ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29.03.2012 N 384

Форма N Н-6.01

Міністерство освіти і науки України

Вінницький національний технічний університет

Кафедра програмного забезпечення

КУРСОВА РОБОТА
з дисципліни ” **Об’єктно-орієнтоване програмування** ”
на тему: РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ МОДЕЛЮВАННЯ ОБ’ЄКТІВ ” КАРТА СВІТУ, КАРТА РІВНЯ, 8-BIT - ГОЛОВНИЙ ГЕРОЙ ТА ВОРОГИ ”
З ВИКОРИСТАННЯМ UML ТА МОВИ ПРОГРАМУВАННЯ JAVA

Студента (ки) __1___ курсу _ групи ЗПІ-19б спеціальності 121 Інженерія програмного забезпечення” Примчука Р.

Керівник: Ст. викл. кафедри ПЗ,
Л.М. Круподьорова

Національна шкала - Відмінно
Кількість балів: 92 Оцінка: ECTS A

Члени комісії _____ доц. каф.ПЗ, к.т.н.
(підпис) Д.І. Катєльніков
(прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Вінниця - 2020 рік

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедри ПЗ, проф., д.т.н.
_____ О.Н. Романюк
(підпис)
Затверджено на засіданні каф. ПЗ протокол №_12 від "28" січня 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу з дисципліни "Об'єктно-орієнтоване програмування"
студенту групи 3ПІ-19б Примчуку Р.

РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ МОДЕЛЮВАННЯ ОБ'ЄКТІВ " КАРТА СВІТУ, КАРТА РІВНЯ, 8-BIT - ГОЛОВНИЙ ГЕРОЙ ТА ВОРОГИ " З ВИКОРИСТАННЯМ UML ТА МОВИ ПРОГРАМУВАННЯ JAVA

Вихідні дані:

- Середовища розробки NetBeans IDE, IntelliJ IDEA або Eclipse, платформа JavaFX.

Вимоги до курсової роботи:

В програмі повинно бути реалізовано: додаток Windows з графічним інтерфейсом. Клас мікрооб'єкта повинен містити не менше 4 елементів змінних і не менше 4 методів (окрім геттерів та сеттерів). Як мінімум одна змінна повинна бути типу int, одна – типу double і як мінімум одна – рядок. Змінні-елементи класу мають бути приватними, для них повинні бути створені відповідні аксесори та мутатори (іншими словами – геттери та сетери. В класі мікрооб'єкта повинен бути присутній конструктор без аргументів. Повинно бути використано делегування конструкторів. Додати до класу мікрооб'єкта в якості елемента посилання на невеличкий службовий об'єкт, щоб зробити необхідним глибинне копіювання. Реалізовано глибинне копіювання шляхом реалізації інтерфейсу Cloneable. Повинне бути продемонстроване використання глибинного копіювання. Продемонструвати використання як мінімум двох функцій-статичних елементів класу java.util.Arrays (можливо для цього потрібно буде створити тимчасовий об'єкт-масив з колекції, яка використовується). Продемонструвати використання інтерфейсу Comparable або Comparer. Мікрооб'єкт: мінімум три графічних примітива (один з яких - текст). Більший макрооб'єкт: мінімум три графічних примітива (один з яких - текст). Менший макрооб'єкт: мінімум три графічних примітива(один з яких - текст). При натискуванні лівої кнопки миші на мікрооб'єкті він повинен ставати активним/неактивним. В програмі повинна бути реалізована активація декількох об'єктів одночасно. При натискувань клавіш-стрілок активні об'єкти/об'єкт повинні рухатись у вікні програми. При натискувань клавіши Delete активні об'єкти повинні знищуватись. Якщо активного об'єкта нема – клавіша Delete ігнорується. Клавіша Esc повинна відміняти активацію обєкта. Повина бути побудована ієрархія класів мікрооб'єктів, яка містить як мінімум три рівня наслідування, які можуть інстанціюватись і зображатись на екрані. Зображення кожного рівня має відрізнятись від всіх інших хоча б одним графічним примітивом. При натискуванні клавіши Insert повинно з'являтись діалогове вікно, яке повинно визначати параметри створюваного мікрооб'єкта. В нього повинна бути додана можливість обирати до якого з класів нащадків у ієрархії наслідування він належить. Крім керуючого елемента Button у діалоговому вікні також повинні бути використані як мінімум три з наступного списку: TextField, CheckBox, {ListBox, ChoiceBox}, RadioButton. Додати у проект клас/класи більшого макрооб'єкта. Більший макрооб'єкт призначений містити всі мікрооб'єкти та всі менші макрооб'єкти. У вікні програми відображається лише певна частина більшого макрооб'єкта, яка не повинна перевищувати 25% його загального розміру. Мінімальний розмір більшого макрооб'єкта 1600 позицій для мікрооб'єкта (тобто на екрані одночасно можна побачити не більше квадрата 20*20 позицій для мікрооб'єкта). Реалізувати можливість змінювати частину більшого макрооб'єкта, яку спостерігає користувач (будь яким способом). Додати у проект клас/класи меншого макрооб'єкта. Менший макрооб'єкт повинен мати можливість містити більше одного мікрооб'єкта. Не обов'язково мати можливості додавати/видаляти менші макрооб'єкти в програму. Цілком достатньо, якщо програма буде мати деяку кількість менших макрооб'єктів сгенеровану при запуску програми і яка буде незмінною протягом роботи програми. Менші макрооб'єкти повинні мати певні позиції в більшому макрооб'єкті і повинні відображатись на екраині. Позиції менших макрооб'єктів в середині більшого макрооб'єкта можуть залишатись незмінними протягом роботи програми. В більшому макрооб'єкті повинно міститись більше одного екземпляра меншого макрооб'єкта. Мікрооб'єкти можуть або належати одному (або більшій кількості) меншому макрооб'єкту, або не належати жодному. Глядачу має бути зрозуміло в чому полягає приналежність мікрооб'єкта меншому макрооб'єкту, тобто чим поведінка такого мікрооб'єкта відрізняється від поведінки мікрооб'єкта, який не належить жодному меншому макрооб'єкту. Повинна обов'язково бути релізовано можливість вилучити мікрооб'єкт з всіх менших макрооб'єктів, в результаті чого він не буде належати жодному меншому макрооб'єкту. Також повинна бути реалізована можливість вручну завести мікрооб'єкт у менший макрооб'єкт. Всі мікрооб'єкти та менші макроб'єкти повинні належати більшому макрооб'єкту і не повинні мати змоги вийти за його межі. Повинен бути реалізований автоматичний рух деяких мікрооб'єктів: 1) при русі деякі мікрооб'єкти взаємодіють між собою (це має бути помітно візуально в тому сенсі, що або щось змінюється у їх зображенні або у характері їх руху). Повинен бути реалізований автоматичний рух деяких мікрооб'єктів: 2) деякі мікрооб'єкти заходять в менші макрооб'єкти та деякі виходять. Повинен бути реалізований автоматичний рух деяких мікрооб'єктів: 3) характер руху мікрооб'єктів повинен змінюватись при натискуванні певної клавіші клавіатури або при виборі команди меню. Під характером руху мається на увазі не просто траєкторія, а те, в який спосіб вони взаємодіють з меншими макрооб'єктами та іншими мікрооб'єктами. Повинна бути реалізована мінікарта, яка дозволяє прискорену навігацію: 1) На мінікарті повинна бути позначена видима область великого макрооб'єкта. 2) Рух видимої області великого макрооб'єкта відповідно змінює мінікарту. 3) Менші макрооб'єкти та мікрооб'єкти позначені на мінікарті. 4) Рух та взаємодія мікрооб'єктів призводить до відповідних змін на мінікарті. 5) Натискування лівої кнопки миші на мінікарті призводить до переміщення видимої області великого макрооб'єкта. Повинні бути продемонстровано використання віртуальних функцій. Повинні бути продемонстровано типу доступу protected. Повинно бути продемонстровано використання динамічного поліморфізму. Повинно бути продемонстровано використання статичного поліморфізму. Повинна бути запрограмована серіалізація/де-серіалізація всіх об'єктів у текстовий/бінарний/XML файл. Серіалізація/де-серіалізація обов'язково повинна зберігати не тільки власне інформацію про стан макро- та мікро-об'єктів, але й про їх позицію на екрані. При серіалізації/де-серіалізації обов'язково повинні використовуватись діалогові вікна, щоб запитати у користувача ім'я файлу та його розташування на диску (наприклад можна використовувати системне діалогове вікно FileChooser).

