КЛАСИЧНИЙ ПРИВАТНИЙ УНІВЕРСИТЕТ

ДО ЗАХИСТУ ДОПУЩЕНО

Зав. кафедри системного аналізу

та програмної інженерії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_І.М. Нацюк

(підпис) (ініціали, прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

КВАЛІФІКАЦІЙНА РОБОТА

РОЗРОБКА ГРИ «GAMESHOOT» НА МОВІ JAVA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Виконав  ст. групи | ЗI - 111 |  | О. С. Цаплюк | | |
|  | (шифр) |  | (підпис) |  | (ініціали, прізвище) |
| Керівник | ст. викл. |  |  |  | Р.Я Шумада |
|  | (посада) |  | (підпис) |  | (ініціали, прізвище) |
| Нормоконтролер | ст. викл. |  |  |  | А.С Шумада |
|  | (посада) |  | (підпис) |  | (ініціали, прізвище) |
| Рецензцнт |  |  |  |  |  |
|  | (посада) |  | (підпис) |  | (ініціали, прізвище) |

Запоріжжя

2016

КЛАСИЧНИЙ ПРИВАТНИЙ УНІВЕРСИТЕТ

Інститут управління

Кафедра системного аналізу та програмної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри системного аналізу

та програмної інженерії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_І.М.Нацюк

(підпис) (ініціали, прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

## ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту групи

ЗІ-111

(шифр)

Цаплюку Олександру Сергійовичу

(прізвище, ім’я та по батькові)

Галузь знань: 0501 – Інформатика та обчислювальна техніка

Спеціальність: 6.050103 – Програмна інженерія

Освітньо-кваліфікаційний рівень – бакалавр

Форма навчання – заочна

1. Тема дипломної роботи: Розробка гри «Gameshoot» на мові Java.

Затверджена наказом по університету № 34-зс від ”09” грудня 2015 р.

2. Термін здачі дипломної роботи ”12” травня 2016 р.

3. Цільова установка та загальний напрямок дипломної роботи: літературні дані за темою дослідження.

Метою даної дипломної роботи є створення гри розважального характеру.

Об’єкт дослідження: технологія розробки додатку на платформі Java.

4. Вихідні дані: літературні дані за темою дослідження.

5. Зміст дипломної роботи:

5.1. На основі аналізу літературних даних визначити основні властивості програмних продуктів та їх основні характеристики.

5.2. Розробити додаток.

5.3. Розробити інтерфейси користувача.

5.4. Перевірити працездатність створеної програми.

5.5. Сформулювати висновки.

6. Календарний план виконання роботи:

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назви частин роботи | Термін виконання | Позначка про виконання |
| 6.1. | Складання бібліографії та вивчення літературних джерел | 19.12.2015 |  |
| 6.2. | Збирання матеріалу. Написання вступу | 05.01.2016 |  |
| 6.3. | Виконання розділу 1 | 06.02.2016 |  |
| 6.4. | Виконання розділу 2 | 20.02.2016 |  |
| 6.5. | Виконання розділу 3 | 02.03.2016 |  |
| 6.6. | Виконання розділу 4 | 01.04.2016 |  |
| 6.7. | Формулювання висновків. Оформлення пояснювальної записки дипломної роботи | 25.04.2016 |  |
| 6.8. | Подання роботи до захисту | 19.05.2016 |  |

Дата видачі завдання ”19” грудня 2015 р.

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

РЕФЕРАТ

Звіт складається з 90 сторінок, 18 рисунків, 18 джерел, 5 таблиць.

Об’єкт дослідження — технологія розробки ігор на платформі Java

Предметом дослідження – розробка гри на мові Java «Gameshoot».

Задачі дослідження:

1. розглянути програмне забезпечення розважального характеру ;
2. визначити складову існуючих жанрів;

2. визначити необхідний функціонал та вимоги до програмного забезпечення розважального характеру;

3. розробити програмний продукт у вигляді розважальної гри «Gameshoot»,;

4. провести випробовування програмного продукту;

5. визначити економічну ефективність випуску програмного продукту.

При виконанні практичних завдань використовувались: операційна система Windows, мова програмування Java, середовище програмування Eclipse. Працює на комп'ютері з ОС сімейства Windows, на якій встановлена JAVA з версією не менше 1.7

ЗМІСТ

[ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 2](#_Toc452410674)

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8](#_Toc452410675)

[ВСТУП 9](#_Toc452410676)

[1. ОГЛЯД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗВАЖАЛЬНОГО ХАРАКТЕРУ 10](#_Toc452410677)

[1.1 Комп'ютерні ігри і навчання 10](#_Toc452410678)

[1.2 Основні ігрові жанри 12](#_Toc452410679)

[1.3 Етапи розробки комп'ютерних ігор. 15](#_Toc452410680)

[2. ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ СТВОРЕННЯ КОМП'ЮТЕРНИХ ІГОР 18](#_Toc452410681)

[2.1 Візуалізація 2D зображення за допомогою технології Java 18](#_Toc452410682)

[2.2 Java-платформа 20](#_Toc452410683)

[2.3 Віртуальна машина Java 23](#_Toc452410684)

[2.4 Середовище розробки Eclipse 24](#_Toc452410685)

[3. ЕТАПИ РОЗРОБКИ ГРИ 28](#_Toc452410686)

[3.1. Постановка завдання на розробку 28](#_Toc452410690)

[3.2. Сценарій гри 28](#_Toc452410691)

[3.3. Складові частини гри 30](#_Toc452410692)

[3.4. Розробка інтерфейсу програми 42](#_Toc452410693)

[3.5 Тестування гри 42](#_Toc452410694)

[4. РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД РОЗРОБКИ І ПРОДАЖУ ПРОГРАМИ 46](#_Toc452410695)

[4.1 Визначення витрат на розробку програми 46](#_Toc452410696)

[4.1 Розрахунок вартості програми 48](#_Toc452410697)

[4.2 Розрахунок економічного ефекту від продажу програми 48](#_Toc452410698)

[5 ОХОРОНА ПРАЦІ КОРИСТУВАЧІВ ПК 50](#_Toc452410699)

[5.1 Вимоги до виробничих приміщень для експлуатації ПК 50](#_Toc452410700)

[5.2 Мікроклімат виробничих приміщень 51](#_Toc452410701)

[5.3 Гігієнічні вимоги до організації та обладнання робочих місць 56](#_Toc452410702)

[5.4 Вимоги до режимів праці та відпочинку при роботі з ПК 57](#_Toc452410703)

[5.5 Вимоги до профілактичних медичних оглядів 59](#_Toc452410704)

[Висновок 61](#_Toc452410705)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 62](#_Toc452410706)

[ДОДАТОК А 64](#_Toc452410707)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IDE - integrated development environment;

SDK - software development kit;

API - application programming interface;

JRE - java runtime environment;

JDK - java development kit;

JVM – java virtual machine;

MMORPG – massively multiplayer online role-playing game;

MMOG – massively multiplayer online game;

MOBA – multiplayer online battle arena .

# ВСТУП

У першому десятилітті 21 століття ігрова індустрія йде вперед шаленим темпом. Щорічно випускається тисячі ігор, що розходяться по всьому світу, мільйонами копій, а оборот їх продажу складає сотні тисяч гривень. Даний процес відбувається у всіх на очах, та повинен бути осмислений у відповідних наукових категоріях, що дозволить більшою мірою зрозуміти специфіку та різноманітність актуальної культури.

Комп’ютерні віртуальні світи як і світи сьогодення володіють власною буденністю і казуальністю. Представлені різноманітними жанрами вони впевнено ввійшли в наше життя. Вони дають нам можливість відпочити від монотонної праці за комп’ютером, другі допомагають оволодіти навичками роботи з комп’ютером, інші розвиваються мислення реакцію, зорову пам'ять.

Актуальною темою є те, що більшість додатків які представлені сьогодні на ринку є досить громіздкими або досить вибагливі до технічних характеристик комп’ютерів, що не можуть задовольнити усіх користувачів, або вимагають від користувача володінню певних навичок.

Головними завданнями до даної роботи є розробка гри, яка би була невибагливою до технічних характеристик пристроїв (комп’ютерів), та не вимагала би в від користувача якихось певних навиків, та не мусила би приділяти досить багато уваги до її зрозуміння. Для виконання поставленого завдання необхідно розглянути жанри ігрової тематики, розібрати технічні засоби для розробки ігор; створити алгоритм роботи гри та розробити сюжетну складову гри, а також в завершальному етапі провести випробування розробленої гри.

# ОГЛЯД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗВАЖАЛЬНОГО ХАРАКТЕРУ

## Комп'ютерні ігри і навчання

Цифрові навчальні ігри - це предмет дослідження педагогіки комп'ютерної гри, яка в свою чергу є частиною медіапедагогіки (або медіаосвіти ). Що відноситься до навчальної грі педагогіка комп'ютерних ігор [1] розглядає три основні завдання:

- вона досліджує і описує роль цифрових навчальних ігор у педагогічній практиці (в школах, під час соціальної роботи, в будинках для людей похилого віку і т.д.). Комп'ютерні ігри поза педагогічного контексту (що відносяться до педагогіки формах, поширенні, діях) і тематизация цифрових навчальних ігор в педагогічному, але не науковому дискурсі (наприклад, в медіа , в професійної медійної літературі, проектах навчальних занять, а також в довідниках по дизайну цифрових навчальних ігор);

- вона критично ставиться до комп'ютерних ігор в педагогічній практиці і педагогічні, ненаукові висловлювання про комп'ютерні ігри. При цьому вона спирається на описові результати педагогічного дослідження комп'ютерних ігор і на форми аргументації з педагогічної етики, наприклад, філософії освіти;

- вона займається методологічним і технічним дослідженням (комп'ютерних ігор) з метою дати науково-обгрунтовані вказівки для створення цифрових навчальних ігор і їх виховно-педагогічного використання (розвитку і оцінки). Методологічне дослідження використовує результати обох названих дослідницьких завдань (наприклад, опис дидактично-методологічного дизайну розважальних комп'ютерних ігор). Наприклад, щоб розробити наукові гіпотези.

Ці три дослідницькі завдання хоч і пов'язані один з одним, але йдуть різної дослідницькою логікою і розрізняються з системної науково-теоретичної позиції. Важливими суміжними науками комп'ютерної педагогіки є комплекс наукових дисциплін про засоби комунікації  та дослідження відеоігор .

Відеоігри можуть бути також дуже корисні в тому сенсі, що вони створюють ефект симуляції дії, але при цьому не несуть будь-якої очевидної небезпеки. Наприклад, коли повітряні війська вчать своїх пілотів літати на літаку, що стоїть мільйони доларів, природно, вони не відправляють пілотів відразу ж на злітну смугу.

Військові повітряні сили використовують симулятор водіння літака, щоб підготувати пілотів до управління цією машиною. Ці віртуальні симулятори призначені для тренувань і підготовки до роботи в реальному світі, при цьому вони попереджають отримання будь-якого шкоди або втрати життя в процесу навчання. Пілот може розбитися в симуляторі, зробити висновки зі своїх помилок і перезапустити програму. Цей процес веде в результаті до високого рівня майстерності в симуляторі, а далі і при управлінні справжнім літаком, на якому пілоти літатимуть в майбутньому. Військові також використовують гри франшизи. Ігри такого роду занурюють гравців в віртуальну реальність. Використовуючи тактичні навички, гравці намагаються досягти в грі будь-яких, поставлених перед ними цілей. Це дозволяє військовим показати своїм солдатам, як справлятися з конкретними ситуаціями, не ризикуючи при цьому отримати поранення на полі бою  .

Цифрові навчальні ігри відрізняються від традиційних навчальних ігор і не заснованого на іграх електронного навчаннятим, що вони використовують методи мотивації розважальних ігор, щоб досягти своїх освітніх цілей . Таким чином, вони досить часто використовують будь-яку історію і парасоціальние відносини між гравцем і неігрових персонажем (англ. Non-player character ) , щоб запустити процес навчання. На основі численних ознак, за якими навчання в групах відрізняється від індивідуального, можна також провести принципова відмінність між цифровими іграми, спрямованими на окремого учня, і тими, які націлені на навчання в групах.

## Основні ігрові жанри

В основі сучасних розподілів ігор на жанри [2] лежить вид активності, який найчастіше здійснює гравець в іграх даного жанру. Так ігри в загальному можуть поділятися на ігри руху, планування і сюжету або спілкування, дії та контролю. В багатьох класифікаціях визначення жанру відбувається за кількома осями. Наприклад, за двома осями сюжет - свобода дії, або трьома абстракція – симуляція - свобода. Проте найчастіше використовуваною класифікацією, хоч і не прийнятою усіма, жанри з якої зустрічаються в більшості існуючих, є наведена нижче, яка виключає осі або багаторівневі поділи:

Екшен. В іграх такого жанру необхідно використовувати рефлекси та швидкість реакції для подолання ігрових обставин. Це один із базових жанрів і водночас найпоширеніший. Як правило екшн-ігри пов'язані із агресивними діями щодо противників або оточення. Персонаж гравця повинен битися, стріляти, переслідувати ціль чи самому уникати переслідування. Стосовно екшн-ігор, де наявні значні елементи пригодницьких ігор, застосовується термін Action-adventure. З-поміж них часто виділяється напрям аркадних ігор, ігровий процес яких вирізняється простотою та легкістю освоєння.

Цей жанр поділяється на велику кількість піджанрів, серед яких основними є:

* Шутери - вимагають від гравця боротися з противниками шляхом стрілянини. Залежно від перспективи, поділяються на шутери від першої (Wolfenstein 3D) чи третьої особи (Макс Пейн). Існують різновиди як тактичні, в яких ігровий персонаж діє у складі команди (SWAT 4), аркади (AlienShooter), стелс-екшн, метою якого є приховані дії для виконання завдань, без прямого знищення противників (серія Hitman). Щодо ігор, де основою ігрового процесу є знищення великих кількостей ворогів, а сама стрілянина в цьому переважає над тактикою і влучністю, застосовується термін Shoot 'emup (R-Type, Touhou, Contra).
* Файтинги - імітують ближній бій, як правило один на один, на спеціальних аренах. Приклади: серія Mortal Kombat, Street Fighter.
* Beat'em up - подібні на файтинги, з тою різницею, що персонажі вільно переміщуються ігровим світом (а не спеціальною ареною) і борються проти багатьох противників одночасно. Приклади: Golden Axe, Battletoads, Double Dragon.
* Платформери - персонаж мусить рухатися, стрибаючи по платформах, та долати перешкоди. Приклади: Mario, Spyro theDragon, Megaman.
* Лабіринти - персонаж рухається лабіринтом з метою знайти вихід, зібрати предмети і/або уникнути пасток і небезпек. Часто в іграх цього жанру є обмеження на час. Приклади: Pac-Man, Boulder Dash.

Стратегія. Сенс стратегічних ігор полягає в плануванні дій та виробленні певної стратегії для досягнення якоїсь конкретної мети, наприклад, перемоги у військовій операції. Гравець керує не одним персонажем, а цілим підрозділом, підприємством чи навіть всесвітом. Відповідно до реалізації ігрового часу, стратегічні відеоігри поділяються на два основних різновиди:

* Покрокові стратегічні ігри - гравець та його противник здійснюють дії один за одним, покроково, маючи змогу за один ігровий хід виконати певну кількість операцій. Приклади: Heroes ofMightandMagic III, Цивілізація.
* Стратегічні ігри в реальному часу - і гравець і противник виконують свої дії одночасно, проте часто часто масштаб часу відрізняється від реального. Наприклад, будівництво триває кілька секунд, а ігрова година складає кілька хвилин реального часу. Приклади: StarCraft, Age of Empires III.
* Tower Defense - похідний жанр, в якому гравець керує оборонними баштами аби не пропустити хвилі ворогів.
* MOBA - похідний жанр, в якому гравець керує одним персонажем з метою захистити свою базу від ворогів та знищити ворожу.
* MMORTS - багатокористувацькі он-лайн стратегії в реальному часі, орієнтовані на суперництво/співпрацю з іншими реальними гравцями, щоб досягнути спільної мети.

Видом стратегічних ігор, які відображають суто бойові дії, є варгейми. Ігри, в яких гравець управляє масштабними державами (імперіями, планетами, галактиками), при цьому займаючись всіма аспектами їх життя, включаючи торгівлю, науки та війни, мають назву глобальних стратегій. Близьким до них є ігри в бога, де гравець виступає в ролі надприродної істоти, управляє світом через чудеса, правителів, керує силами природи.

