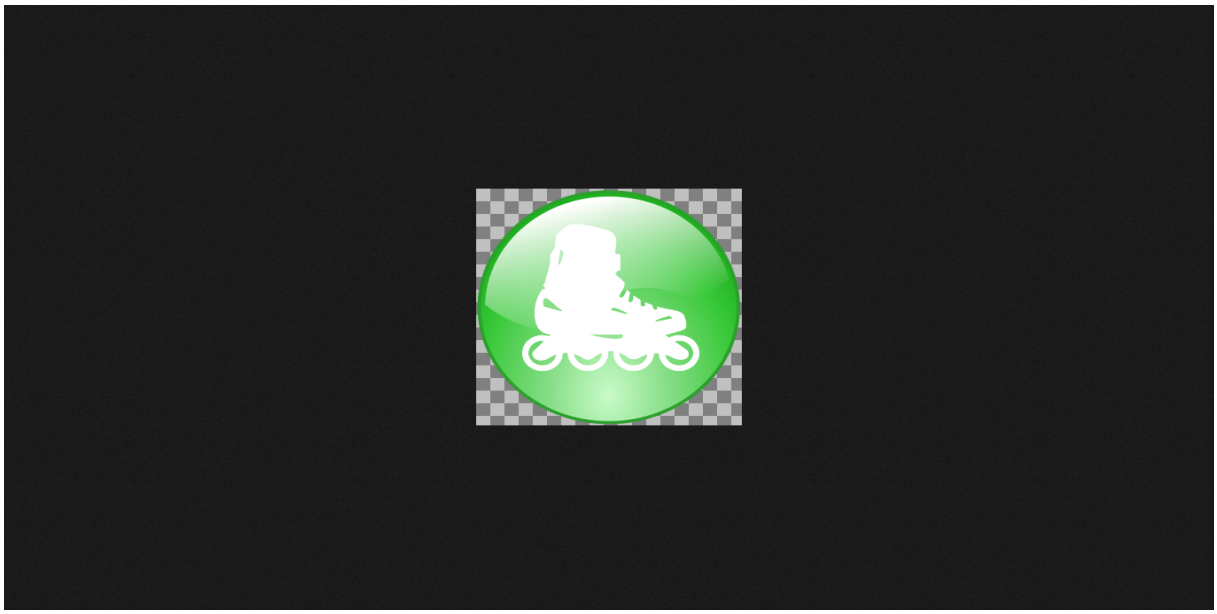
Интерфейсът в основния екран на играта може да се реализира под формата на решетка, върху който да се поставят изображенията на пуловете. В случая много по-удачен вариант е да се маркират 16 полета, във формата на решетка с размери 4x4, така че да създават илюзията за решетка.



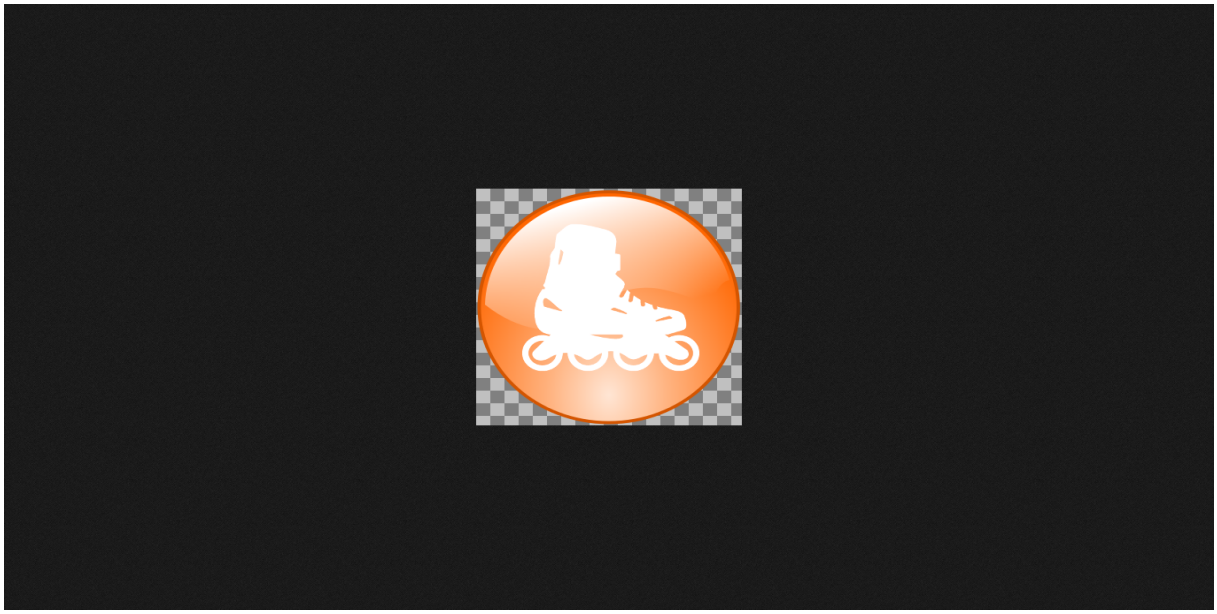
Фиг. 52 Изображение използвано за син пул



Фиг. 53 Изображение използвано за розов пул

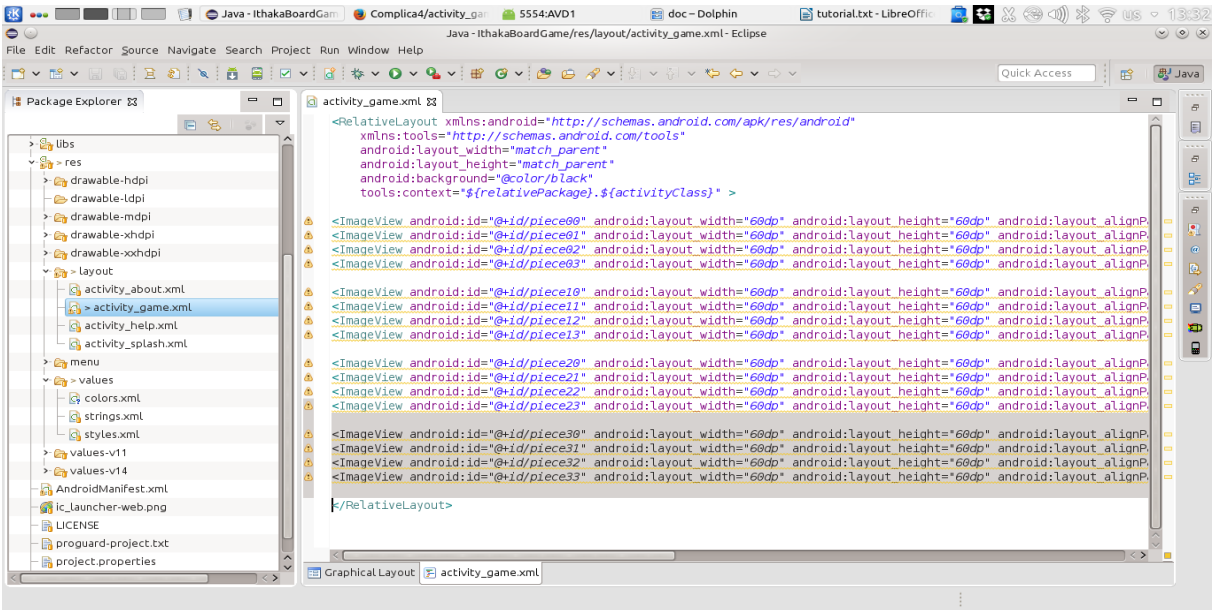


Фиг. 54 Изображение използвано за зелен пул



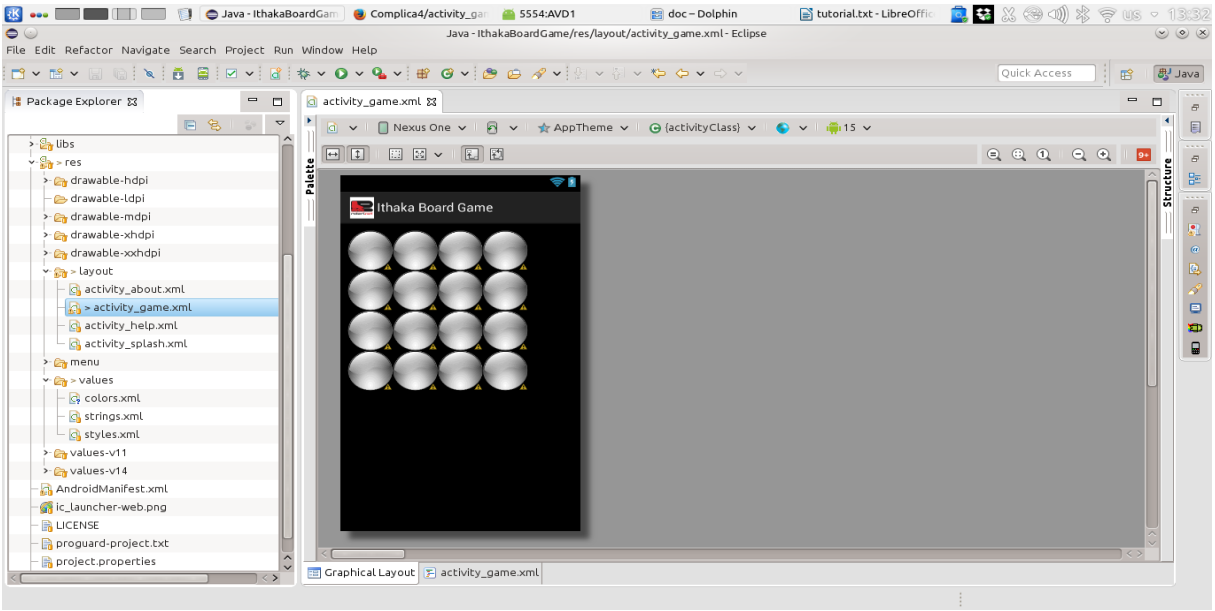
Фиг. 55 Изображение използвано за оранжев пул

Тези пет графични артефакти са напълно достатъчни да се реализира целият графичен потребителски интерфейс в основния екран на играта. На практика е използвано едно и също базово изображение, за което са изработени четири варианта в различен цвят. Важно е да се забележи използването на слой за прозрачност. Слоят за прозрачност се нарича алфа канал и служи за определяне каква част от изображението да бъде визуализирана и каква част да бъде заместена от фона, върху който се визуализира изображението. По този начин с правоъгълни изображения могат да се пресъздадат обекти с неправилна форма като се наслагват едни върху други.



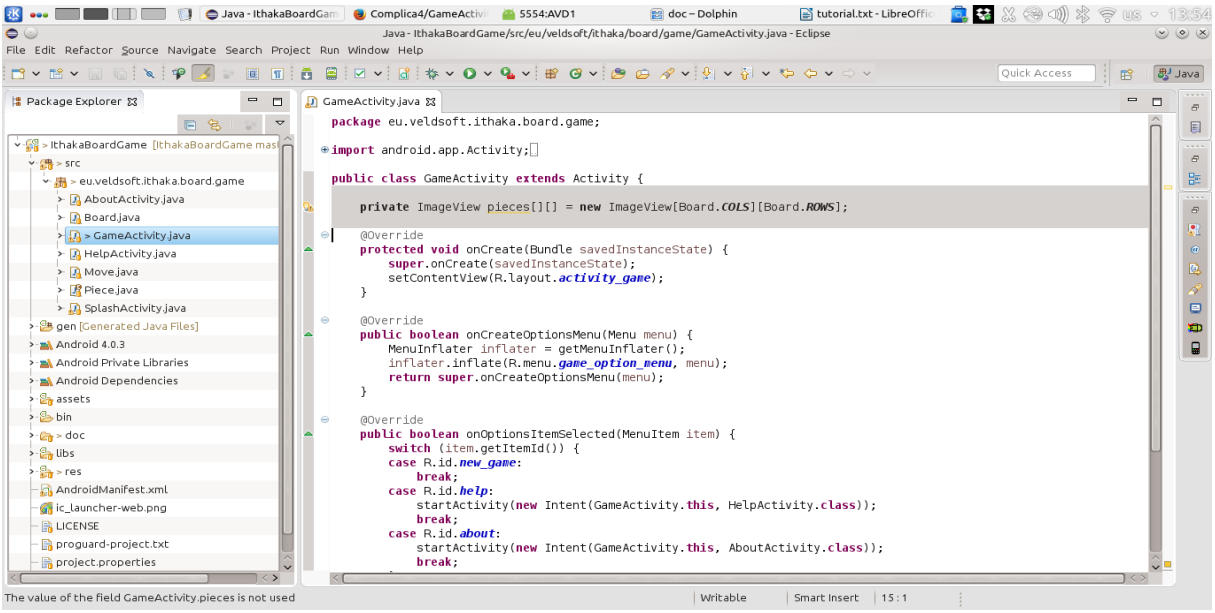
Фиг. 56 Основен екран описан като XML ресурс

За нуждите на основния екран в играта е достатъчно да се подредят 16 компонента от тип ImageView. Визуалните компоненти биват оразмерени, така че максимално ефективно да запълват визуалното пространство. Също така всеки визуален компонент получава подходящи координати спрямо горния ляв ъгъл. При тази подредба се получава своеобразна матрица. Най-съществената част в описанието на визуалните компоненти са техните идентификатори (свойството id). Идентификаторите служат за получаване на програмен достъп до визуалните компоненти, описани в ресурсните XML файлове. При настоящата реализация идентификаторите са така подбрани, че да символизират индексите в двумерен масив на Java.



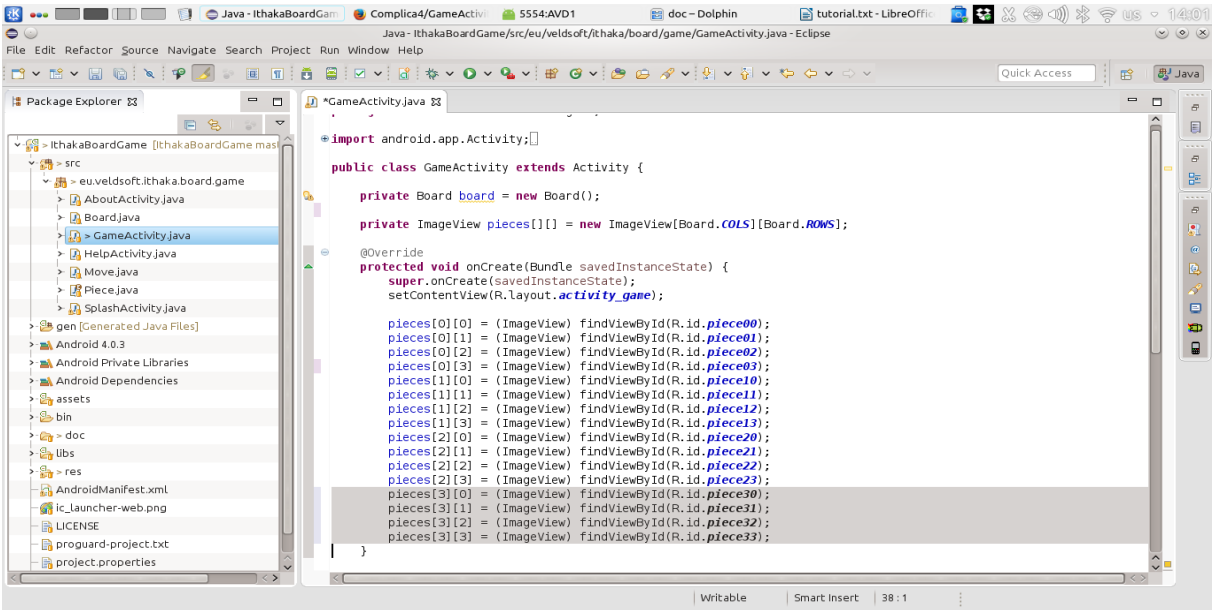
Фиг. 57 Първоначален изглед на празно играло табло

Така подредени, визуалните компоненти биха отразявали вътрешното състояние на обектния модел. Единствената задача на визуалния потребителски интерфейс е да представя пред потребителя моментното състояние на обектния модел.



Фиг. 58 Референции за достъп до компонентите описани в XML файла

След като графичният интерфейс е описан под формата на XML код, то в програмния код трябва да се осигури достъп до визуалните компоненти с помощта на Java референции. Най-удачният вариант за съхраняване на референциите е в двумерен масив, който да повтаря организацията на ImageView компонентите, описани под формата на XML.

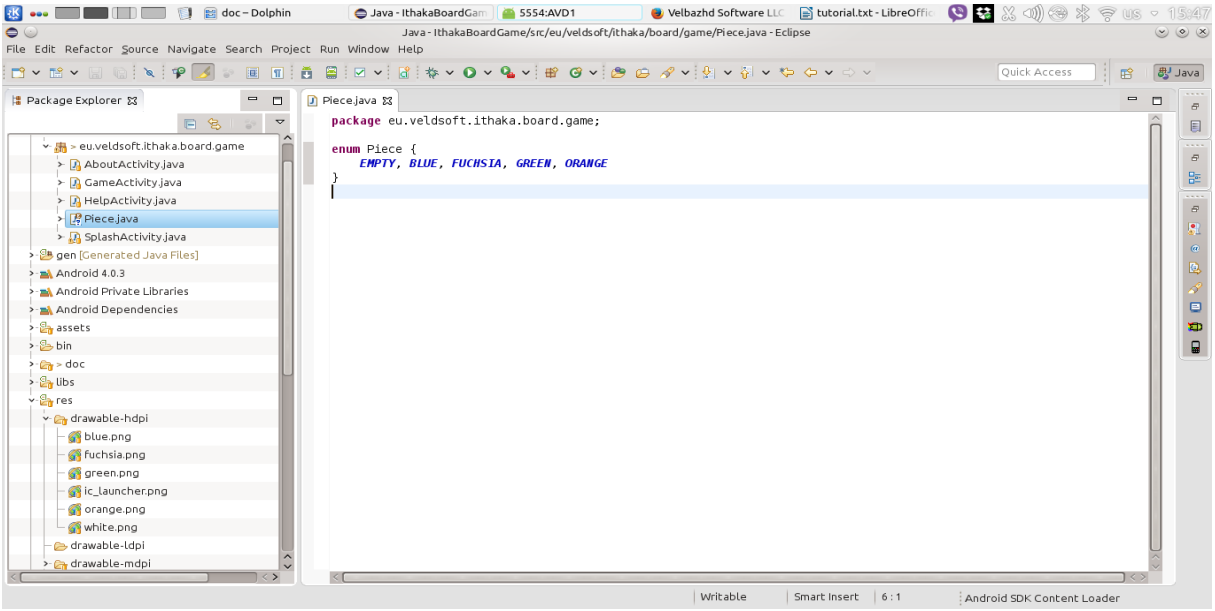


Фиг. 59 Съхраняване на референции с помощта на фактори функция

Най-съществено е да се забележи начинът, по който се съпоставят идентификаторите от XML описанието към съответните клетки на двумерния масив. Използва се съответствие, при което pieces[X][Y] се съпоставя на XML компонент R.id.pieceXY.

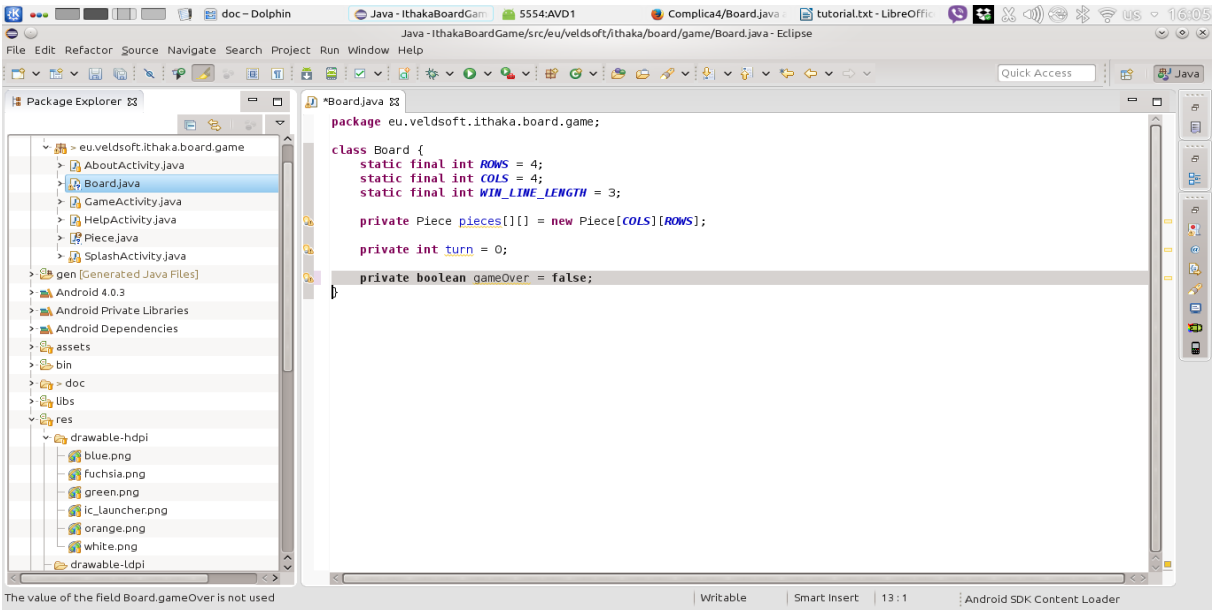
7. Обектноориентиран модел

В средния слой е поместен обектноориентираният модел, който по своето същество представлява група от обекти, взаимно свързани по между си. Графичният интерфейс служи като моментна снимка на вътрешното състояние, в което се намират обектите. В клас ориентирани езици, какъвто е и езикът Java, обектите се описват със своите класове. За успешно ОО моделиране е от съществено значение правилно да се идентифицират съществителните имена в заданието и част от тях да бъдат избрани за класове в системата. В играта Ithaka основно съществува игрално поле, върху което са разположени играли пулове. На практика игралното поле е в has релация спрямо пуловете. По отношение на това кои обекти притежават други обекти, се прилагат два основни подхода за моделиране - 1. композиция, 2. агрегация.



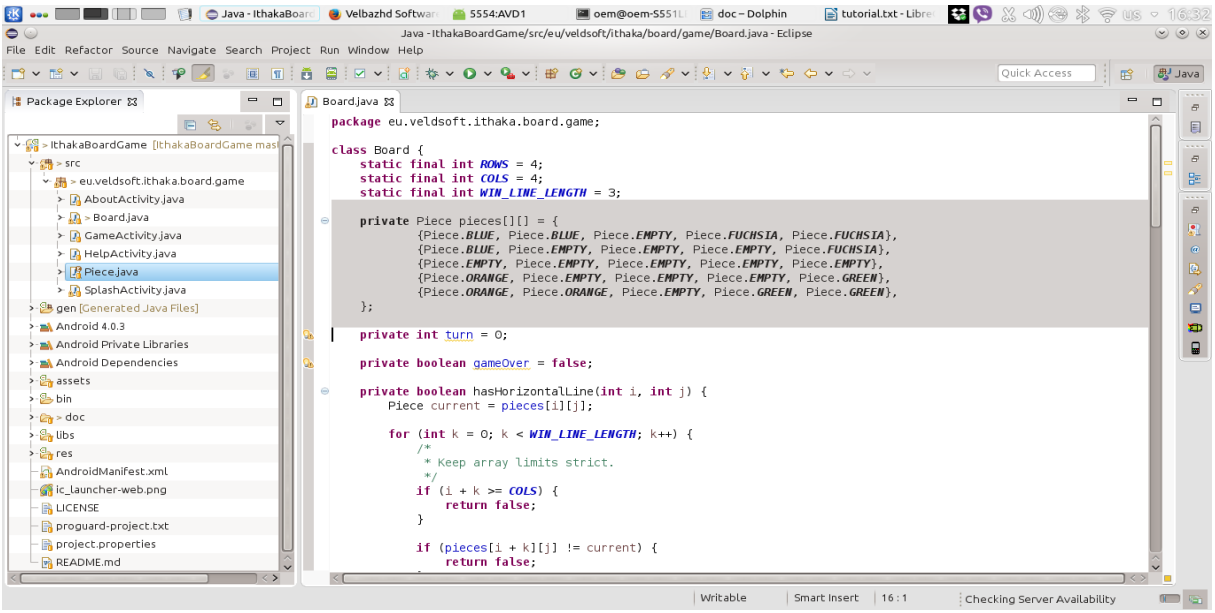
Фиг. 60 Моделиране на пуловете

За моделирането на пуловете има различни възможности, но една от най-удачните е използването на изброените типове в Java. В процеса на компилация изброените типове се трансформират до стандартни Java класове, но по време на самото моделиране изброените типове налагат серия ограничения за начина по който вътрешните им член-променливи и методи могат да се използват. В оригиналния вариант на играта пуловете са само с четири цвята, но в последствие е възможно да се добавят и други цветове. Моделиране с изброени константи ще позволи софтуерът с лекота да бъде разширен в тази посока. Неестествено е в изброените пулове да се включи и понятието за празна клетка, но от програмна гледна точка е изключително рационално, най-малкото защото в графичния програмен интерфейс артефактите съответстват на пет различни елемента, а не на четири. От друга страна, чисто условно може да се смята, че действително има бели пулове, които се разменят с цветните, така че да се формират съответните комбинации върху игралното табло.



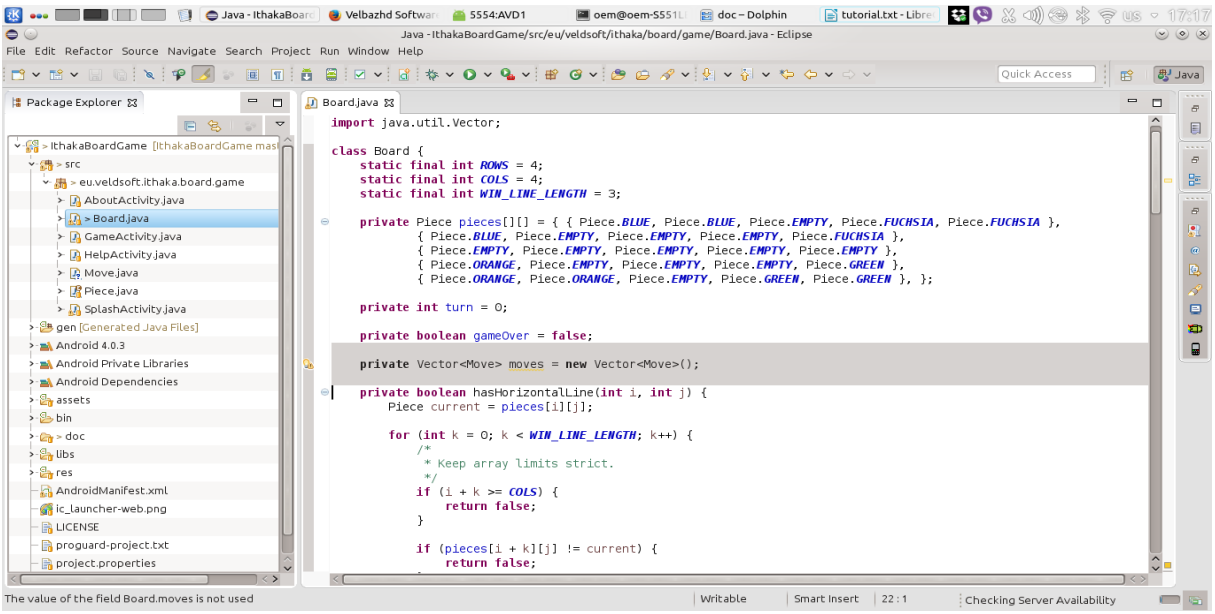
Фиг. 61 Модел на игралното табло

Най-важният обект в разработвания софтуерен продукт е игралното табло. За да бъде представено в Java ОО модел, игралното табло се описва под формата на самостоятелен клас Board. Обект от класа Board ще бъде вграден в програмния интерфейс, така че да се осъществява комуникация между визуалния слой и слоя на обектния модел. Традиционно, класовете в обектноориентираните езици имат група от характеристики (представени под формата на член променливи и/или свойства). Най-важната характеристика на игралното табло е неговият размер. В оригиналните условия на играта игралното табло е с размери 4x4, но не е изключено, в бъдещи версии на софтуерната реализация, размерите да бъдат променени (както и броят на цветовете пулове). Поради тази причина и за да се избегне феноменът, познат под названието „магически числа“, най-удачно е размерите на игралното табло да бъдат представени под формата на статични константи. Друга важна характеристика на игралното табло е дължината на линията, която формира печеливша комбинация (в оригиналния вариант това е 3). Тъй като играта се играе на ходове, важно е да се съхранява информация кой по ред е текущият ход. Това може да се постигне с целочислена променлива (в случая променливата turn), която да служи като брояч. Полезно би било да се знае в кой момент играта е приключила и това се постига с допълнителен булев флаг (в случая променливата gameOver). Повечето компютърни игри се описват с група обекти, които изпадат в различни състояния. Поради тази причина много често се прилагат способите от теория на крайните автомати. От първоначалния ОО анализ най-съществена е характеристиката на класа, определяща пуловете. Най-удачно е пуловете да бъдат моделирани с двумерен масив от типа Piece. Състоянието на този двумерен масив ще бъде отразявано в графичния потребителски интерфейс, където вече е предвиден друг двумерен масив от тип ImageView. Практически, всички изчисления ще се извършват в обектния модел, но резултатът от тях ще става видим в потребителския интерфейс. Също така потребителят ще взаимодейства с елементите на графичния интерфейс, а информацията ще бъде предавана към по-долния слой на обектния модел.