Зміст ПЗ до курсової роботи

Індивідуальне завдання
АНОТАЦІЯ
ВСТУП
1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ ЗАВДАННЯ НА РОБОТУ
2 РОЗРОБКА ІНТЕРФЕЙСУ ПРОГРАМИ ТА ПРИКЛАДІВ ВИКОРИСТАННЯ
3 РОЗРОБКА ПІДСИСТЕМИ ГРАФІЧНОГО ВІДОБРАЖЕННЯ
4 РОЗРОБКА ДІАГРАМ КЛАСІВ, ОБ'ЄКТІВ ТА СТАНУ
5 ВИКОРИСТАННЯ ЗАСОБІВ ПРОГРАМУВАННЯ JAVAFX
6 РОЗРОБКА ПІДСИСТЕМИ СЕРІАЛІЗАЦІЇ/ДЕСЕРІАЛІЗАЦІЇ ДАНИХ
7 КЕРІВНИЦТВО КОРИСТУВАЧА
ВИСНОВКИ
ПЕРЕЛІК ПОСИЛАНЬ
Додатки (за необхідністю)

Дата видачі "6" лютого 2020 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

АНОТАЦІЯ

Основної метою курсової роботи є розробка програмного продукту – додатку (комп’ютерної гри) для Windows ОС. Під час розробки була спроектована система для моделювання об’єктів «Карта світу, карта рівня, 8-bit – головний герой та вороги» з використанням UML та мови програмування Java, зокрема графічної бібліотеки JavaFX. Програма містить у собі такі поняття, як робота з класами, файлами, графічними вікнами, принципи інкапсуляції, наслідування та різних видів поліморфізму. В результаті виконання курсової роботи отримано повністю готовий програмний продукт, а саме аналог усім відомої гри «Contra».

ЗМІСТ

ВСТУП

В даний час відбувається стрімкий розвиток глобального процесу інформатизації суспільства. При цьому кардинальним чином змінюється все інформаційне середовище суспільства. Нові автоматизовані інформаційні технології проникають практично у всі сфери соціальної практики і стають невід'ємною частиною нової, інформаційної культури людства.

ІТ – це сфера, що розвивається все більше і більше з кожним днем, тому саме сьогодні ця тема є найбільш актуальною. У сучасному світі люди можуть з легкістю використовувати хмарні сховища для збереження своєї інформації в надійному місці. Це дає змогу отримати доступ до даних з будь-якої точки світу. Інтернет дозволяє передавати інформацію за тисячі кілометрів в один натиск. За допомогою комп'ютерів можна обробляти величезні об'єми інформації за кілька секунд. Це дає змогу бути нереально продуктивними у будь-якій сфері.

Чого тільки коштує новітня розробка у сфері VR-технологій (віртуальна реальність). Це створене комп'ютером тривимірне середовище, з яким може взаємодіяти людина. Якщо говорити простою мовою – завданням окулярів віртуальної реальності є перехитрити мозок таким чином, щоб він сприймав видиме за реальне за допомогою спеціальних технологій. Насправді VR-можливості дуже широкі: абітурієнти можуть ознайомитися із своїми майбутніми університетами та прогулятися їхніми територіями, навіть не виходячи з дому. Люди можуть на власні очі побачити точне відтворення квартир, які вони збираються придбати, та зайти в будь-яку кімнату ще до початку будівництва. Будь-хто може стрибнути з парашутом або покататися на лижах, а діти навіть можуть опинитися в центрі самої казки та взаємодіяти з персонажами.

ІТ-технології розвиваються не тільки у навчальних сферах, але й в ігровій індустрії. На сьогодні є величезна кількість продуктів для споживання у всьому світі, але хочу поговорити саме про Україну. Індустрія українських комп'ютерних ігор має давню і багату історію, співтовариство гейм девелоперів включає в себе як фрілансерів-одинаків, так і великі компанії, відомі у всьому світі. Експерти

оцінюють український ринок розробки ігор в десятки, а то й сотні мільйонів доларів. Серед таких компаній є: GSC GAME WORLD, VOSTOK GAMES, FLYING CAFE FOR SEMIANIMALS, PERSHA STUDIA, ILOGOS і тд. Звичайно, такі компанії розробляють ігри на вищому рівні і заробляють на них чималі кошти. Але чи кожен готовий платити за те, щоб просто розважитись, погравши шутер в телефоні чи на ПК. Не впевнений у цьому. Ось і головна проблема таких великих компаній – доступність. Для того, щоб оцінити продукт розробників, його потрібно спочатку придбати, а така змога є не у всіх. Інколи, ціни на ігри досягають реально космічних цін для простого середньостатистичного працівника. Але цю проблему можна легко вирішити шляхом моделювання ігор, наприклад, з рекламою та внутрішньоігровими покупками. Під час цього можна використати такі ресурси, як UnityAds або GoogleAds. Завдяки ним можна легко підключити рекламу до певної програми та отримувати реальні кошти, які будуть нараховуватись за перегляди цієї ж advertising.