Рольова гра. Гравець асоціюється з конкретним персонажем або лідером команди, які діють відповідно до правил своїх ролей. Наприклад, лицар не може того, що чарівник, кожна роль має свої особливості, а часом від неї залежить і розвиток сюжету. Мета ігрового процесу полягає у виконанні різноманітних завдань (квестів) для розвитку одного персонажа або групи. Можливі варіанти дій залежать від обраного образу персонажа, попередньо визначеного чи формованого самим гравцем.

Рольові відеоігри часто поєднуються з іншими жанрами, утворюючи Action-RPG, тактичні рольові ігри, MUD-и(текстова онлайн-гра) і т.п.. Особливим випадком рольової гри є MMORPG, багатокористувацькі рольові онлайн-ігри, в яких живі гравці взаємодіють одне з одним у віртуальному світі через мережу Інтернет, що визначає специфіку ігрового процесу.

Іноді рольові відеоігри поділяються за дизайном і побудовою сюжету. Так існує умовний поділ на рольові ігри західного зразка та східного (японські рольові ігри).

Симулятор. В широкому розумінні всі ігри є симуляторами. У вужчому значенні це відеоігри, призначені для складання уявлення про дійсність за допомогою відображення певних реальних явищ та властивостей у віртуальному середовищі. Існує чимало піджанрів, як технічні (управління складними технічними пристроями, авіаційною технікою та інші, наприклад гра Іл-2 Штурмовик), аркадні (відрізняються від аркад наявністю спрощеної фізичної моделі. Наприклад, X-Wing), спортивні, економічні, побачень та інші.

Пригоди. В пригодницьких іграх гравець керує ігровим персонажем, який рухається по сюжету та виконує зумовлені сценарієм завдання, покладаючись на свою уважність та логіку, здійснює пошуки підказок і вирішує загадки. Всередині жанру виділяються основні піджанри: інтерактивна література, інтерактивні фільми  та візуальні романи. Часто за аналогією до пригодницьких фільмів пригодницькими називаються ті відеоігри, сюжет яких динамічно розгортається, насичений яскравими подіями, швидкою зміною обстановки, а персонажі проявляють кмітливість та сміливість. Приклади: серія про Індіану Джонса,  Disney's Aladdin Nasira's Revenge, Syberia.

## Етапи розробки комп'ютерних ігор.

Розглянемо етапи розробки типової комп'ютерної гри:

* підготовка до виробництву-це перший етап роботи над грою. Завдання розробників на цьому етапі - розробити концепцію гри, дизайн персонажів, вибрати засоби для реалізації проекту, створити прототип гри, підготувати план, за яким буде створюватися гра, і узгодити цей план з начальством, або - з компанією, яка планує видавати гру. Як правило, всі сучасні ігри пишуться під конкретного видавця, який часто вкладає в розробку чималі кошти. Коли всі адміністративні питання вирішені, гра набуває на етап виробництва;
* виробництво - це ключовий етап у створенні гри. Розробники займаються реалізацією раніше створеного плану. Однак початковий план гри піддається змінам - іноді ці зміни відбуваються дуже часто - аж до щоденних коригувань.

В ході виробництва гри - особливо це стосується комерційних версій - періодично влаштовується розгляд поточних результатів розробки, до яких команда повинна представити проект, який досяг певного рівня розвитку. Тобто, наприклад, до одного з таких моментів повинна бути готова працююча демо-версія гри, до іншому - перший рівень і так далі. Як правило, ці проміжні результати служать відмінною рекламою нових ігрових проектів - демо-версії публікують на ігрових сайтах, геймери «приміряють» до цих версіями можливості свого обладнання;

* випуск. Після того, як гра створена, протестована і налагоджена, настає час її випуску. Як правило, інтерес до цієї події посилено підігрівається видавцем гри - адже не варто забувати, що головна мета видавця - прибуток. Як правило, найбільш успішні ігри з лишком виправдовують очікування видавців;
* підтримка -Ігри для ПК часто виходять з помилками - вся справа в тому, що розробникам вічно не вистачає часу, щоб все як слід налагодити. Благо, є можливість виправляти помилки на вже встановлених іграх, встановлюючи патчі (від англійського patch - заплатка). Цим користуються розробники, випускаючи сируватий гру і, після цього, цілу низку латочок для неї. Така практика не поширена для консольних ігор - тут розробники змушені відповідальніше підходити до своєї роботі і випускати повністю робочу гру, яка не потребує втручань.

Звичайно, вище приведена лише приблизна схема роботи над грою, однак практично всі ігри проходять через однакові етапи.

До ігор такого типу пред‘являються такі вимоги [7]:

1. Використання простих засобів управління[11] :

* користувач має легко знаходити необхідні елементи керування;
* кількість елементів керування, яка необхідна для успішної роботи з програмою, не повинна перевищувати 7 одиниць;
* основні елементи керування повинні бути доступні в будь- якому стані програми;
* користувач має легко знаходити необхідні елементи керування;
* має існувати можливість керування програмою лише за допомогою клавіатури, не користуючись мишею.

1. Зручний графічний інтерфейс:

* інтерфейс програми має бути простим та зрозумілим, якщо користувач не прочитав інструкцію до програми;
* елементи керування повинні бути доступними на моніторах будь-якого розміру;
* користувач повинен мати можливість зупинити поточну операцію або можливість вимкнути програму;
* звуки повинні бути простими, не відволікаючими.

1. Поступове ускладнення гри при наборі певної кількості очок:

* старт програми має тривати не більше 10 секунд;
* перехід між рівнями має тривати не більше 10 секунд;
* наявність бонусів, що можуть спростити гру.

1. Під час виконання програма повинна виконуватися коректно і не приводити до збоїв.

# ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ СТВОРЕННЯ КОМП'ЮТЕРНИХ ІГОР

## Візуалізація 2D зображення за допомогою технології Java

Методи комп'ютерної графіки знаходять широке застосування в сучасному проектуванні високотехнологічних промислових виробів. Двовимірна графіка, в свою чергу ділиться на растрову і векторну. Двовимірна графіка (2D) використовується більшою мірою, зокрема при роботі з фото-банками, фотографіями, ілюстраціями та інколи з флеш - кінцевий результат рендера виводиться у форматі JPG, а не як вихідний файл. У кожного виду є свої гідність і недоліки при роботі з фото-банками.

Растрова графіка - це класичний спосіб представлення комп'ютерної графіки, за своєю суттю - це масив різнокольорових пікселів, кожен з яких вимагає зберігання для нього кольору і координат (прообразом растрової графіки можна вважати вишивку хрестом, де зображення складається з кольорових хрестиків). Векторна графіка - це спосіб представлення зображення за допомогою математичних символів і примітивів. Растрова і векторна графіки в силу своїх відмінностей не є взаємозамінними, так як за допомогою растрової графіки найзручніше представляти фото-реалістичні зображення, наприклад, фотографії, а за допомогою векторної - малюнки та схеми. З одного боку - безперечною перевагою векторної графіки є можливість масштабування без втрати якості і невелика вага файлу з простими об'єктами, а з іншого - реалістичне зображення з дрібними деталями буде неймовірно великим за рахунок складності фігур, за допомогою яких вона створена.

Середовище розробки програмного забезпечення, в якому найбільш поширені блоки програмного коду представляють у вигляді графічних об'єктів і застосовуються вони в основному для створення прикладних програм і розробки графічного інтерфейсу користувача (GUI). Слід враховувати, що деякі візуальні середовища розробки мають власний формат зберігання проекту і при переході в інше середовище може виникнути неперенесення властивостей проекту і деяких частин його, таких, як власні бібліотеки використовуваного середовища розробки. Середовищами розробки являються [3] : Eclipse, IntelliJ IDEA, NetBeans , MyEclipse, Oracle WebLogic Workshop, IBM WebSphere Studio, BlueJ, DrJava, Greenfoot, JCreator, JDeveloper, JBuilder , JGRASP.

Останні кілька років розробники докладали масу зусиль, щоб інтегрувати графіку і анімацію в свої аплети і додатки Java. Однак спочатку включені в Java графічні пакети AWT Java мали обмежені кошти для вирішення таких завдань. Зараз, використовуючи інтерфейси прикладного програмування Java 2D і Java 3D, розробники можуть реалізовувати набагато більш складні графічні додатки, включаючи ігри, зберігаючи екран, екранні заставки та тривимірний графічний користувальницький інтерфейс.

У технології Java графіка ускладнюється тим, що додатки Java повинні працювати з потрібною бібліотекою класів, незалежно від конкретної графічної системи. Класи бібліотеки AWT реалізують ці інтерфейси для створення додатків. Програми Java використовують ці методи для розміщення і переміщення графічних об'єктів, зміни їх розмірів, взаємодії об'єктів. Бібліотека класів Java, заснована на peer-інтерфейсах, отримала назву AWT (Abstract Window Toolkit). При виведенні об'єкта, створеного в додатку Java і заснованого на peer-інтерфейсі, на екран створюється парний йому (peer-to-peer) об'єкт графічної підсистеми операційної системи, який і відображається на екрані. Бібліотека "легких" компонентів Java, названа Swing, може використовуватися самостійно, незважаючи на те, що всі класи з неї розширюють класи бібліотеки AWT. Всі ці засоби Java 2: AWT, Swing, Java 2D, DnD, InputMethod Framework і Accessibility склали бібліотеку графічних засобів Java, названу JFC (Java Foundation Classes).

## Java-платформа

На теперішній час комп'ютерному світу відомо багато різних платформ, серед яких такі як: Microsoft Windows, Macintosh, OS/2, UNIX® і система Netware®. Для встановлення на кожну з них програмного забезпечення, воно повинне бути відкомпільоване окремо. Двійковий файл додатка, що виконується на одній платформі, не може бути запущений на іншій, тому як двійковий файл – специфічний, залежно від платформи.

Платформа Java – нова програмна платформа для транспортування й виконання високо інтерактивних, динамічних і безпечних аплетів і додатків на системах мережевих комп'ютерів. Основною якістю Java-платформи, що виділяє її серед інших, є те, що вона розташовується на самому верхньому рівні в інших платформах, що дозволяє їй робити компіляцію в байт-коди, не прив'язані до кожної з фізичних машин і представляють собою машинні інструкції для віртуальної машини (virtual machine). Програма, написана мовою Java, компілюється у файл байт-коду, що може працювати скрізь, де присутня Java-платформа, на кожній з основних операційних систем. Інакше кажучи, той самий файл буде виконуватися на будь-якій операційній системі, на якій присутня Java-платформа. Подібна мобільність стає можливою завдяки тому, що в основі Java-платформи лежить віртуальна Java-машина.

У той час як кожна платформа має у своєму розпорядженні свою власну реалізацію віртуальної Java-машини, всі віртуальні машини задовольняють вимогу єдиної специфікації. Завдяки цьому, платформа Java може реалізовувати єдиний стандарт – універсальний програмний інтерфейс для аплетів і додатків на будь-яких апаратних засобах. Тому платформа Java є ідеальною для Інтернету, де та сама програма повинна бути здатна до виконання на будь-якому комп'ютері у світі.

Розробники використовують мову Java при написанні вихідного коду для Java-додатків. Вони компілюють свій код один раз і позбуваються тим самим від необхідності компілювати його для кожної системи окремо. Вихідний текст мови Java компілюється в проміжну, переносну форму байт-коду, що запуститься скрізь, де є присутнім Java-Платформа.

Розробники можуть писати об’єктно-орієнтовані, багатопоточні, динамічно зв'язані додатки, використовуючи мову Java. Платформа має вбудовані системи захисту, обробки виняткових ситуацій і автоматичного «збору сміття». Крім того, існує можливість використовувати JIT (just-in-time) компілятори (компілятори «на льоту») і прискорити виконання програм за допомогою перетворення байт – кодів Java у машинну мову. Також розробники можуть записувати й викликати так звані нативні методи – методи C, C++ або інших мов, відкомпільовані для певної операційної системи - для підвищення швидкості виконання або для застосування спеціальних функціональних можливостей.

Програми, написані мовою Java і потім відкомпільовані, будуть запускатися на Java-Платформі. Платформа Java має дві основних частини:

* Java Virtual Machine (віртуальна Java-Машина);
* Java API (прикладний програмний інтерфейс Java).

У сукупності ці частини забезпечують оперативні засоби керування роботою програми для кінцевого користувача при установці інтернет-додатків.

The Java Base Platform – «мінімальна» Java платформа, створена для запуску Java-Аплетів і додатків, що розробники можуть без проблем встановити й використовувати. Дана платформа призначається для мережевих, настільних комп'ютерів і робочих станцій. Платформа містить у собі ту ж віртуальну машину, що описувалася вище й при цьому має мінімальний комплект API, необхідний для запуску основних аплетів і додатків. Згаданий мінімальний комплект відомий, як Java Applet API або Java Base API. Розробники, які пишуть для цього комплекту, можуть бути впевнені в тім, що програма запуститься скрізь без необхідності в підключенні додаткових бібліотек- класів.

Java API– визначає стандартний інтерфейс для аплетів і додатків, не звертаючи уваги на встановлену на комп'ютер операційну систему. Java API – основа, каркас при розробці додатка. Даний API визначає набір стандартних інтерфейсів для використання в ключових областях, кількість котрих збільшується, у яких програмісти звичайно вибудовують свої Java додатки.

Java Base API (Основний прикладний програмний інтерфейс Java) забезпечує можливість роботи з різними допоміжними класами, з уведенням / виведенням, з мережею, з GUI (графічний інтерфейс користувача), і аплетами. Компанії-виробники операційних систем, які мають ліцензію Java, підписали контракти, що зобов'язують їх включати JavaBase API у будь-яку Java-Платформу, що вони встановлюють.

Java Standard Extension API (стандартний розширений прикладний програмний інтерфейс Java)-є розширенням описаного вище Java Base API. Передбачається, що деякі розширення будуть згодом мігровані в Java Base API. Інші нестандартні API будуть підтримуватися в додатках, аплетах і основних операційних системах. При публікації специфікації будь-якого розширювального API до його остаточного виходу, в обов'язковому порядку публікуються промислові огляди з можливістю зворотного зв'язку з розроблювачами.

Деякі ліцензіати платформи Java уклали контракти про включення приватних реалізацій Java Base API в Java – платформі. В міру розробки бібліотек- класів, Java Base Platform розростаються й нові класи регулярно мігрують у встановлену на кожну ліцензійну операційну систему Java Base Platform.

Інший набір API, що називається Standard Extension API, визначений Java Soft у партнерстві з провідними промисловими компаніями, створений для розширення основних функціональних можливостей. Найближчим часом планується мігрувати деяку підмножину Standard Extension API в Java Base Platform.

Embedded Java Platform була розроблена для споживача, що використовує прилади з малими ресурсами й з більш спеціалізованою функціональністю, ніж мережевий комп'ютер,наприклад: принтери, ксерокси, мобільні телефони й ін. Подібна апаратура може мати деякі специфічні властивості, а саме невеликий обсяг пам'яті, відсутність дисплея або неможливість зв'язку по мережі.

API, розроблений для такої платформи, називається Java Embedded API. Java Embedded API – найменший із прикладних програмних інтерфейсів, які можуть бути впроваджені в описані вище прилади й при цьому ефективно працювати. Оскільки дана платформа усе ще допрацьовується, Java Embedded API дотепер не може розглядатися як стандарт. Тому існує деяка невизначеність, пов'язана із складом API. Приблизно, він буде містити в собі пакети java .lang і java.util.Java-Додатки, написані для одного окремого пристрою, зберігають працездатність на широкому діапазоні подібних за своєю специфікою пристроїв.

## Віртуальна машина Java

Java Virtual Machine є ключем до незалежності від основної операційної системи й апаратних засобів. Це платформа, що приховує основну операційну систему від Java аплетів і додатків. Крім того, перенос Java Virtual Machine на браузер або іншу операційну систему не становить ніяких труднощів.

Віртуальна Машина визначає машинно-незалежний формат для двійкових файлів, що зберігаються з розширенням class.Цей формат включає команди для віртуального комп'ютера у формі байт-кодів. Подання байт-коду будь-якої програми, написаної мовою Java, символічне, у тому розумінні, що зсуви й індекси усередині методів не є константами, а представляються символічно, як строкові імена. При першому виклику методу, він відшукується у файлі class по імені, і в цей момент визначається числове значення його зсуву для більш швидкого доступу при наступних викликах. Тому, будь-який новий метод або метод, що перевантажує старий, може бути визначений у будь-якому місці в структурі класу. Він буде переданий у символічній формі, і програма належним чином виконається без порушень при виконанні коду.