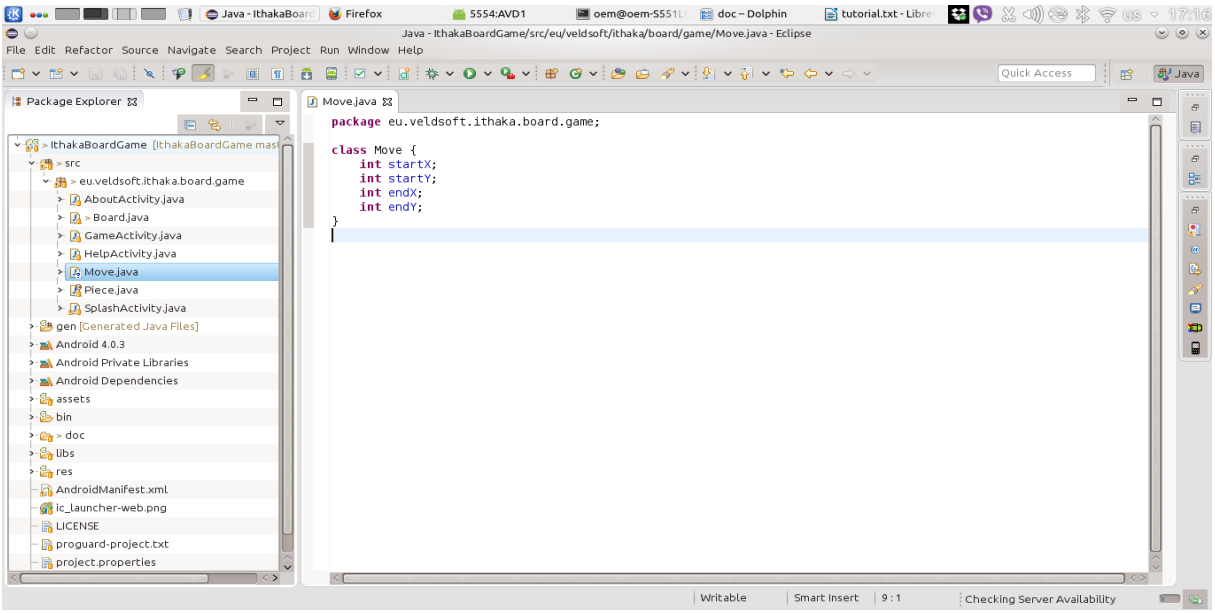
Фиг. 62 Изходно състояние на игралното табло

В изходно състояние игралното табло е подредено според указанията в правилата на играта. Точно поради тези причина, в обектния модел, двумерният масив, съдържащ пуловете, бива инициализиран подходящо в момента, в който бива конструиран обект от класа Board.



Фиг. 63 Запазване на история с изиграните ходове

Едно от правилата на играта изисква да се следят трите последни хода. При наличие на три идентични хода, играчът губи играта. Поради тази причина възниква необходимостта да се съхранява историята на изиграните ходове. Тъй като едно разиграване може да продължи много ходове, то най-удачно е използването на контейнер от тип Vector, в който да се записват обекти от клас Move.

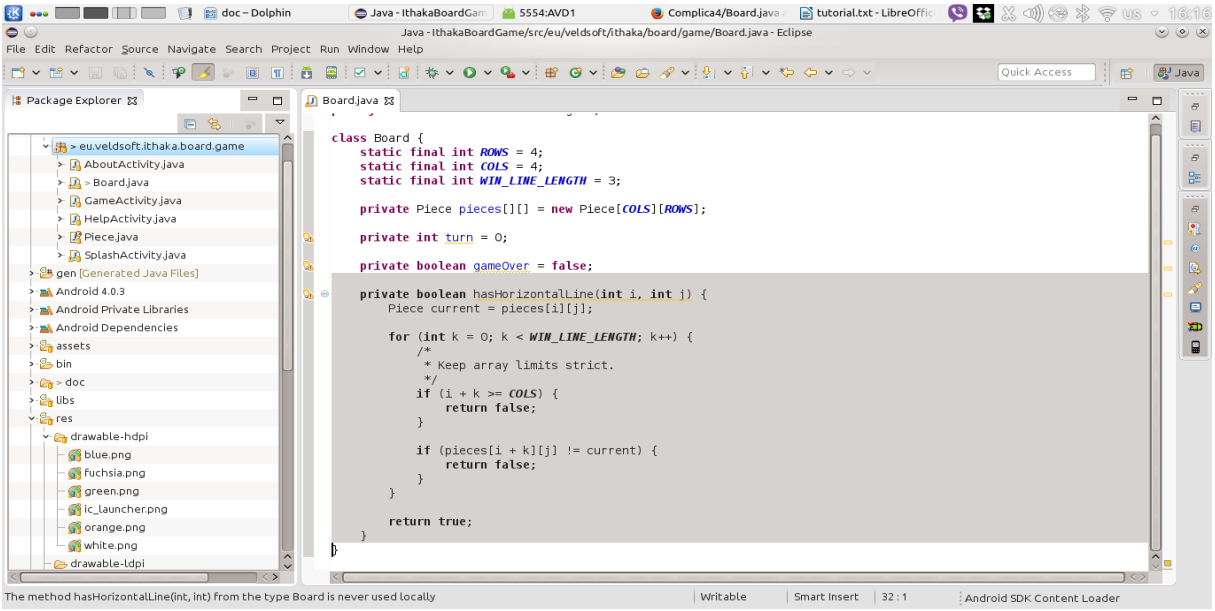


Фиг. 64 Моделиране на отделен ход в играта

За да се моделират ходовете в играта удачен вариант е използването на отделен клас, наречен Move. Всеки ход в играта се описва с начални координати (пул, който предстои да бъде местен) и крайни координати (клетка, в която пулът трябва да попадне). Съществено за ходовете в играта е, че те може да са два основни типа – валидни и невалидни. При настоящия модел валидността на ходовете е изнесена извън класа Move. Добрите практики за ОО моделиране изискват всички вътрешни променливи на класовете да бъдат скрити (частно ниво на достъп), но в случая е очевидно, че полетата са с пакетното ниво на достъп, което езикът Java позволява. Нарушаването на концепциите за капсулиране са допустими, когато класът е с пакетно ниво на достъп и инстанции от него няма да се ползват извън рамките на пакета. В настоящата разработка класовете на обектния модел и на графичния интерфейс се намират в един пакет (тъй като приложението е твърде елементарно), но добрата практика за писане на модулен софтуер изисква класовете с различно предназначение да бъдат оформени в тематични пакети. Също така, може да се забележи, че всички класове от групата на обектния модел са с пакетно ниво на достъп, докато всички класове (предимно наследници на Activity) са с публично ниво на достъп. Причината за това е, че операционната система трябва да има достъп до интерфейсните класове и да има възможност за създаване на обекти от тях. Това съвсем не е така с класовете от обектния модел, които трябва да служат като вътрешна анатомия на софтуерния продукт. Макар и малко на брой трите класа (Pice, Board и Move) напълно покриват първоначалните нужди за моделиране на играта Ithaka. След определяне на структурите от данни, следва стъпката за изготвяне на алгоритмите, които ще обработват данните.

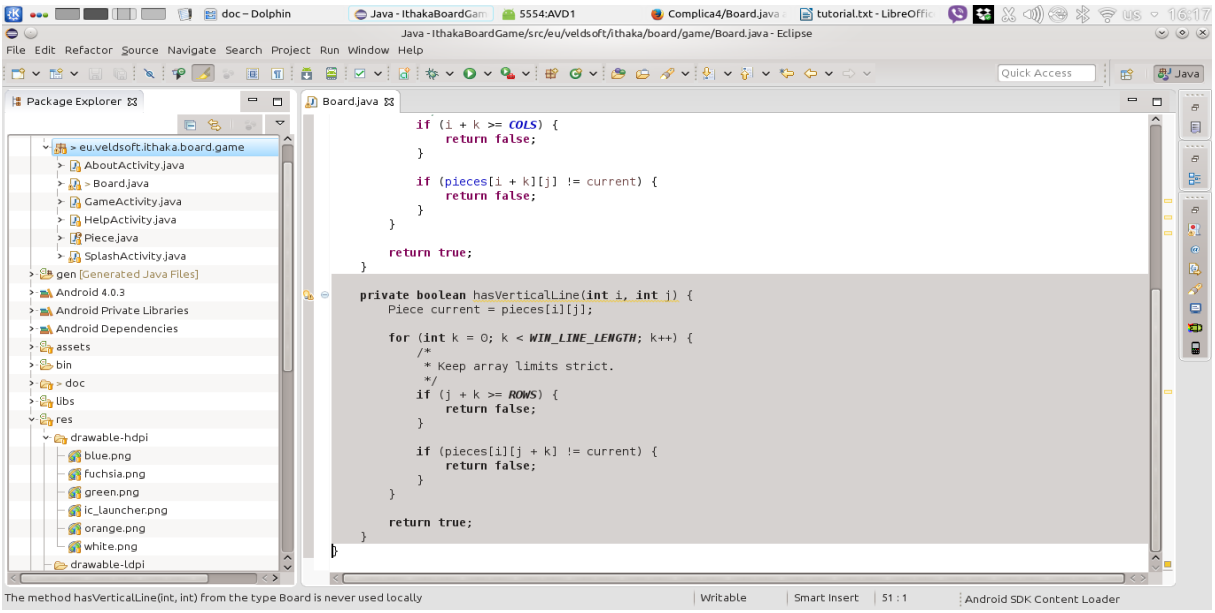
8. Извършване на пресмятания

Java е изцяло обектноориентиран език и позволява писане на програмен код само в рамките на методи (тоест не съществуват самостоятелни функции, както е примерно в езика C). Най-съществениият елемент на тази фаза от софтуерния дизайн е коя функция в кой клас ще бъде реализирана. Най-разумният подход при реализирането на програмен текст е първо да се изработят тези функции, които са най-ясни и най-лесни за реализация. Този подход невинаги е приложим, особено ако се използва работен процес, базиран на FDD (Feature Driven Development), при който възложителят определя коя функционалност да бъде реализирана с приоритет. Тъй като в настоящата разработка чисто формално не съществува възложител, то без проблем може да се започне с най-лесните функции.



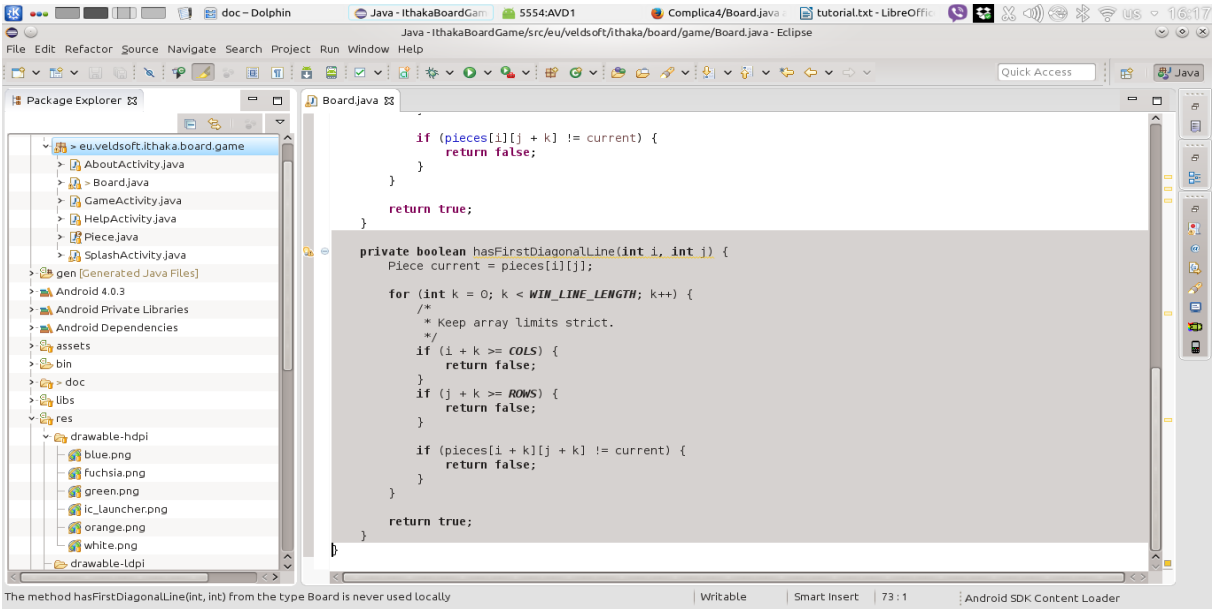
Фиг. 65 Намиране на хоризонтална печеливша комбинация

Едно от най-значимите събития в играта е намирането на ситуациите, в които единият играч побеждава другия играч.



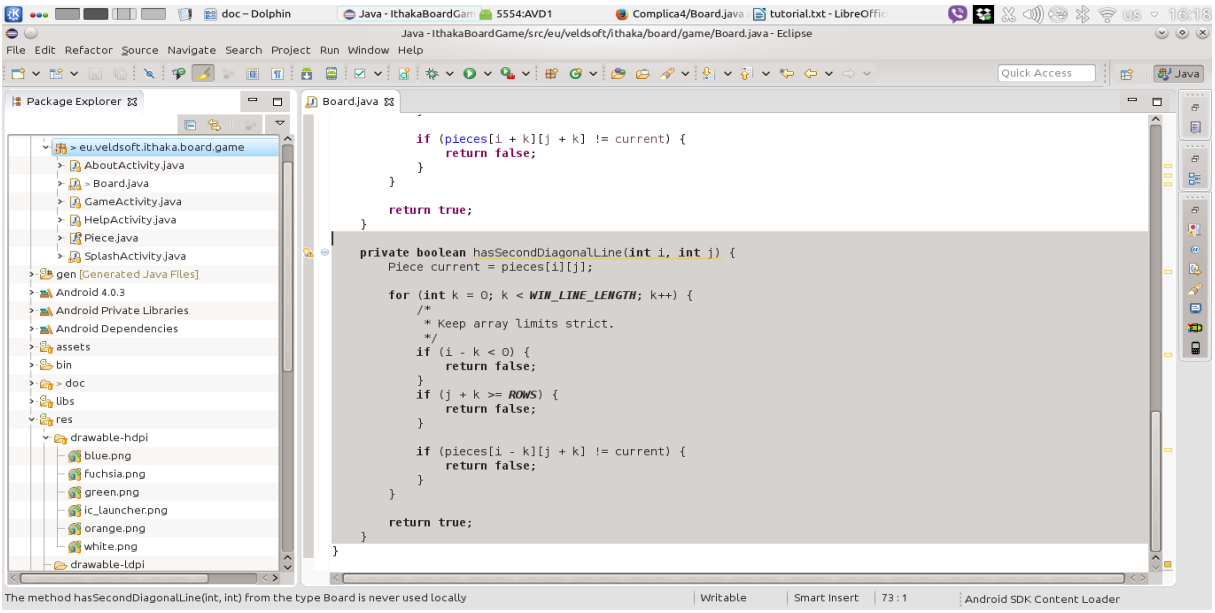
Фиг. 66 Намиране на вертикална печеливша комбинация

Най-често срещаната ситуация за формиране на печеливша комбинация е подреждането на три пула от един цвят в права линия.



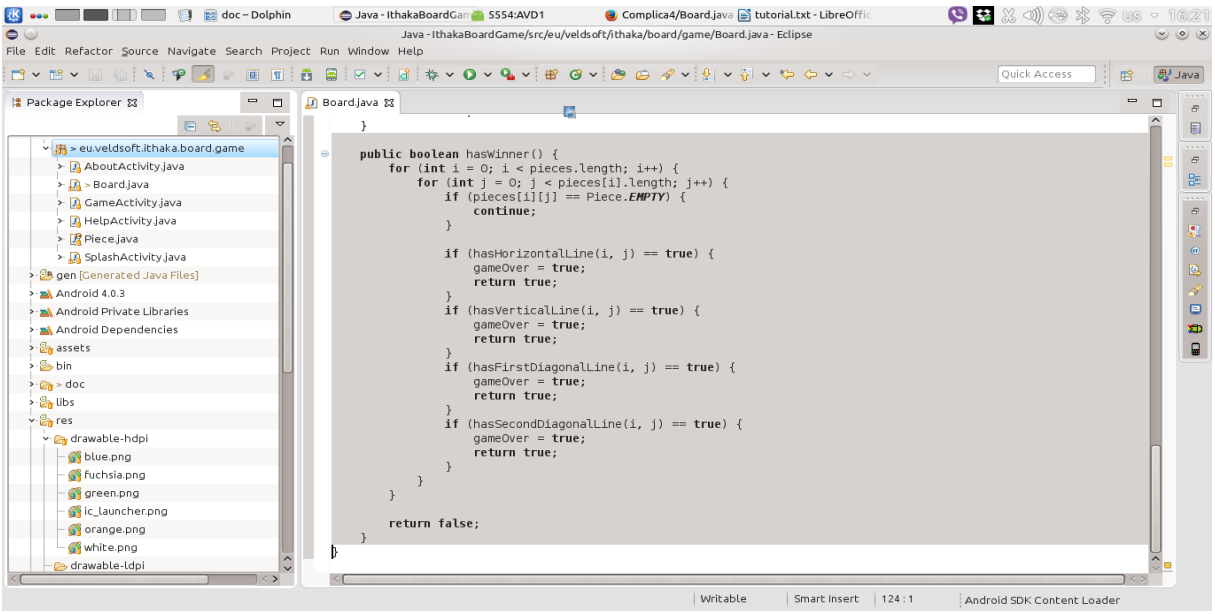
Фиг. 67 Намиране на печеливша комбинация по главния диагонал

Печелившите комбинации могат да са ортогонално или диагонално.



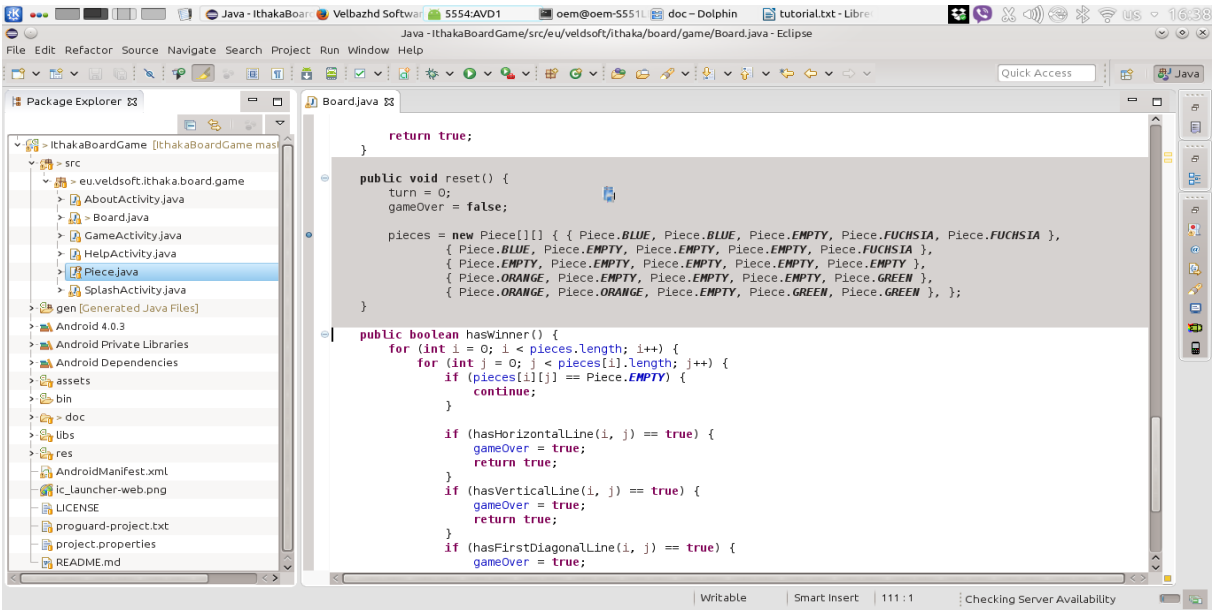
Фиг. 68 Намиране на печеливша комбинация по вторичния диагонал

При избора как да бъдат проектирани отделните функции, най-важен е принципът, използван още от средновековието – разделяй и владей. Поради тази причина, намирането на печеливша комбинация е разбито в четири помощни функции като всяка една търси в определена посока. Това, което трябва да се съобрази е, че игралното табло има размери 4x4, докато печелившата комбинация е с дължина 3. Поради тази причина, печелившата комбинация може да започва от различни клетки и да завършва на различни клетки, без значение, че направлението е едно и също.



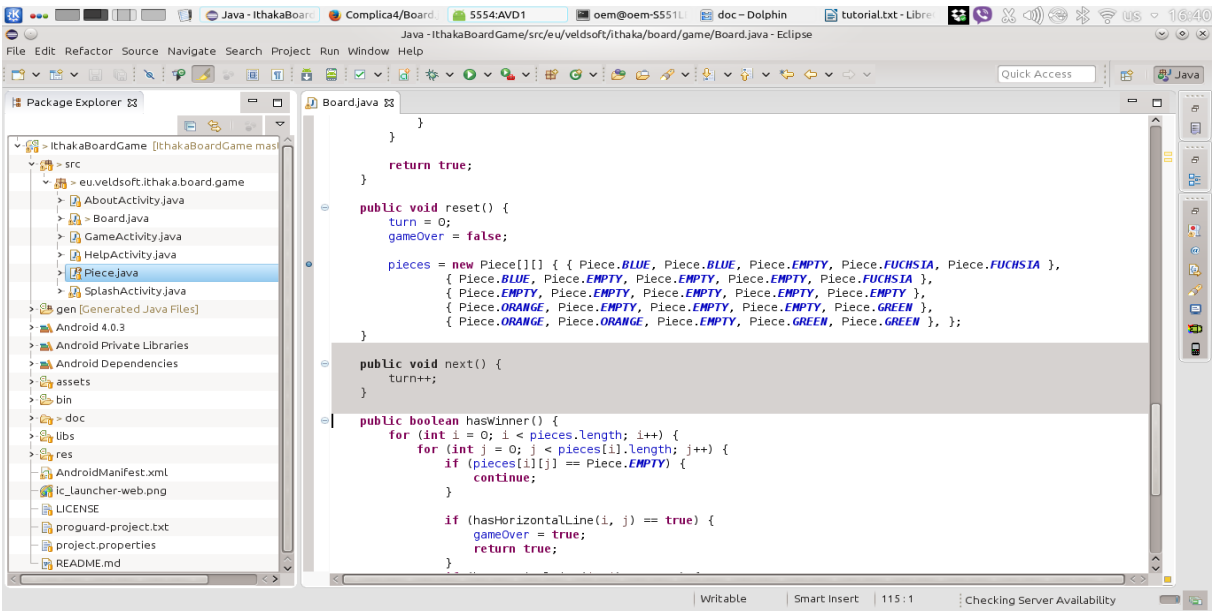
Фиг. 69 Обединение на четирите помощни функции

След разбиването на четири помощни функции, следва резултатът от всичките тях да бъде обработен в една по-главна функция (тоест овладяването на проблема).



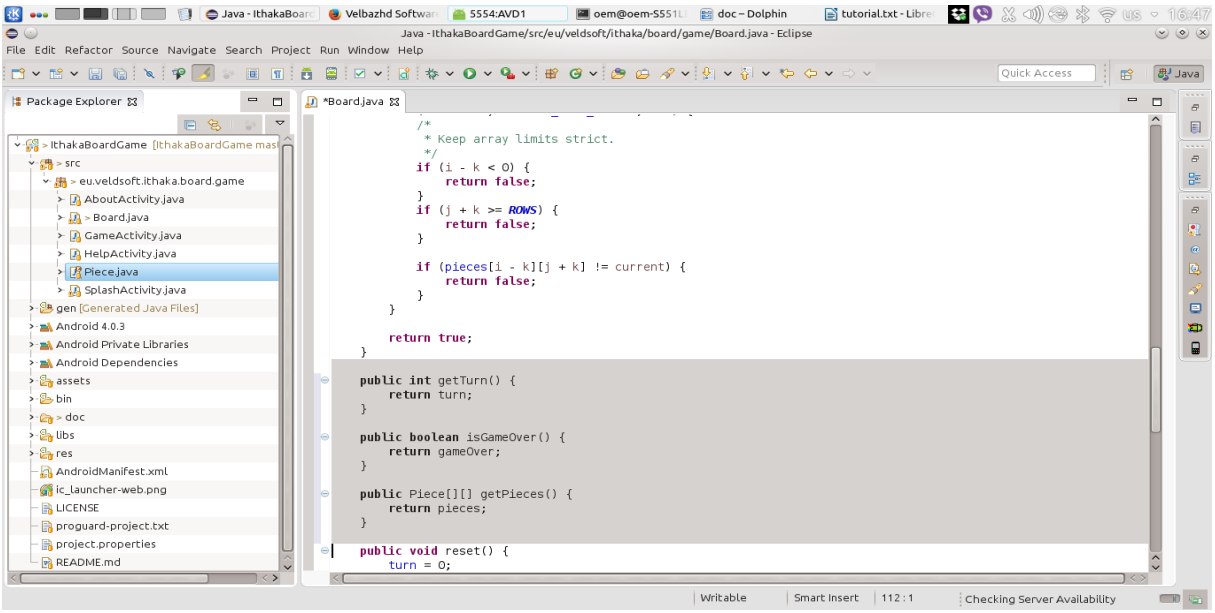
Фиг. 70 Установяване на изходно състояние за обекта на игралното табло

При проектирането на класове има два основни подхода – обектите им да бъдат непроменяеми (immutable objects) или променяеми (mutable objects). Смята се, че непроменяемите обекти са значително по-безопасни за работа, но те са свързани с множество заделяния на инстанции и последващото им освобождаване от модула за управление на паметта (garbage collector). За обекта на игралното табло се използва принципът за променяем обект, което налага съществуването на функция, която да установява обекта в изходно състояние.



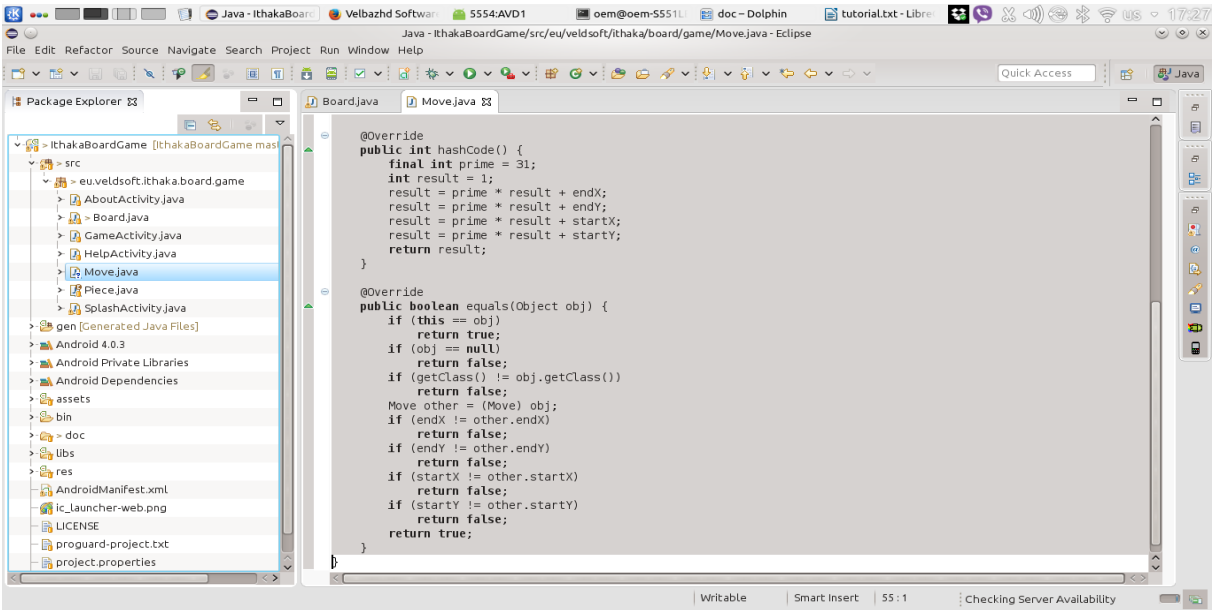
Фиг. 71 Броене на ходовете

Броенето на изиграните ходове се осъществява с променлива брояч.