Під час виконання курсової роботи можна з легкістю використовувати різноманітні IT-ресурси, які надаватимусь змогу робити все легко і просто. В даній грі доречно було б використати різноманітні бібліотеки Java, а зокрема JavaFX. Саме завдяки цьому можна легко створити віконні програми для ПК та користуватись ними. Можливо, можна було б використати певні VR-ресурси у написанні цієї гри, але JavaFX цього не дозволяє.

Підсумовуючи всі вище наведені факти, і було взято за тему курсової гру-шутер, а саме аналог усім відомої «Contra», яку ще наші батьки грали у дитинстві на Dendy. Звичайно ж, цей додаток буде безкоштовним і в нього зможе пограти кожен!

Документація

Тема: Аналог гри «Contra»

Великий макрооб’єкт: Карта світу гри

Малий макрооб’єкт: Карта рівня

Мікрооб’єкти: 8-bit - головний герой та вороги.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ

1.1 Предметна область

У цьому розділі описано гру «Contra», яку й було взято в основу курсової роботи. Ґрунтуючись на цьому додатку, розроблену новий аналог цієї гри, яка й представлена в даній роботі.

1.1.1 Геймплей

Персонаж матиме змогу переміщуватись в усі сторони (в оригінальній версії – тільки вліво та вправо), знищуючи різноманітних ворогів на своєму шляху, стаціонарні гармати, турелі, танки, падаючі гранати. Грати можна буде тільки одному гравцю, але є змога керувати одразу кількома героями. На рівнях можна підбирати корисні бонуси, такі як відновлення здоров’я героя, збільшення шкоди зброї та ін. Зброя та поліпшення знаходяться в контейнерах на карті. Є можливість створити нового героя та керувати ним. Відновити здоров’я можна також й у середині малого макрооб’єкту – бараці.

1.1.2 Мікрооб’єкт – 8-bit (головний герой)

Взагалі тим, хто вже колись зустрічався з цією грою, зрозуміло, що в оригінальній версії «Contra» персонажів є доволі багато і можна вибирати, за кого ти будеш проходити рівень. Я ж у свою чергу планую зробити головним персонажем воїна, якого звуть 8-bit (*рис 1.1*), хоча також будуть й інші герої, якими буде змога керувати.

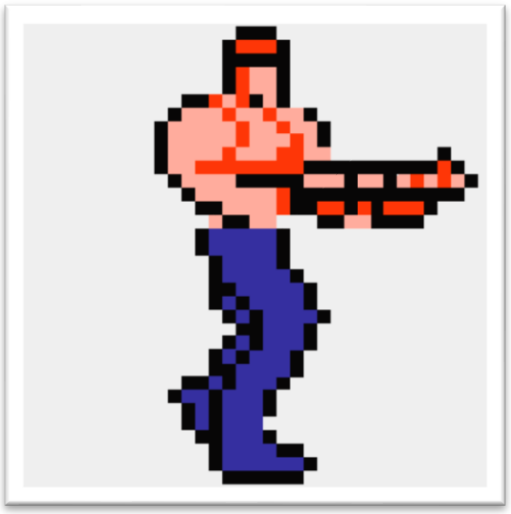

Рисунок 1.1 - «Герой «Contra» - 8-bit»

Показники головного героя:

- Ім’я
- Кількість здоров’я
- Швидкість
- Зброя

Види зброї:

- '(M)' (Machinegun, кулемет) — стріляє автоматично чергою куль, не вимагаючи натискань кнопки стрілянини.
- '(S)' (Spread, Super, Shotgun або Salute, дробовик) — випускає п'ять куль середньої швидкості за постріл. Кулі летять віялом, що дозволяє стріляти не прицільно, вражати кількох ворогів одночасно або влучати в ціль кількома кулями. Якщо на кнопку вогню натискати дуже часто, то «віяла» можуть створювати дуже щільний потік вогню.
- '(F)' (Fire або Fougasse, вогнемет) — стріляє повільними згустками полум'я, які рухаються в напрямку пострілу широкими колами, захоплюючи значну площу. [1]

Всього є три рівні ієрархії героїв. Перший рівень – Hero, який містить у собі усі характеристики та методи взаємодії з макрооб’єктами. Другий рівень – Ninja, який містить перевизначений метод створення героя. Третій рівень – Hero, який також містить перевизначений метод створення героя.

Бонуси, що зустрічаються на карті (в контейнерах):

- Відновлення здоров’я героя
- Покращення швидкості героя
- Збільшення швидкості героя
- Покращення шкоди зброї

Вороги

- Бійці-вороги (мають ті ж характеристики, що і головний герой)
- Стаціонарні гармати
- Турелі
- Танки
- Міни
- Інші герої

1.1.3 Великий макрооб’єкт – карта світу

Великий макрооб’єкт представляє собою карту світу. Якщо ж вдаватись в подробиці, то це місце, де знаходиться список усіх рівнів, в які ти можеш пограти.

У чому ж різниця між рівнями?

Рівні відрізняються складністю ворогів, кількістю контейнерів та зброєю, яка видається на початку бою. Якщо ж бути простіше, то різниця тільки в складності, тому треба дуже постаратись, щоб пройти усе.

Обирай рівень та вперед за перемогою!

1.1.4 Малий макрооб’єкт – карта рівня

Малий макрооб’єкт представляє собою карту рівня. Якщо ж бути точнішим, то це місце, де знаходяться всі об’єкти та відбуваються усі події. Головний герой стикається з ворогами та має подолати їх. На карті також є бонусні контейнери, у яких можуть знаходитись зброя або покращення характеристик героя. Також є змога відкрити міні-карту та подивитись на розташування усіх об’єктів та ворогів.

1.2 Існуючі реалізації

Аналізуючи тему курсової роботи, можна легко зрозуміти, що я буду писати аналог всім відомої гри «Contra», яку ви ще могли грати в дитинстві на приставці Dendy.

Чому я вибрав саме цю гру?

«Я ще пам’ятаю з дитинства, як ми з моїм братом грали в цю прекрасну гру, яка приносила море емоцій та задоволення. Ми збирались кожен вечір разом та проводили час за приставкою. Тим більше, з нами інколи розважались і батьки. А відпочинок з рідними – це завжди круто! Тому я, вмотивований спогадами, і вирішив обрати таку тему для курсової роботи. Думаю, це того варте!»