Байт-Коди – високорівневі подання програми, які проходять оптимізацію й машинну генерацію об'єктного коду за допомогою just-in-time компілятора, після чого можуть бути виконані. Крім того, всередині Virtual Machine може запуститися «збір сміття», під час якої змінні втримуються в стеках в адресному просторі Java-Платформи.

## Середовище розробки Eclipse

Якщо ви уважно стежите за розвитком OpenSource проектів або програмуванням із застосуванням Java, ви вже могли чути частину того шуму, який супроводжує Eclipse. Eclipse - це розширювання середовища розробки (Integrated Development Environment, далі IDE) з відкритим вихідним кодом (далі OpenSource). Проект був запущений в листопаді 2001 року, коли IBM передала вихідний код свого Websphere Studio Workbench вартістю 40 мільйонів доларів на OpenSource і сформувала консорціум Eclipse [4] для управління триваючої розробкою.

Мета створення Eclipse була сформульована таким чином: " розробити багату, повнофункціональну індустріальну платформу комерційної якості для розробки сильно - інтегрованих інструментів ". Для досягнення цієї мети консорціум націлений на три головних проекту:

1) проект The Eclipse безпосередньо Eclipse IDE ("платформи", що містить і виконуючою інструменти Eclipse), інструментів розробки для Java (Java Development Tools, далі JDT) і середовища розробки Plug -In (Plug – In Development Environment, далі PDE), що дозволяють розширювати платформу;

2) проект Eclipse Tools має своєю метою створення інструментів для платформи Eclipse (у поточній розробці знаходяться підпроекти створення IDE для Cobol, IDE для C/C++, а також інструменту для побудови EMF моделей).

3) платформа Eclipse в поєднанні з JDT включає багато можливостей, які включаються в комерційні IDE : редактор з підсвічуванням синтаксису, інкрементальних компіляція коду, потокобезпечна, відладчик, навігатор по класах, менеджери файлів і проектів, а також інтерфейси до стандартних систем контролю вихідних текстів, таким як CVS і Clear Case.

4) проект Eclipse являє собою першу настільки потужно підтриману світовим IT - спільнотою спробу створення єдиної відкритої інтегрованої платформи розробки додатків, що володіє надійністю, функціональністю і рівнем якості комерційного продукту. Фактично ця платформа призначена для всього і ні для чого конкретно : вона являє собою основу, що має блочну структуру і інтегруючу інструменти розробки ПО різних виробників для створення додатків на будь-якій мові, з використанням будь-яких технологій і для будь-якої програмної платформи. Навколо проекту Eclipse в даний час сформовано співтовариство найбільших IT -компаній, серед яких IBM, SAP AG, RedHat та інші [4].

Також Eclipse пропонує безліч унікальних можливостей, наприклад рефакторінг коду, автоматичне оновлення встановлення коду (за допомогою Менеджера Оновлень), список поточних завдань, налагодження модулів за допомогою JUnit та інтеграцію з інструментом компоновки Jakarta Ant.

Незважаючи на велику кількість стандартних можливостей, Eclipse відрізняється від традиційних IDE по ряду особливостей. Напевно найцікавіше в Eclipse те, що вона повністю незалежна від платформи і мови. Крім мов, підтримуваних консорціумом зараз (Java, Cobol, C / C + +), ведуться розробки по додаванню в Eclipse підтримки таких мов, як Python, Eiffel, PHP, Ruby, і C #.

Консорціум надає готові виконувані файли для Windows, Linux, Solaris, HP- UX, AIX, QNX і Mac OS X. Великий інтерес в Eclipse являє plug - in архітектура, а також багатий API, що надається PDE, що дозволяє розширювати Eclipse. Додавання підтримки для нового редактора, подання або мови програмування є досить простим, завдяки грамотно розробленим API і великим будівельним блокам, що надаються Eclipse.

Переваги:

Eclipse служить в першу чергу платформою для розробки розширень, чим він завоював популярність: будь-який розробник може розширити Eclipse своїми модулями. Вже існують Java Development Tools (JDT), C/C++ Development Tools (CDT), розробляються інженерами QNX спільно з IBM, і кошти для мови Ada (GNATbench, Hibachi), COBOL, FORTRAN, PHP і ін.. від різних розробників.

Багато розширень доповнює середовище Eclipse менеджерами для роботи з базами даних, серверами додатків і т.д. Eclipse JDT (Java Development Tools) - найбільш відомий модуль, націлений на групову розробку: середовище інтегроване з системами керування версіями - CVS, GIT в основній поставці, для інших систем (наприклад, Subversion, MS SourceSafe) існують плагіни. Також пропонує підтримку зв'язку між IDE і системою управління завданнями.

У основне постачання включена підтримка трекера помилок Bugzilla, також є безліч розширень для підтримки інших трекерів (Trac, Jira і ін). В силу безкоштовності і високої якості, Eclipse у багатьох організаціях є корпоративним стандартом для розробки додатків. Eclipse написана на Java, тому є платформо-незалежним продуктом, за винятком бібліотеки SWT, яка розробляється для всіх поширених платформ. Бібліотека SWT використовується замість стандартної для Java бібліотеки Swing. Вона повністю спирається на нижче розміщену платформу (операційну систему), що забезпечує швидкість і натуральний зовнішній вигляд користувацького інтерфейсу, але іноді викликає на різних платформах проблеми сумісності та стійкості додатків.

Враховуючи сотні проектів по розробці plug - in, що ведуться в даний час, таких індустріальних гігантів як IBM, HP і Rational, що надають ресурси, а також проектних важковаговиків як Erich Gamma, що допомагають спрямовувати процес еволюції в потрібне русло - у Eclipse велике майбутнє.

Чому розробники програмного забезпечення вибирають Java. Технологія Java протестована, вдосконалена, розширена і перевірена учасниками спільноти розробників Java, архітекторів і ентузіастів. Java дозволяє розробляти високопродуктивні портативні програми практично на всіх комп'ютерних платформах. Доступність додатків в різнорідних середовищах дозволяє компаніям надавати більш широкий спектр послуг, сприяє підвищенню продуктивності, рівня взаємодії і спільної роботи кінцевих користувачів і істотного зниження вартості спільного володіння корпоративними і споживчими додатками. Java стала незамінним інструментом для розробників і відкрила для них такі можливості:

написання програмного забезпечення на одній платформі і його запуск практично на будь-який інший платформі;

створення програм, що працюють в веб-браузері і мають доступ до веб-служб;

розробка додатків на стороні сервера для форумів в Інтернеті, магазинів, опитувань, обробки форм HTML і багато іншого;

об'єднання додатків або служб з використанням мови Java для створення високоспеціалізованих додатків або служб;

створення багатофункціональних і ефективних додатків для мобільних телефонів, віддалених процесорів, мікроконтролерів, бездротових модулів, датчиків, шлюзів, споживчих продуктів і практично будь-яких інших категорій електронних пристроїв.

Проанілізувавши все вищесказане, можна зробити висновок, що Java повністю задовольняє нас у створенні нашої гри.

# ЕТАПИ РОЗРОБКИ ГРИ



## Постановка завдання на розробку

При тривалій роботі за комп'ютером виникає бажання трохи відпочити. Для цієї мети не підходять ігри типу стратегій, квестів, бо для них потрібно багато часу, а для 5-10 хвилинного відпочинку цілком підійде саме «Gameshoot». Метою даної роботи є розробка гри „Gameshoot ”. Вона відноситься до розважальних ігор, які допомагають розвинути реакцію, тренують логіку. Простий дизайн повинен привести до не складних алгоритмів та структур даних.

У стандартній поставці операційних систем даної гри немає, тому виникла необхідність у її створенні.

## Сценарій гри

Загальний цикл роботи програми можна представити таким чином. Після запуску виникає вікно програми з порожнім ігровим полем. Після цього треба дати команду на початок гри. На полі з’являється гравець та генеруються хвилі супротивників, і починається ігровий процес.

Супротивники відрізняються за типом та рангом. Після знищення кожної хвилі генерується нова, але з біль сильнішими супротивниками. Гра триває до тих пір, поки не будуть знищені усіх хвилі супротивників, або гравець не втратить усі життя. Втрата життя відбувається при доторканні гравця до супротивника. Для того, щоб знищити супротивника, потрібно стріляти по ньому набоями. Після знищення супротивника гравець отримує бали, та з'являється можливість випадання бонусів у вигляді: посилення зброї, уповільнення швидкості ворогів або збільшення кількості життів гравця. Блок-схема алгоритму ігрового процесу наведена на Рис 3.1

Початок гри

Генерація хвиль супротивників

Кінець гри

Зниження життів гравця

N =5

n=0

Знищені усі хвилі

Знищення супротивників

Збираннябонусів

Гральний процес

Рис. 3.1 Блок-схема алгоритму ігрового процесу

## Складові частини гри

Проект складається з дев’яти класів, які зображено на Рис. 3.1. Кожний клас відповідає об’єкту у грі, які мають певні характеристики. Відношення між класами зображено на Рис 3.2.

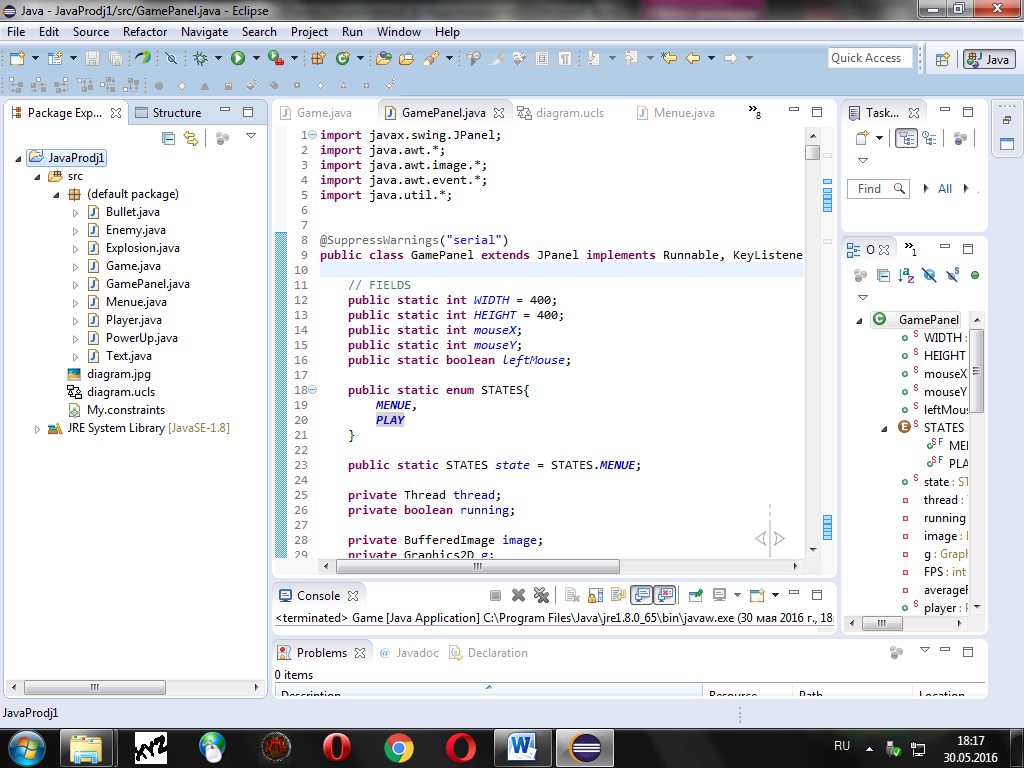


Рис. 3.2 Складові класи проекту

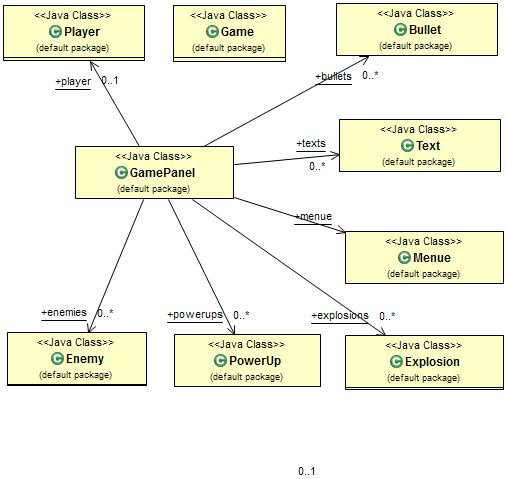


Рис. 3.3 Діаграма взаємодії класів

Клас Game є головним класом проекту, містить метод public static void main (String[] args) який власне запускає гру.

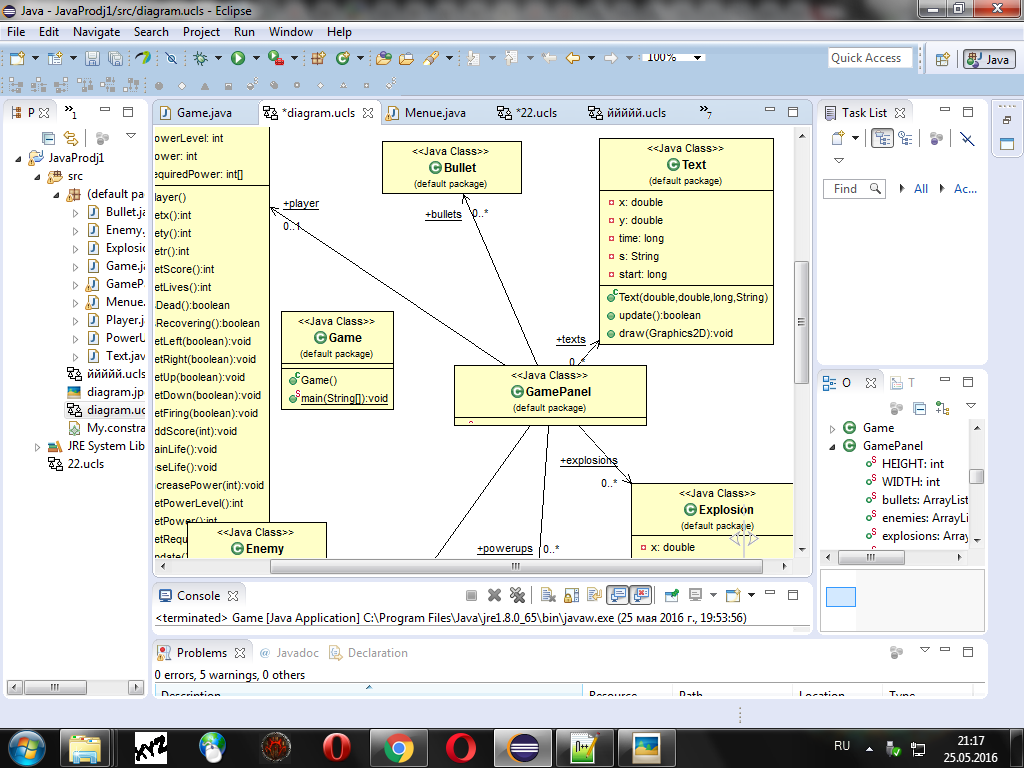


Рис. 3.4 структура класу Game

Клас Bullet- клас набоїв, який містить дані щодо розмірів, розміщення швидкості. Містить наступні методи:

* метод public double getx() - отримує положення координат набою по осі х;
* метод public double gety() - отримує положення координат набою по осі у;
* метод public double getr() -отримує розмір кулі;
* метод public bullet (double angle, int x, int y) – конструктор набоїв;
* метод public void draw (graphics2dg)відповідає за зображення набоїв на ігровому полі.