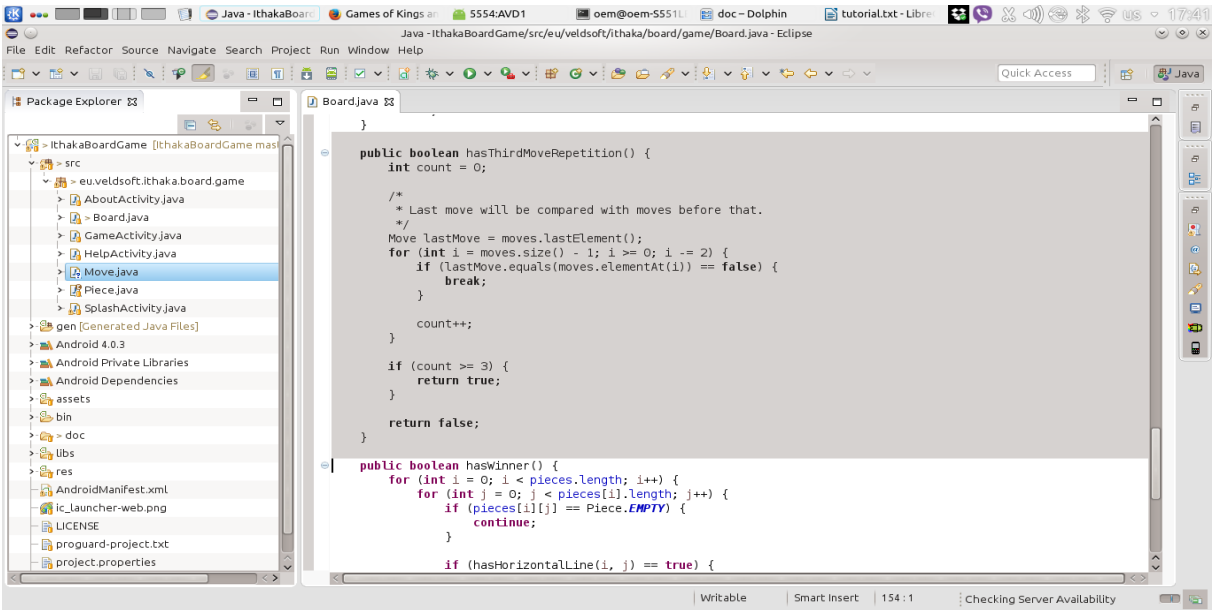
Фиг. 72 Достъп до скритите полета на обекта

Достъпът до скритите полета на обекта се осъществява с функции, наречени аксесори. Тяхната единствена задача е да предоставят контролиран достъп до скритите променливи на обекта.



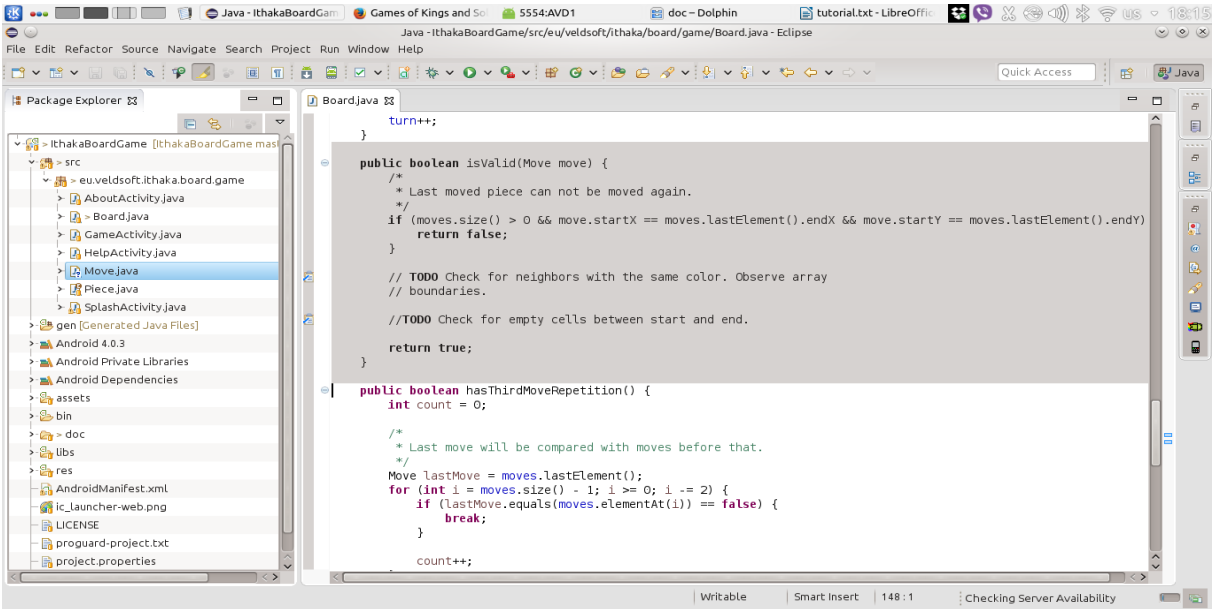
Фиг. 73 Служебни функции за сравняване на обекти от тип Move

Тъй като едното условие за край на играта изисква да се следи тройното повтаряне на един ход, то най-разумната реализация на тази проверка е във Vector контейнер, който да съхранява всички изиграни ходове. Също така е важно ходовете да могат да се сравняват два по два. В езика Java за тази цел са предвидени два служебни метода hashCode и equals.



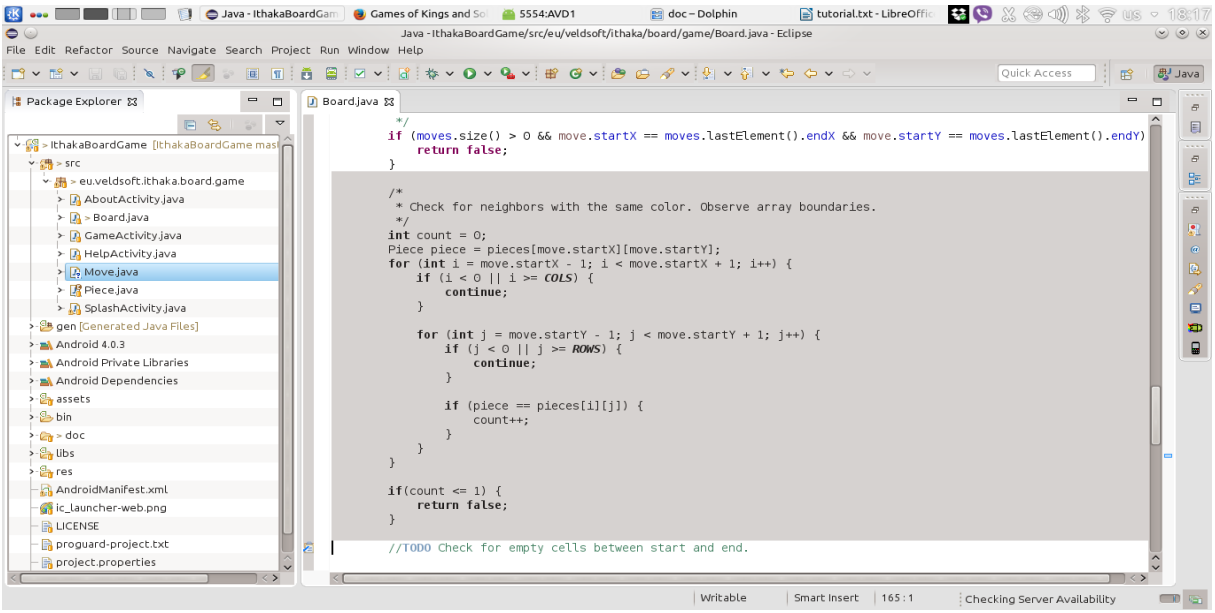
Фиг. 74 Проверка за три повторения на ход

При наличието на обекти, описващи един ход и хронология на играта е достатъчно да се проверят трите последни хода, изиграни от конкретния играч.



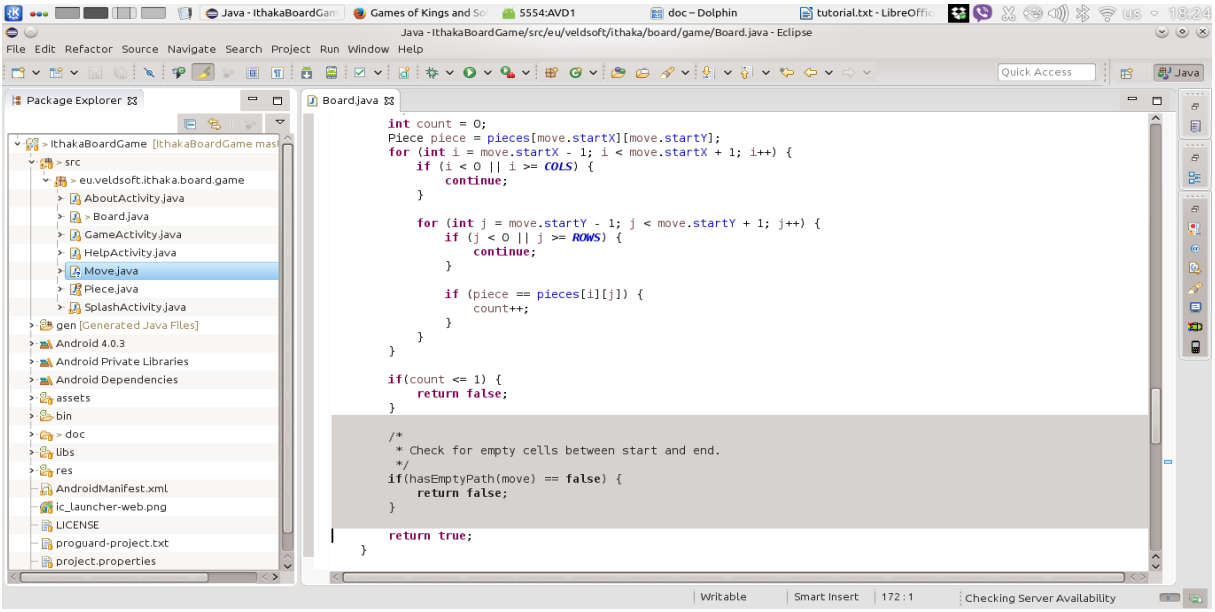
Фиг. 75 Валидност на ходовете

Когато бъде избран определен ход (независимо дали от човека или от компютърния опонент), има две възможности – ходът да е валиден или ходът да не е валиден.



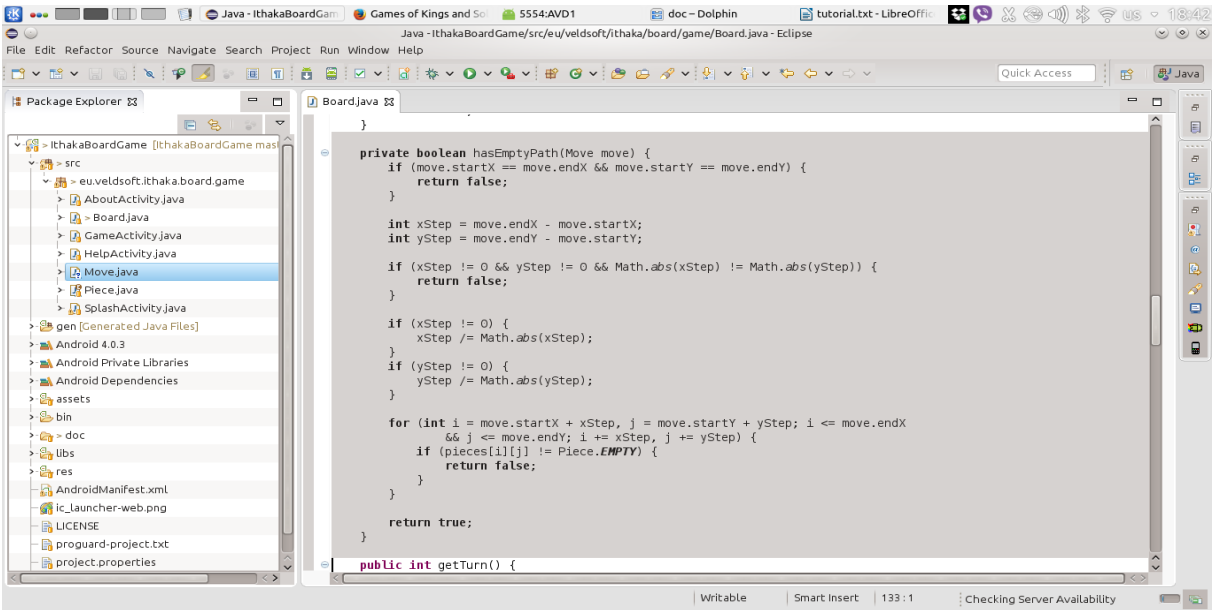
Фиг. 76 Проверка за съседен пул от същия цвят

Според правилата на играта валиден е само ход, при който се мести пул, имащ за съсед поне още един пул от същия цвят.



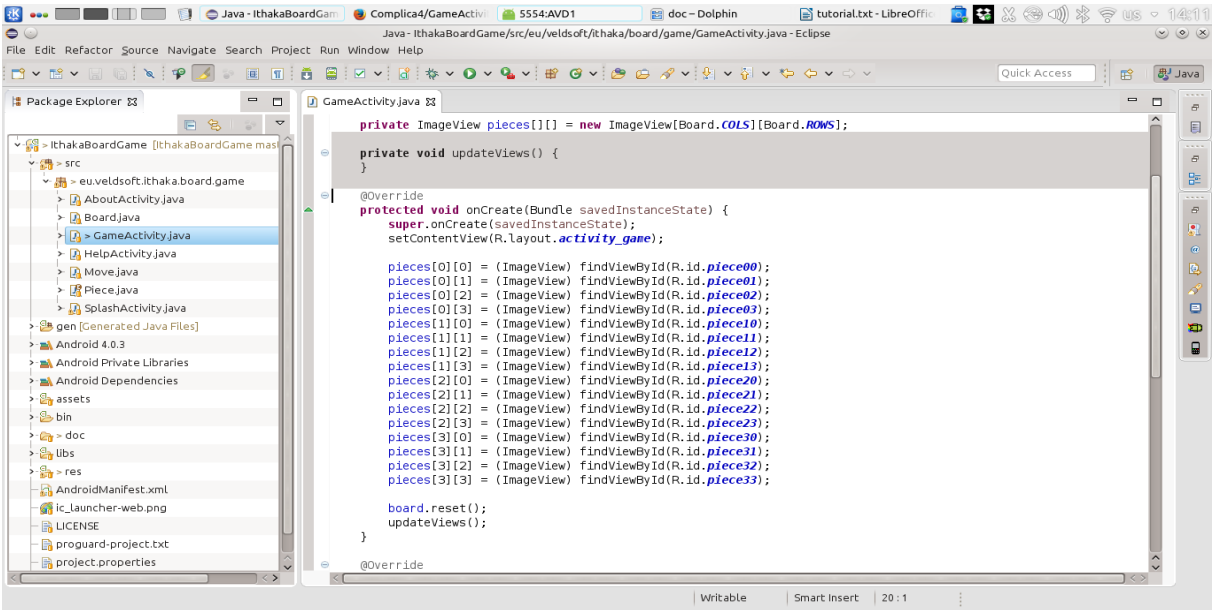
Фиг. 77 Наличие на пряк път от пула до клетката на която трябва да бъде поставен

Второто условие за валидност на хода според правилата, е наличието на път по права линия до клетката, върху която пулът трябва да бъде поставен. Това условие включва и изискването клетката, върху която трябва да бъде поставен, да бъде празна клетка.



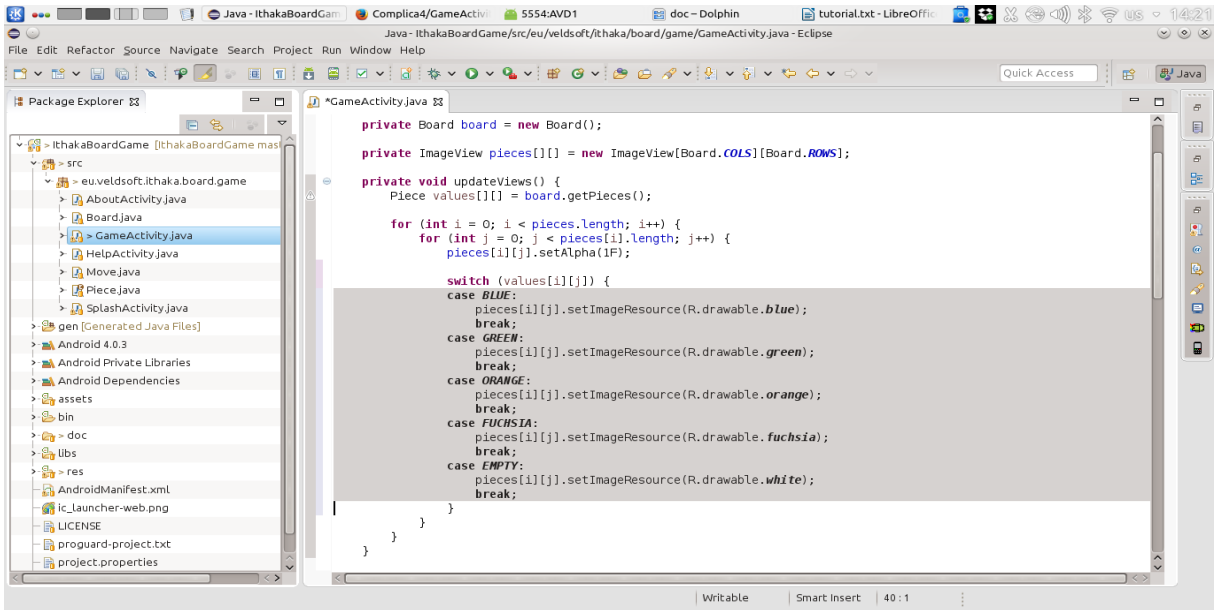
Фиг. 78 Помощна функция за търсене на директен път

Търсене на път от пула до същата клетка, върху която вече се намира, се смята за невалиден път. По условията на играта пътят може да бъде по права линия – хоризонтално, вертикално или диагонално.



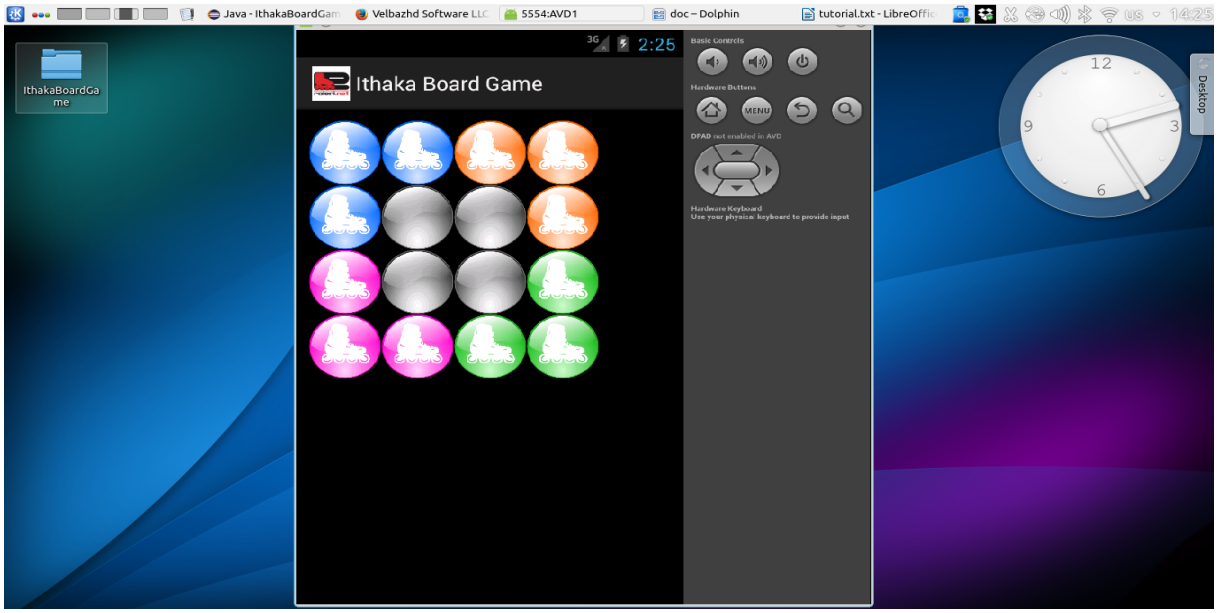
Фиг. 79 Обновяване на визуалните контроли

Използването на многослойна софтуерна архитектура (в случая трислоен модел) налага изграждане на връзки между слоевете. Връзката между слоя на графичния потребителски интерфейс и обектния модел се осъществява с променлива board от клас Board, която е вътрешно поле за класа GameActivity. На практика, компонент от графичния интерфейс съдържа компонент от обектния модел.



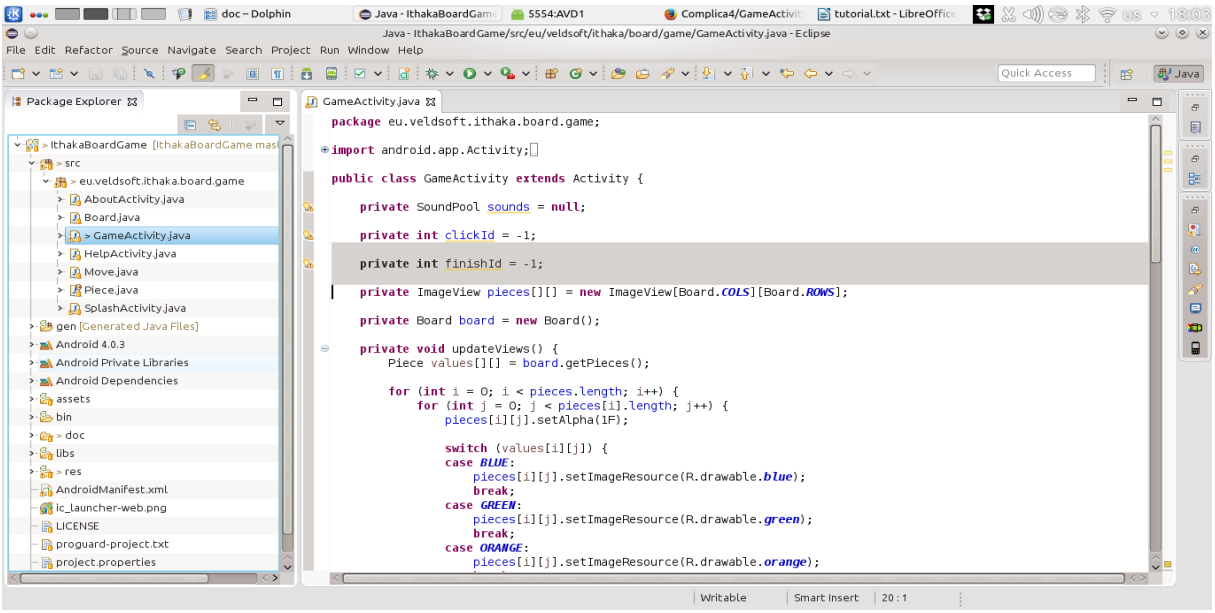
Фиг. 80 Отразяване на вътрешното състояние върху екрана

Основната задача на графичния интерфейс е да изобрази моментна снимка на вътрешното състояние (в случая обекта board). За тази цел се получава матрица с информация за състоянието на игралното табло. След това матрицата се обхожда и във всяка клетка се изрисува изображение, според инструкциите в матрицата. Възможностите за програмно настройване на канала за прозрачност ще се използват за онагледяване на избран от потребителя пул.



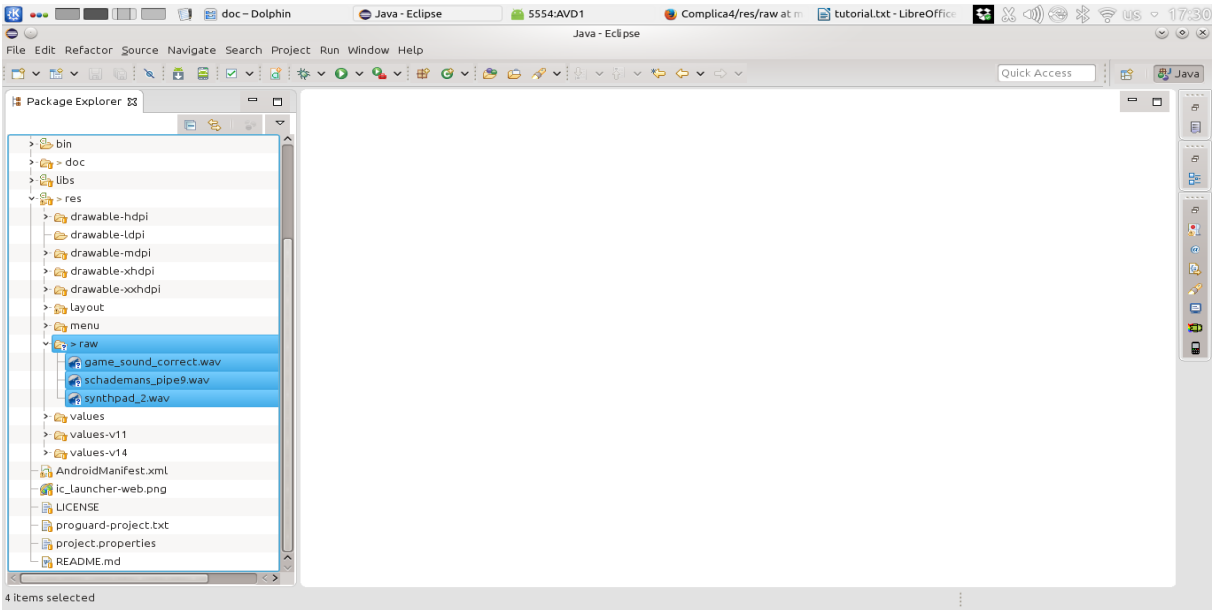
Фиг. 81 Основен екран на играта, визуализиращ информация взета от обектния модел

Двете основни сетива, с които хората възприемат света, са зрението и слухът. Визуализирането на игралното табло дава добра представа на потребителя как протича процесът на игра, но усещането може да се подсили с подходящо избрани звуци.



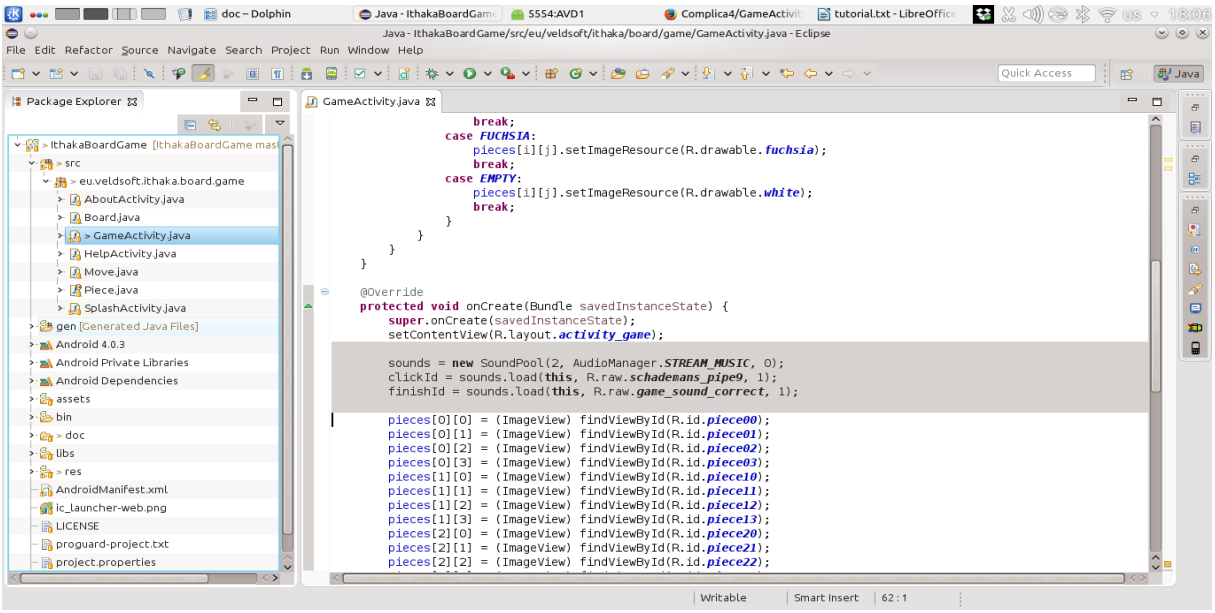
Фиг. 82 Структури от данни за използване на звуци

Платформата Android предоставя възможност за работа със звуци, под формата на пул от звукови ресурси. Две са основните събития, които си заслужава да бъдат допълнително подсилени със звуково оформление – избор на клетка за игра и край на разиграването.