Трішки довідки:

Contra розповсюджувалася в Європі під назвою Gryzor — відеогра жанру біжи і стріляй розроблена і видана компанією Konami. Спочатку випущена для аркадних ігрових автоматів 20 лютого 1987 року. Домашня версія була як порти для різних персональних комп'ютерних систем та ігрових консолей. Домашні версії були локалізовані для регіону PAL під назвою Gryzor на різних комп'ютерних форматах, і також під назвою Probotector гра була випущена пізніше для NES. Contra є першою грою із однойменної серії.

Геймплей цієї частини став класичним для серії: персонаж гравця в більшості рівнів переміщується зліва направо, знищуючи різноманітних ворогів, що нападають з різних сторін, стаціонарні гармати, турелі, танки, падаючі гранати. Грати можна одному або вдвох за одним пристроєм. У кінці кожного рівня знаходиться свій оригінальний бос — особливо сильний і великий противник.

На рівнях можна підібрати нову зброю (яка замінює наявну), корисні бонуси, такі як захисний бар'єр чи пришвидшення стрілянини. Зброя та її поліпшення знаходяться в контейнерах, які іноді пролітають по екрану або розташованих на місці. Щоб вміст контейнера випав, його потрібно підстрелити. На початку дається 3 життя, де запасні представлено на екрані значками медалей. Можливо ще 3 рази продовжити гру в разі загибелі, але вже з 2-ма життями кожного разу. Після загибелі вся підібрана зброя і бонуси втрачаються.

Сьогодні буде дуже важко знайти консоль, щоб насолодитись смаком спогадів, тому було вирішено написати дану гру для ПК, щоб кожен користувач зміг згадати: «Яке ж воно ДИТИНСТВО!». Звичайно, багато хто вже випустив гру для Web, але в неї не можна буде порубатись без доступу в Інтернет і, до того ж, її навіть не відкриєш на ввесь екран. Треба з цим розібратись!

1.3 Розробка технічного завдання на роботу

В програмі повинно бути реалізовано: додаток Windows з графічним інтерфейсом.

Клас мікрооб’єкта повинен містити не менше 4 елементів змінних і не менше 4 методів (окрім геттерів та сеттерів). Як мінімум одна змінна повинна бути типу int, одна – типу double і як мінімум одна – рядок. Змінні-елементи класу мають бути приватними, для них повинні бути створені відповідні аксесори та мутатори (іншими словами – геттери та сетери. В класі мікрооб’єкта повинен бути присутній конструктор без аргументів. Повинно бути використано делегування конструкторів. Додати до класу мікрооб’єкта в якості елемента посилання на невеличкий службовий об’єкт, щоб зробити необхідним глибинне копіювання. Реалізовано глибинне копіювання шляхом реалізації інтерфейсу Cloneable. Повинне бути продемонстроване використання глибинного копіювання. Продемонструвати використання як мінімум двох функцій-статичних елементів класу java.util.Arrays (можливо для цього потрібно буде створити тимчасовий об’єкт-масив з колекції, яка використовується). Продемонструвати використання інтерфейсу Comparable або Comparer. Мікрооб'єкт: мінімум три графічних примітива (один з яких - текст). Більший макрооб'єкт: мінімум три графічних примітива (один з яких - текст). Менший макрооб'єкт: мінімум три графічних примітива(один з яких - текст). При натискуванні лівої кнопки миші на мікрооб’єкті він повинен ставати активним/неактивним. В програмі повинна бути реалізована активація декількох об’єктів одночасно. При натискувань клавіш-стрілок активні об’єкти/об’єкт повинні рухатись у вікні програми. При натискувань клавіши Delete активні об’єкти повинні знищуватись. Якщо активного об’єкта нема – клавіша Delete ігнорується. Клавіша Esc повинна відміняти активацію обєкта. Повина бути побудована ієрархія класів мікрооб'єктів, яка містить як мінімум три рівня наслідування, які можуть інстанціонуватись і зображатись на екрані. Зображення кожного рівня має відрізнятись від всіх інших хоча б одним графічним примітивом. При натискуванні клавіши Insert повинно з’являтись діалогове вікно, яке повинно визначати параметри створюваного мікрооб’єкта. В нього повинна бути додана можливість обирати до якого з класів нащадків у ієрархії наслідування він належить. Крім керуючого елемента Button у діалоговому вікні також повинні бути використані як мінімум три з наступного списку: TextField, CheckBox,

{ListBox, ChoiceBox}, RadioButton. Додати у проект клас/класи більшого макрооб’єкта. Більший макрооб’єкт призначений містити всі мікрооб’єкти та всі менші макрооб’єкти. У вікні програми відображається лише певна частина більшого макрооб’єкта, яка не повинна перевищувати 25% його загального розміру. Мінімальний розмір більшого макрооб’єкта 1600 позицій для мікрооб’єкта (тобто на екрані одночасно можна побачити не більше квадрата 20*20 позицій для мікрооб’єкта). Реалізувати можливість змінювати частину більшого макрооб’єкта, яку спостерігає користувач (будь яким способом). Додати у проект клас/класи меншого макрооб’єкта. Менший макрооб’єкт повинен мати можливість містити більше одного мікрооб’єкта. Не обов’язково мати можливості додавати/видаляти менші макрооб’єкти в програму. Цілком достатньо, якщо програма буде мати деяку кількість менших макрооб’єктів сгенеровану при запуску програми і яка буде незмінною протягом роботи програми. Менші макрооб’єкти повинні мати певні позиції в більшому макрооб’єкті і повинні відображатись на екраині. Позиції менших макрооб’єктів в середині більшого макрооб’єкта можуть залишатись незмінними протягом роботи програми. В більшому макрооб’єкті повинно міститись більше одного екземпляра меншого макрооб’єкта. Мікрооб’єкти можуть або належати одному (або більшій кількості) меншому макрооб’єкту, або не належати жодному. Глядачу має бути зрозуміло в чому полягає приналежність мікрооб’єкта меншому макрооб’єкту, тобто чим поведінка такого мікрооб’єкта відрізняється від поведінки мікрооб’єкта, який не належить жодному меншому макрооб’єкту. Повинна обов'язково бути релізовано можливість вилучити мікрооб'єкт з всіх менших макрооб'єктів, в результаті чого він не буде належати жодному меншому макрооб'єкту. Також повинна бути реалізована можливість вручну завести мікрооб’єкт у менший макрооб’єкт. Всі мікрооб'єкти та менші макроб'єкти повинні належати більшому макрооб'єкту і не повинні мати змоги вийти за його межі. Повинен бути реалізований автоматичний рух деяких мікрооб’єктів: 1) при русі деякі мікрооб'єкти взаємодіють між собою (це має бути помітно візуально в тому сенсі, що або щось змінюється у їх зображенні або у характері їх руху). Повинен бути реалізований автоматичний рух деяких