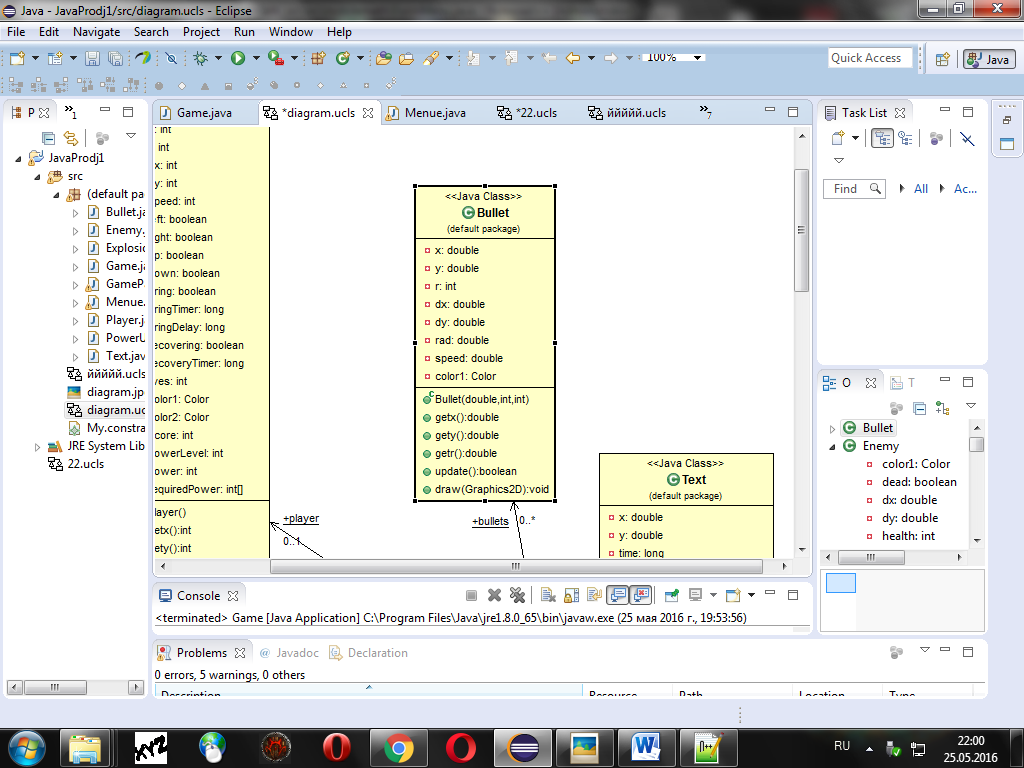


Рис. 3.5 структура класу Bullet

Клас Enemy - клас супротивників, що містить притаманні їм властивості – ранги та типи. Наші супротивники поділяються на три типи та чотири ранги. З кожним збільшенням рангу розмір супротивника та кількість необхідних попадань для його знищення збільшується, але зменшується його швидкість. Містить наступні методи:

* метод public Enemy (int type, int rank) –являє собою конструктор супротивників;
* метод public double getx() - отриму положення коордант супротивника по осі Х;
* public double gety() - отрумує положення координат супротивника по осі У;
* метод public int getr() – отриму значення розміру супротивника
* метод public int gettype() – отримує тип супротивника;
* метод public int getrank() - отримує ранг супротивника;
* метод public void setslow(boolean b) – відповідальній за сповільнення супротивників;
* метод public boolean isdead() – виконує перевірку чи знищений супротивник;
* метод public void hit() відповідальний за попадання набоями по супротивниках;
* метод public void explode() відподальний за знищення супротивника вищого рангу та формування супротивників нижчого ранку на його місці;
* метод public void update () відповідальній за переміщення супротивників на ігровому полі;
* метод public void draw(Graphics2Dg) зображення супротивників на ігровому полі;

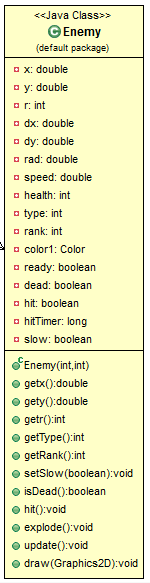


Рис. 3.6 Структура класу Enemy

Клас Text являє собою конструктор, який відповідає за формування тексту на ігровому полі, та має наступні методи:

- метод public Text(double x, double y, long time, String s) є конструктором тексту;

- метод public boolean update() відповідальній за час відображення тексту;

- метод public void draw(Graphics2D g) відповідальний за зображення тексту на ігровому полі.

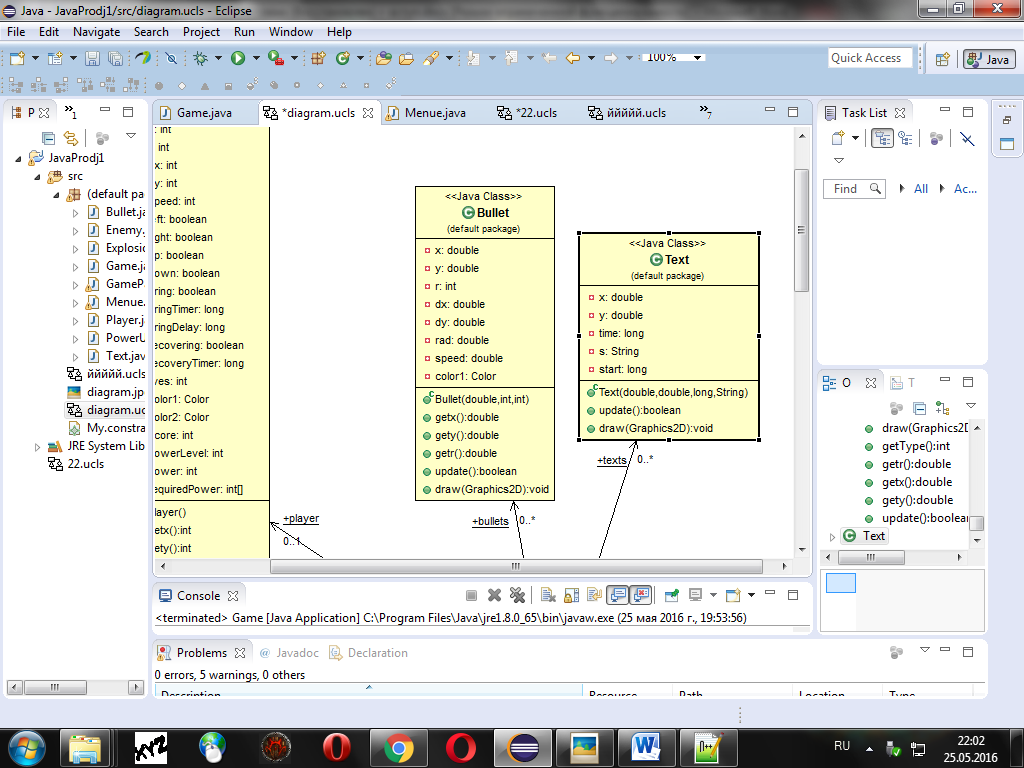


Рис 3.7 Структура класу Text

Клас Player, клас гравця, який містить інформацію щодо гравця, та має такі методи :

* public Player()
* метод public int getx() отримує координат гравця по осі Х;
* метод public int gety() отримує координати гравця по осі У;
* метод public int getr() – отримує розмір гравця;
* метод public int getScore() отримує значення (кількість) очок;
* метод public int getLives() відповільаний за кількість життя;
* метод public void addScore () відповідальний за підрахунок ігрових очок;
* метод public boolean isDead() перевіряє чи знищений гравец
* метод public boolean isRecovering() поновлює гравця після зіткнення з супротивником;
* метод public void setLeft(boolean b) відповідальний за переміщення гравця ліворуч;
* метод public void setRight(boolean b) відповідальний за переміщення гравця праворуч;
* метод public void setUp(boolean b) відповідальний за переміщення гравця вгору;
* метод public void setDown(boolean b) { down = b; } відповідальних за переміщення гравця униз
* метод public void gainLife() відповідальний за отримання життя з піднятих бонусів;
* метод publicvoidloseLife() відповідальний за зменшення життів гравця;
* метод public void increasePower (inti) відповідальний за збільшення потужності гравця;
* метод public void update () відповідний за переміщення та стріляння гравцем;
* метод public void draw (Graphics2Dg) відповідальний за зображення гравця на ігровому полі.

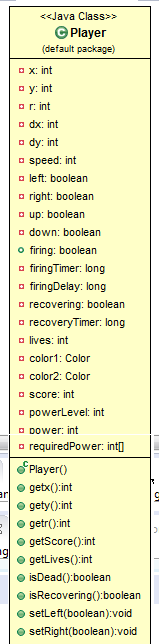


Рис. 3.8 Структура класу Player

Клас PowerUp - клас бонусів, який являє собою метод конструктор в який передаються тип бонусу, та який відповідає за їх зображення на ігровому полі, та містить наступні методи:

* метод public PowerUp(int type, double x, double y) конструктор бонусів;
* метод public double getx() – отримує положення координат бонусів по осі Х;
* метод public double gety() – отримує положення координат бонусів по осі У;
* метод public double getr() – отримує розмір бонусу;
* метод public int getType() отримує типу бонусу;
* метод public boolean update() відповідальний за переміщення бонусів по ігровому полі;
* метод public void draw(Graphics2D g) відповідальний за зображення бонусів на ігрому полі

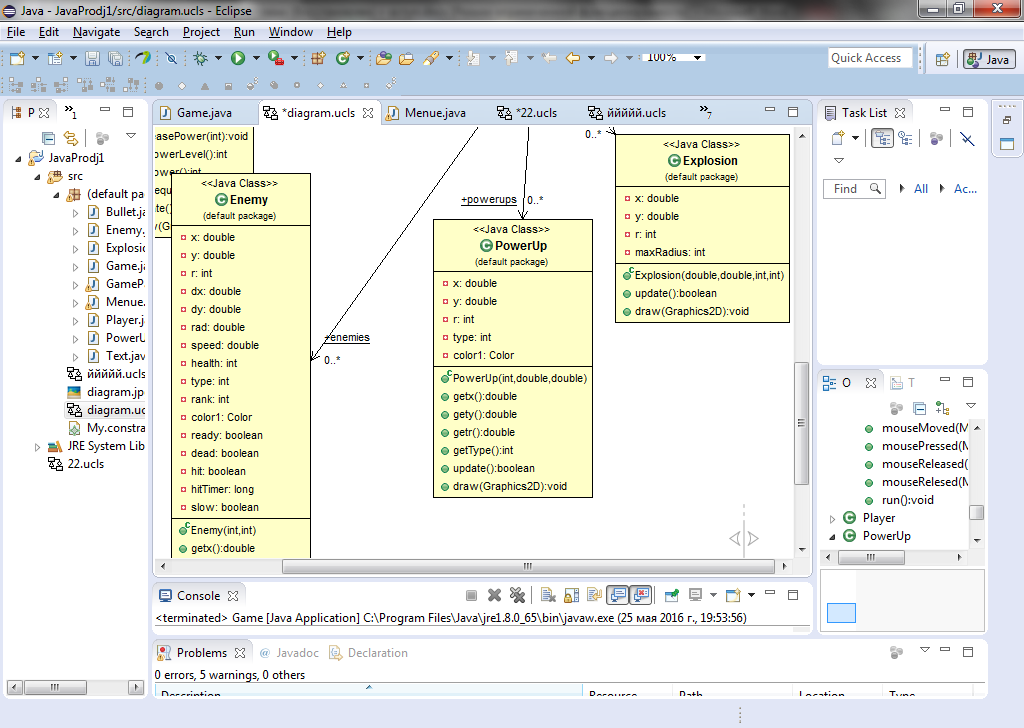


Рис. 3.10 Структура класу PowerUp

Клас GamePanel - клас, який відповідає за роботу гри, що взаємодіє з рештою класів. Даний клас мітить наступні методи:

* метод public GamePanel() конструктор відповідальний за створення ігрового поля по заданим параметрам
* метод public void keyTyped(KeyEventkey), який відповідає за переміщення гравця на ігровому полі при натисканні клавіш (таблиця 3.3) та передає дані в клас Player;
* метод public void keyPressed(KeyEvent key) відповідає за натиснення клавіш клавіатури;
* метод public void keyReleased(KeyEvent key) відповідає зі дію при відпусканню клавіші на клавіатурі;
* метод private void create NewEnemies() відповідає за генерацію хвиль супротивників;
* метод private void game Render () відповідає за відображення ігрового поля та усіх об’єктів на ньому;
* метод private void game Update() відповідає за поновлення даних ігрових об’єктів на ігровому полі;
* метод public void run()відповідає за цикл гри.
* Метод public void mousePressed (MouseEvent e) відповідальний за натискання клавіш миші;
* Метод public void mouseRelesed(MouseEvent e) відповідальний за відтискання кнопок миші;

Таблиця 3.3

Призначення клавіш керування

|  |  |
| --- | --- |
| Назва клавіші | Виконувана функція |
| Стрілка вправо | Рух гравця вправо |
| Стрілка вліво | Рух гравця вліво |
| Стрілка вгору | Рух гравця вгору |
| Z, ПКМ | Вогонь |
| Ecs | Вихід у меню |

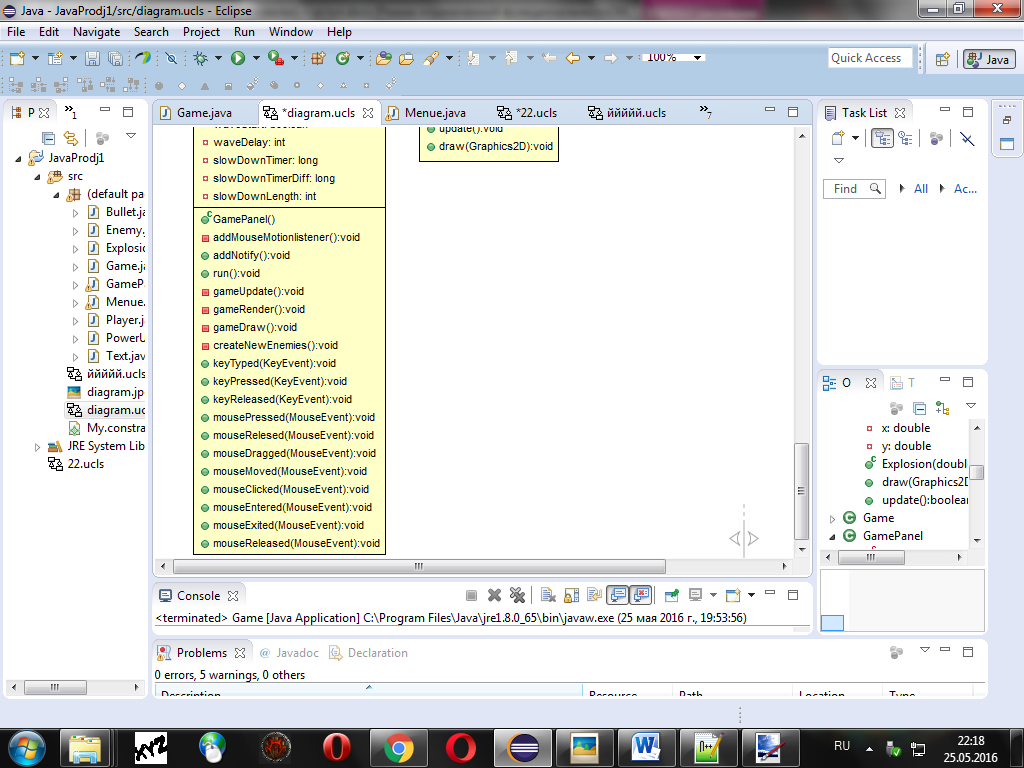
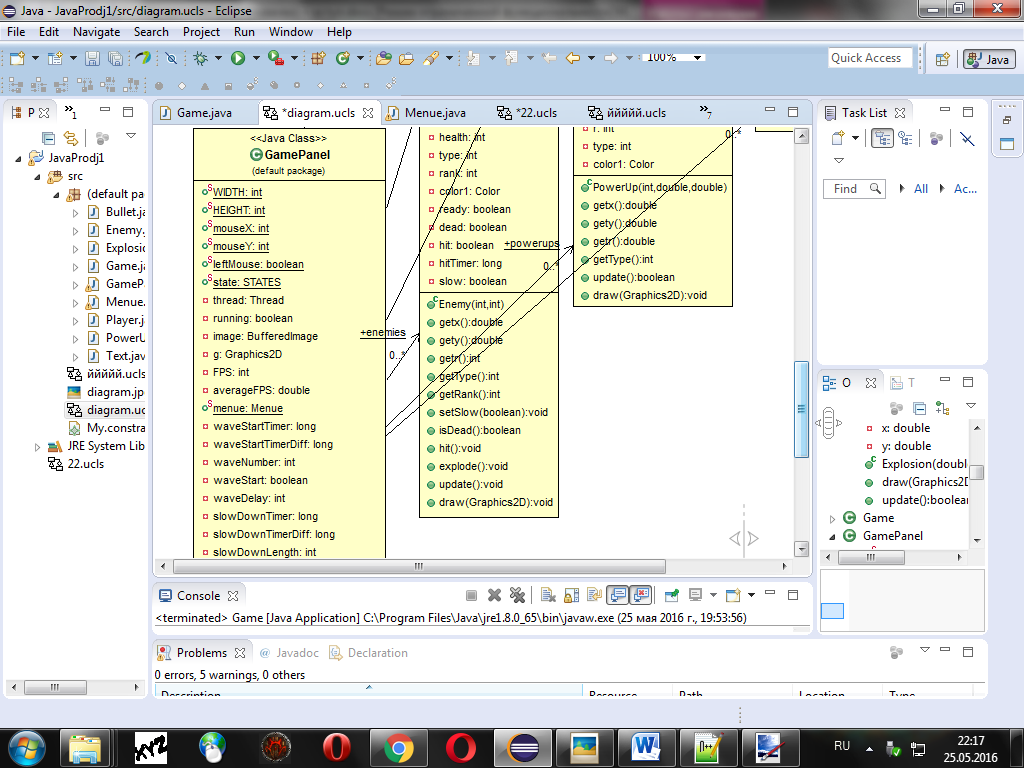