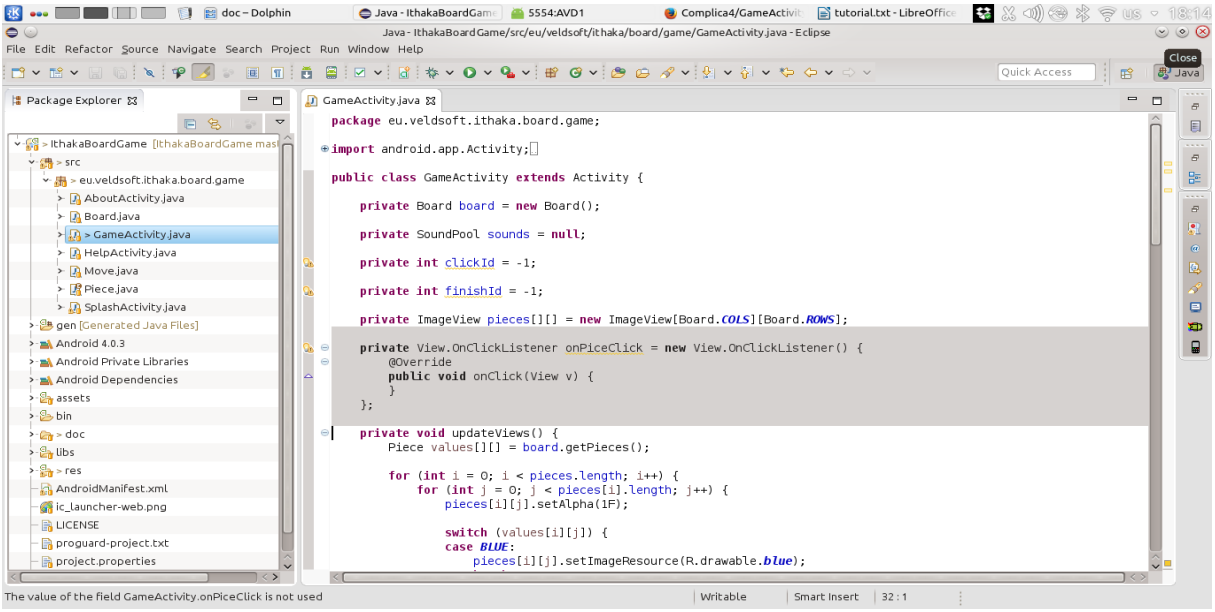
Фиг. 83 Звукови файлове

По подобие на графичните ресурси проектът на Eclipse позволява включването на звукови ресурси. Един от най-подходящите файлови формати е форматът WAVE. Този формат няма компресия и дава достатъчно добро качество на звука. Подходящ е за малки по размер звукови ресурси.



Фиг. 84 Програмен достъп до звуковите ресурси

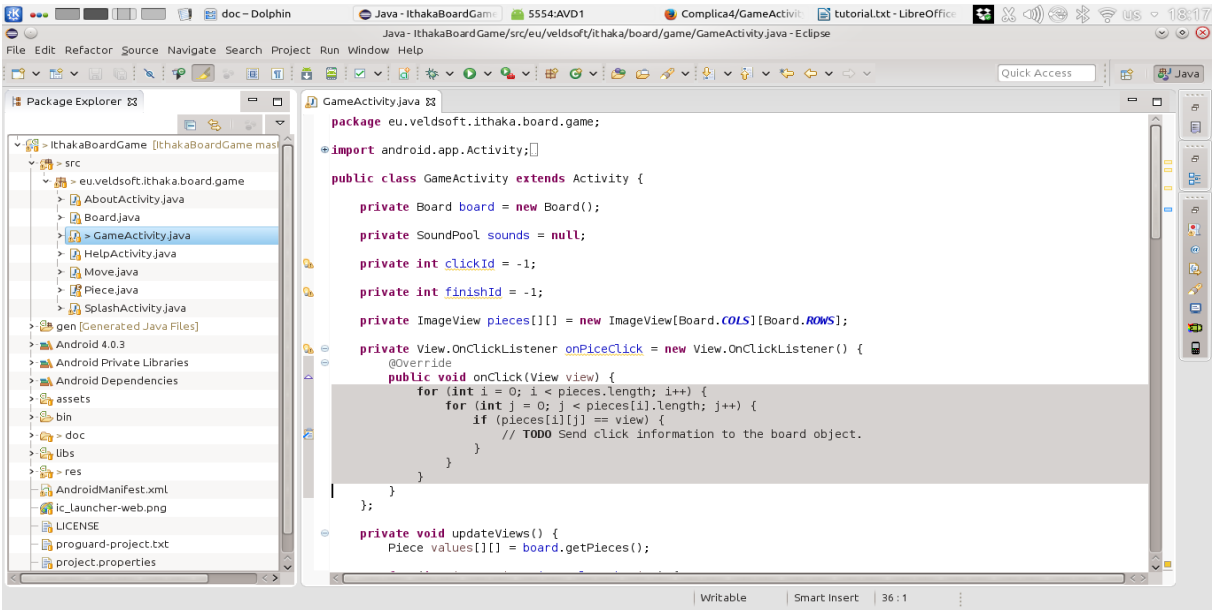
Звуковите ресурси се използват програмно чрез зареждане от фактори функция. Веднъж заредени, звуците се използват с техните идентификатори в пула от звуци.



Фиг. 85 Обработка на потребителския вход

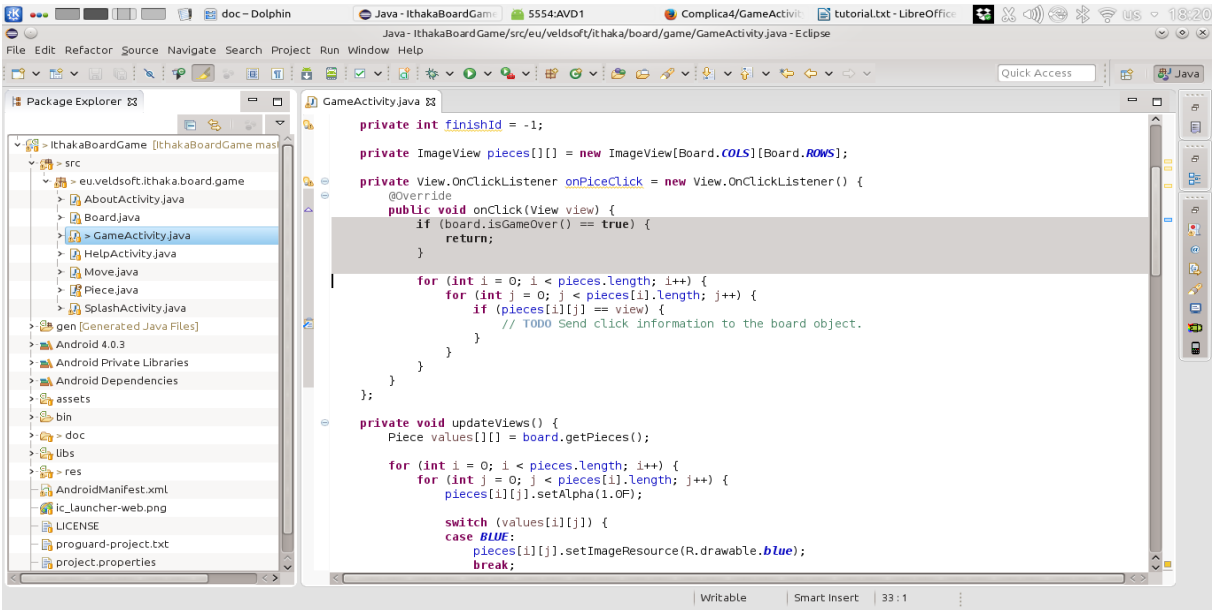
Java е език разчитащ на събитийно ориентирано написан програмен код. Тази философия е спазена и в платформата Android. Приложните програмисти вписват кода, който приложението им трябва да изпълнява във функции, наречени event handlers. За да бъде възможна тази обработка, всеки визуален контрол трябва да се абонира за обработка на възникнало в него събитие към обект-слушател (listener object). Възможни са различни конфигурации между визуалните компоненти и обектите-слушатели. Най-често използваният подход е към всеки визуален контрол да бъде закачен само един отделен обект-слушател. Програмно е позволено да се закачат множество обекти-слушатели към един визуален контрол, но това създава проблем за начина, по който ще бъдат уведомени различните обекти, при настъпване на събитие. Друг много популярен

подход е един и същи обект-слушател да бъде прикачен към множество визуални компоненти. При тази стратегия възниква допълнително усложнение да се идентифицира кой визуален компонент е предизвикал възникването на съответното събитие. Това, разбира се, е напълно възможно, тъй като функцията, обработваща събитието, получава входящ параметър с референса на визуалния компонент, който е отговорен за възникване на събитието.



Фиг. 86 Определяне на визуалния компонент, предизвикал събитието

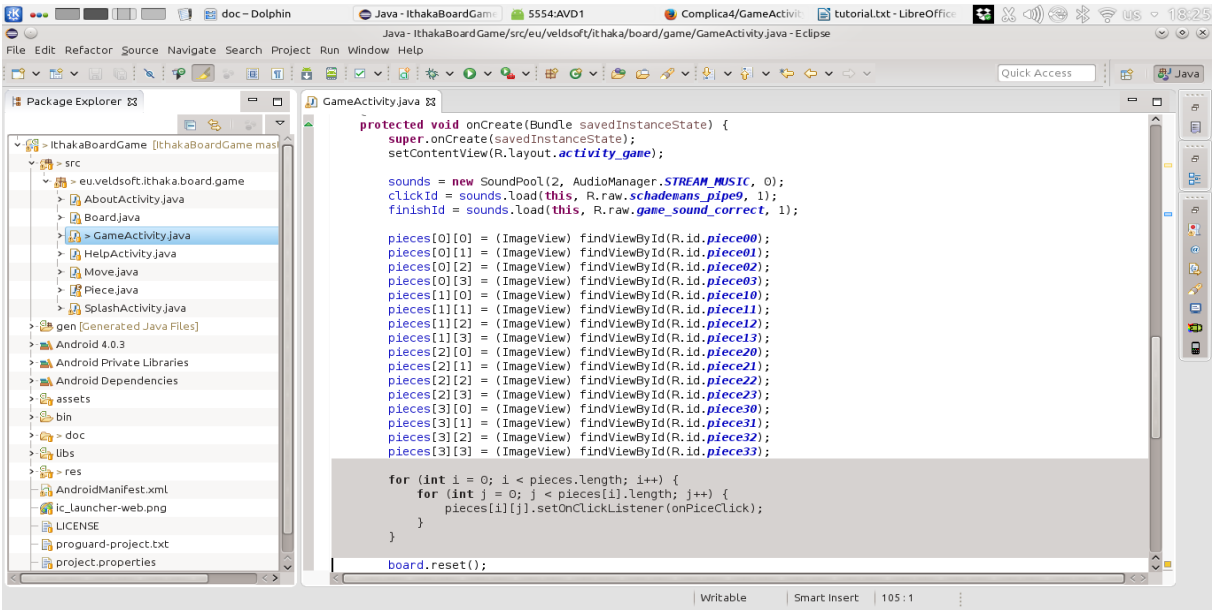
При наличие на референция към визуалния компонент, който е предизвикал събитието, е достатъчно да се проверят всички визуални компоненти и референциите им да бъдат сравнени, за да се установи кой елемент точно е натиснал потребителят.



Фиг. 87 Ако играта е приключила, потребителят няма контрол над игралното табло.

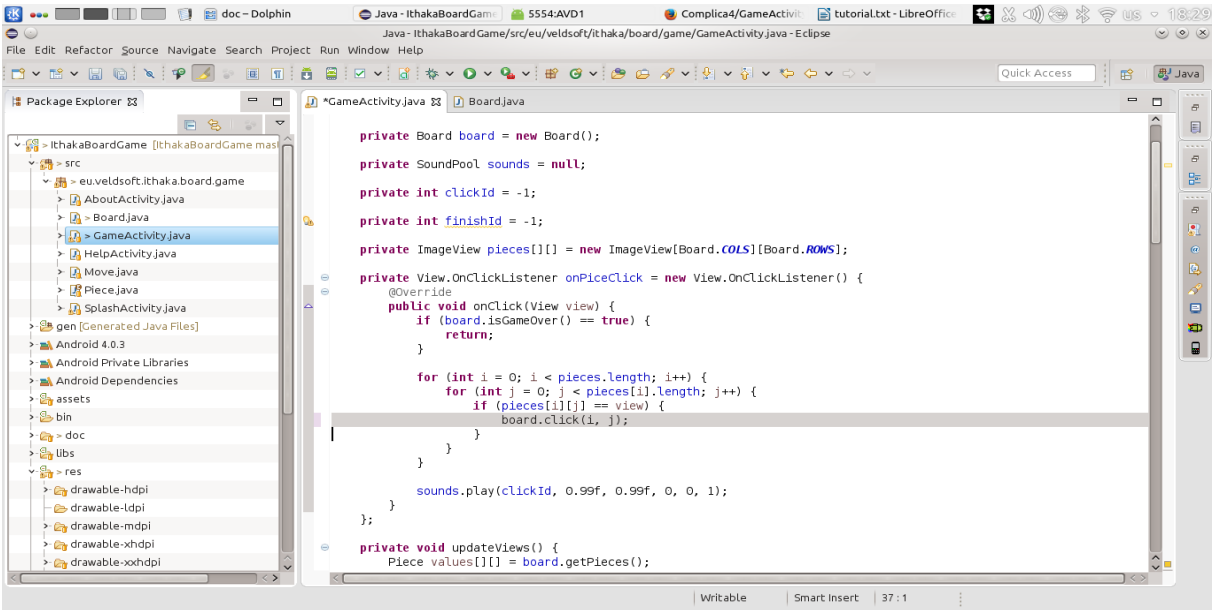
Когато играта приключи, следва потребителят да не може по никакъв начин да модифицира игралното табло. Той би могъл да започне нова игра като избере

съответстващата опция в менюто на екран.



Фиг. 88 Прикачване на обекта-слушател към визуалните компоненти

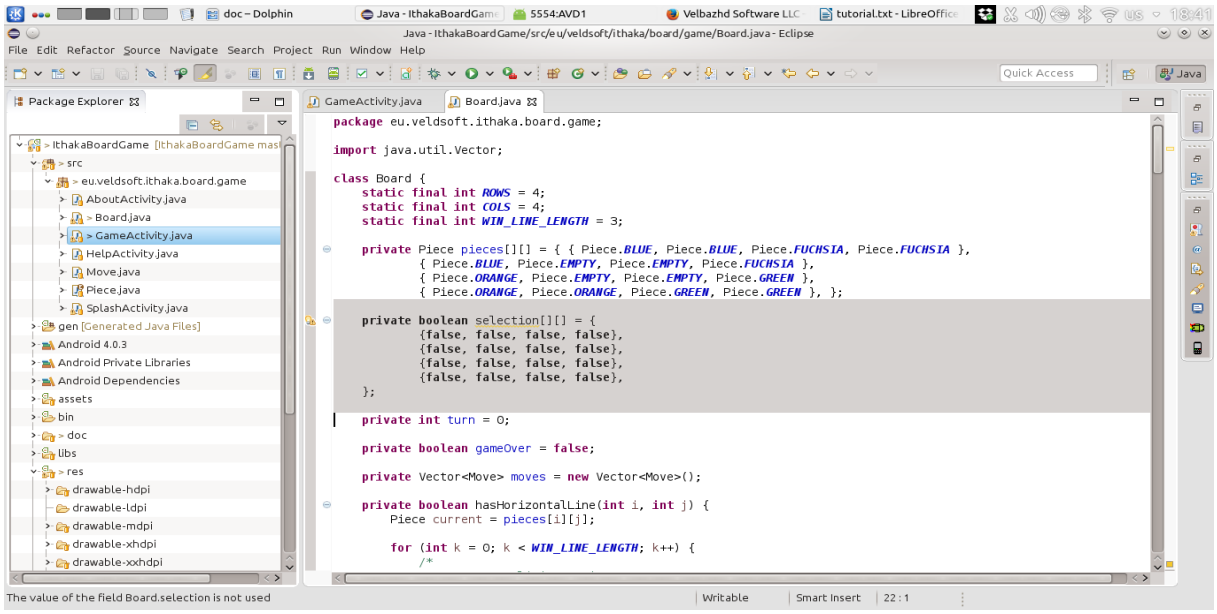
За обектите-слушатели има няколко различни стратегии. Те могат да бъдат анонимни обекти, от анонимни класове или пък обекти с имена, но от анонимни класове (използваният в конкретния случай), а също така могат да бъдат обекти с имена от класове с имена. Въпрос на експертен избор е дали класът на обекта-слушател да бъде анонимен, вътрешен скрит или пакетен външен. В настоящата разработка е създаден само един обект с име от анонимен слушателски клас. Този обект се закача за всичките 16 визуални компонента от ImageVeiew тип, които формират решетката на игралното табло.



Фиг. 89 Предаване на потребителския вход към обектния модел

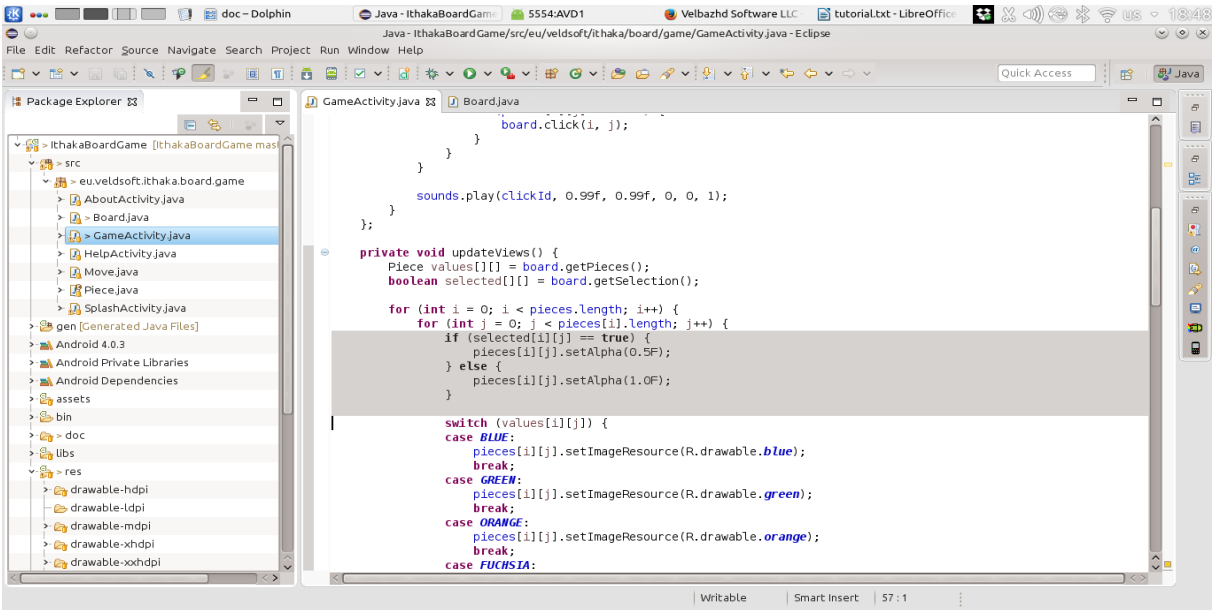
Втората основна задача на потребителския интерфейс е да предава действията, избрани от потребителя към обектния модел на приложението. За тази цел обектът-слушател

определя коя клетка в решетката е била избрана и извиква, с подходящи параметри, функция в обектния модел, която да обработи възникналата промяна вследствие на действие от външния свят.



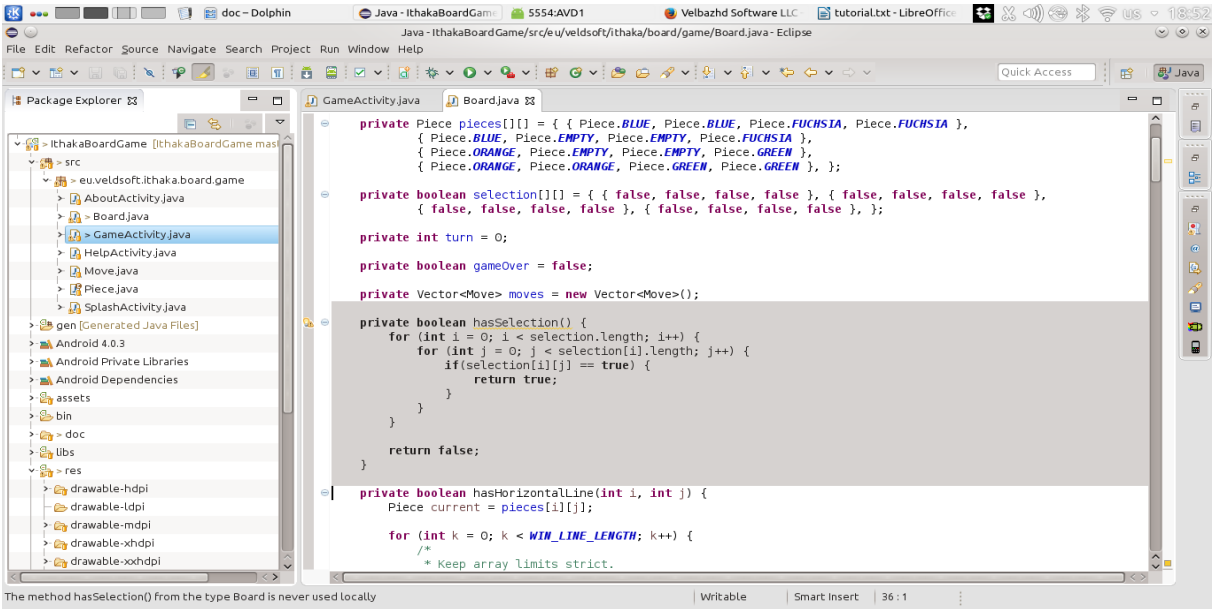
Фиг. 90 Структура данни, отчитаща избора на клетки върху игралното поле

Появата на възможност за избор на клетка от игралното поле налага допълнително разширяване на възможностите, заложени в обектния модел. Има различни начини, по които да се съхранява информацията за избора на клетка в матрицата. Възможен вариант е две цели числа да съхраняват координатите на избраната клетка. Един такъв подход за моделиране на ситуацията с избрана клетка би бил удачен за първоначалната постановка на играта, но ще се окаже неудачен, ако в последствие се появи възможност повече от една клетка на игралното табло да бъде в състояние избрана. Точно поради тази причина, най-разумно е всички клетки бъдат онагледени със състояние избрана/неизбрана. При такъв избор на структура от данни, масивите pieces и selection се наричат паралелни масиви. На по-следващ етап от развитието на проекта е разумно вместо два паралелни масива да бъде направен един масив от тип Cell, като класът Cell би съдържал две вътрешни полета – пул и състояние на избиране. Новият двумерен масив също подлежи на първоначално установяване при рестартиране на играта, а също така се съпровожда от getter функция, която да го връща на обкръжаващата го среда.



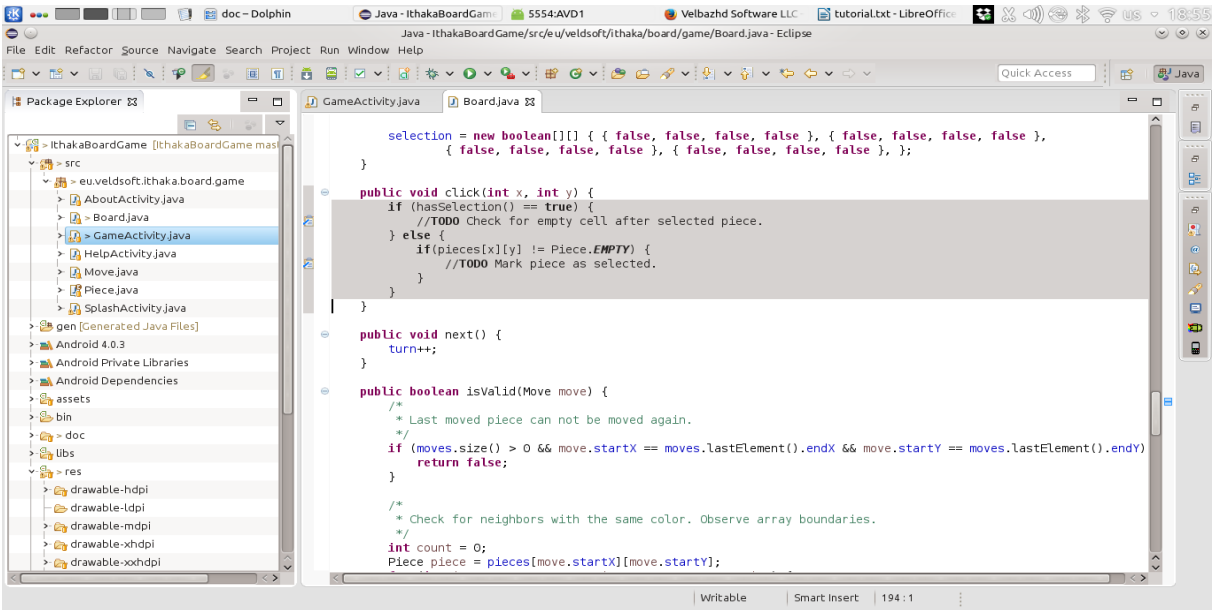
Фиг. 91 Визуално представяне на избраните клетки

Един то най-елегантните начини да се визуализира състоянието на избраните клетки е като се модифицира каналът за прозрачност на съответните визуални контроли.



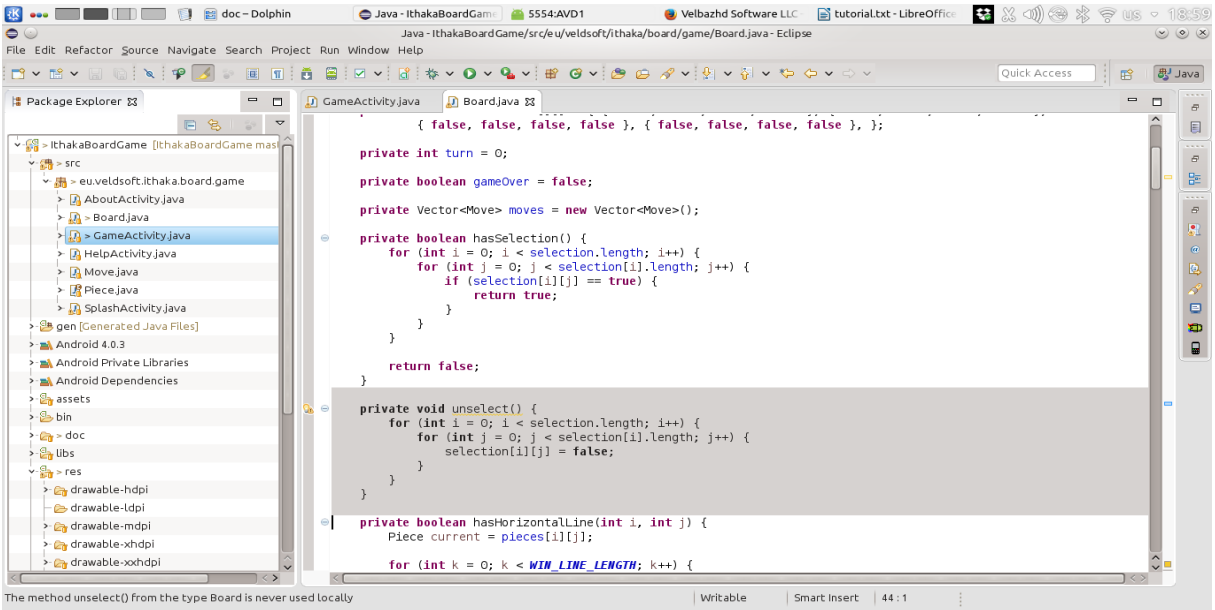
Фиг. 92 Проверка за наличие на избрана клетка

Потребителят избира клетка от решетката на играта, когато иска да определи кой пул да бъде преместен. Поради тази причина е важно в обектния модел да се знае дали клетката вече е избрана или все още предстои да бъде избрана.