мікрооб’єктів: 2) деякі мікрооб'єкти заходять в менші макрооб'єкти та деякі виходять. Повинен бути реалізований автоматичний рух деяких мікрооб’єктів: 3) характер руху мікрооб'єктів повинен змінюватись при натискуванні певної клавіші клавіатури або при виборі команди меню. Під характером руху мається на увазі не просто траєкторія, а те, в який спосіб вони взаємодіють з меншими макрооб'єктами та іншими мікрооб'єктами. Повинна бути реалізована мінікарта, яка дозволяє прискорену навігацію: 1) На мінікарті повинна бути позначена видима область великого макрооб’єкта. 2) Рух видимої області великого макрооб’єкта відповідно змінює мінікарту. 3) Менші макрооб’єкти та мікрооб’єкти позначені на мінікарті. 4) Рух та взаємодія мікрооб’єктів призводить до відповідних змін на мінікарті. 5) Натискування лівої кнопки миші на мінікарті призводить до переміщення видимої області великого макрооб’єкта. Повинні бути продемонстровано використання віртуальних функцій. Повинні бути продемонстровано типу доступу protected. Повинно бути продемонстровано використання динамічного поліморфізму. Повинно бути продемонстровано використання статичного поліморфізму. Повинна бути запрограмована серіалізація/де-серіалізація всіх об’єктів у текстовий/бінарний/XML файл. Серіалізація/де-серіалізація обов’язково повинна зберігати не тільки власне інформацію про стан макро- та мікро-об’єктів, але й про їх позицію на екрані. При серіалізації/де-серіалізації обов’язково повинні використовуватись діалогові вікна, щоб запитати у користувача ім’я файлу та його розташування на диску (наприклад можна використовувати системне діалогове вікно FileChooser).

Для нормального виконання програми необхідно наступне апаратне забезпечення (мінімум):

- комп’ютер від фірм Asus, Lenovo, Samsung і тд.
- 4 ГБ оперативної пам’яті;
- NVIDIA GeForce 940MX
- відеокарта об’ємом пам’яті не менше 1ГБ;
- клавіатура, ОС Windows 10/8/7.

- Розмір дискового простору, що займає програма: 300 000 Кб.
- Розмір оперативної пам’яті, що займає програма: 30 000 Кб.

1.4 Обгрунтування вибору мови програмування

Яку ж мову вибрати для написання гри? Звичайно ж, Java. Чому? Якщо ви прочитаєте деякі форуми і статті про розробку ігор, у вас може скластися враження, що Java не є хорошим вибором для цього. Однак ця думка дещо застаріла і є неточною. Світ відеоігор дуже різноманітний. Умовно ви можете розділити ігри на наступні категорії: [2]

«Великі» ігри. Такі як 3D-шутери, масштабні екшн-пригоди / action RPG. До цієї категорії також входять проекти рівня AAA. Зазвичай, це високобюджетні ігри, призначені для масової аудиторії, такі як Red Dead Redemption 2, Assassin Creed Origins і так далі. Ігри такого типу, зазвичай, написані на C ++ і супроводжуються ігровими движками. Java є рідкісним гостем в розробці ігор такого типу через особливості JVM. Точніше, його можна використовувати, наприклад, для створення внутрішніх деталей.

Інді-ігри. Один геймер відноситься до інді-проектам з презирством, а інший любить їх з пристрастю. Інді-гра означає незалежну гру, створену невеликою командою або навіть одним розробником.

Серед інді багато аматорських проектів, деякі з них не приносять грошей своїм творцям. Інді більше про оригінальну ідею і ігровий процеси, хороші сценарії і суб'єктивне бачення, а не про продуктивність і високоякісну графіку.

Проте, такі інді-проекти прориваються на ринок і стають мегапопулярними. Хорошим прикладом є Minecraft, спочатку створений однією людиною, Маркусом Перссоном. Станом на травень 2019 року було продано понад 176 мільйонів копій Minecraft. Це робить її однією з найбільш продаваних відеоігор всіх часів. У 2014 році Microsoft купила Minecraft і Mojang за 2,5 мільярда доларів США. До речі, Маркус Перссон є Java-розробником, тому він написав гру на цій мові. Так що, так, Java дійсно хороший для інді-проектів.

Чому не C++ ?

Якщо ж порівнювати з C++, то ця мова є значно застарілою на цей час, і тому навіть такі бібліотеки, як SFML уже не зможуть виконати усі ті завдання, які потрібно. Маючи досвід програмування на цій мові, було прийняте рішення обрати щось сучасніше. Тим паче, що зараз є така прекрасна альтернатива, як Java. Тому варто одразу відкинути цей варіант.

Що ж стосовно C# ? [3]

Повністю з вами згоден, що сьогодні більшість ігор пишеться саме на мові програмування C#, але це не дає змоги нам відкинути інші мови. Розглянемо переваги Java:

- Сувора статична типізація – вона значно зменшує кількість помилок і покращує можливість підтримки коду, особливо при використанні статичного аналізатора.
- Велика історія - мова Java існує на ринку з 1995 року, тому сьогодні він широко вивчений і пропонує сотні рішень і технологій.
- Спрощена версія C # - всі розробники, які вивчають C ++ і Java, дуже близькі до цієї мови. Java вивчається на основі C ++ і не є складною. Більш того, Java є простим, передбачуваним і мінімалістичним. Нові функції представлені дуже ґрунтовно.(А так, як я вже вивчав C++ в університеті, то мені буде набагато простіше програмувати саме на цій мові).

Тому ці факти дозволяють мені відкинути C# та навіть не задумуватись у своєму виборі.

У висновку хочу сказати, що Java хороший для розробки мобільних ігор і інді-проектів (ці ринки кілька перетинаються) і для розробки серверної частини високонавантажених онлайн-ігор.

Більш того, Java є універсальною мовою. Вона використовується при розробці на стороні сервера, в великих даних, мобільних і веб-додатках, торгових додатках, вбудованому просторі і т.д. Отже, Java дає вам свободу вибору, може бути, навіть

більше, ніж будь-яка інший мова програмування і саме цю мову я обираю для написання курсової роботи та й подальшого розвитку в сфері програмування.

1.5 Висновки

Пройшовши через усі етапи, можна зробити висновок, що тема курсової є досить цікавою, і вона сподобається кожному. Головна мета цієї гри – принести задоволення та приплив спогадів з минулого, доторкнутись до дитинства і побувати там, хоч під час гри.

Гра прийдеться до душі, як і малим, так і дорослим, тому певної вікової категорії тут немає. Кожен може обрати все на свій смак: зброю, рівень та навіть кількість життів. Завдяки цьому усі зможуть знайти щось своє. Пройшовши всі етапи, гравець має отримати всі відповіді на свої питання, адже сюжет також буде присутній. Незважаючи на те, що зараз на світовому ринку є досить круті ігри з просто нереальною графікою, але, вважаю, що саме цей витвір не залишить користувачів байдужими!