Рис. 3.11 Структура класу GamePanel

Клас Explosion являє собою конструктор, в який передаються дані щодо розміщення та розміру супротивника, який реалізує знищення: метод public boolean update() та відображення на ігровому полі,. Та містить наступні методи:

* метод public Explosion (double x, double y, int r, int max)конструктор знищення супротивника;
* метод public boolean update() відповідальній за поновлення даних при знищенні;
* метод public void draw (Graphics 2D g) відповідальний за зображення знищення на ігровому полі;

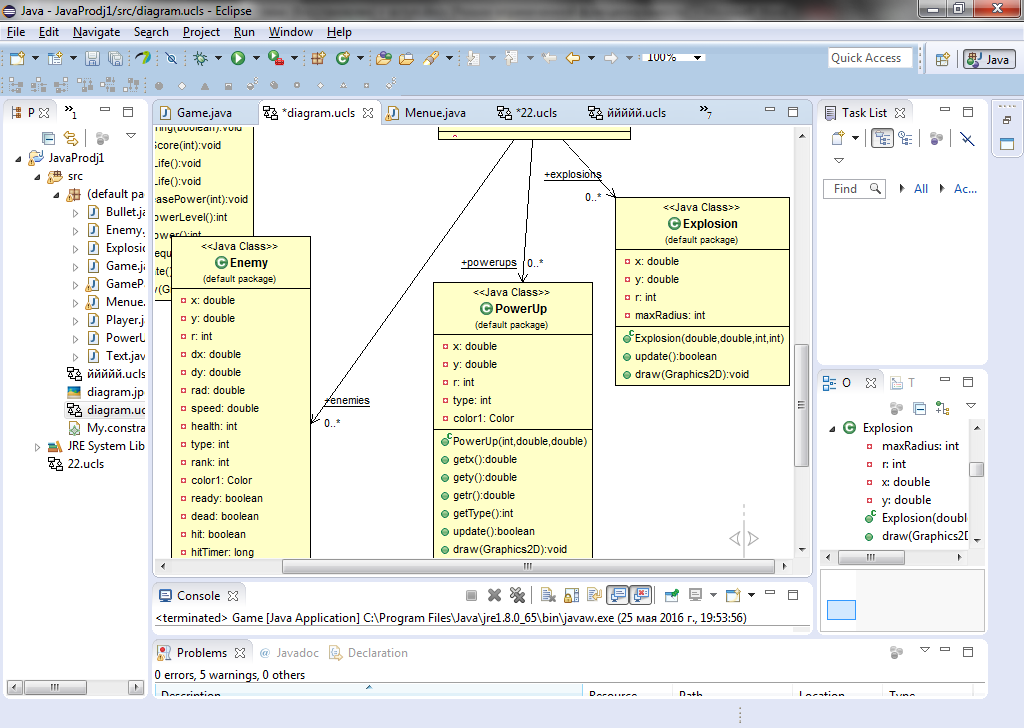


Рис 3.12 структура класу Explosion

Клас Menue – є класом ігрового меню, що містить данні щодо його контексту, та обробник подій при використанні меню. Містить наступні методи:

* метод public Menue()- є конструктором меню;
* метод public void update()- відповідальниій за динамічну зміну елементів меню
* метод public void dwaw(Graphics2D g) - відповідає за зображення меню на ігровому полі

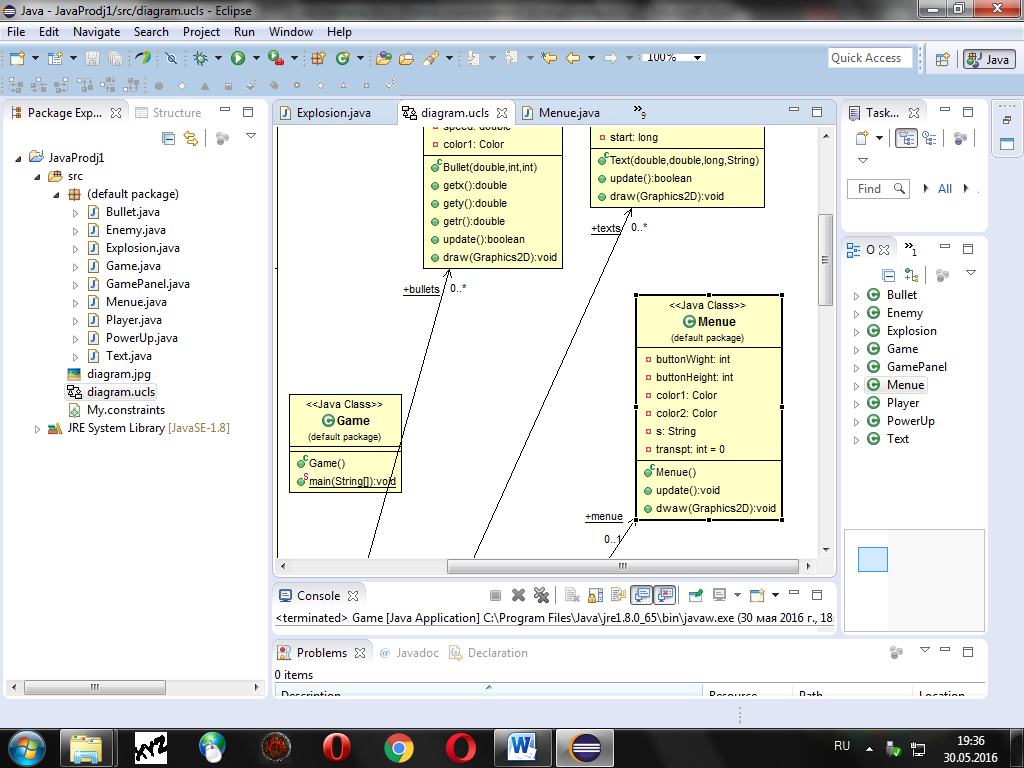


Рис. 3.13 Структура класу Menue

## Розробка інтерфейсу програми

Аналізуючи потоки вхідних і вихідних даних, розробляємо загальний інтерфейс програми, уточнюємо вимоги до функцій, що виконуються програмою, визначаємо їх структуру.

На етапі ескізного проектування були виділені такі об’єкти, що входять до складу програми: власне гра «Gameshoot»; ігрове вікно, яке включає ігрове поле, поле кнопок і ігрове меню. Графічно це відображається в ієрархії об’єктів на рис 3.4.

Гра

«Gameshoot»

Ігрове вікно

Поле індікації

Ігрове меню

Ігрове поле

Життя

Рахунок

Супротивники

Поле кнопок

Продов-жити гру

Зупинити гру

Почати гру спочатку

Зупинити гру

Вийти з гри

бонусів

бонус

Гравець

Рис 3.14 Ієрархія об’єктів

Основною умовою при створенні інтерфейсу для даної гри було використання спокійних відтінків нейтральних кольорів (біле зображення гравця на блакитному фоні)

## Тестування гри

За підсумками розроблених розв'язуваних завдань можна назвати тестові приклади необхідні для перевірки працездатності гри. Якщо з проходження всіх пунктів тестування отримають задовільні результати, можна буде заявити у тому, що програма працює вірно.

1. Після того як набої виходять за межі ігрового поля або зіштовхуються з супротивником вони видаляються з пам’яті комп’ютера.

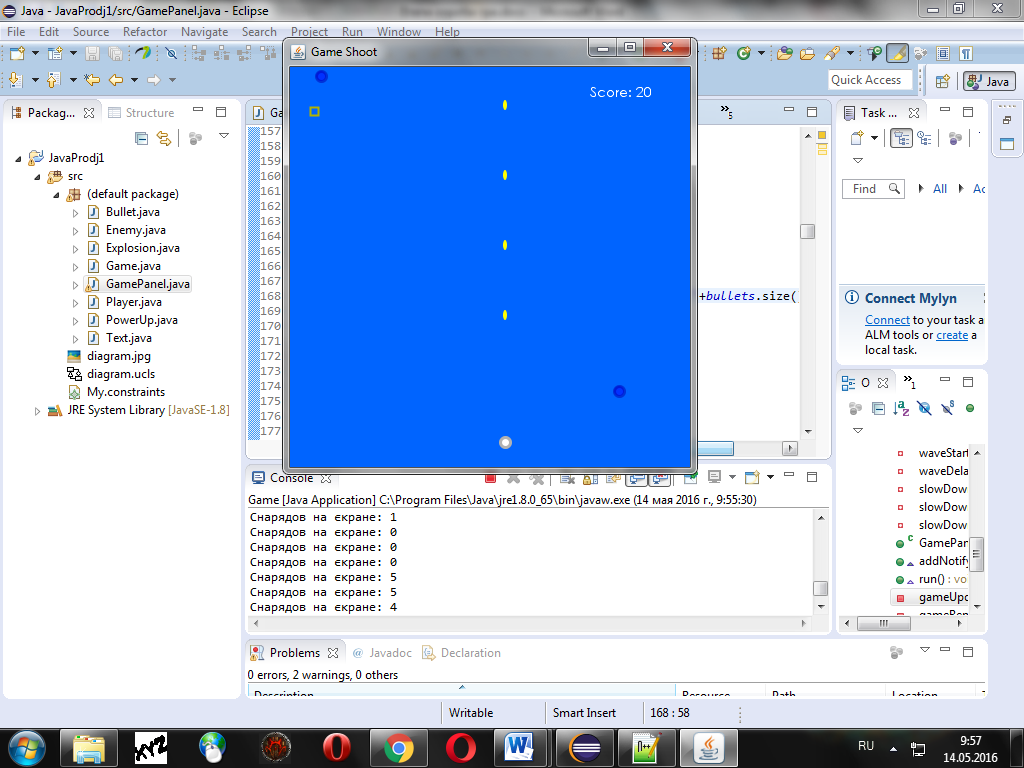


Рис 3.15 Лічильник куль на ігровому полі

1. Після знищення ворога вищого типу відбудеться видалення його з пам’яті і розпаданні на три ворога меншого типу:

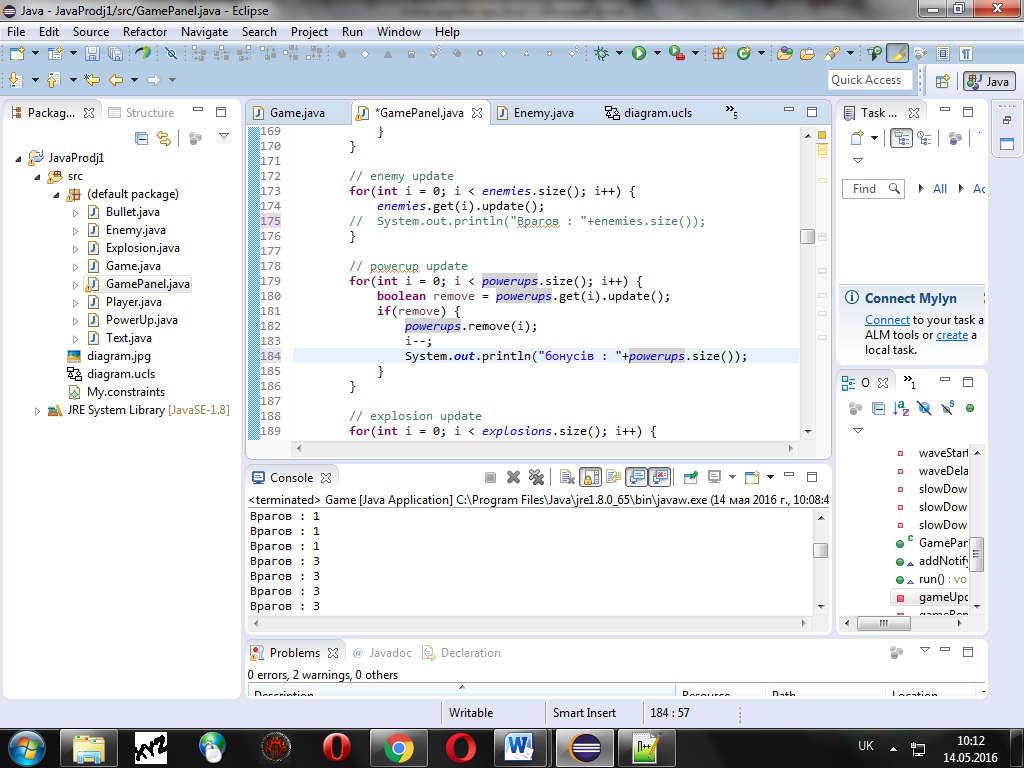
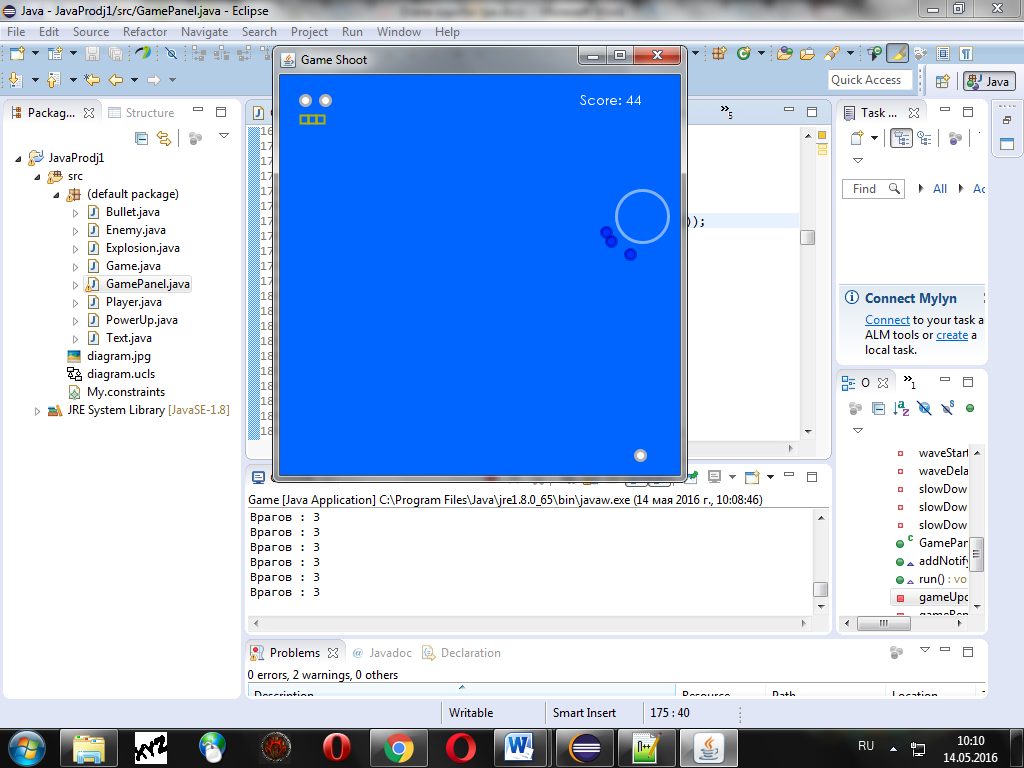


Рис 3.16 Лічильник ворогів на ігровому полі

1. Після підбирання бонусів на ігровому полі відбувається їх видалення з пам’яті

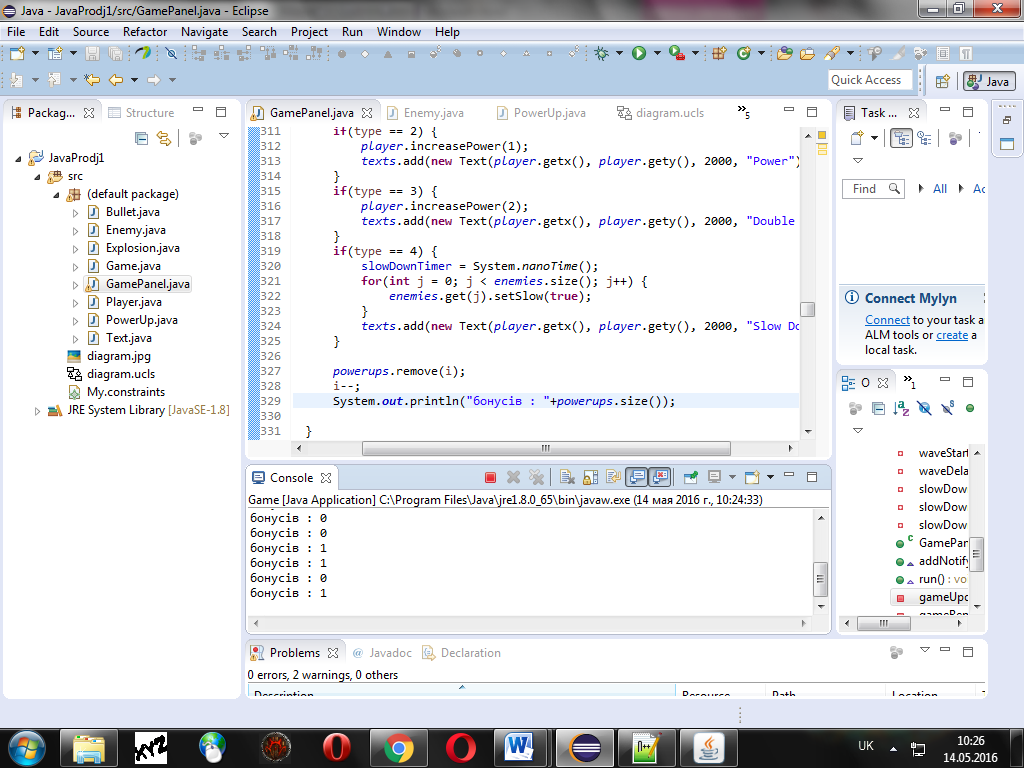


Рис. 3.17 Лічильник бонусів на ігровому полі

1. Гра повністю функціональна та може запускатись головним класом через командну строку. Для надання більш зручного використання скомпільована в єдиний файл. Перевіримо та запустимо програму як єдиний файл рис 3.5.4.