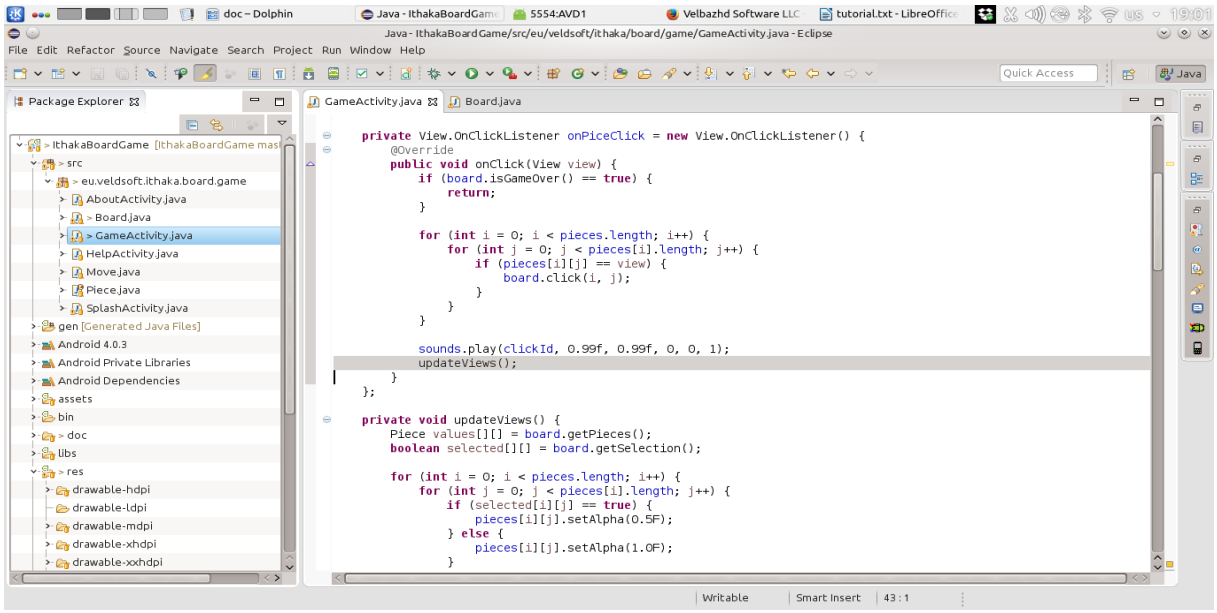
Фиг. 93 Избор на действие, според това дали е избрана клетка

Според това дали е избрана клетка от игралното табло, в обектния модел се извършват различни действия. При неизбрана клетка се прави опит за избиране на клетка. При избрана клетка се прави опит за преместване на пула.



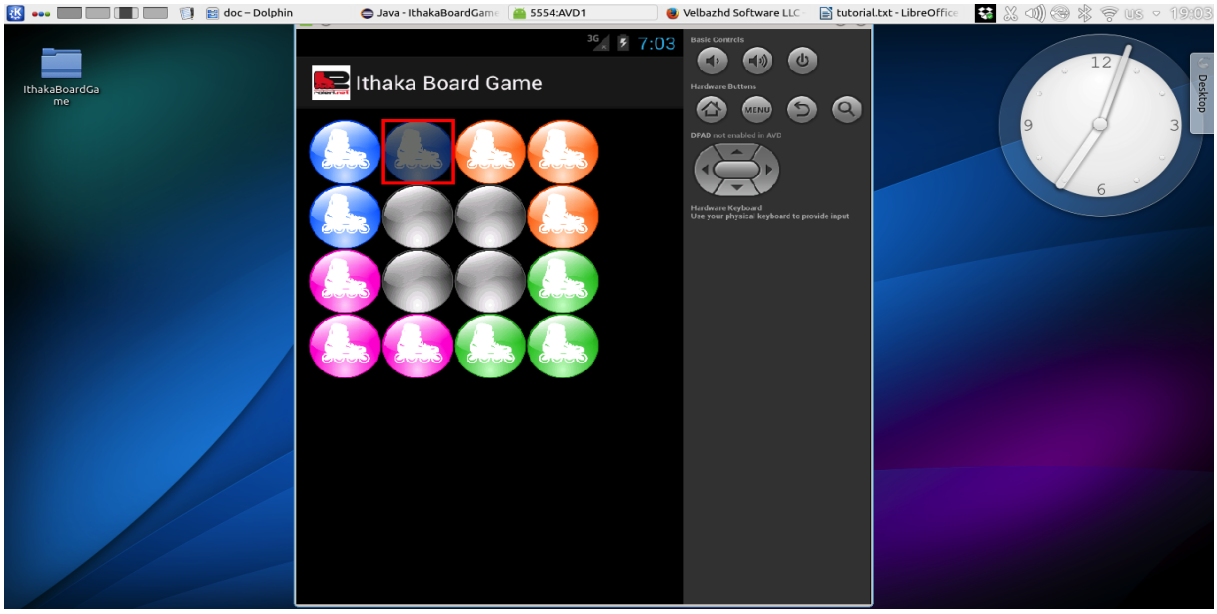
Фиг. 94 Премахване на избраните клетки

След като ходът на съответния играч приключи е важно маркировката на клетките да бъде премахната.



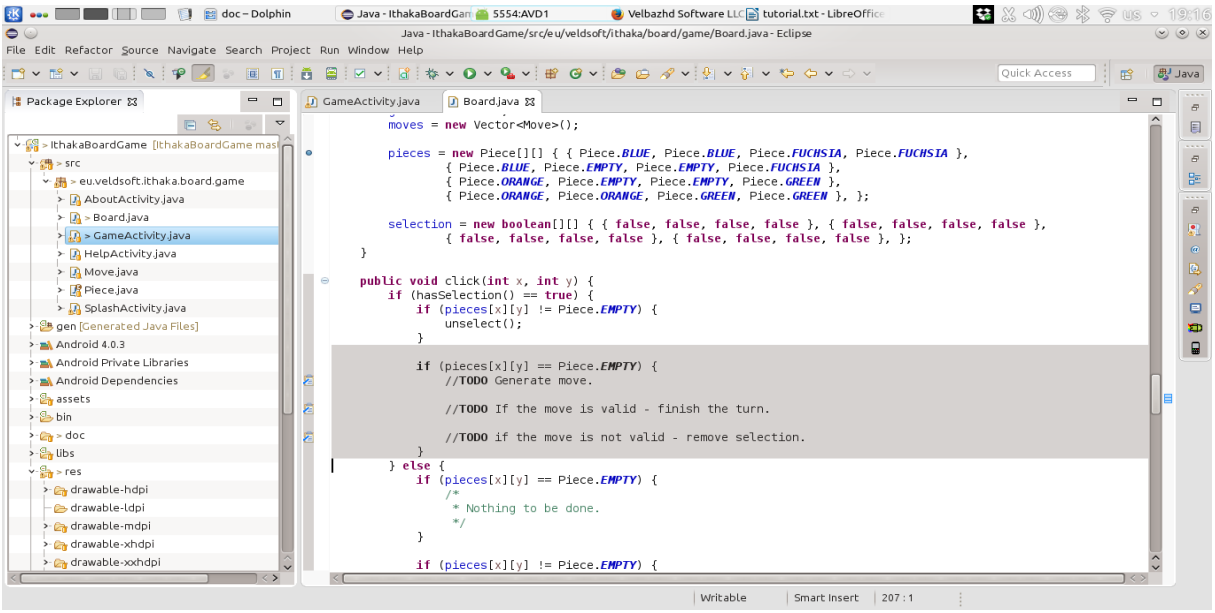
Фиг. 95 Обновяване на визуалните компоненти, след взаимодействие на потребителя с устройството

Когато потребителят извърши определено действие на екрана, то се отразява в обектния модел, но това трябва да доведе и до обновяване на визуалните компоненти. Един от класическите подходи при реализиране на тези процедури е използването на дизайн шаблона Model-View-Controller (MVC). Тъй като разработваната игра е твърде елементарна, то MVC шаблонът би бил едно ненужно усложнение, тъй като той е подходящ за по-мащабни софтуерни решения.



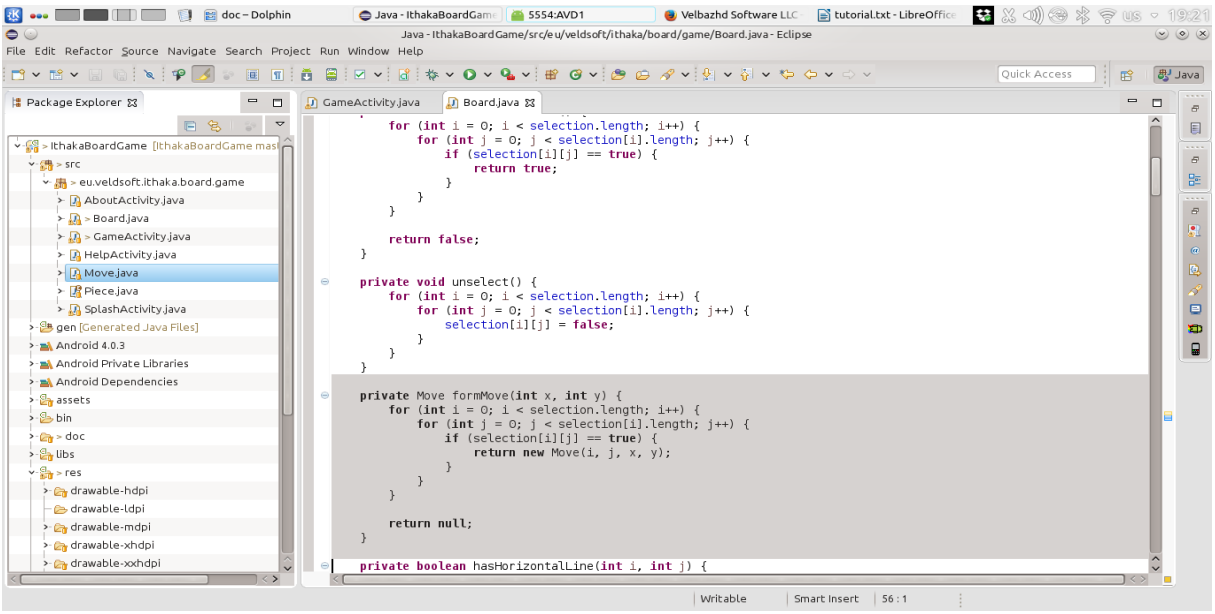
Фиг. 96 Пул в избрано състояние (синият пул, на първия ред, втора колона)

След като бъде избран пул, то трябва да се случат серия изчислителни стъпки.



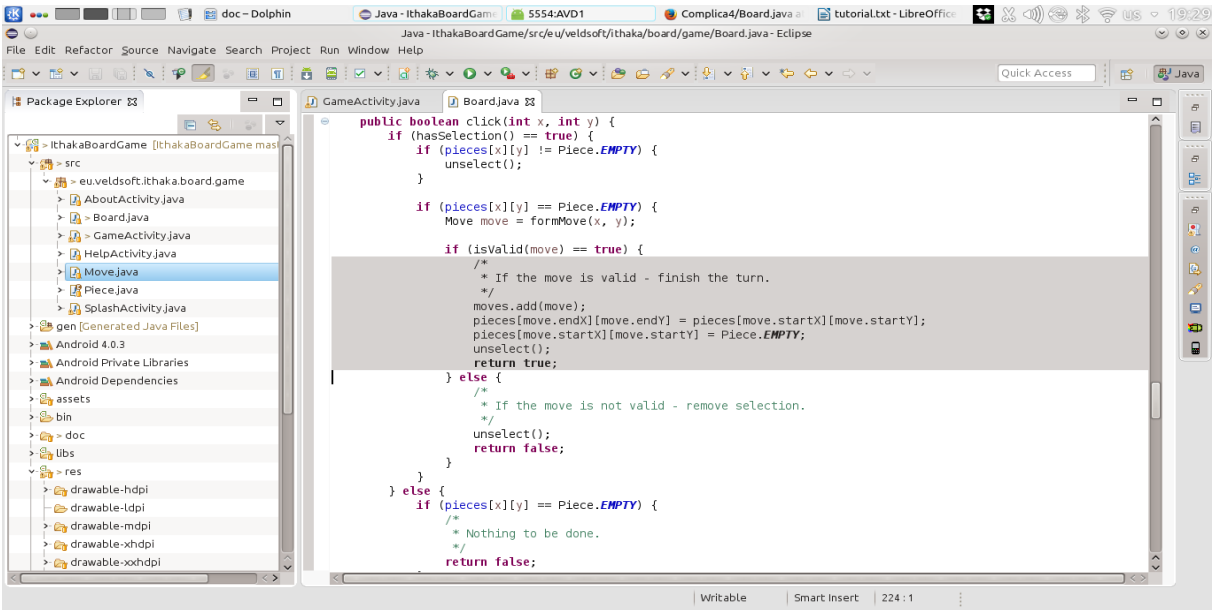
Фиг. 97 Действия за осъществяването на ход

На първо място трябва да се генерира обект от тип Move. Ако генерираният ход е валиден, то той трябва да бъде изпълнен и съответният играч да приключи хода си. Ако генерираният ход е невалиден, то изиграването на ход трябва да се рестартира, като се премахне направената селекция и се даде следващ шанс на играча да избира.



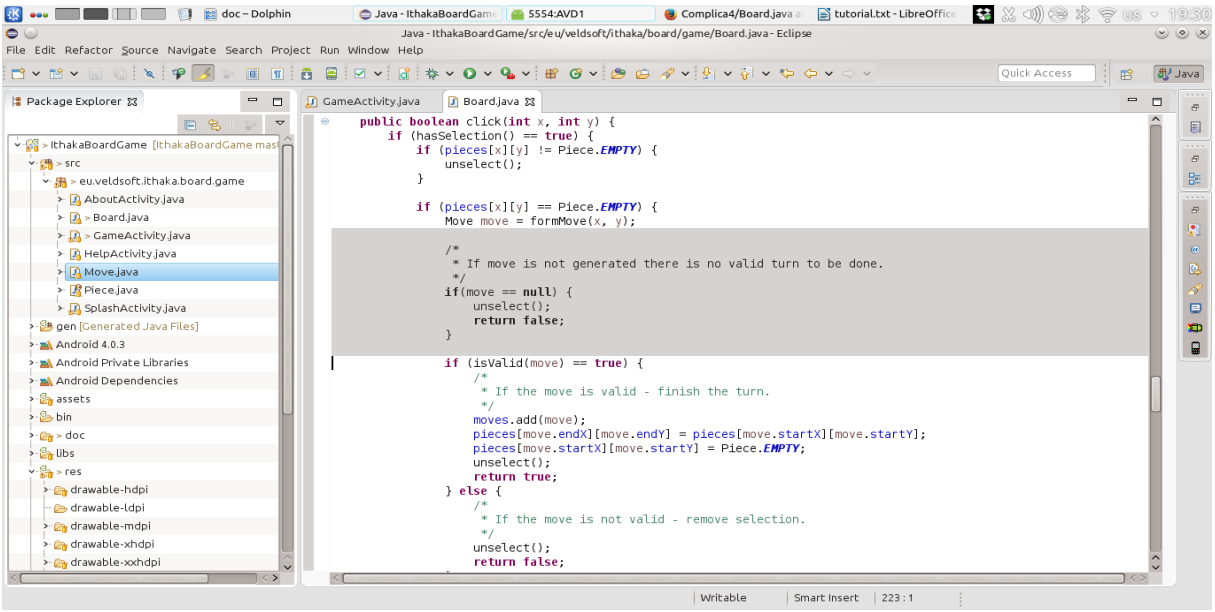
Фиг. 98 Изработване на обект от тип Move

Изработването на обект от тип Move се извърша на база предходната селекция на пул и текущия избор на празна клетка, където пулът трябва да се премести.



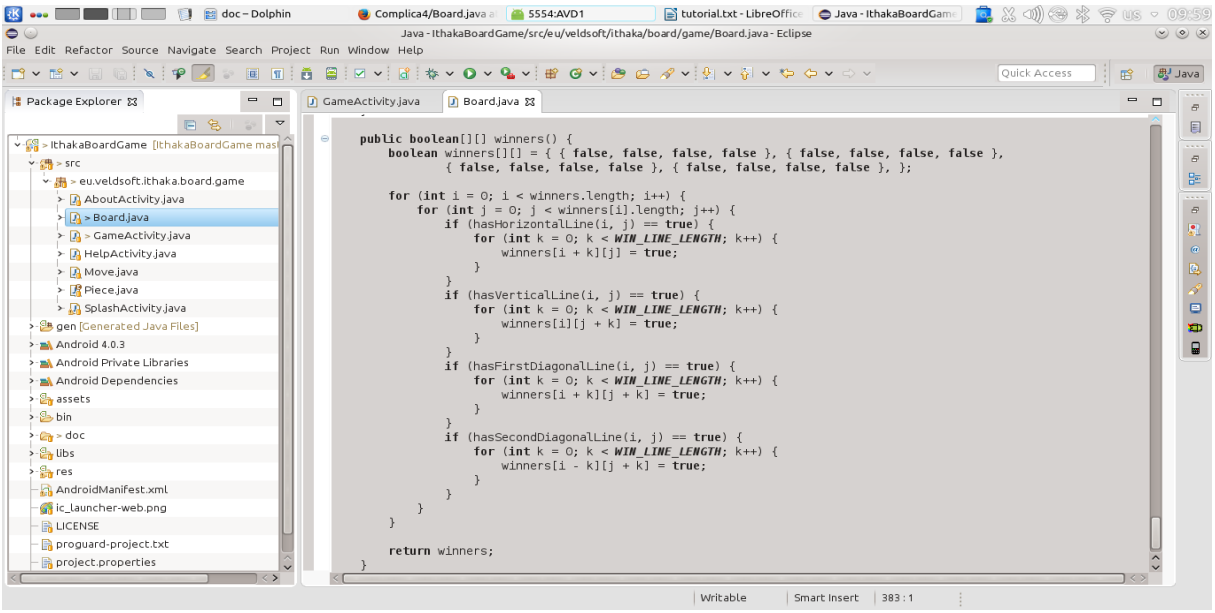
Фиг. 99 Проверка за валидността на хода

След формирането на хода следва проверка за неговата валидност.



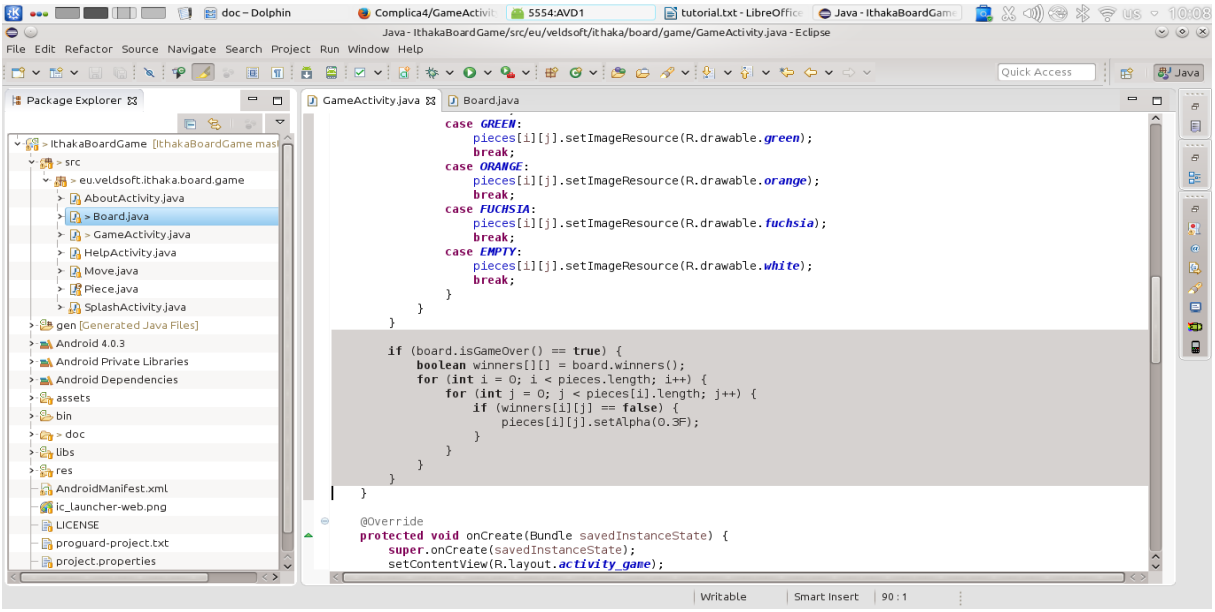
Фиг. 100 Проверка за неуспешно генериране на ход

Ако поради някаква причина генерирането на ход не е било успешно, то процедурата по изиграване на ход отново трябва да се рестартира.



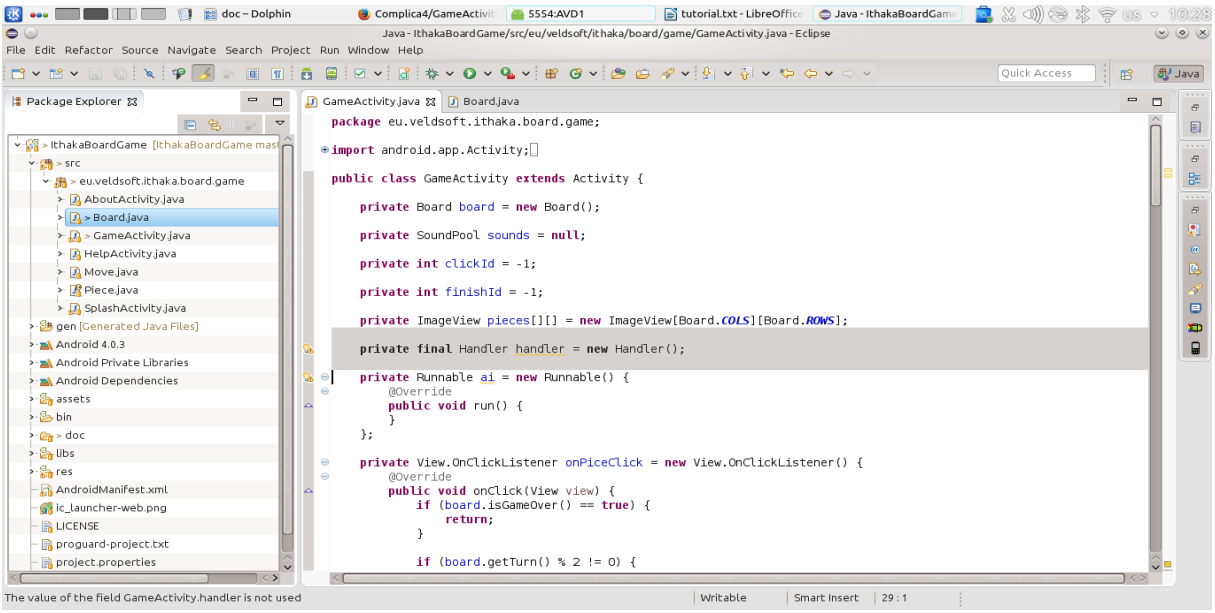
Фиг. 101 Определяне на победителя

При определянето на победителя най-рационално е да се маркират точните пулове, които формират печелившата комбинация.



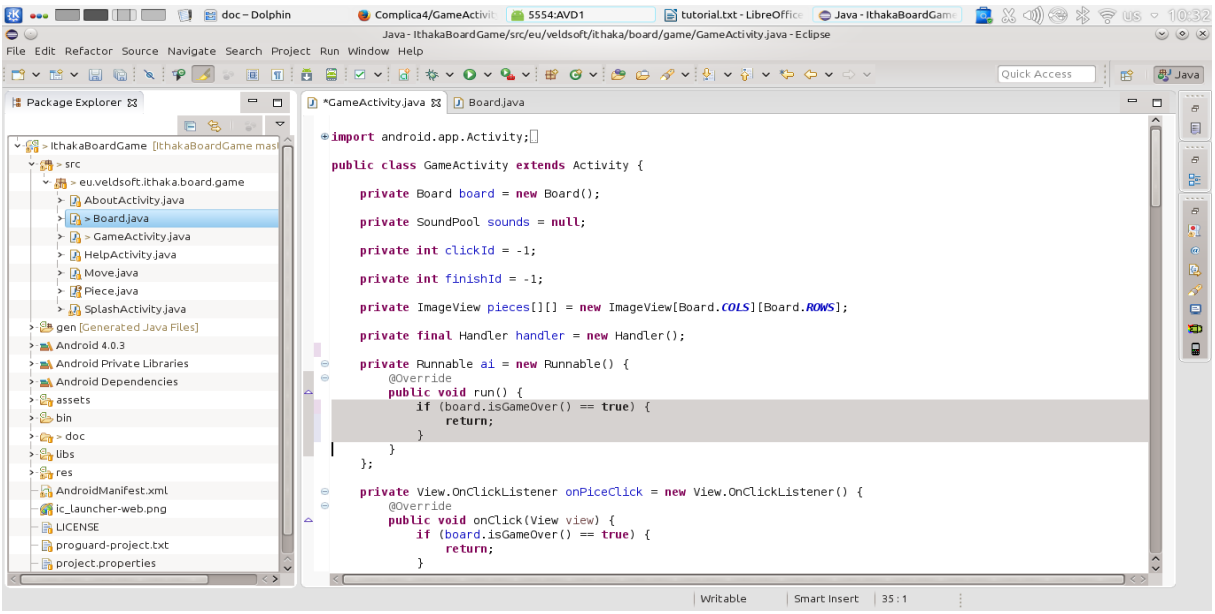
Фиг. 102 Определяне на победителя

Разполагайки с нужната информация за пуловете, участващи във формирането на печелившата линия, тази информация може подходящо да бъде отразена в графичния потребителски интерфейс. За тази цел, всички печеливши пулове се изобразяват с пълна стойност на канала за прозрачност, а пуловете, които не участват в печелившата комбинация, се затъмняват.



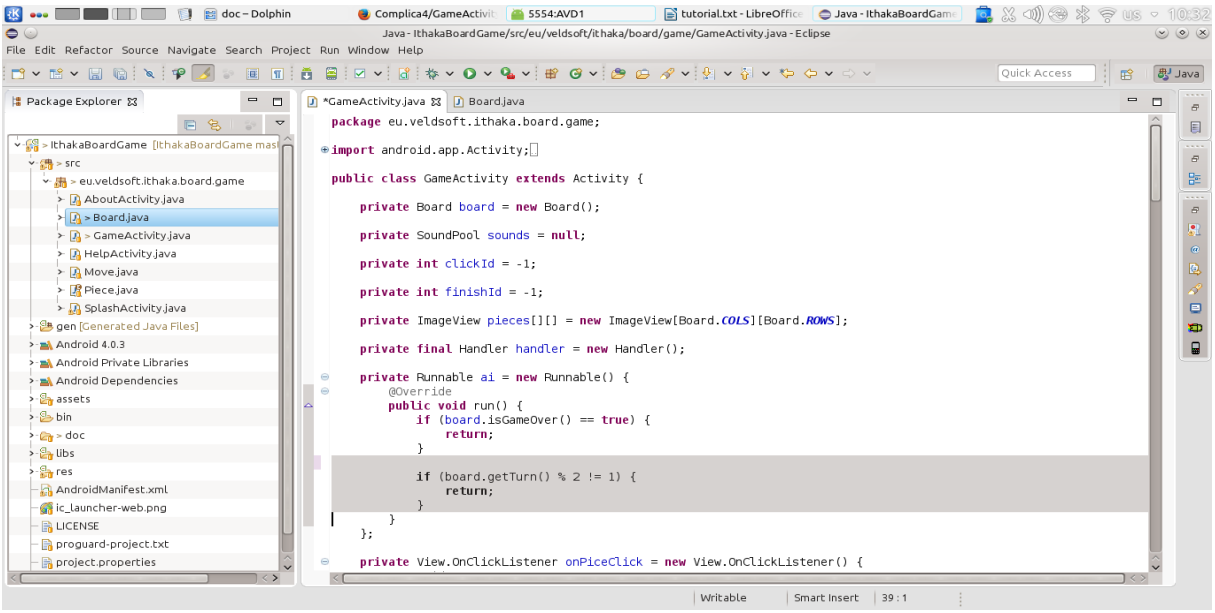
Фиг. 103 Използване на нишка за компютърния опонент

Човекът, играещ на мобилното устройство предизвиква събитията, свързани с неговите решения за игра. Това няма как да се случи по отношение на компютърния опонент. Поради тази причина един от най-удачните подходи за генериране на събития от компютърния опонент е чрез постоянно работеща Java нишка.