2 РОЗРОБКА ІНТЕРФЕЙСУ ПРОГРАМИ ТА ПРИКЛАДІВ ВИКОРИСТАННЯ

2.1 Розробка інтерфейсу програми

Увесь інтерфейс гри реалізується на базі операційної системи Windows, що дає змогу запустити даний додаток на будь-якому ПК, де встановлена ОС. Розробка програми відбувається в середовищі IntelliJ IDEA на базі мови програмування Java. Інтерфейс програми віконний, тому відкривається, як звичайна програма в ОС Windows. Користувач починає знайомство з додатком з його запуску. Достатньо просто встановити усі необідні компоненти та запустити. Якщо ж встановлені не всі компоненти, то на екран буде виведено повідомлення з помилкою, яку потрібно виправити. Кожний компонент є надзвичайно важливим для правильного функціонування програми, так як усе зв’язано досить тісно.

Після запуску програми відкривається вікно з інтерфейсом програми.

Структура вікна (*рис 2.1*):

- Назва вікна – Contra
- Робоча область – великий макрооб’єкт
- Міні-карта – лівий верхній куток
- Мікрооб’єкти – Hero, Ninja, Hunter
- Макрооб’єкти – Hero camp, Ninja Camp, Hunter Camp
- Активні клавіші – Insert, Delete, Ecs, C, T

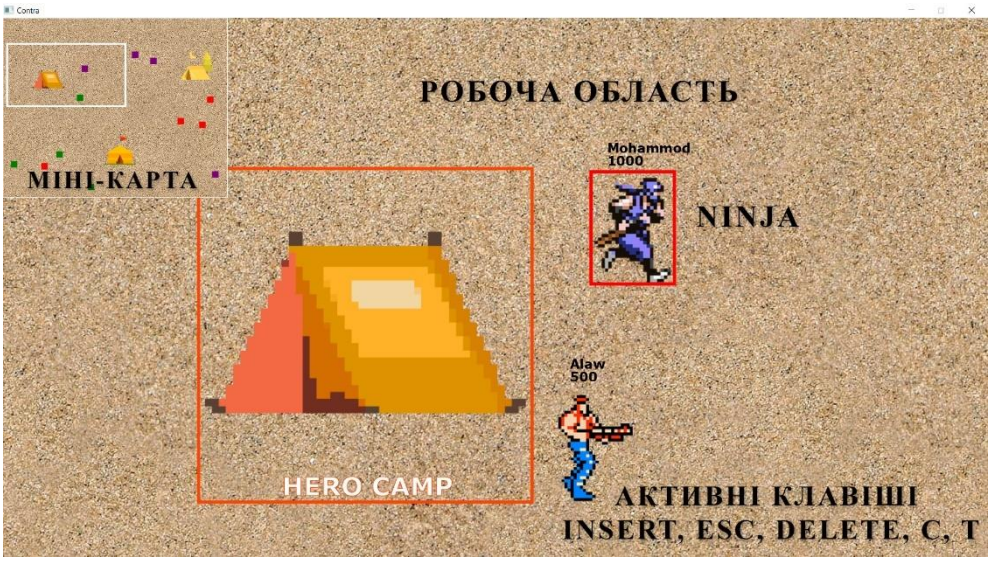

Рисунок 2.1 – Інтерфейс програми

Робоча область вікна слугує для відображення інформаційних об’єктів: карти світу, карти рівня та героїв. Мікрооб’єкти відображаються за допомогою трьох графічних примітивів: зображення, текст та форма. Малий макрооб’єкт відображується за допомогою зображення, тексту та форми. При натискуванні лівої кнопки миші на міні-карту здійснюється пересування основного вікна на вказану координату. При пересуванні камери стрілками змінюється й видима область вікна. При натисканні лівої кнопки миші на героя об’єкт стає, або активним, або неактивним (виділяється за допомогою червоного прямокутника).

2.2 Приклади використання

Приклади використання наведені в цьому розділі надають змогу розробнику покращувати свою програму та підтримувати її на високому рівні. Це розкриває можливості взаємодії користувача з програмою та надає варіанти її покращення.

Приклад 1. Створення нового героя

Рисунок 2.2 – Створення нового персонажа

Натискаючи на клавішу Insert відкривається діалогове вікно з можливістю додавання нового героя на карту. У вікні можна задати такі характеристики герою, як ім’я, тип, випадковий бонус. Також можна побачити, як герой буде виглядати. Після натискування клавіші Cancel у діалоговому вікні, усі введені характеристики будуть видалені та вікно буде закрите. Після натискуванні клавіші OK буду створено нового персонажа в основному вікні гри з введеними характеристиками.

Якщо ж не всі характеристики будуть введені, або будуть введені з помилками, то на екрані з’явиться вікно з повідомленням про помилку введення. При закритті діалогового вікна на хрестик програма продовжить свою роботу в звичайному режимі. Усі вище наведені характеристики зображенні на рисунку 2.2.

Приклад 2. Бій

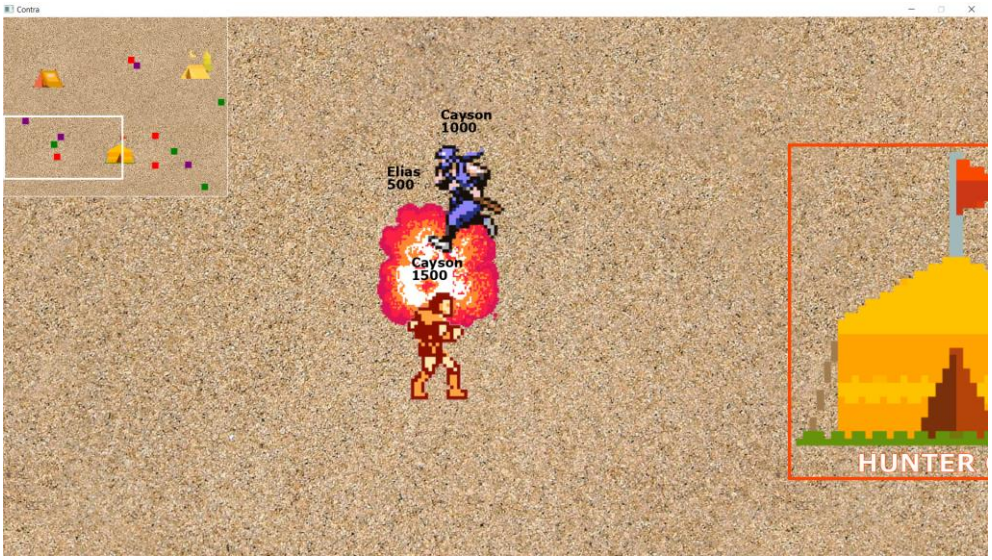

Рисунок 2.3 – Бій

Під час запуску програми на карті з’являється 10 нових героїв, які ворогують між собою. У грі існує три рівня героїв: Hero, Ninja та Hunter. Кожен з ним має різні характеристики. Коли координати зображення одного героя пересікаються з координатами зображення іншого героя, то між ними відбувається бій. На карті з’являється зображення бою у вигляді вибухів. Після програшу одного з героїв зникає його зображення, а переможець вирушає у свій барак відновлювати здоров’я на максимум. Після відновлення здоров’я герой повертається на карту. Зображення бою героїв представлене на рисунку 2.3.