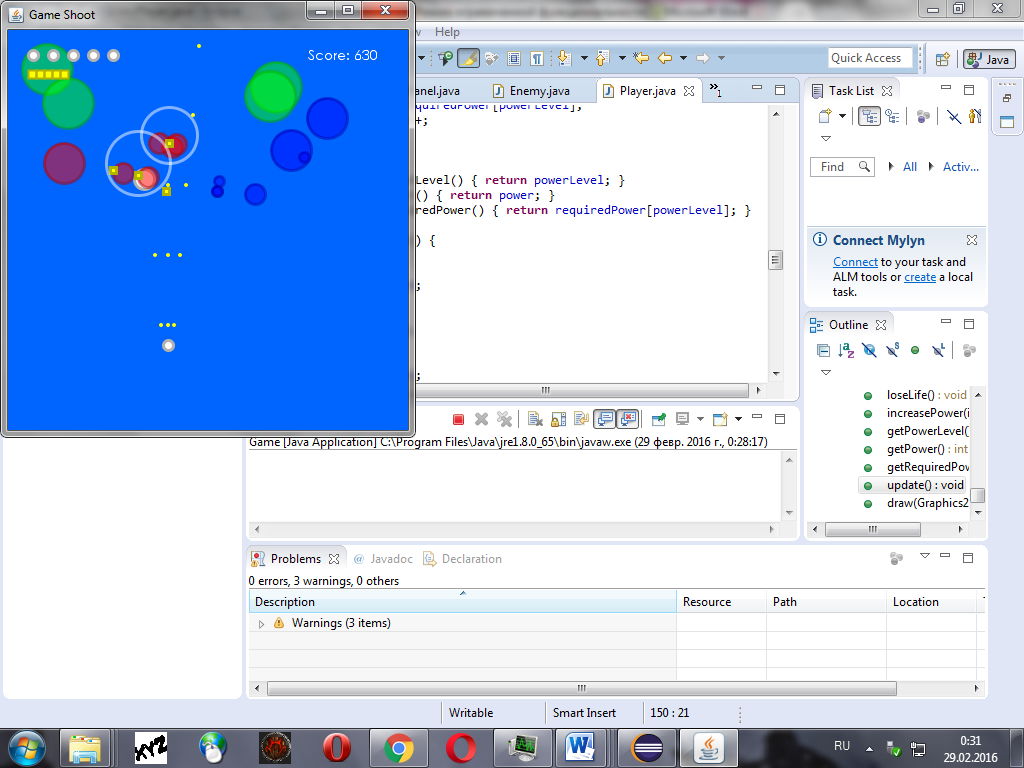


Рисунок 3.18 – Ігровий процес

Протестувавши програму щодо критеріїв можна зазначити, що дана програма:

- відповідає призначенню, тобто, призначена для гри;

- має завершеність, тобто даний продукт має всі риси, необхідні для гри;

- є інтуїтивно зрозумілим і простим інтерфейсом;

- має стабільність;

- в достатній мірі протестована, зроблені виправлення виявлених помилок;

- має гарну адаптованість і незалежність від пристроїв (здатна працювати на різноманітному апаратному забезпеченні);

- продукт не вимагає інсталяції, тобто здійснюється запуск .jar-файлу.

# РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД РОЗРОБКИ І ПРОДАЖУ ПРОГРАМИ

## Визначення витрат на розробку програми

Витрати на розробку програми визначаються по формулі:

З = ФОТ + Отч + Накл + Зпевм., (4.1)

де ФОТ – фонд оплати праці, грн;

Отч – відрахування до позабюджетних фондів, грн;

Накл – накладні витрати, пов'язані із створенням програми, грн;

Зпевм – витрати, пов'язані з експлуатацією програми, грн.

З = 52,90 + 14,50,66 + 14,90+ 22 = 104,30 (грн).

Розмір фонду оплати праці розробника розраховується по формулі:

ФОТ = Зппрям× (1 + Кр / 100) (4.2)

де Зппрям – пряма заробітна плата, грн;

Кр – районний коефіцієнт % (Кр = 15 %).

ФОТ = 46 × (1+15/100)= 52,90(грн)

Зппрям=СР×Окл., (4.3)

де СР – повний термін розробки програми, міс. (днів);

Окл – оклад (денна тарифна ставка) розробника, грн.

Оклад = 40 / 20 = 2 (грн)

Зппрям = 22×2 = 44 (грн)×

Час, що реально витрачається розробником на створення системи приведено в табл. 4.1.

Таблиця 4.1Терміни на розробку програми

|  |  |  |
| --- | --- | --- |
| Стадія розробки | Термін, відведений на стадію розробки (днів) | |
|  | Всього | В т.ч. машинного часу |
| 1. Передпроектна стадія | 2 |  |
| 2. Проектування | 4 | 3 |
| 3. Програмування | 14 | 16 |
| 4. Випробування | 3 | 3 |
| Разом: | 22 | 22 |

Розрахунок страхових внесків до позабюджетних фондів проводиться по формулі:

Отч = ФОТ × (Котч / 100), (4.4)

де Котч – відсоток відрахувань % (Котч = 26%);

Отч = 52,90 × (26 / 100) = 13,75 (грн).

Накладні витрати, пов'язані із створенням програми, складають 35% від ФОТ і розраховуються по формулі:

Накл = ФОТ × 0,35. (4.5)

Накл = 52,90 × 0,35 = 18,51 (грн)

Витрати пов'язані з експлуатацією ПЕВМ, розраховуються по формулі:

Зпевм = tмаш×Смаш.час (4.6)

де tмаш – повний машинний час, витрачений в процесі розробки програми;

Смаш. година – вартість машино-години, грн/година.

Зпевм = 23 × 1 = 23 (грн)

## Розрахунок вартості програми

Вартість розробленої програми визначається по формулі:

Ціна = З × (1 + Нормпр / 100) (4.7)

де Нормпр – норма прибутку %, (Нормпр = 25 %).

Ціна = 104,30× (1 + 25 / 100) = 130,40 (грн)

## Розрахунок економічного ефекту від продажу програми

Економічний ефект від продажу програми визначається за формулою:

ЕФ = Кіл × Ціна – Кіл × З – Кіл × Ціна Зрекл / 100 (4.8)

де Зрекл – витрати на рекламу % на одиницю проданої продукції (Зрекл = 1 %).

Кіл – гадана кількість проданих одиниць продукту, шт/рік.

ЕФ = 100 × 130,40 – 100 × 104,30 – 100 × 130,40 × 1 / 100 = 1867,1(грн).Таким чином економічний ефект від продажу розробленої програми складає 1867,1 гривень на рік.

# 5 ОХОРОНА ПРАЦІ КОРИСТУВАЧІВ ПК

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загострило проблеми збереження власного та суспільного здоров'я, вимагає вдосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів [1, c. 232].

Зараз у нашій країні проводиться розробка національних нормативних документів, спрямованих на охорону праці користувачів ПК. Найбільш повним нормативним документам щодо забезпечення охорони пращ користувачів ПК є «Державні санітарні правила й норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин» ДСанШН 3.3.2.007–98.

## Вимоги до виробничих приміщень для експлуатації ПК

Об'ємно-планувальні рішення будівель та приміщень для роботи з ПК мають відповідати вимогам ДСанПіН 3.3.2.007–98.

Розміщення робочих місць з ПК у підвальних приміщеннях, на цокольних поверхах заборонено.

Площа на одне робоче місце становить не менше ніж 6,0 м3, а об'єм – не менше ніж 20,0 м3.

Приміщення для роботи з ПК повинні мати природне та штучне освітлення відповідно до СНиП П-4–79/

Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче, ніж 1,5%.

Виробничі приміщення повинні обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо, з урахуванням вимог до площі приміщень.

У приміщеннях з ПК слід щоденно робити вологе прибирання.

Приміщення із ПК мають бути оснащені аптечками першої медичної допомоги.

При приміщеннях із ПК мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження. В кімнаті психологічного розвантаження слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04. – 87) [2, c. 17].

## Мікроклімат виробничих приміщень

Складові мікроклімату виробничих приміщень та їх вплив на працівника. На підприємствах на самопочуття, стан здоров’я людини впливає мікроклімат виробничих приміщень, який визначається дією на організм людини температури, вологості, рухомості повітря і теплового випромінювання.

Виробничий мікроклімат, як правило, відрізняється значною мінливістю, нерівномірністю по горизонталі та вертикалі, різноманітністю сполучень температури, вологості, рухомості повітря, інтенсивності випромінювання залежно від особливостей технології виробництва, кліматичних особливостей місцевості, конструкцій споруд, організації повітрообміну із зовнішнім середовищем.

Мікроклімат виробничих приміщень - умови внутрішнього середовища цих приміщень, що впливають на тепловий обмін працюючих з оточенням шляхом конвекції, кондукції, теплового випромінювання та випаровування вологи.

Мікроклімат виробничих приміщень характеризується температурою, вологістю повітря, швидкістю переміщення повітряних мас, а також тепловим випромінюванням від нагрітих обладнання, машин, предметів праці. Від комплексного впливу цих елементів залежать теплові відчуття і зумовлені ними фізіологічні та психічні стани працівників.

Визначальним метеорологічним елементом є температура повітря, дія якої може посилюватися або послаблюватися іншими факторами. Посилення несприятливого впливу одного фактора дією інших факторів характеризується як їхня взаємодія. При антагоністичній взаємодії несприятливий вплив одного фактора послаблюється іншим, що діє в цей час.

Мікроклімат виробничих приміщень зумовлюється технологіч­ним процесом і певною мірою зовнішніми метеорологічними умовами.

Цехи, в яких тепловиділення від обладнання, матеріалів, людей, сонця перевищують 20 ккал на 1 м3 за годину, відносяться до гарячих. Як правило, основними при цьому є теплові випромінювання від нагрітих поверхонь обладнання і матеріалів.

Так, 500-тонна мартенівська піч віддає 15 млн ккал/год, з яких 10 млн ккал припадає на випромінювання. Теплове навантаження складає 250—300 ккал на 1 м3 за годину. Біля 100 ккал на 1 м3 за годину становлять тепловиділення в сушильних цехах заводів будівельних матеріалів, на деяких дільницях виробництва штучного волокна, капрону, в легкій, нафтопереробній, харчовій промисловості. Виконання роботи в умовах теплового випромінювання і високих температур викликає різко виражені фізіологічні зрушення в організмі працюючих.

Таблиця. 5.1

Оптимальні відчуття залежно від температури і вологості повітря

|  |  |  |
| --- | --- | --- |
| Температура °С | Відносна вологість повітря,  % | Стан |
| 21 | 40 | Найбільш приємний стан |
|  | 75 | Відсутні неприємні відчуття |
| 91 | Втома, пригнічений стан |
| 24 | 20 | Неприємні відчуття |
|  | 65 | Потреба у відпочинку |
| 80 | Втома, пригнічений стан |
|  | 100 | Неможливе виконання важкої роботи |
| 34 | 25 | Відсутні неприємні відчуття |

Їхня працездатність в таких умовах знижується на 50 %. Фізіологами встановлено, що температура 22°С є тією межею, за якою починається прогресивне зниження працездатності. Так, при підвищенні температури до 26°С вона зменшується на 4 % з кожним градусом, а при подальшому її підвищенні до 30°С — на 6 %. А. В. Васильєвою виділені три температурних режими: від 0 до 25°С; від 25 до 35°С і від 35 до 50°С.

Кожному режиму властивий певний характер зрушень фізіологічних функцій працівника. При першому режимі ці зрушення знаходяться в зоні допустимих показників; при другому відмічаються підвищені показники, а при третьому — парадоксальні реакції. В умовах, коли температура повітря дорівнює або перевищує температуру тіла працівника, віддача тепла організмом відбувається шляхом випаровування вологи.

Так, при виконанні важкої фізичної роботи при високій температурі повітря кількість виділеного поту може сягати 1,0 – 1,5 л/год. Тепловий удар супроводжується раптовою втратою свідомості, підвищенням температури тіла до 40 – 41°С, слабим частим пульсом, припиненням потовиділення. Судомна хвороба характеризується порушенням водно-солевого обміну, судомами м’язів, кінцівок, діафрагми, потовиділенням, згущенням крові. Теплова рівновага працівника залежить також від вологості повітря, тобто вмісту у повітрі водяних парів. Найсприятливішою для організму є відносна вологість повітря від 35 до 60 %.

Таблиця. 5.2

Оптимальні норми в робочій зоні виробничих приміщень

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Категорія роботи | Холодний і перехідний періоди року (температура зовнішнього повітря нижча за +10°С) | | | Теплий період року (температура зовнішнього повітря вища за +10°С) | | |
|  | Температура  повітря, °С | Відносна вологість повітря,  % | Швидкість руху повітря, м/с | Температура  повітря, °С | Відносна вологість повітря,  % | Швидкість руху повітря, м/с |
| Легка | 20-22 | 60-30 | не більш як 0,2 | 22-25 | 60-30 | 0,2-0,5 |
| Середньої важкості | 17-19 | 60-30 | не більш як 0,3 | 20-23 | 60-30 | 0,2-0,5 |
| Важка | 16-18 | 60-30 | не більш як 0,3 | 18-21 | 60-30 | 0,3-0,7 |

Якщо вологість менша 35 %, то повітряне середовище характеризується сухістю, яка посилює випаровування води з поверхні шкіри. При підвищенні вологості повітря (понад 60 %) випаровування поту утруднене.

Так, при температурі 25°С в умовах дуже сухого повітря організм втрачає через шкіру і легені 75,4 г вологи за годину, а в умовах дуже вологого повітря – лише 23,9 г/год. Для оцінки комфортності умов праці залежно від температури і вологості повітря використовується показник ефективних температур. Ефективною вважається температура, яку відчуває людина при певній вологості повітря і відсутності його руху. Рух повітря у приміщенні також справляє різний вплив на організм працівника, посилюючи або послаблюючи дію інших метеорологічних факторів.

Так, при високій температурі і високій вологості повітря сприятливішою є вища швидкість руху повітря порівняно з комбінацією високої температури і низькою вологістю повітря. Для оцінки комфортності умов праці залежно від температури, вологості і руху повітря використовується показник ефективно-еквівалентної температури.

Ефективно-еквівалентною вважається температура, яка відчувається людиною при певній відносній вологості і швидкості руху повітря. Заходами, які забезпечують створення оптимальних мікрокліматичних умов на виробництві, є:

* механізація важких робіт у гарячих цехах;
* застосування дистанційного управління тепловипромінювальними процесами і апаратами;
* теплоізоляція гарячих поверхонь обладнання;
* застосування теплових повітряних завіс на вході до виробничих приміщень;
* вентиляція і кондиціювання повітря, регулювання вологості повітря.