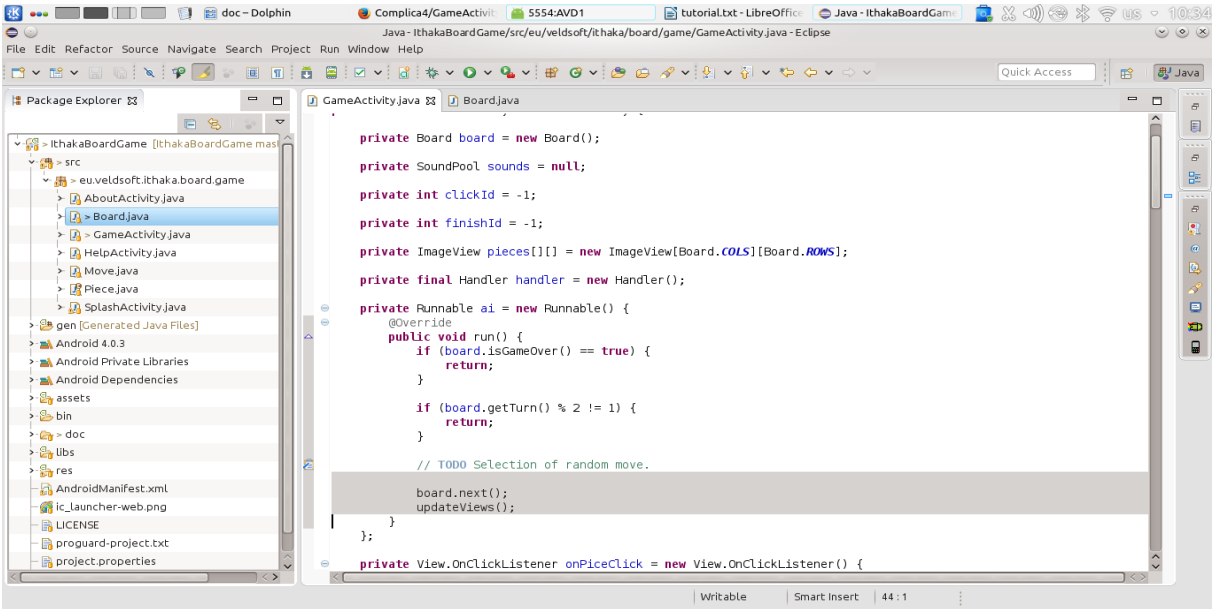
Фиг. 104 Компютърният опонент не прави ход, ако играта е завършила.

Ако играта вече е приключила, то няма причина изкуственият интелект да бъде активиран.



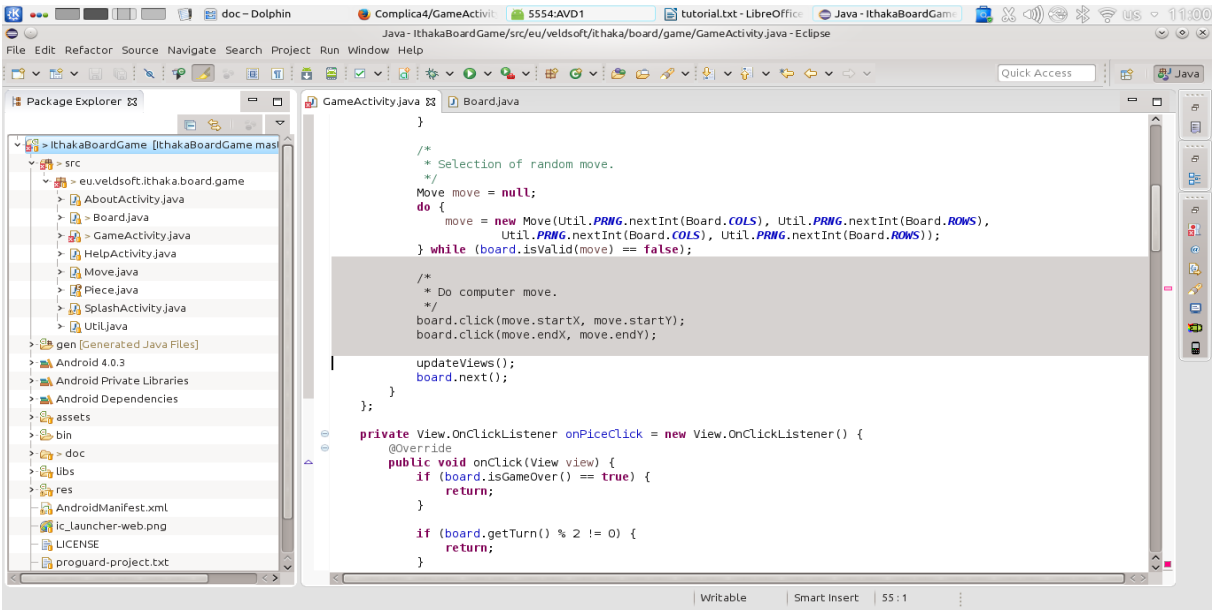
Фиг. 105 Следене на ходовете в нишката за компютърния опонент

Ако на ход е играчът човек, също няма причина изкуственият интелект да бъде активиран.



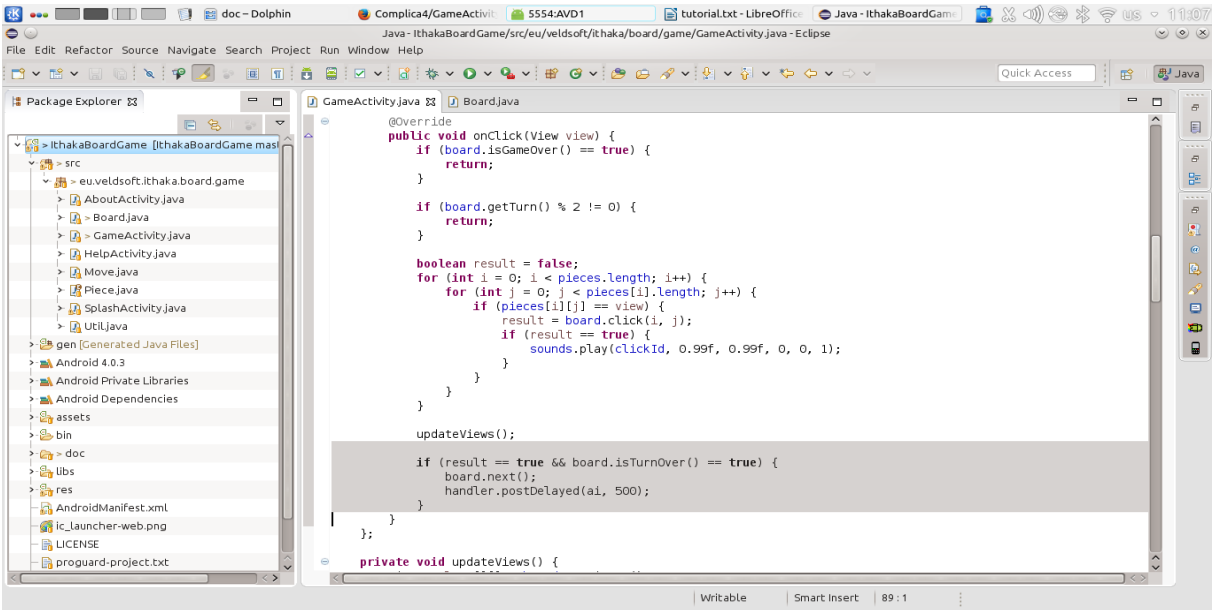
Фиг. 106 Маркиране за приключване на хода и обновяване на графичния интерфейс

Тъй като създаването на изкуствен интелект за подобен вид игри има сложност, надминаваща нивото на настоящото изложение, в разработката ще бъде избран най-елементарният вариант, а именно избор на случаен валиден ход.



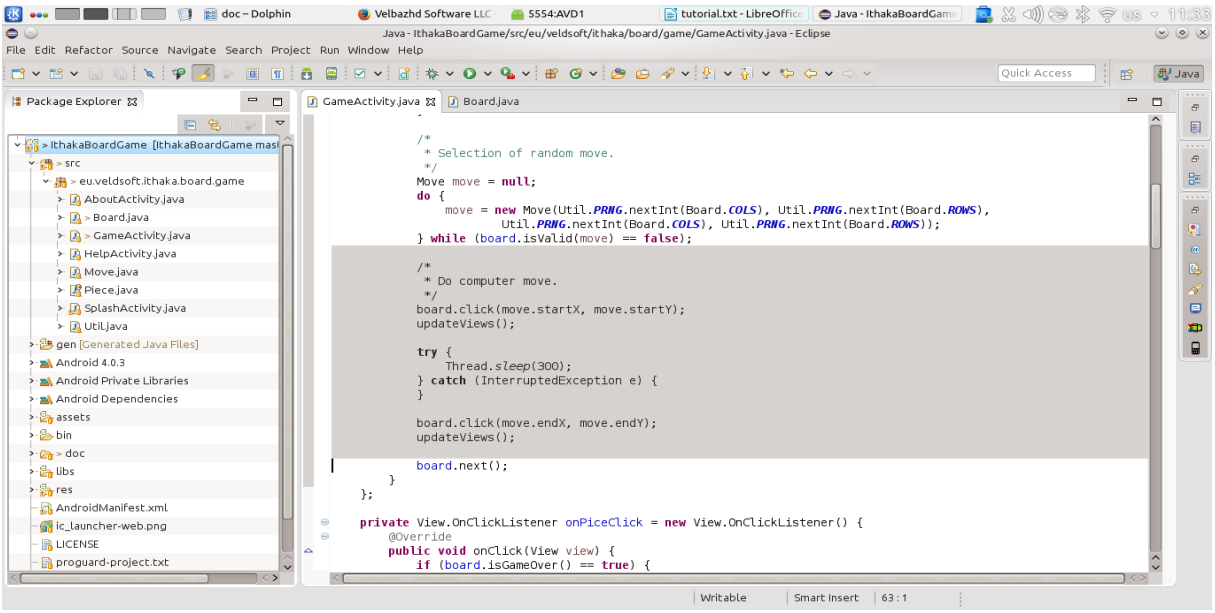
Фиг. 107 Избор на случаен валиден ход

Избирането на случаен валиден ход става с генериране на двойки координати върху игралното табло, докато не се уцели такава двойка, която да е валидна, след което следва двойно избиране, аналогично на начина, по който избира човекът играч.



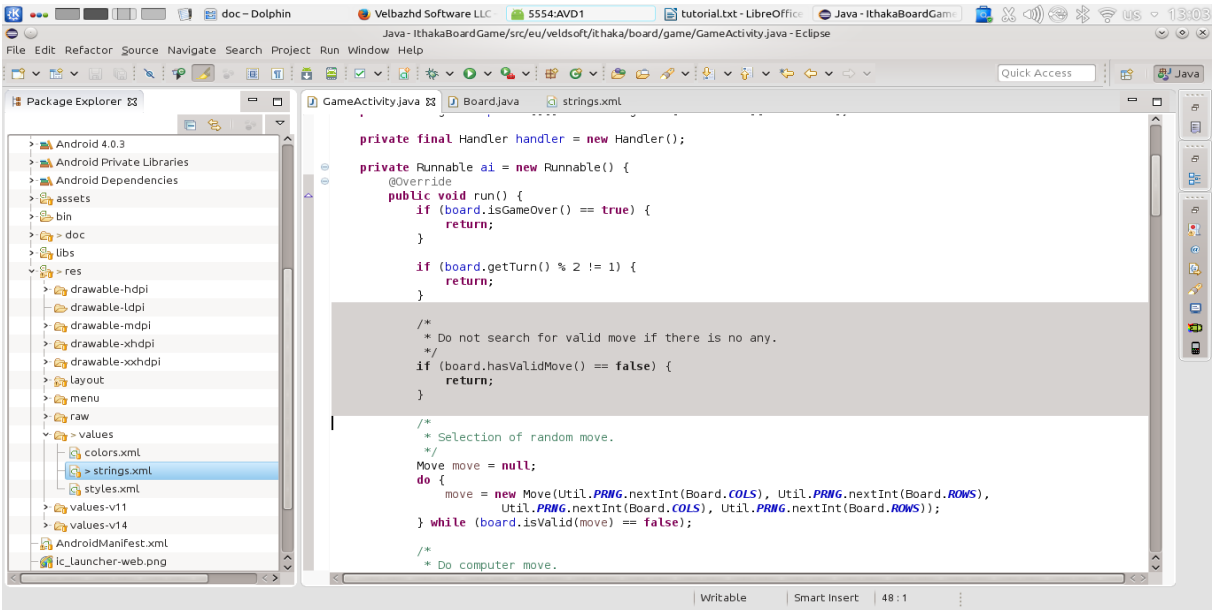
Фиг. 108 Залагане на отложено събитие за активиране на компютърния опонент

След като човекът приключи своя ход, трябва да заложи отложен старт на събитието, което ще предизвика активиране на компютърния опонент.



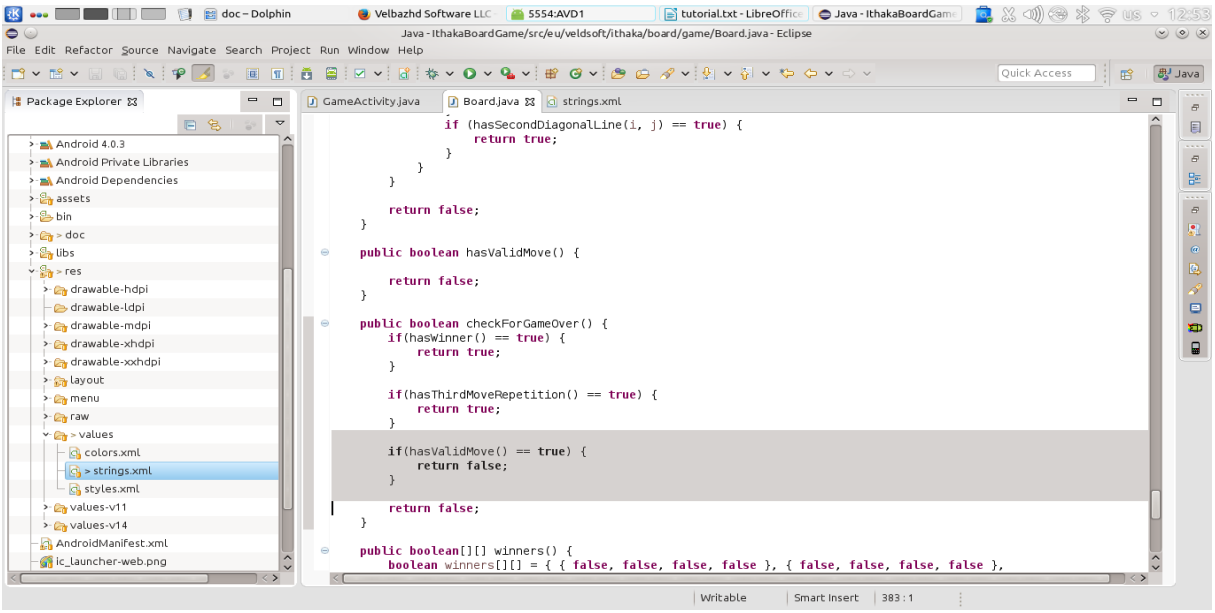
Фиг. 109 Забавяне на компютърния опонент

Тъй като компютърният опонент изпълнява твърде бързо избора на пул и клетка за преместване, то е удачно да се сложи известно забавяне, за да може човекът играч да види кой пул избира компютърът и да проследи на коя клетка е станало преместването.



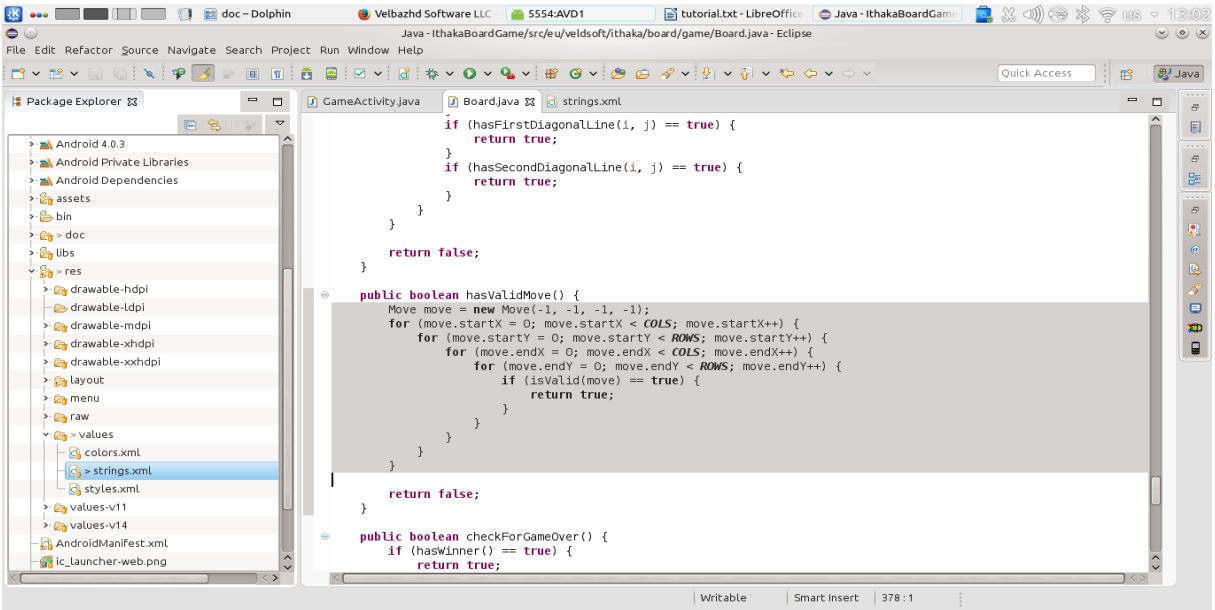
Фиг. 110 При липса на валиден ход от страна на компютърния опонент

Възможно е да не съществува валиден ход, когато е ред за игра на компютърния опонент. Тази възможност трябва да се проверя, защото в противен случай цикълът за търсене на случаен валиден ход би се превърнал в безкраен цикъл, а това е сериозна логическа програмна грешка.



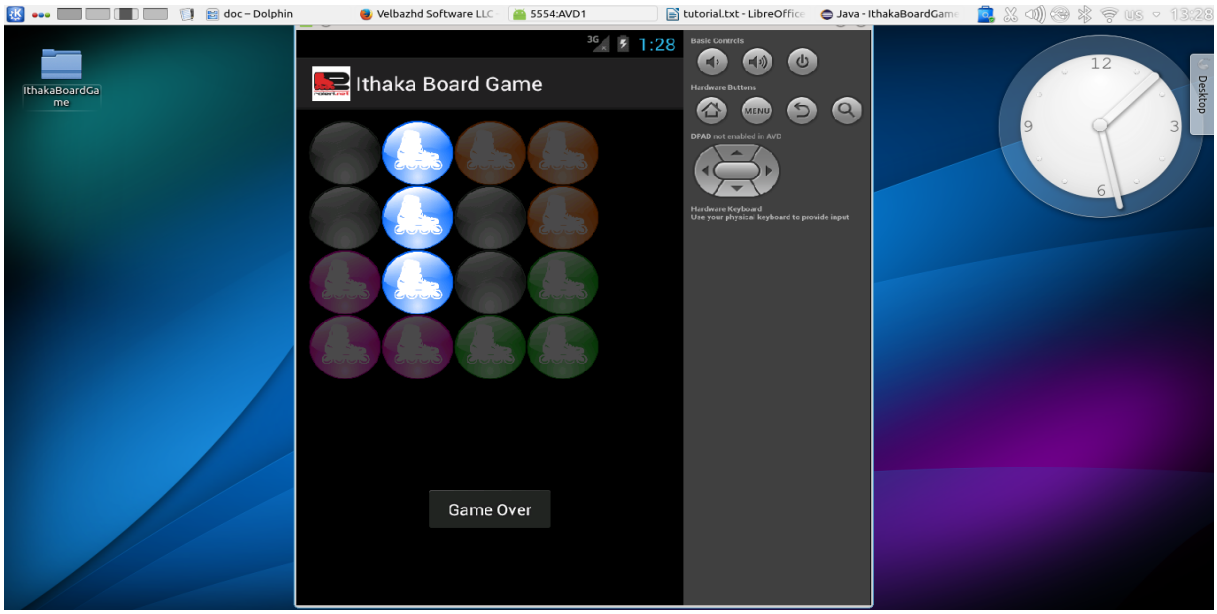
Фиг. 111 Проверка за край на играта

След всеки направен ход (независимо дали е от човека или от компютърния опонент) е възможно да настъпят условията, определящи края на играта. Точно поради тази причина след всеки ход трябва да се прави проверка за край на играта.



Фиг. 112 Проверка за наличие на валиден ход

Едно от условията за край на играта е опонентът да няма валиден ход. Най-лесен начин, това условие да бъде проверено, е на принципа на пълното изчерпване. Изпробва се всяка клетка в двойка с всяка друга и ако нито една не позволява валиден ход, то тогава се маркира краят на играта поради липса на валиден ход. Алгоритъмът е бавен, защото се състои от четири вложени цикъла, но пък размерът на игралното пространство е 16, което позволява прилагане на принципа на грубата сила.

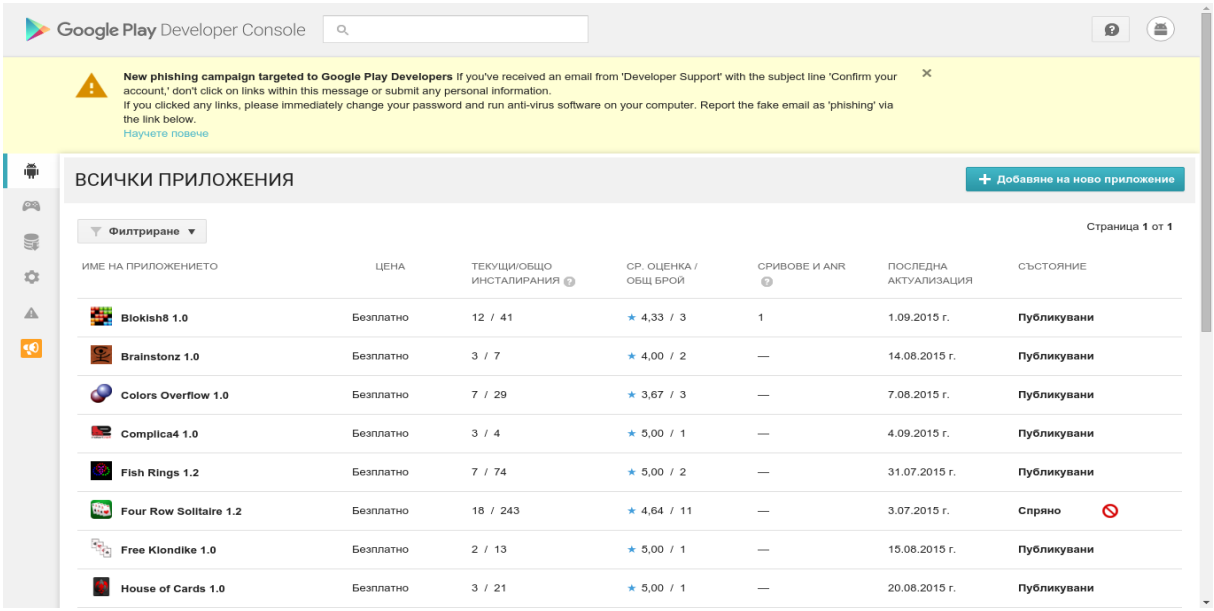


Фиг. 113 Ситуация при която играта е приключена

С това базовата разработка на играта е приключена. При този род игри е от голямо значение да се създаде достатъчно силен компютърен опонент, така че интересът на играещия да се задържи по-дълго и той да намира мотивация за изиграването на повече игри. Android е платформа, която дава много възможности за реализацията на мрежова комуникация. Игра като играта Ithaka, е идеален кандидат за добавяне на възможности за игра в мрежа по Wi-Fi и/или по Bluetooth. И двете направления изискват значително повече познания и умения спрямо нивото на текущото изложение. Поради тази причина, разработката на изкуствения интелект и играта в мрежа ще бъдат представени в изложение за по-напреднали потребители.

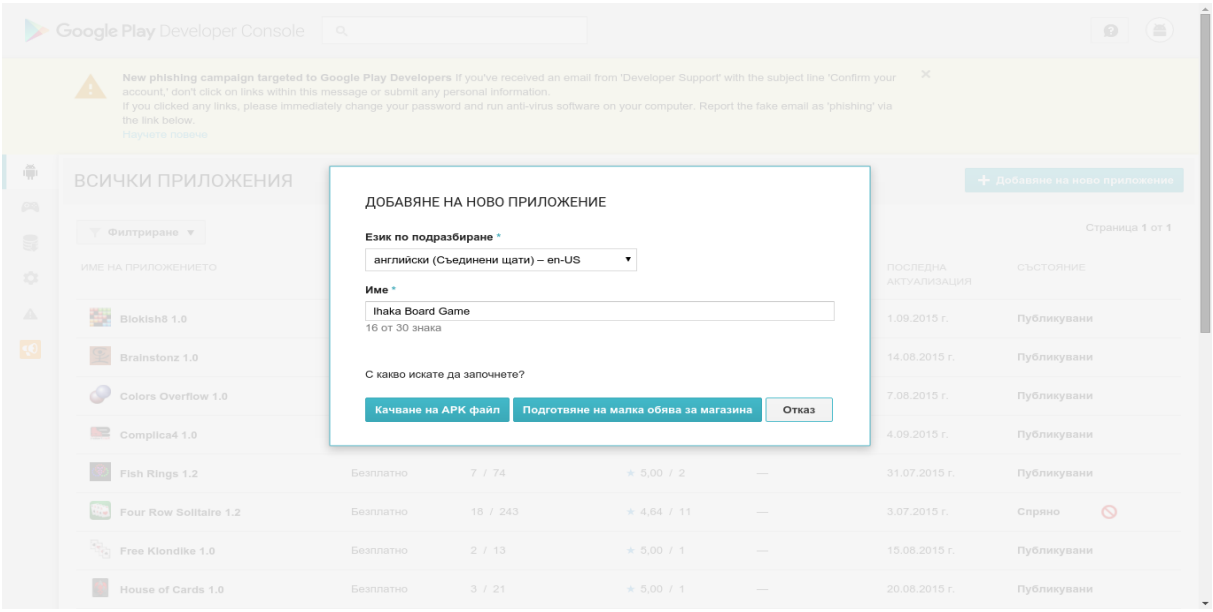
9. Разпространение на продукта

След като бъде създаден един софтуерен продукт, то той трябва да достигне до своите потребители. В случая на Android най-достъпният канал за разпространение на приложения е Google Play Store. Когато приложението е с отворен код, може да се използва и услуга за разпространение каквато е F-Droid.



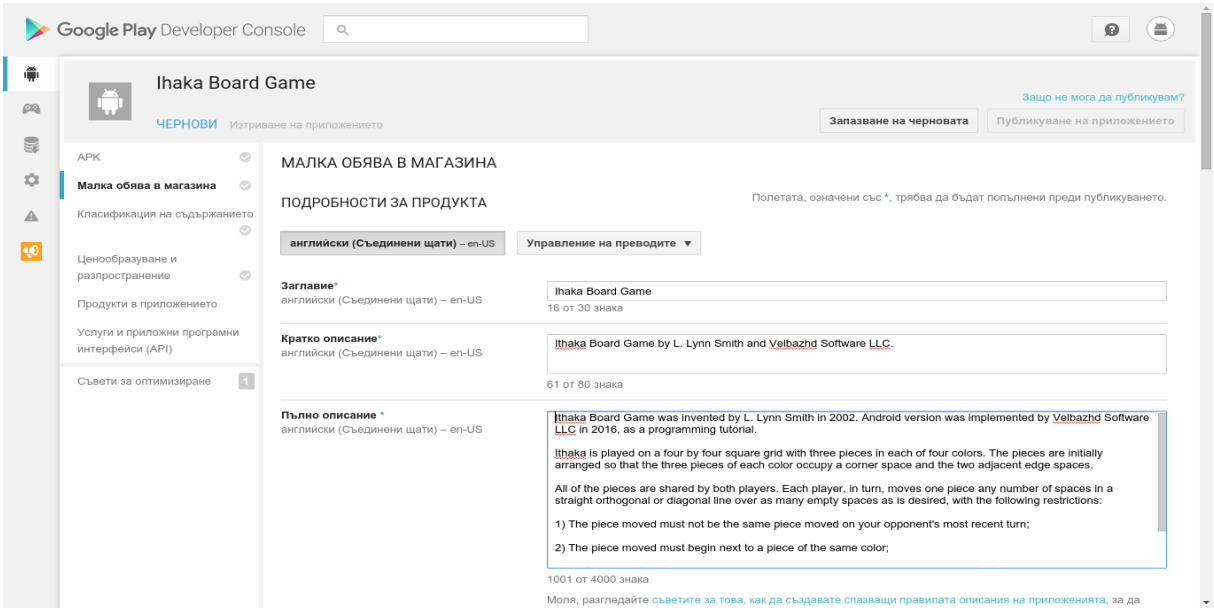
Фиг. 114 Начален екран при публикуване на ново приложение

Публикуването започва с избор на заглавие и език за разпространение.