Приклад 3. Відновлення здоров’я

Після успішного бою переможець повертається у свій барак для відновлення здоров’я. Там він проводить деякий час, відновлюючи свої показники та після цього повертається назад на карту. Виділивши героя, його можна віднести до бараку самостійно за допомогою стрілок на клавіатурі. Якщо ж герой має максимальне здоров’я, то він не зможе зайти до казарми.

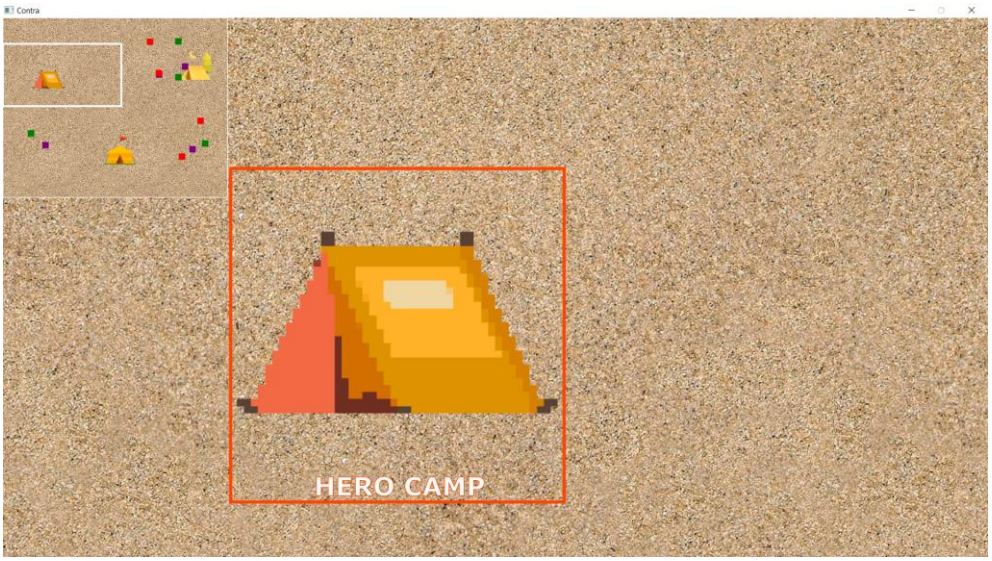

Рисунок 2.4 – Відновлення здоров’я героя

Приклад 4. Повернення до точки збору

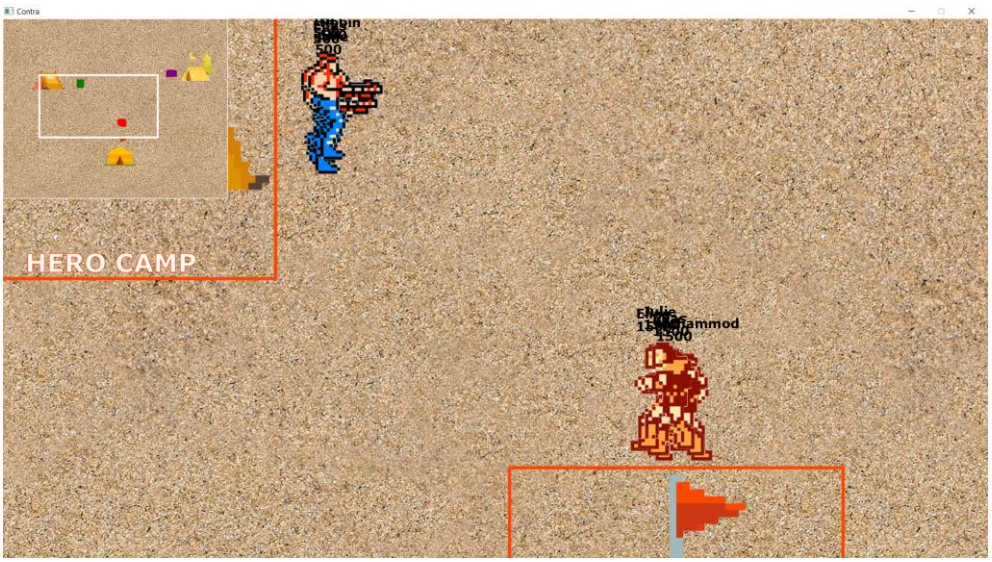

Рисунок 2.5 – Збір героїв

Після запуску програми герої з’являються на випадкових позиціях та починають своє життя на карті. Кожен герой може взаємодіяти з ворогами, але при натисканні на клавішу T усі персонажі починають рухатись у точку збору, яке знаходить біля бараків. Ця дія працює, поки користувач ще раз не натисне на клавішу T на клавіатурі.

3 РОЗРОБКА ПІДСИСТЕМИ ГРАФІЧНОГО ЗООБРАЖЕННЯ

3.1 Модель графічного відображення

Кожен об’єкт в роботі складається з декількох графічних примітивів, що разом створюють повноцінний образ, яким можна керувати. За допомогою такого прийому продукт стає більш зрозумілішим та простішим для використання. Нижче наведена детальна характеристика мікро- та макрооб’єктів.