## Гігієнічні вимоги до організації та обладнання робочих місць

Обладнання й організація робочого місця із БДТ мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру й особливостей трудової діяльності (ГОСТ 12,2.032–78, ГОСТ22.269–76, ГОСТ 21.889–76) [3, c. 177].

Конструкція робочого місця користувача ПК має забезпечити підтримання оптимальної робочої пози.

Робочі місця із ПК слід так розташовувати відносно світлових прорізів, щоб природне світло падало збоку, переважно зліва.

При розміщенні робочих столів із ПК слід дотримуватись таких відстаней: між бічними поверхнями БДТ – 1,2 м; від тильної поверхні одного ПК до екрана іншого – 2,5 м.

Екран ПК має розташовуватися на оптимальній відстані тещ очей користувача, що становить 600…700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Розташування екрана ПК має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30\* до нормальної лінії погляду працюючого.

Клавіатуру слід розташовувати на поверхні столу на відстані 100…300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури в межах 5… 15°.

Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосовувати при екранні фільтри, локальні світлофільтри (засоби індивідуального захисту очей) та інші засоби захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат [3, c. 175].

При оснащенні робочого місця із ВДТ лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам ДСанПіН 3.3.2.007–98.

## Вимоги до режимів праці та відпочинку при роботі з ПК

При організації праці, пов'язаної з використанням ПК, для збереження здоров'я працюючих, запобігання професійним захворюванням І підтримки працездатності передбачаються внутрішньо змінні регламентовані перерви для відпочинку [4, c. 220].

Внутрішньозмінні режими праці й відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення й зниження працездатності.

При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ПК слід вважати таку, що займає не менше 50% робочого часу. Впродовж робочої зміни мають передбачатися:

* перерви для відпочинку і вживання їжі (обідні перерви);
* перерви для відпочинку й особистих потреб (згідно із трудовими нормами);
* додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

За характером трудової діяльності розрізняють три професійні групи, згідно з діючим класифікатором професій [4, c. 223–224]:

1) розробники програм інженери-програмісти) виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну Інформацією з ЕОМ і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в режимі діалогу з ПК у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

2) оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної із ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи й характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

3) оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ПК при 8-годинній денній робочій зміні в залежності від характеру праці:

* для розробників програм із застосуванням ПК слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ПК;
* для операторів із застосуванням ПК слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;
* для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ПК.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ПК не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин робота аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

Для зниження нервово-емоційного напруження, стомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які наведені у Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСаяПІН 3.3.2.007–98 [4, c. 226].

## Вимоги до профілактичних медичних оглядів

Працюючі з ПК підлягають обов'язковим медичним оглядам: попереднім – при влаштуванні на роботу і періодичним – протягом трудової діяльності, відповідно до наказу МЗ України N45 від 31.03.94 р.

Періодичні методичні огляди мають проводитися раз на два роки комісією в складі терапевта, невропатолога та офтальмолога.

До складу комісії, що проводить попередні та періодичні медичні огляди, при необхідності (за наявністю медичних показань), можуть залучатись до оглядів лікарі інших спеціальностей.

Основними критеріями оцінки придатності до роботи з ПК мають бути показники стану органів зору: гострота зору, показники рефракції, акомодації, стану бінокулярного апарату ока тощо. При цьому необхідно враховувати також стан організму в цілому [1, c. 239].

Жінки, що працюють з ВДТ, обов'язково оглядаються акушером-гінекологом один раз на два роки.

Жінки з часу встановлення вагітності та в період годування дитини грудьми до виконання всіх робіт, пов'язаних з використанням ПК, не допускаються.

Виконання вимог, наведених в Правилах, в комплексі з практичним здійсненням первинних та спеціальних заходів повинно стати нормою діяльності всіх фахівців, безпосередньо пов'язаних з навчальними та виробничими колективами.

# Висновок

В даній роботі була реалізована та описана комп’ютерна програма мовою Java. В результаті виконання завдання створена робоча програма в середовищі Eclipse.

Були здобуті навички створення алгоритмів, вивчені основи алгоритмізації та мови Java, навички роботи з різними типами даних, засвоєна специфіка роботи в середовищі програмування.

Досягнута мета та завдання, на прикладі розробки програмного забезпечення у вигляді розважальної гри «Gameshoot», було закріплено вивчений курс програмування мовою Java.

Створена комп’ютерна гра дає можливість людині проводити вільний час, розвиваючи своє логічне мислення.

Інтерфейс програми простий і зручний в користуванні. Тестування програми підтвердило, що програма коректно виконує взаємодію об'єктів різних класів.

Застосування розроблено за допомогою мови програмування Java та середовища розробки Eclpse. Працює на комп'ютері з ОС сімейства Windows, на якій встановлена JAVA з версією не менше 1.7

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* + 1. Статті про програмуванні [Електронний ресурс] URL https://ru.wikipedia.org/wiki/Компьютерная\_игра
    2. Статті про програмуванні [Електронний ресурс] URL https://ru.wikipedia.org/wiki/Категория:Компьютерные\_игры\_по\_жанрам
    3. Статті про середовища розробки [Електронний ресурс] URL https://uk.wikipedia.org/wiki/Інтегроване\_середовище\_розробки
    4. Статті про програмуванні [офіційний сайт] URL https://wiki.eclipse.org/Main\_Page
    5. Статті про програмуванні для [Електронний ресурс] /Режим доступу: URL http://flashbot.ru/Java-dev
    6. Офіційна довідка середовищі програмування [Електронний ресурс] / URL Режим доступу: http://www.jetbrains.com
    7. Статті про програмуванні [Електронний ресурс] URL http://life-prog.ru/komputernaya\_grafika.php
    8. Форум про програмування [Електронний ресурс] / Режим доступу: URL http://www.cyberforum.ru/Java-dev/
    9. Буч Г., Роберт А. Максимчук, Майкл У. Енгл, Боббі Дж. Янг, Д. Коналл, Келлі А. Х'юстон / Об'єктно-орієнтований аналіз і проектування з прикладами додатків, 3-тє вид. : Пер. з англ. - М .: ТОВ «І.Д. Вільямс », 2010. - 720 с.
    10. Підручник, уроки для початківців [Електронний ресурс] / Режим доступу: URL  http://startandroid.ru/ru/
    11. Хашими С. Разработка приложений для Java / Хашими С, Коматинени С, Маклин Д. — Пітер, 2011. — 265с.
    12. Майер Р. Программирование приложений для планшетныхкомпьютеров и смартфонов / Майер Р. — Эксмо, 2011. — 671с.
    13. Khalid A. Mughal - A Programmer’sGuidetoJava™ SCJP Certification - Addison-Wesley -2009.– 1089с.
    14. Michael Sicor-EJB 3 DeveloperGuide -Packt publishing -2008. -277с.
    15. Вязовик Н. А. Програмування на Java. Курс лекцій. - М. Інтернет-університет інформаційних технологій. -2003. -592с.
    16. Карабін П. Мова програмування Java: Створення інтерактивних додатків для Internet. - М. Прес-прес-бук-прес. -2006. -224с.
    17. Знакомьтесь: Java. Самоучитель -Аккуратов Е. Е. -2006. -230с.
    18. ПрименениешаблоновJava. - СтивенСтелтинг, ОлавМаасен -2002. -576с.

# ДОДАТОК А

GamePanel.java

import javax.swing.JPanel;

import java.awt.\*;

import java.awt.image.\*;

import java.awt.event.\*;

import java.util.\*;

public class GamePanel extends JPanel implements Runnable, KeyListener {

// FIELDS

public static int WIDTH = 400;

public static int HEIGHT = 400;

private Thread thread;

private boolean running;

private BufferedImage image;

private Graphics2D g;

private int FPS = 30;

private double averageFPS;

public static Player player;

public static ArrayList<Bullet> bullets;

public static ArrayList<Enemy> enemies;

public static ArrayList<PowerUp> powerups;

public static ArrayList<Explosion> explosions;

public static ArrayList<Text> texts;

private long waveStartTimer;

private long waveStartTimerDiff;

private int waveNumber;

private boolean waveStart;

private int waveDelay = 2000;

private long slowDownTimer;

private long slowDownTimerDiff;

private int slowDownLength = 6000;

// CONSTRUCTOR

public GamePanel() {

// super();

setPreferredSize(new Dimension(WIDTH, HEIGHT));

setFocusable(true);

requestFocus();

}

// FUNCTIONS

public void addNotify() {

super.addNotify();

if (thread == null) {

thread = new Thread(this);

thread.start();

}

addKeyListener(this);

}

public void run() {

running = true;

image = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE\_INT\_RGB);

g = (Graphics2D) image.getGraphics();

g.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g.setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING, RenderingHints.VALUE\_TEXT\_ANTIALIAS\_ON);

player = new Player();

bullets = new ArrayList<Bullet>();

enemies = new ArrayList<Enemy>();

powerups = new ArrayList<PowerUp>();

explosions = new ArrayList<Explosion>();

texts = new ArrayList<Text>();

waveStartTimer = 0;

waveStartTimerDiff = 0;

waveStart = true;

waveNumber = 0;

long startTime;

long URDTimeMillis;

long waitTime;

long totalTime = 0;

int frameCount = 0;

int maxFrameCount = 30;

long targetTime = 1000 / FPS;

// hue = 0;

// GAME LOOP

while (running) {

startTime = System.nanoTime();

gameUpdate();

gameRender();

gameDraw();

URDTimeMillis = (System.nanoTime() - startTime) / 1000000;

waitTime = targetTime - URDTimeMillis;

try {

Thread.sleep(waitTime);

} catch (Exception e) {

}

frameCount++;

if (frameCount == maxFrameCount) {

averageFPS = 1000.0 / ((totalTime / frameCount) / 1000000);

frameCount = 0;

totalTime = 0;

}

}

g.setColor(new Color(0, 100, 255));

g.fillRect(0, 0, WIDTH, HEIGHT);

g.setColor(Color.WHITE);

g.setFont(new Font("Century Gothic", Font.PLAIN, 16));

String s = "G A M E O V E R";

int length = (int) g.getFontMetrics().getStringBounds(s, g).getWidth();

g.drawString(s, (WIDTH - length) / 2, HEIGHT / 2);

s = "Final Score: " + player.getScore();

length = (int) g.getFontMetrics().getStringBounds(s, g).getWidth();

g.drawString(s, (WIDTH - length) / 2, HEIGHT / 2 + 30);

gameDraw();

}

private void gameUpdate() {

// new wave

if (waveStartTimer == 0 && enemies.size() == 0) {

waveNumber++;

waveStart = false;

waveStartTimer = System.nanoTime();

} else {

waveStartTimerDiff = (System.nanoTime() - waveStartTimer) / 1000000;

if (waveStartTimerDiff > waveDelay) {

waveStart = true;

waveStartTimer = 0;

waveStartTimerDiff = 0;

}

}

// create enemies

if (waveStart && enemies.size() == 0) {

createNewEnemies();

}

// player update

player.update();

// bullet update

for (int i = 0; i < bullets.size(); i++) {

boolean remove = bullets.get(i).update();

if (remove) {

bullets.remove(i);

i--;

}

}

// enemy update

for (int i = 0; i < enemies.size(); i++) {

enemies.get(i).update();

}

// powerup update

for (int i = 0; i < powerups.size(); i++) {

boolean remove = powerups.get(i).update();

if (remove) {

powerups.remove(i);

i--;

}

}

// explosion update

for (int i = 0; i < explosions.size(); i++) {

boolean remove = explosions.get(i).update();

if (remove) {

explosions.remove(i);

i--;

}

}

// text update

for (int i = 0; i < texts.size(); i++) {

boolean remove = texts.get(i).update();

if (remove) {

texts.remove(i);

i--;

}

}

// bullet-enemy collision

for (int i = 0; i < bullets.size(); i++) {

Bullet b = bullets.get(i);

double bx = b.getx();

double by = b.gety();

double br = b.getr();

for (int j = 0; j < enemies.size(); j++) {

Enemy e = enemies.get(j);

double ex = e.getx();

double ey = e.gety();

double er = e.getr();

double dx = bx - ex;

double dy = by - ey;

double dist = Math.sqrt(dx \* dx + dy \* dy);

if (dist < br + er) {

e.hit();

bullets.remove(i);

i--;

break;

}

}

}

// check dead enemies

for (int i = 0; i < enemies.size(); i++) {

if (enemies.get(i).isDead()) {

Enemy e = enemies.get(i);

// chance for powerup

double rand = Math.random();

if (rand < 0.001)

powerups.add(new PowerUp(1, e.getx(), e.gety()));

else if (rand < 0.020)

powerups.add(new PowerUp(3, e.getx(), e.gety()));

else if (rand < 0.120)

powerups.add(new PowerUp(2, e.getx(), e.gety()));

else if (rand < 0.130)

powerups.add(new PowerUp(4, e.getx(), e.gety()));

player.addScore(e.getType() + e.getRank());

enemies.remove(i);

i--;

e.explode();

explosions.add(new Explosion(e.getx(), e.gety(), e.getr(), e.getr() + 30));

}

}

// check dead player

if (player.isDead()) {

// running = false;

}

// player-enemy collision

if (!player.isRecovering()) {

int px = player.getx();

int py = player.gety();

int pr = player.getr();

for (int i = 0; i < enemies.size(); i++) {

Enemy e = enemies.get(i);

double ex = e.getx();

double ey = e.gety();

double er = e.getr();

double dx = px - ex;

double dy = py - ey;

double dist = Math.sqrt(dx \* dx + dy \* dy);

if (dist < pr + er) {

player.loseLife();

}

}

}

// player-powerup collision

int px = player.getx();

int py = player.gety();

int pr = player.getr();

for (int i = 0; i < powerups.size(); i++) {

PowerUp p = powerups.get(i);

double x = p.getx();

double y = p.gety();

double r = p.getr();

double dx = px - x;

double dy = py - y;

double dist = Math.sqrt(dx \* dx + dy \* dy);

// collected powerup

if (dist < pr + r) {

int type = p.getType();

if (type == 1) {

player.gainLife();

texts.add(new Text(player.getx(), player.gety(), 2000, "Extra Life"));

}

if (type == 2) {

player.increasePower(1);

texts.add(new Text(player.getx(), player.gety(), 2000, "Power"));

}

if (type == 3) {

player.increasePower(2);

texts.add(new Text(player.getx(), player.gety(), 2000, "Double Power"));

}

if (type == 4) {

slowDownTimer = System.nanoTime();

for (int j = 0; j < enemies.size(); j++) {

enemies.get(j).setSlow(true);

}

texts.add(new Text(player.getx(), player.gety(), 2000, "Slow Down"));

}

powerups.remove(i);

i--;

}

}

// slowdown update

if (slowDownTimer != 0) {

slowDownTimerDiff = (System.nanoTime() - slowDownTimer) / 1000000;

if (slowDownTimerDiff > slowDownLength) {

slowDownTimer = 0;

for (int j = 0; j < enemies.size(); j++) {

enemies.get(j).setSlow(false);

}

}

}

}

private void gameRender() {

// draw background

g.setColor(new Color(0, 100, 255));

g.fillRect(0, 0, WIDTH, HEIGHT);

// draw slowdown screen

if (slowDownTimer != 0) {

g.setColor(new Color(255, 255, 255, 64));

g.fillRect(0, 0, WIDTH, HEIGHT);

}

// draw player

player.draw(g);

// draw bullet

for (int i = 0; i < bullets.size(); i++) {

bullets.get(i).draw(g);

}

// draw enemy

for (int i = 0; i < enemies.size(); i++) {

enemies.get(i).draw(g);

}

// draw powerups

for (int i = 0; i < powerups.size(); i++) {

powerups.get(i).draw(g);

}

// draw explosions

for (int i = 0; i < explosions.size(); i++) {

explosions.get(i).draw(g);

}

// draw text

for (int i = 0; i < texts.size(); i++) {

texts.get(i).draw(g);

}

// draw wave number

if (waveStartTimer != 0) {

g.setFont(new Font("Century Gothic", Font.PLAIN, 18));

String s = "- W A V E " + waveNumber + " -";

int length = (int) g.getFontMetrics().getStringBounds(s, g).getWidth();

int alpha = (int) (255 \* Math.sin(3.14 \* waveStartTimerDiff / waveDelay));

if (alpha > 255)

alpha = 255;

g.setColor(new Color(255, 255, 255, alpha));

g.drawString(s, WIDTH / 2 - length / 2, HEIGHT / 2);