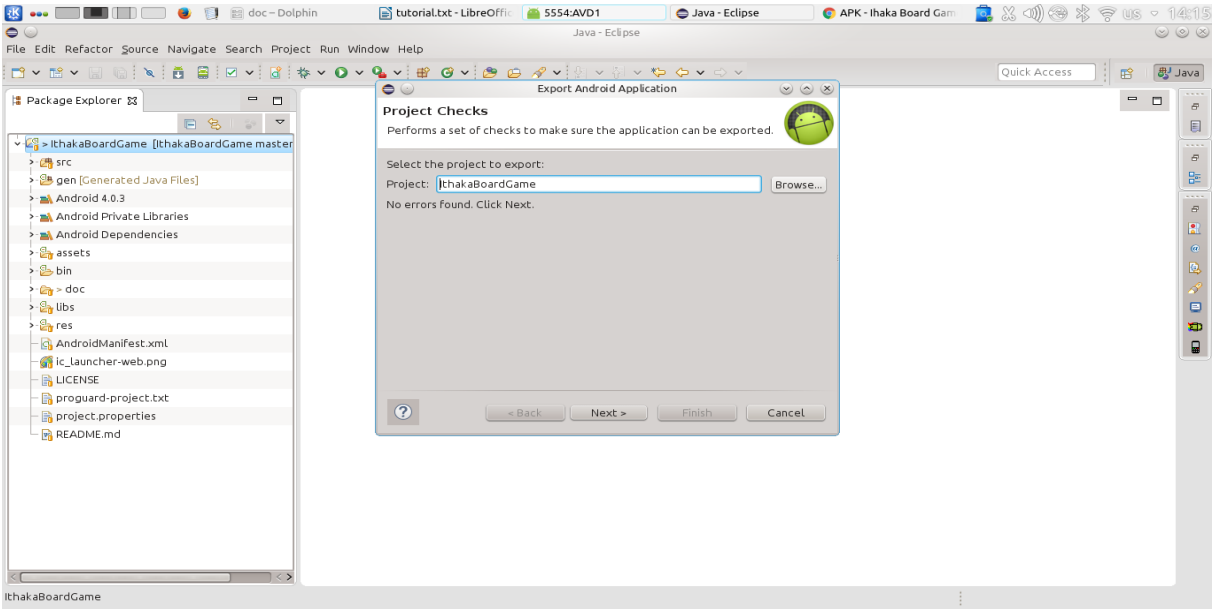
Фиг. 115 Название на приложението

Към заглавието на продукта се добавя кратко и детайлно описание.



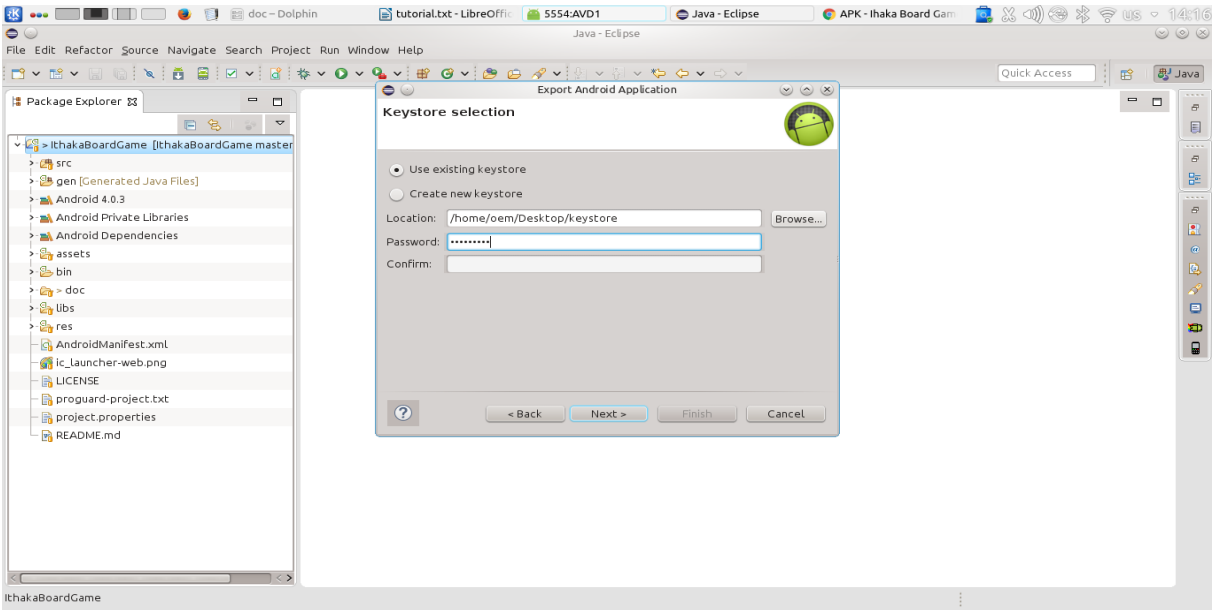
Фиг. 116 Описание на продукта

Най-важната част от публикуването е създаването на дигитално подписан изпълним файл.



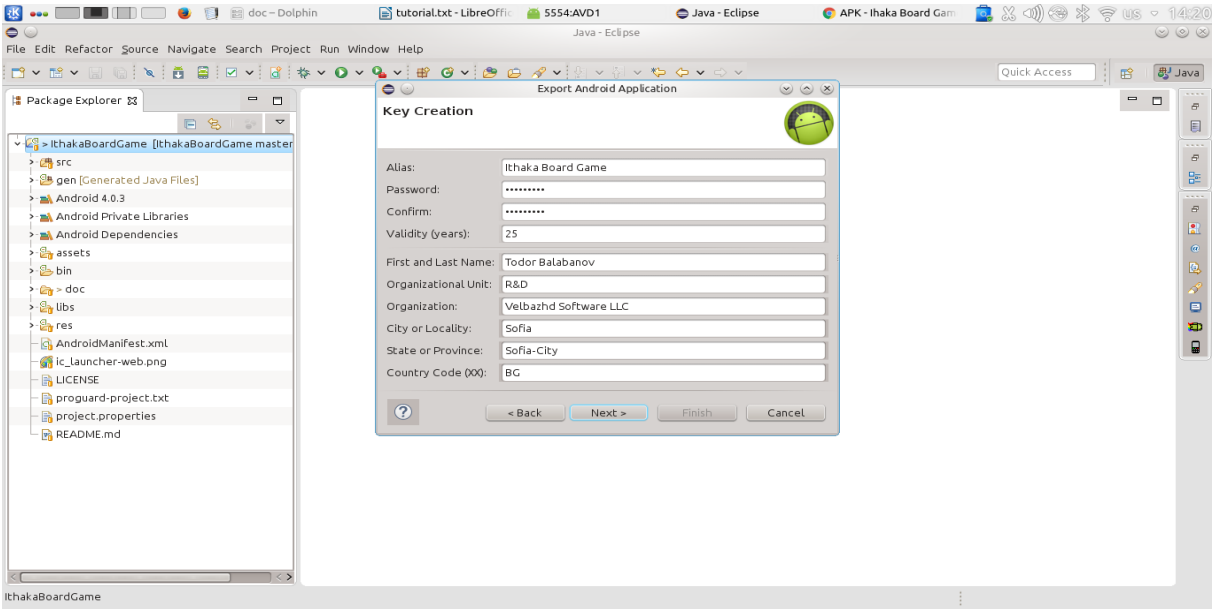
Фиг. 117 Избор на проект за експортиране

Дигиталното подписване става от контекстно зависимо меню на папката с проекта в Eclipse.



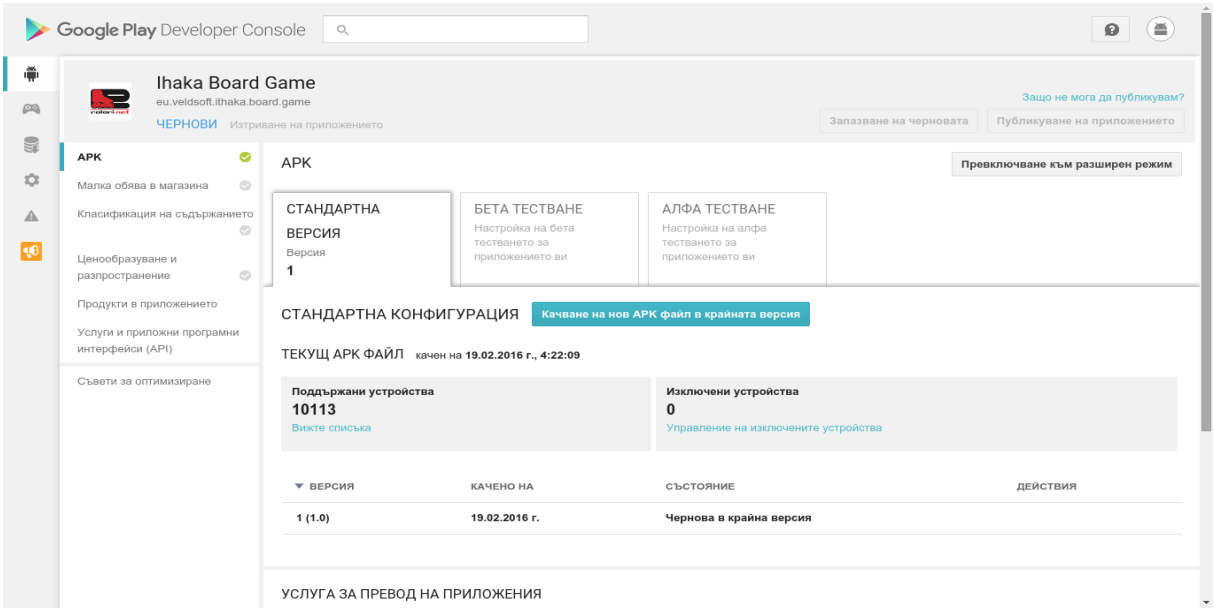
Фиг. 118 Хранилище за дигиталния ключ

Подписването се извършва с дигитален ключ, който се съхранява в специално създадено за тази цел хранилище.



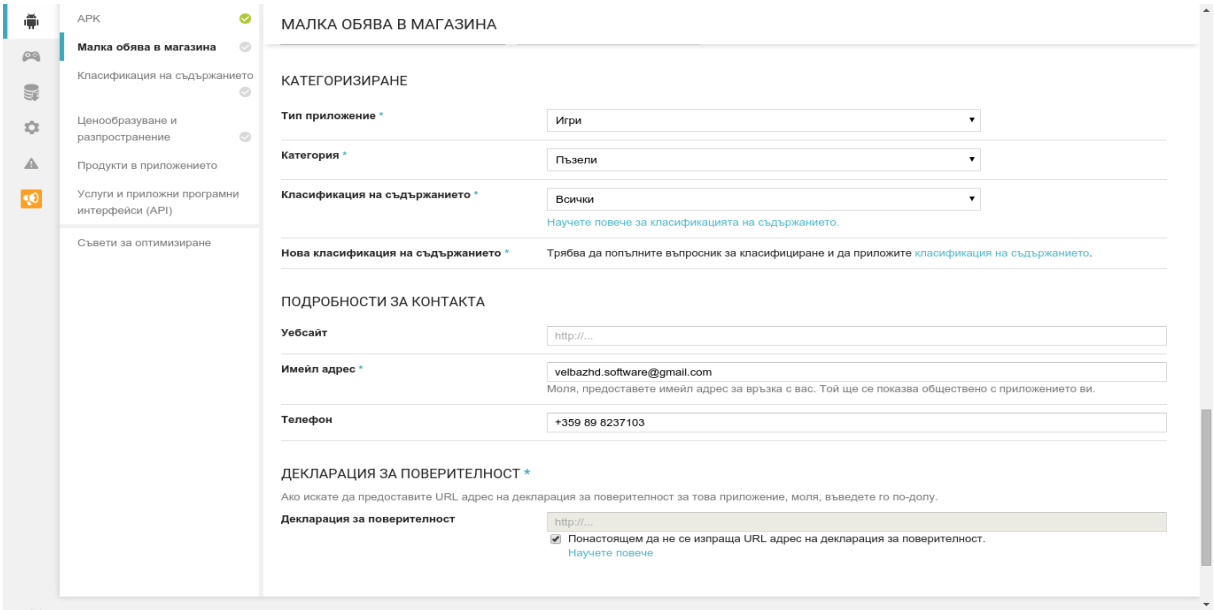
Фиг. 119 Информация за дигиталния ключ

Добрата практика за поддръжка на Android приложения е всяко приложение да бъде подписано с отделен дигитален ключ.



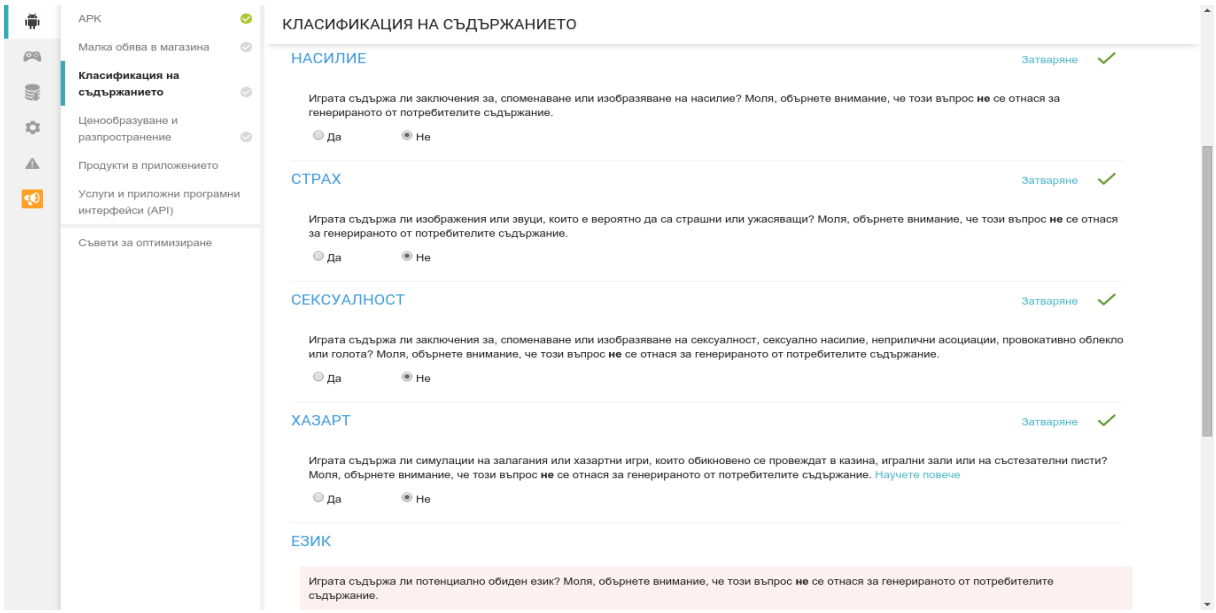
Фиг. 120 Дигитално подписаният изпълним файл, след публикуване

Google Play Store публикува единствено файлове, които са дигитално подписани от създателя им. Това се налага от съображения за сигурност и да бъде напълно известно кой точно е отговорен за съдържанието в изпълнимия файл.



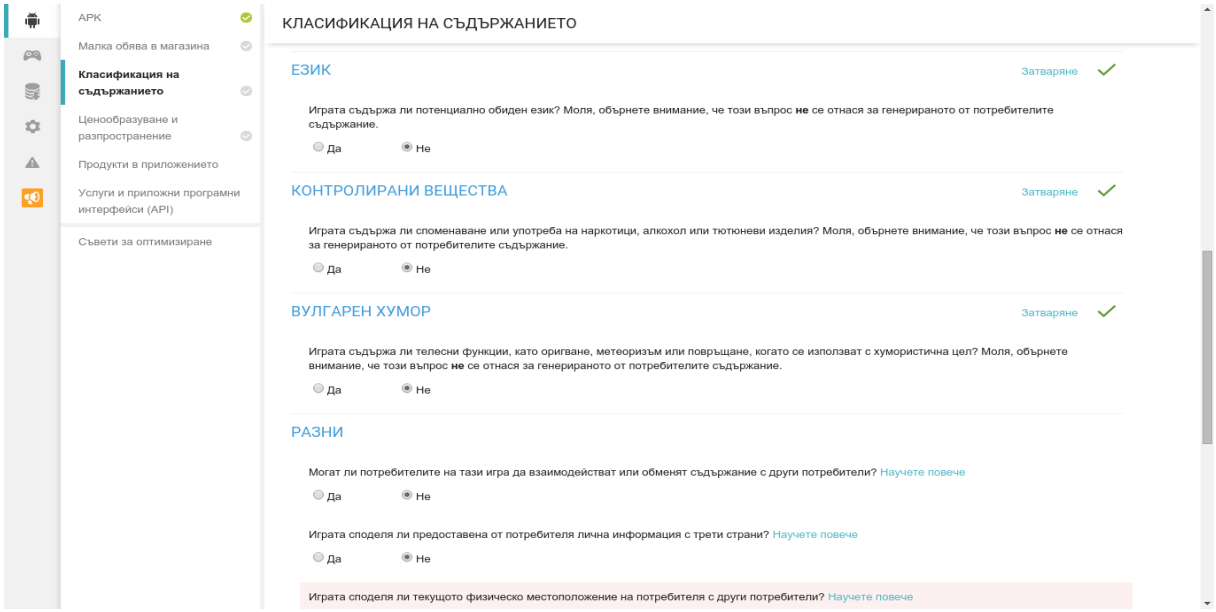
Фиг. 121 Категоризиране на приложението

Освен име и описание, приложението трябва да бъде определено по категория и зрялост на аудиторията, която има право да го използва.



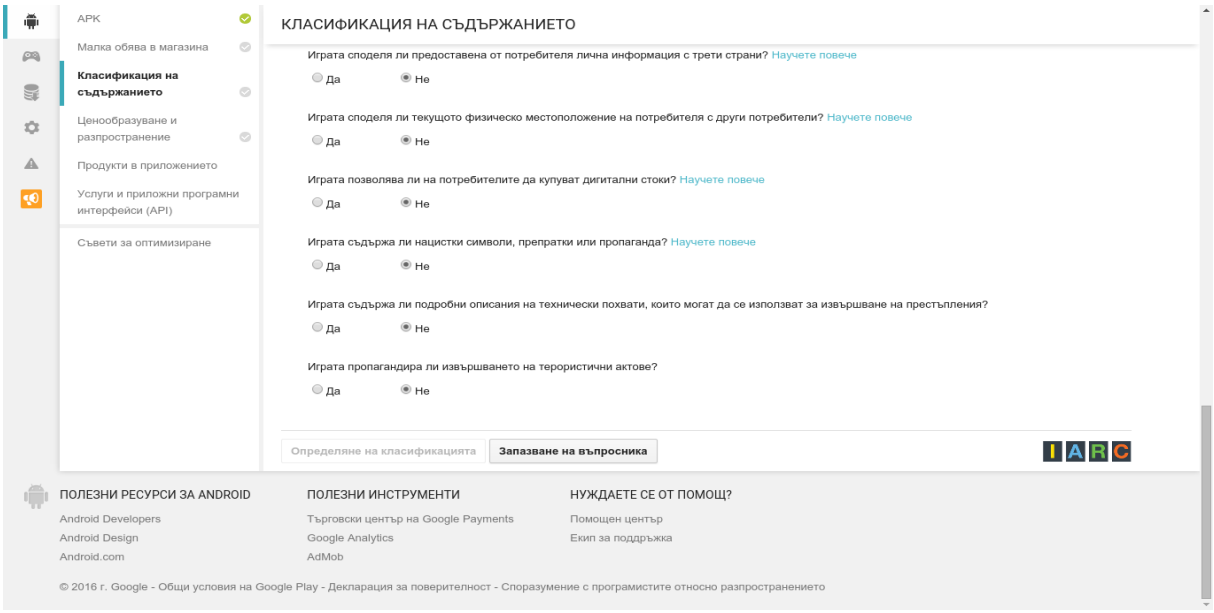
Фиг. 122 Компоненти за зрялост на аудиторията – насилие, сексуалност

Различните компоненти за зрялост на аудиторията са по отношение на насилие, сексуално съдържание, вулгарен език и т.н.



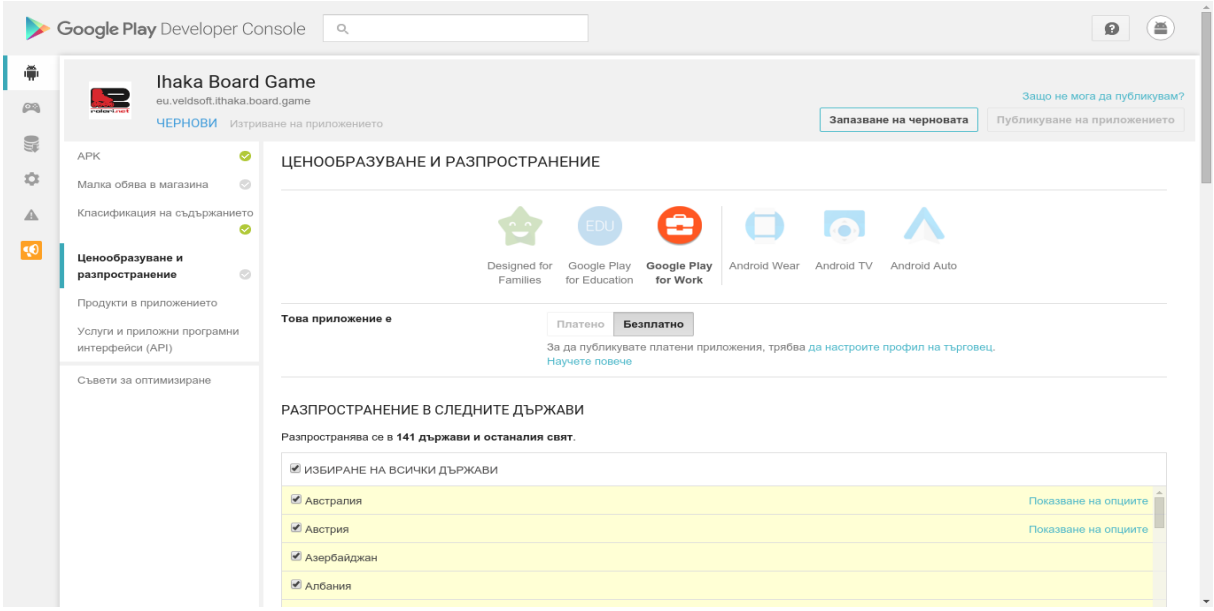
Фиг. 123 Компоненти за зрялост на аудиторията – вулгарен език или хумор

Детайлността на категоризирането засяга дори актуални теми, като тероризма.



Фиг. 124 Компоненти за зрялост на аудиторията – локализиране и проследяване

Не всяко приложение е подходящо за разпространение в цял свят, за това има отделна секция, в която се определя цената и териториите, на които приложението може да се използва.

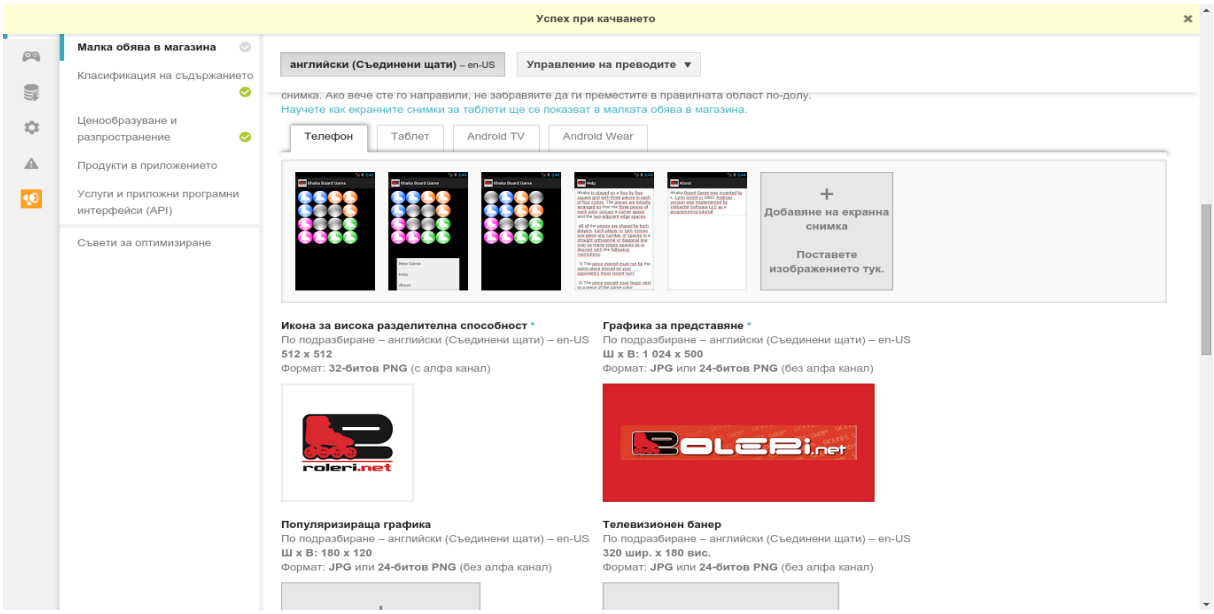


Фиг. 125 Цена и територия на разпространение

За територията на САЩ е необходимо допълнително деклариране на обстоятелствата около публикуването на приложението.

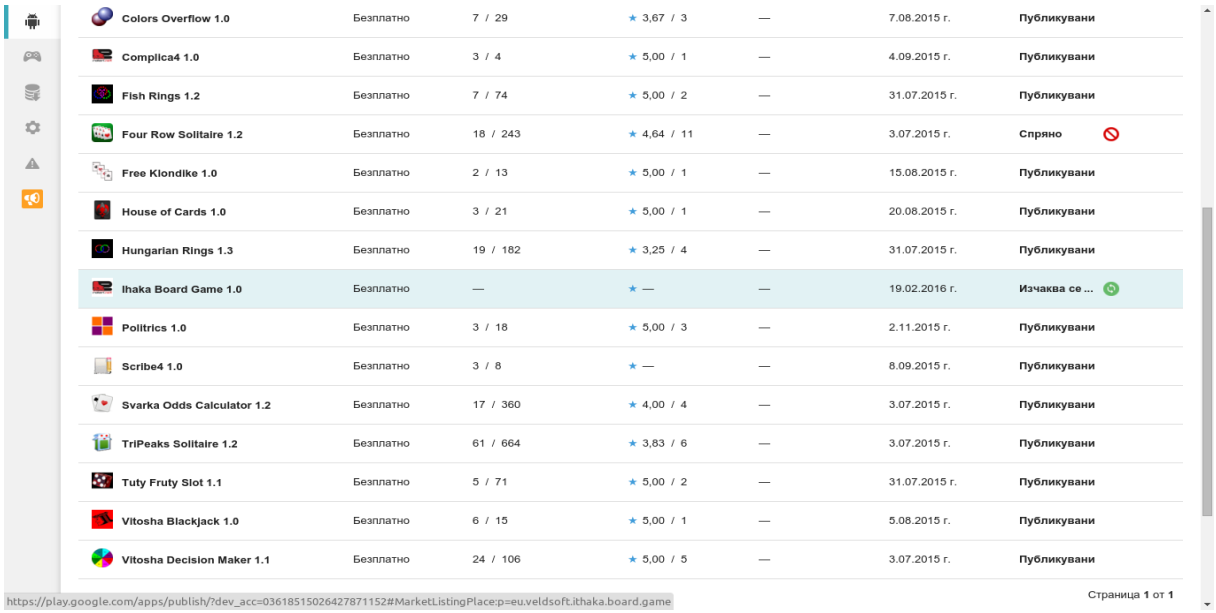
Фиг. 126 Съгласуване с експортните закони на САЩ

За да се увеличат максимално шансовете, потребителите да изберат точно това приложение, когато търсят какво да използват, е необходимо да се направи добро графично представяне.