}

// draw player lives

for (int i = 0; i < player.getLives(); i++) {

g.setColor(Color.WHITE);

g.fillOval(20 + (20 \* i), 20, player.getr() \* 2, player.getr() \* 2);

g.setStroke(new BasicStroke(3));

g.setColor(Color.WHITE.darker());

g.drawOval(20 + (20 \* i), 20, player.getr() \* 2, player.getr() \* 2);

g.setStroke(new BasicStroke(1));

}

// draw player power

g.setColor(Color.YELLOW);

g.fillRect(20, 40, player.getPower() \* 8, 8);

g.setColor(Color.YELLOW.darker());

g.setStroke(new BasicStroke(2));

for (int i = 0; i < player.getRequiredPower(); i++) {

g.drawRect(20 + 8 \* i, 40, 8, 8);

}

g.setStroke(new BasicStroke(1));

// draw player score

g.setColor(Color.WHITE);

g.setFont(new Font("Century Gothic", Font.PLAIN, 14));

g.drawString("Score: " + player.getScore(), WIDTH - 100, 30);

// draw slowdown meter

if (slowDownTimer != 0) {

g.setColor(Color.WHITE);

g.drawRect(20, 60, 100, 8);

g.fillRect(20, 60, (int) (100 - 100.0 \* slowDownTimerDiff / slowDownLength), 8);

}

}

private void gameDraw() {

Graphics g2 = this.getGraphics();

g2.drawImage(image, 0, 0, null);

g2.dispose();

}

private void createNewEnemies() {

enemies.clear();

Enemy e;

if (waveNumber == 1) {

for (int i = 0; i < 4; i++) {

enemies.add(new Enemy(1, 1));

}

}

if (waveNumber == 2) {

for (int i = 0; i < 8; i++) {

enemies.add(new Enemy(1, 1));

}

}

if (waveNumber == 3) {

for (int i = 0; i < 4; i++) {

enemies.add(new Enemy(1, 1));

}

enemies.add(new Enemy(1, 2));

enemies.add(new Enemy(1, 2));

}

if (waveNumber == 4) {

enemies.add(new Enemy(1, 3));

enemies.add(new Enemy(1, 4));

for (int i = 0; i < 4; i++) {

enemies.add(new Enemy(2, 1));

}

}

if (waveNumber == 5) {

enemies.add(new Enemy(1, 4));

enemies.add(new Enemy(1, 3));

enemies.add(new Enemy(2, 3));

}

if (waveNumber == 6) {

enemies.add(new Enemy(1, 3));

for (int i = 0; i < 4; i++) {

enemies.add(new Enemy(2, 1));

enemies.add(new Enemy(3, 1));

}

}

if (waveNumber == 7) {

enemies.add(new Enemy(1, 3));

enemies.add(new Enemy(2, 3));

enemies.add(new Enemy(3, 3));

}

if (waveNumber == 8) {

enemies.add(new Enemy(1, 4));

enemies.add(new Enemy(2, 4));

enemies.add(new Enemy(3, 4));

}

if (waveNumber == 9) {

running = false;

}

}

public void keyTyped(KeyEvent key) {

}

public void keyPressed(KeyEvent key) {

int keyCode = key.getKeyCode();

if (keyCode == KeyEvent.VK\_LEFT) {

player.setLeft(true);

}

if (keyCode == KeyEvent.VK\_RIGHT) {

player.setRight(true);

}

if (keyCode == KeyEvent.VK\_UP) {

player.setUp(true);

}

if (keyCode == KeyEvent.VK\_DOWN) {

player.setDown(true);

}

if (keyCode == KeyEvent.VK\_Z) {

player.setFiring(true);

}

}

public void keyReleased(KeyEvent key) {

int keyCode = key.getKeyCode();

if (keyCode == KeyEvent.VK\_LEFT) {

player.setLeft(false);

}

if (keyCode == KeyEvent.VK\_RIGHT) {

player.setRight(false);

}

if (keyCode == KeyEvent.VK\_UP) {

player.setUp(false);

}

if (keyCode == KeyEvent.VK\_DOWN) {

player.setDown(false);

}

if (keyCode == KeyEvent.VK\_Z) {

player.setFiring(false);

}

}

}

Game.java

import javax.swing.JFrame;

//import java.awt.\*;

publicclass Game {

publicstaticvoid main(String[] args) {

JFrame window = new JFrame("Game Shoot");

window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

window.setContentPane(new GamePanel());

window.pack();

window.setVisible(true);

}

}

Bullet.java

import java.awt.\*;

publicclass Bullet {

// FIELDS

privatedouble x;

privatedouble y;

privateint r;

privatedouble dx;

privatedouble dy;

privatedouble rad;

privatedouble speed;

private Color color1;

// CONSTRUCTOR

public Bullet(double angle, int x, int y) {

this.x = x;

this.y = y;

r = 2;

rad = Math.toRadians(angle);

speed = 10;

dx = Math.cos(rad) \* speed;

dy = Math.sin(rad) \* speed;

color1 = Color.YELLOW;

}

// FUNCTIONS

publicdouble getx() { return x; }

publicdouble gety() { return y; }

publicdouble getr() { return r; }

publicboolean update() {

x += dx;

y += dy;

if(x < -r || x > GamePanel.WIDTH + r ||

y < -r || y > GamePanel.HEIGHT + r) {

returntrue;

}

returnfalse;

}

publicvoid draw(Graphics2D g) {

g.setColor(color1);

g.fillOval((int) (x - r), (int) (y - r), 2 \* r, 5 \* r);

}

}

Enemy.java

import java.awt.\*;

publicclass Enemy {

// FIELDS

privatedouble x;

privatedouble y;

privateint r;

privatedouble dx;

privatedouble dy;

privatedouble rad;

privatedouble speed;

privateint health;

privateint type;

privateint rank;

private Color color1;

privateboolean ready;

privateboolean dead;

privateboolean hit;

privatelong hitTimer;

privateboolean slow;

// CONSTRUCTOR

public Enemy(int type, int rank) {

this.type = type;

this.rank = rank;

// default enemy

if(type == 1) {

//color1 = Color.BLUE;

color1 = new Color(0, 0, 255, 128);

if(rank == 1) {

speed = 5;

r = 5;

health = 1;

}

if(rank == 2) {

speed = 4;

r = 10;

health = 2;

}

if(rank == 3) {

speed = 2.5;

r = 20;

health = 3;

}

if(rank == 4) {

speed = 1.5;

r = 30;

health = 4;

}

}

// stronger, faster default

if(type == 2) {

//color1 = Color.RED;

color1 = new Color(255, 0, 0, 128);

if(rank == 1) {

speed = 3;

r = 5;

health = 2;

}

if(rank == 2) {

speed = 3;

r = 10;

health = 3;

}

if(rank == 3) {

speed = 2.5;

r = 20;

health = 3;

}

if(rank == 4) {

speed = 2.5;

r = 30;

health = 4;

}

}

// slow, but hard to kill

if(type == 3) {

//color1 = Color.GREEN;

color1 = new Color(0, 255, 0, 128);

if(rank == 1) {

speed = 1.5;

r = 5;

health = 3;

}

if(rank == 2) {

speed = 1.5;

r = 10;

health = 4;

}

if(rank == 3) {

speed = 1.5;

r = 25;

health = 5;

}

if(rank == 4) {

speed = 1.5;

r = 45;

health = 5;

}

}

x = Math.random() \* GamePanel.WIDTH / 2 + GamePanel.WIDTH / 4;

y = -r;

double angle = Math.random() \* 140 + 20;

rad = Math.toRadians(angle);

dx = Math.cos(rad) \* speed;

dy = Math.sin(rad) \* speed;

ready = false;

dead = false;

hit = false;

hitTimer = 0;

}

// FUNCTIONS

publicdouble getx() { return x; }

publicdouble gety() { return y; }

publicint getr() { return r; }

publicint getType() { return type; }

publicint getRank() { return rank; }

publicvoid setSlow(boolean b) { slow = b; }

publicboolean isDead() { return dead; }

publicvoid hit() {

health--;

if(health <= 0) {

dead = true;

}

hit = true;

hitTimer = System.nanoTime();

}

publicvoid explode() {

if(rank > 1) {

int amount = 0;

if(type == 1) {

amount = 3;

}

if(type == 2) {

amount = 3;

}

if(type == 3) {

amount = 4;

}

for(int i = 0; i < amount; i++) {

Enemy e = new Enemy(getType(), getRank() - 1);

e.setSlow(slow);

e.x = this.x;

e.y = this.y;

double angle = 0;

if(!ready) {

angle = Math.random() \* 140 + 20;

}

else {

angle = Math.random() \* 360;

}

e.rad = Math.toRadians(angle);

GamePanel.enemies.add(e);

}

}

}

publicvoid update() {

if(slow) {

x += dx \* 0.3;

y += dy \* 0.3;

}

else {

x += dx;

y += dy;

}

if(!ready) {

if(x > r && x < GamePanel.WIDTH - r &&

y > r && y < GamePanel.HEIGHT - r) {

ready = true;

}

}

if(x < r && dx < 0) dx = -dx;

if(y < r && dy < 0) dy = -dy;

if(x > GamePanel.WIDTH - r && dx > 0) dx = -dx;

if(y > GamePanel.HEIGHT - r && dy > 0) dy = -dy;

if(hit) {

long elapsed = (System.nanoTime() - hitTimer) / 1000000;

if(elapsed > 50) {

hit = false;

hitTimer = 0;

}

}

}

publicvoid draw(Graphics2D g) {

if(hit) {

g.setColor(Color.WHITE);

g.fillOval((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(3));

g.setColor(Color.WHITE.darker());

g.drawOval((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(1));

}

else {

g.setColor(color1);

g.fillOval((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(3));

g.setColor(color1.darker());

g.drawOval((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(1));

}

}

}

Explosion.java

import java.awt.\*;

publicclass Explosion {

// FIELDS

privatedouble x;

privatedouble y;

privateint r;

privateint maxRadius;

// CONSTRUCTOR

public Explosion(double x, double y, int r, int max) {

this.x = x;

this.y = y;

this.r = r;

maxRadius = max;

}

publicboolean update() {

r += 2;

if(r >= maxRadius) {

returntrue;

}

returnfalse;

}

publicvoid draw(Graphics2D g) {

g.setColor(new Color(255, 255, 255, 128));

g.setStroke(new BasicStroke(3));

g.drawOval((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(1));

}

}  
Player.java

import java.awt.\*;

publicclass Player {

// FIELDS

privateint x;

privateint y;

privateint r;

privateint dx;

privateint dy;

privateint speed;

privateboolean left;

privateboolean right;

privateboolean up;

privateboolean down;

privateboolean firing;

privatelong firingTimer;

privatelong firingDelay;

privateboolean recovering;

privatelong recoveryTimer;

privateint lives;

private Color color1;

private Color color2;

privateint score;

privateint powerLevel;

privateint power;

privateint[] requiredPower = {

1, 2, 3, 4, 5

};

// CONSTRUCTOR

public Player() {

x = GamePanel.WIDTH / 2;

y = GamePanel.HEIGHT / 2;

r = 5;

dx = 0;

dy = 0;

speed = 5;

lives = 8;

color1 = Color.WHITE;

color2 = Color.RED;

firing = false;

firingTimer = System.nanoTime();

firingDelay = 200;

recovering = false;

recoveryTimer = 0;

score = 0;

}

// FUNCTIONS

publicint getx() { return x; }

publicint gety() { return y; }

publicint getr() { return r; }

publicint getScore() { return score; }

publicint getLives() { return lives; }

publicboolean isDead() { return lives <= 0; }

publicboolean isRecovering() { return recovering; }

publicvoid setLeft(boolean b) { left = b; }

publicvoid setRight(boolean b) { right = b; }

publicvoid setUp(boolean b) { up = b; }

publicvoid setDown(boolean b) { down = b; }

publicvoid setFiring(boolean b) { firing = b; }

publicvoid addScore(int i) { score += i; }

publicvoid gainLife() {

lives++;

}

publicvoid loseLife() {

lives--;

recovering = true;

recoveryTimer = System.nanoTime();

}

publicvoid increasePower(int i) {

power += i;

if(powerLevel == 4) {

if(power > requiredPower[powerLevel]) {

power = requiredPower[powerLevel];

}

return;

}

if(power >= requiredPower[powerLevel]) {

power -= requiredPower[powerLevel];

powerLevel++;

}

}

publicint getPowerLevel() { return powerLevel; }

publicint getPower() { return power; }

publicint getRequiredPower() { return requiredPower[powerLevel]; }

publicvoid update() {

if(left) {

dx = -speed;

}

if(right) {

dx = speed;

}

if(up) {

dy = -speed;

}

if(down) {

dy = speed;

}

x += dx;

y += dy;

if(x < r) x = r;

if(y < r) y = r;

if(x > GamePanel.WIDTH - r) x = GamePanel.WIDTH - r;

if(y > GamePanel.HEIGHT - r) y = GamePanel.HEIGHT - r;

dx = 0;

dy = 0;

// firing

if(firing) {

long elapsed = (System.nanoTime() - firingTimer) / 1000000;

if(elapsed > firingDelay) {

firingTimer = System.nanoTime();

if(powerLevel < 2) {

GamePanel.bullets.add(new Bullet(270, x, y));

}

elseif(powerLevel < 4) {

GamePanel.bullets.add(new Bullet(270, x + 5, y));

GamePanel.bullets.add(new Bullet(270, x - 5, y));

}

else {

GamePanel.bullets.add(new Bullet(270, x, y));

GamePanel.bullets.add(new Bullet(275, x + 5, y));

GamePanel.bullets.add(new Bullet(265, x - 5, y));

}

}

}

if(recovering) {

long elapsed = (System.nanoTime() - recoveryTimer) / 1000000;

if(elapsed > 2000) {

recovering = false;

recoveryTimer = 0;

}

}

}

publicvoid draw(Graphics2D g) {

if(recovering) {

g.setColor(color2);

g.fillOval(x - r, y - r, 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(3));

g.setColor(color2.darker());

g.drawOval(x - r, y - r, 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(1));

}

else {

g.setColor(color1);

g.fillOval(x - r, y - r, 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(3));

g.setColor(color1.darker());

g.drawOval(x - r, y - r, 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(1));

}

}

}

PowerUp.java

import java.awt.\*;

publicclass PowerUp {

// FIELDS

privatedouble x;

privatedouble y;

privateint r;

privateint type;

private Color color1;

// 1 -- +1 life

// 2 -- +1 power

// 3 -- +2 power

// 4 -- slow down time

// CONSTRUCTOR

public PowerUp(int type, double x, double y) {

this.type = type;

this.x = x;

this.y = y;

if(type == 1) {

color1 = Color.PINK;

r = 3;

}

if(type == 2) {

color1 = Color.YELLOW;

r = 3;

}

if(type == 3) {

color1 = Color.YELLOW;

r = 5;

}

if(type == 4) {

color1 = Color.WHITE;

r = 3;

}

}

// FUNCTIONS

publicdouble getx() { return x; }

publicdouble gety() { return y; }

publicdouble getr() { return r; }

publicint getType() { return type; }

publicboolean update() {

y += 2;

if(y > GamePanel.HEIGHT + r) {

returntrue;

}

returnfalse;

}

publicvoid draw(Graphics2D g) {

g.setColor(color1);

g.fillRect((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(3));

g.setColor(color1.darker());

g.drawRect((int) (x - r), (int) (y - r), 2 \* r, 2 \* r);

g.setStroke(new BasicStroke(1));

}

}

Text.java

import java.awt.\*;

publicclass Text {

// FIELDS

privatedouble x;

privatedouble y;

privatelong time;

private String s;

privatelong start;

// CONSTRUCTOR

public Text(double x, double y, long time, String s) {

this.x = x;

this.y = y;

this.time = time;

this.s = s;

start = System.nanoTime();

}

publicboolean update() {

long elapsed = (System.nanoTime() - start) / 1000000;

if(elapsed > time) {

returntrue;

}

returnfalse;

}

publicvoid draw(Graphics2D g) {

g.setFont(new Font("Century Gothic", Font.PLAIN, 12));

long elapsed = (System.nanoTime() - start) / 1000000;

int alpha = (int) (255 \* Math.sin(3.14 \* elapsed / time));

if(alpha > 255) alpha = 255;

g.setColor(new Color(255, 255, 255, alpha));

int length = (int) g.getFontMetrics().getStringBounds(s, g).getWidth();

g.drawString(s, (int) (x - (length / 2)), (int) y);

}

}