Фиг. 127 Рекламни банери и работни екрани

След приключване на процеса, на публикуване са нужни няколко дни приложението да бъде одобрено от рецензентите в Google Play Store.



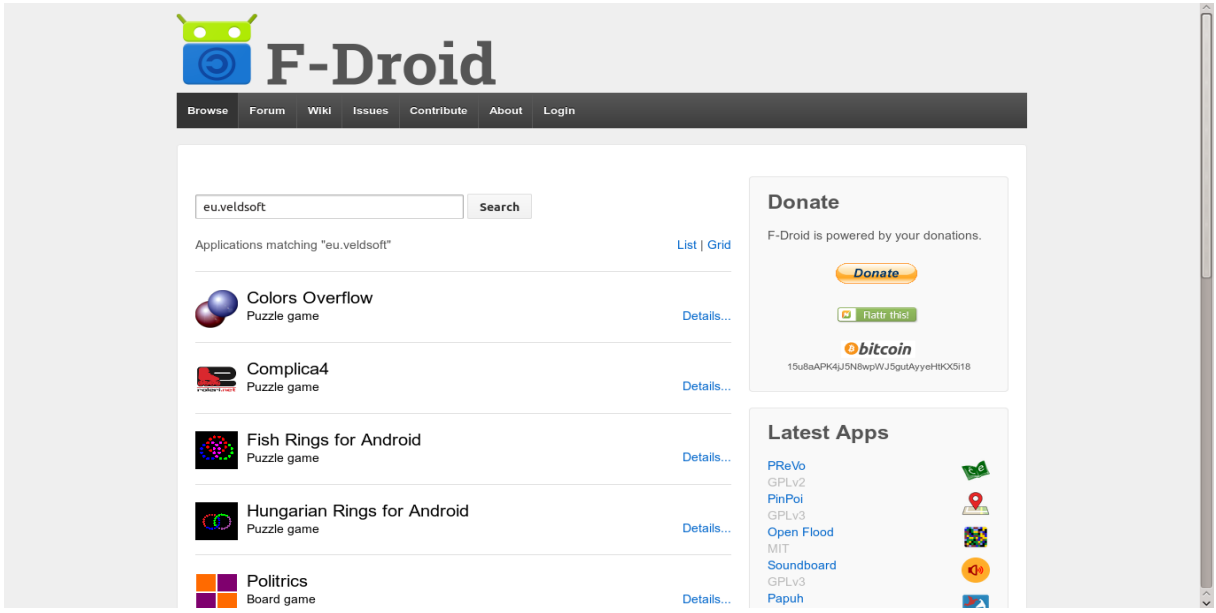
Colors Overflow 1.0	Безплатно	7 / 29	★ 3,67 / 3	—	7.08.2015 г.	Публикувани
Complica4 1.0	Безплатно	3 / 4	★ 5,00 / 1	—	4.09.2015 г.	Публикувани
Fish Rings 1.2	Безплатно	7 / 74	★ 5,00 / 2	—	31.07.2015 г.	Публикувани
Four Row Solitaire 1.2	Безплатно	18 / 243	★ 4,64 / 11	—	3.07.2015 г.	Спряно
Free Klondike 1.0	Безплатно	2 / 13	★ 5,00 / 1	—	15.08.2015 г.	Публикувани
House of Cards 1.0	Безплатно	3 / 21	★ 5,00 / 1	—	20.08.2015 г.	Публикувани
Hungarian Rings 1.3	Безплатно	19 / 182	★ 3,25 / 4	—	31.07.2015 г.	Публикувани
Ihaka Board Game 1.0	Безплатно	—	★ —	—	19.02.2016 г.	Изчаква се ...
Politrics 1.0	Безплатно	3 / 18	★ 5,00 / 3	—	2.11.2015 г.	Публикувани
Scribe4 1.0	Безплатно	3 / 8	★ —	—	8.09.2015 г.	Публикувани
Svarka Odds Calculator 1.2	Безплатно	17 / 360	★ 4,00 / 4	—	3.07.2015 г.	Публикувани
TriPeaks Solitaire 1.2	Безплатно	61 / 664	★ 3,83 / 6	—	3.07.2015 г.	Публикувани
Tuty Fruty Slot 1.1	Безплатно	5 / 71	★ 5,00 / 2	—	31.07.2015 г.	Публикувани
Vitosha Blackjack 1.0	Безплатно	6 / 15	★ 5,00 / 1	—	5.08.2015 г.	Публикувани
Vitosha Decision Maker 1.1	Безплатно	24 / 106	★ 5,00 / 5	—	3.07.2015 г.	Публикувани

https://play.google.com/apps/publish/?dev_acc=03618515026427871152#MarketListingPlace:p=eu.veldsoft.ithaka.board.game

Страница 1 от 1

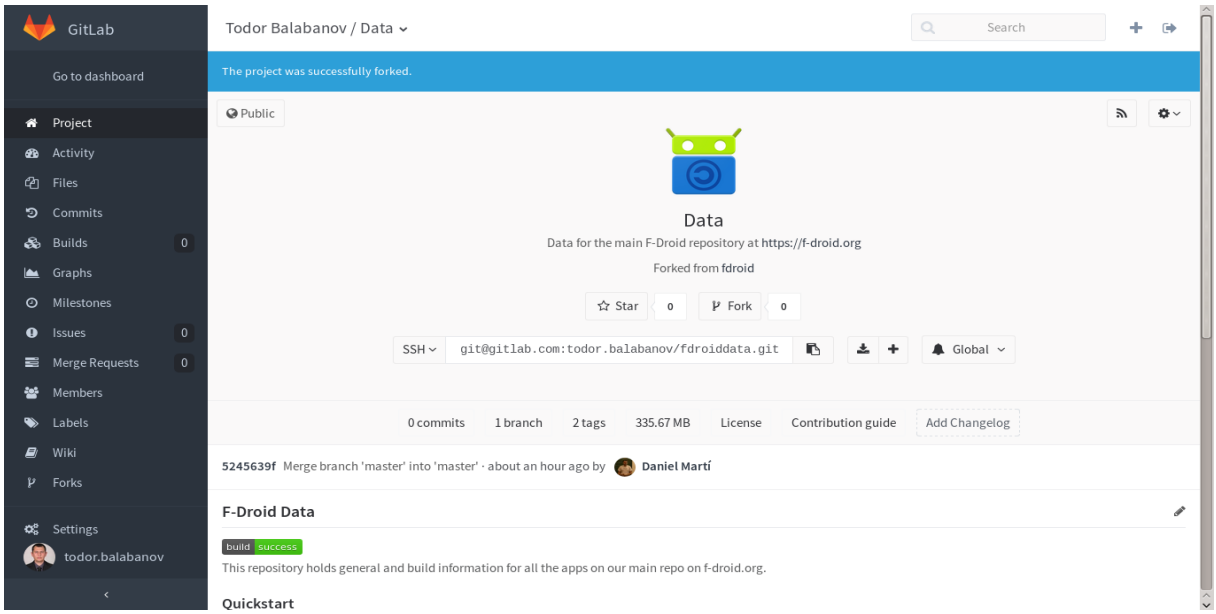
Фиг. 128 Изчакване на одобрение за публикуване

Втората възможност за публикуване е в най-голямото хранилище за Android продукти с отворен код, наречено F-Droid.



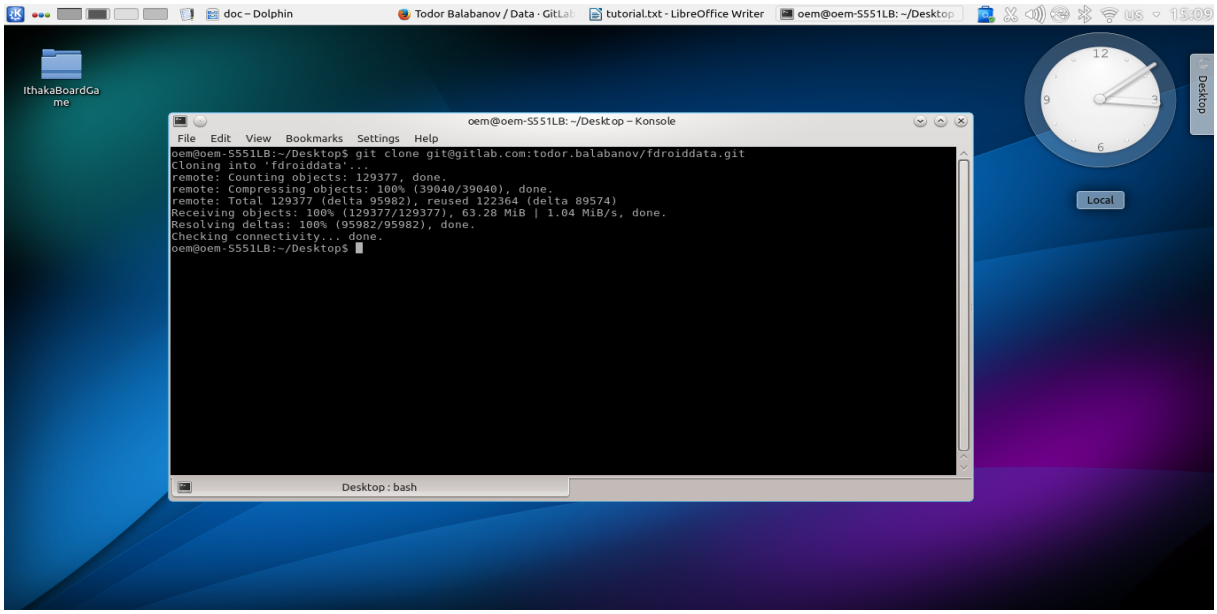
Фиг. 129 F-Droid.

Процесът за публикуване в F-Droid е коренно различен, спрямо този в Google Play Store. Основната разлика се състои в това, че продуктът трябва да е с отворен код и да е публично достъпен в хранилище като GitHub. Това изискване е задължително, тъй като в F-Droid не се публикува изпълним файл. Системата на F-Droid автоматично изтегля програмния код и ресурсите, извършва компилацията и изготвя изпълним файл, който е дигитално подписан с ключове, управлявани от инфраструктурата на F-Droid платформата.



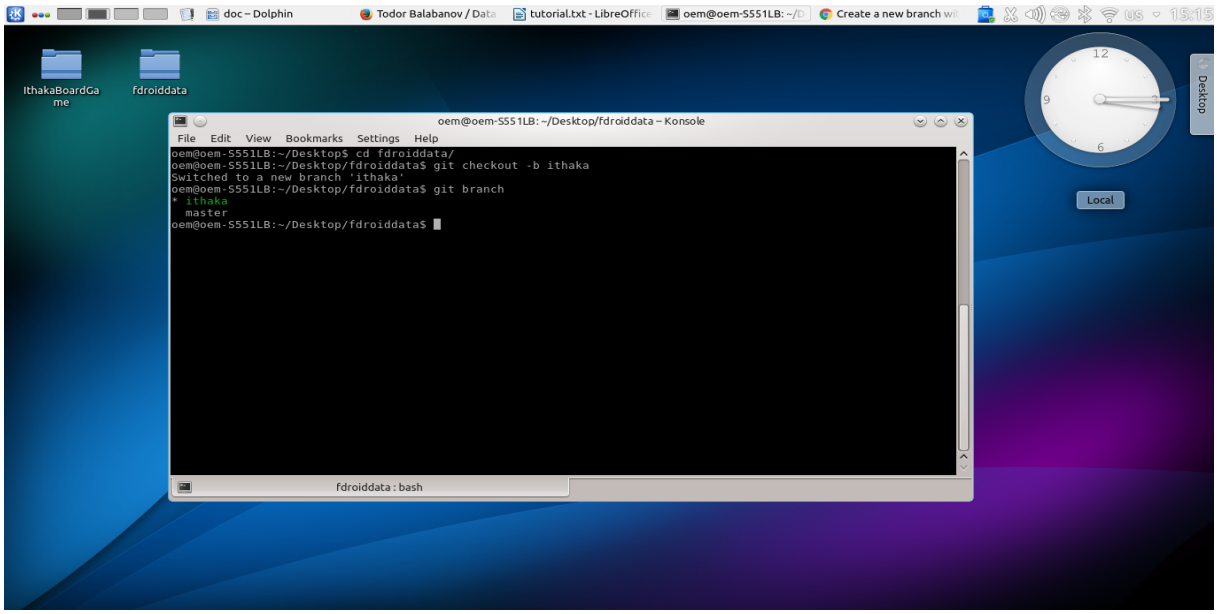
Фиг. 130 Клониране на проекта с описателните данни на F-Droid

Процесът за публикуване във F-Droid започва с клониране на проекта, съдържащ описателните данни на платформата F-Droid. По настояще този проект се съхранява в GitLab хранилище.



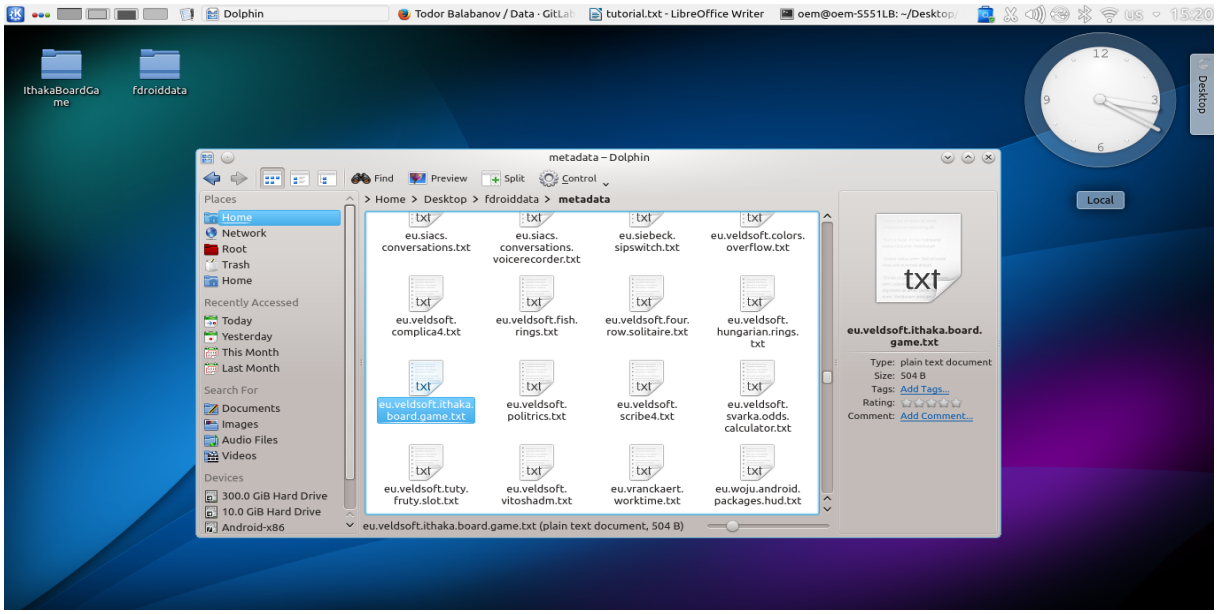
Фиг. 131 Локален клонинг на проекта с описателните данни за F-Droid

От разклонението хранилище в GitLab се изтегля локално копие.



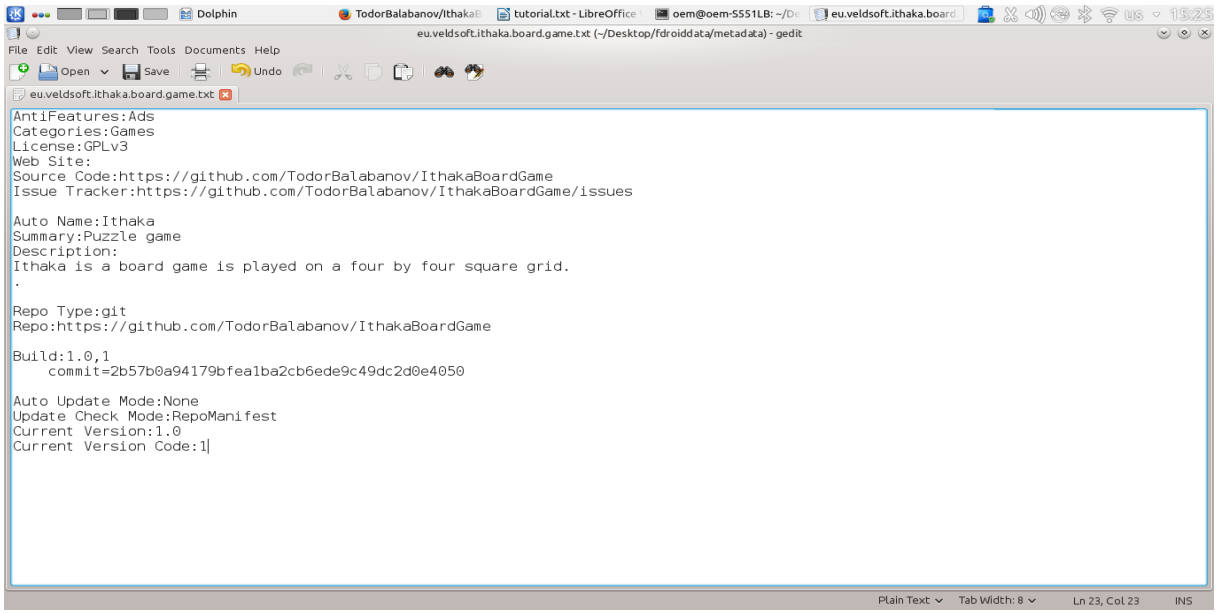
Фиг. 132 Допълнителен клон в локалното копие на проекта

За да протече гладко синхронизирането с основния проект в GitLab, в локалното копие се създава специално отделен клон, където ще бъде поместен описателният файл на приложението.



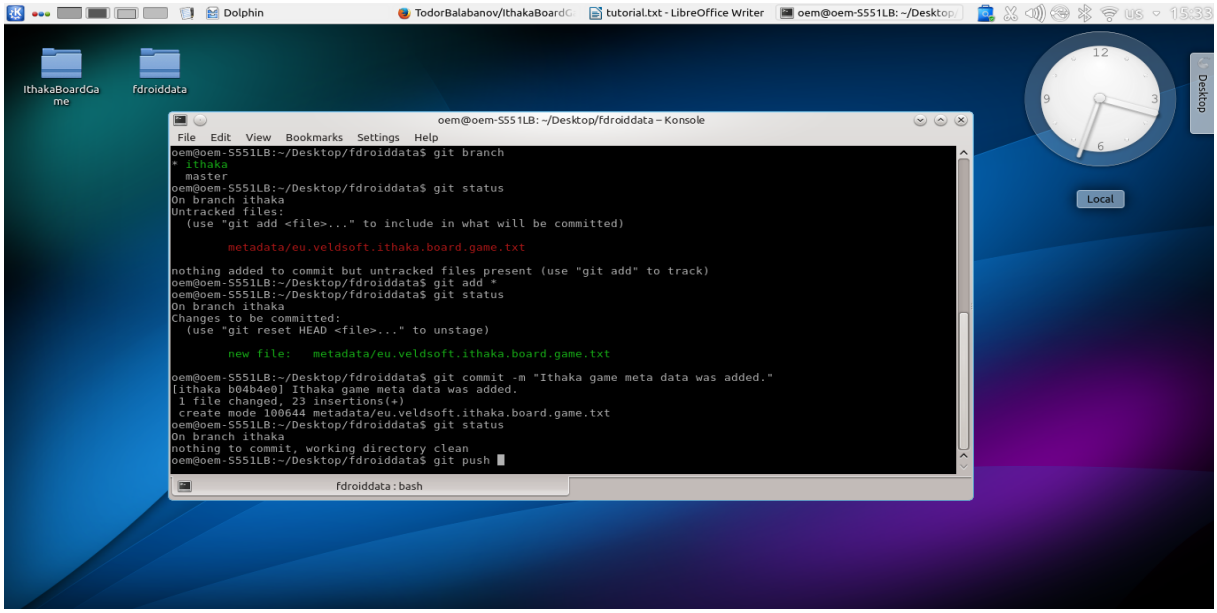
Фиг. 133 Описателен файл на приложението.

За да се извърши публикуването, е необходимо да се създаде описателен текстов файл в папката metadata.



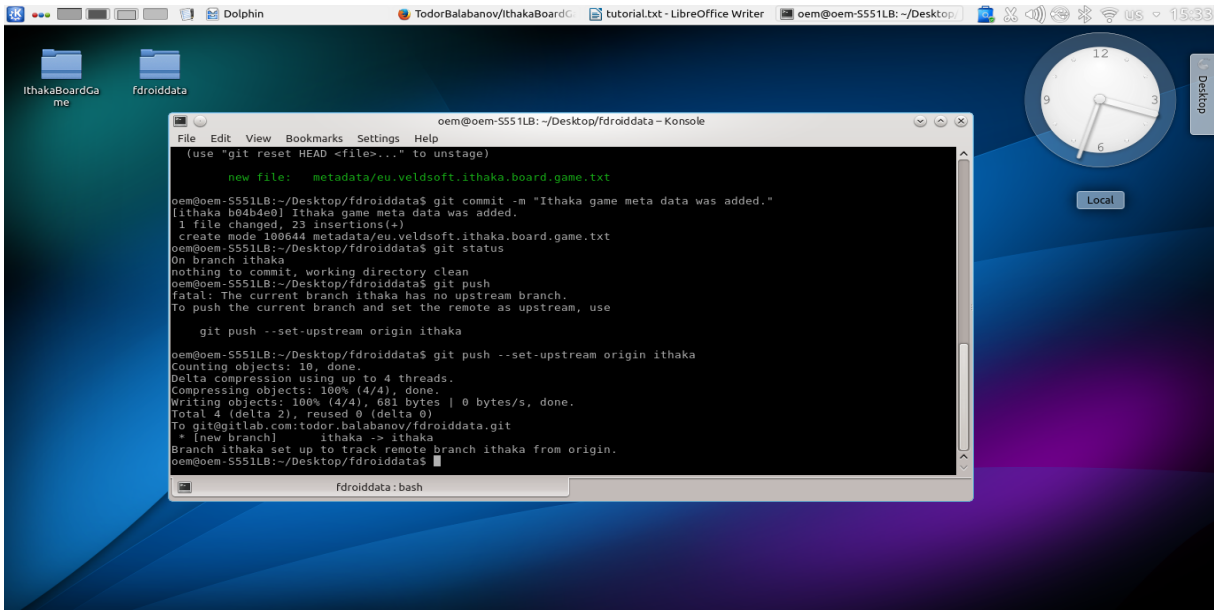
Фиг. 134 Описание на приложението

В описателния файл най-важни са следните атрибути – кратко описание на приложението, адрес на хранилището, от което да бъде изтеглен кодът, категория на приложението, дали има негативни за потребителя елементи (в случая има реклами), код на версия и най-важното хеш кодът на публикацията в хранилището, която да бъде използвана за изграждане на изпълнимия файл.



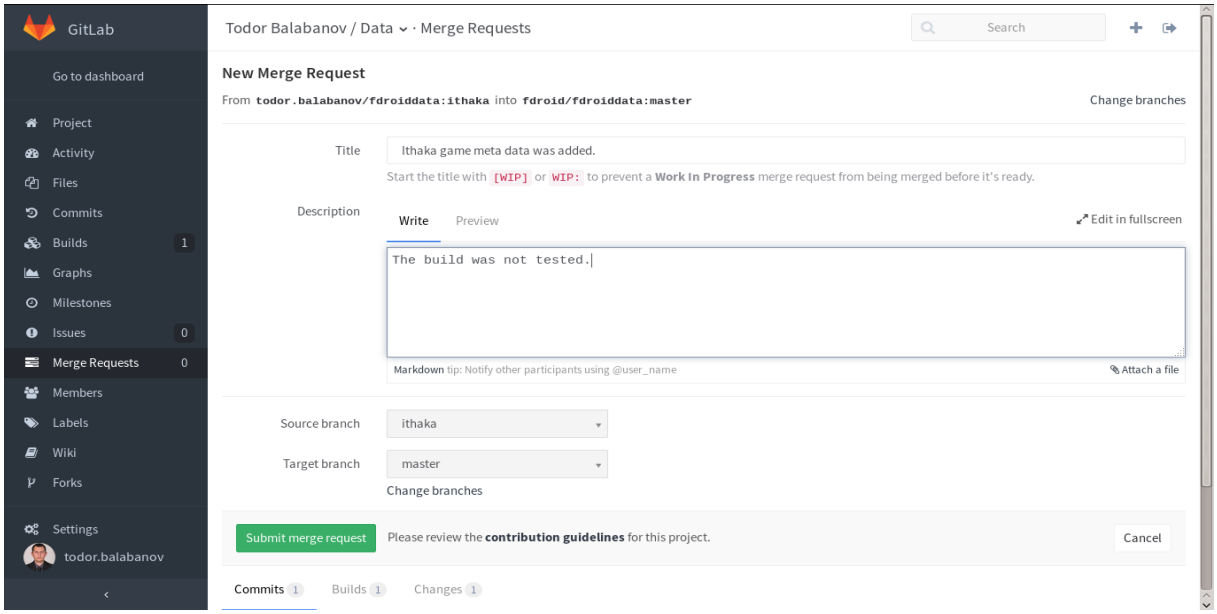
Фиг. 135 Публикуване на описателния файл в локалното хранилище

След като бъде изготвен описателният файл, то той трябва да бъде публикуван в хранилището на GitLab.



Фиг. 136 Публикуване от локалното копие към глобалното хранилище на GitLab

Git е двустепенна система за контрол на версиите. Първоначално кодът се синхронизира в локално хранилище, а на втора фаза кодът се синхронизира с глобалното хранилище.



Фиг. 137 Създаване на merge request

Процесът на публикуване завършва със създаването на заявка за сливане на описателния файл към основния проект с описателни данни за платформата F-Droid.

10. Заключение

Разработването на софтуерни приложения за мобилни устройства е една от най-динамично развиващите се сфери на съвременния софтуер. Популярността и все по-снижаващата се цена на мобилните технологии правят този пазар все по-голям и все по-перспективен. В същото време, развитието на развойните инструменти дава възможност дори на хора с малък опит да се включат в глобалната икономика на софтуерните продукти. В настоящото изложение, макар и ограничено по обем, бе демонстрирано какви са стъпките за преминаване от идея за софтуерен продукт до реално изработено и публикувано в Глобалната мрежа приложение. Макар и с минимална функционалност, представеното приложение е завършен продукт, кодът на който е публично достъпен и може да бъде доразвиван от всеки имащ желание да се включи в процеса. Двете основни неща, които значително биха подобрили предложеното софтуерно решение са модул за изкуствен интелект и възможности за мрежова игра.

Литература

1. Goodwill J., Matlock W., Beginning Swift Games Development for iOS, Apress, 2015, ISBN 978-1-4842-0400-9
2. DiMarzio J. F., Practical Android 4 Games Development, Apress, 2011, ISBN 978-1-4302-4030-3
3. Jackson W., Beginning Java 8 Game Development, Apress, 2014, ISBN 978-1-4842-0415-3
4. Astle D., Hawkins K., Beginning OpenGL Game Programming, Premier Press, 2004, ISBN 1-59200-369-9
5. Chin R., Beginning Android 3D Game Development, Apress, 2014, ISBN 978-1-4302-6548-1
6. Novak J., Game Development Essentials: An Introduction, Third Edition, Delmar Cengage Learning, 2012, ISBN 978-1-1113-0765-3
7. Bethke E., Game development and production, Wordware Publishing Inc, 2003, ISBN 1-55622-951-8
8. Шикаланов А., Балабанов Т., Многоезична интегрирана среда за разработка на приложения Eclipse, За буквите - О писменехь, 2014, ISBN 978-619-185-101-0

Настоящото учебно помагало съдържа основни примери по дисциплината „Увод в проектирането на електронни игри“, която е част от бакалавърската програма „Компютърни науки“ на Университета по библиотекознание и информационни технологии.

Помагалото е базирано на практически примери, покриващи цялостния цикъл за производството на електронни игри, който обхваща - идея, анализ на изискванията, проектиране, разработка, тестване и внедряване. Така подбраната структура на учебното помагало позволява отделни части от него да послужат при практическата реализация на курсови задачи или дипломни работи.

Свободният достъп до помагалото поражда нарастващ интерес за въвеждането на тази дисциплина в учебните планове и на други учебни заведения, както и евентуални възможности за формиране на цялостна магистърска програма, насочена към софтуерното производство на развлекателни мултимедийни продукти.

Използваните информационни източници имат не само справочен характер, но и представляват добра основа за значително разширяване на знанията и уменията в областта.

eISSN 2367-8666