3.1.1 Мікрооб’єкт

Мікрооб’єкт в програмі складається із п’яти графічних примітивів (*рис. 3.1*):

- текст імені – Text heroNameText
- текст здоров’я – Text heroHealthText
- зображення – ImageView imageViewOfHero
- іконка активності – ImageView imageViewOfActiveIcon
- видима область – Rectangle rectangle

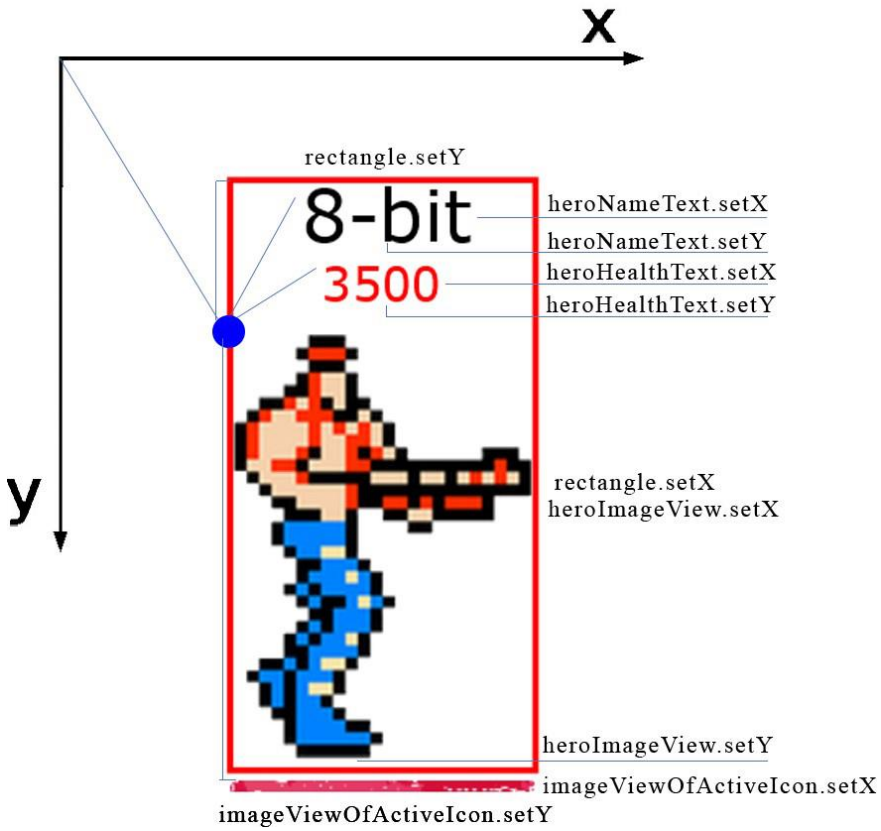

Рисунок 3.1 - Графічні примітиви мікрооб’єкта

Координати heroImageView вираховуються відносно початку системи координат (x = 0, y=0), у свою чергу heroNameText, heroHealthText, imageViewOfActiveIcon та rectangle вираховуються уже відносно heroImageView, а саме верхнього лівого кута (див. рис 3.1). Це дає змогу одночасно переміщати усі графічні примітиви та рухати об’єкт.

3.1.2 Великий макрооб’єкт

Великий макрооб’єкт являє собою карту світу, на якій знаходять усі інші об’єкти. Іншими словами, це просто фон для програми, на якому будуть відбуватись усі процеси. Великий макрооб’єкт задається за допомогою зображення. Ширина зображення, яке містить у собі макрооб’єкт, дорівнює 3000 пікселів, а висота – 2250.

Рисунок 3.2 – Великий макрооб’єкт

3.1.3 Малий макрооб’єкт

Малий макрооб’єкт складається із трьох графічний примітивів: текст, форма та зображення. Він належить більшому макрооб’єкту та розташований на ньому. Координати розташування макрооб’єкта вираховуються відносно початку вісі координат. На карті розташовано три макрооб’єкта: heroCamp, ninjaCamp, hunterCamp. Кожен з них знаходить в окремому класі, який міститиме менші мікрооб’єкти. Зображення макрооб’єкта представлено на рисунку 3.3.

Рисунок 3.3 – Малий макрооб’єкт

3.2 Графічні процедури підсистеми графічного відображення

Основним класом для відображення всього графічного контенту є class Scene (java.scene.Scene). Він представляє собою контейнер для усіх графічних елементів всередині об’єкта Stage (Image, ImageView, Text, Button та ін.), який називається Stage primaryStage. [4] Stage у своє чергу є точкою входу програми. Він представляє собою головне вікно та передається в якості аргументу метода start(). Також для відображення створюється об’єкт класу Group group. Цей клас також являється контейнером для графічних примітивів, який не приймає розміщення (Layout) для своїх підкомпонентів. Його використання дає змогу накладати об’єкти та переміщати їх як завгодно. Ціль Group – згрупувати Control в одну групу і виконати певну задачу. [5] У моєму випадку було згруповано усі графічні примітиви класу Hero, Ninja та Hunter (*рис. 3.4*). Тут зображено групування об’єктів в класі Main. Також в програмі було створено ще один Group group2 для діалогового вікна. Схема групування графічних примітивів показана на рисунку 3.5. Мікрооб’єкт у курсовій роботі малюється за допомогою методу drawHero(). Ця функція приймає так параметри, як String name, int health, boolean isActive, double XPosition, double YPosition.

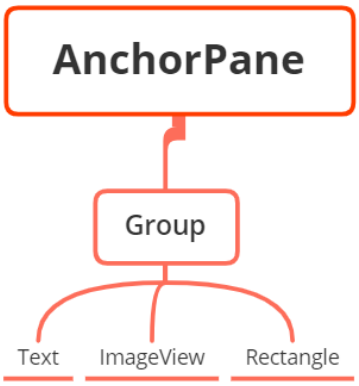

Рисунок 3.4 - Scene scene (головне вікно)

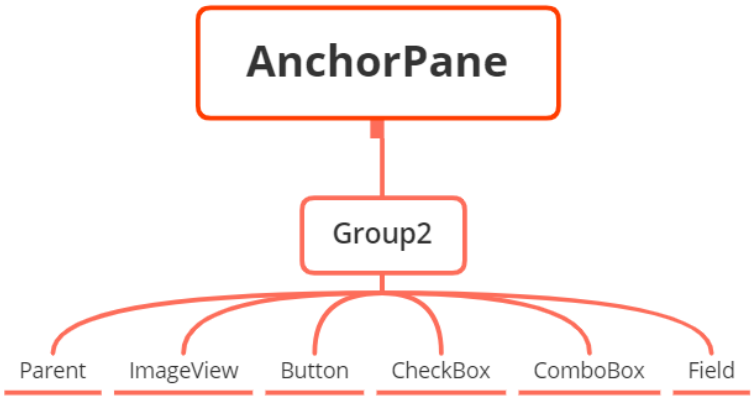

Рисунок 3.5 - Scene scene (діалогове вікно)

Цей метод призначений для того, щоб створити усі графічні примітиви, загрузити усі зображення, встановити координати елементів відносно один одного та додати усе до group, за допомогою якого й буде виведена вся графіка на екран.

4 РОЗРОБКА ДІАГРАМ КЛАСІВ, ОБ'ЄКТІВ ТА СТАНУ

4.1 Діаграми класів

4.1.1 Діаграми наслідування

Клас Hero містить у собі логіку програми, параметри об’єктів та є першим класом ієрархії, яких є три Hero – Ninja – Hunter. Методи класу Hero є віртуальними.

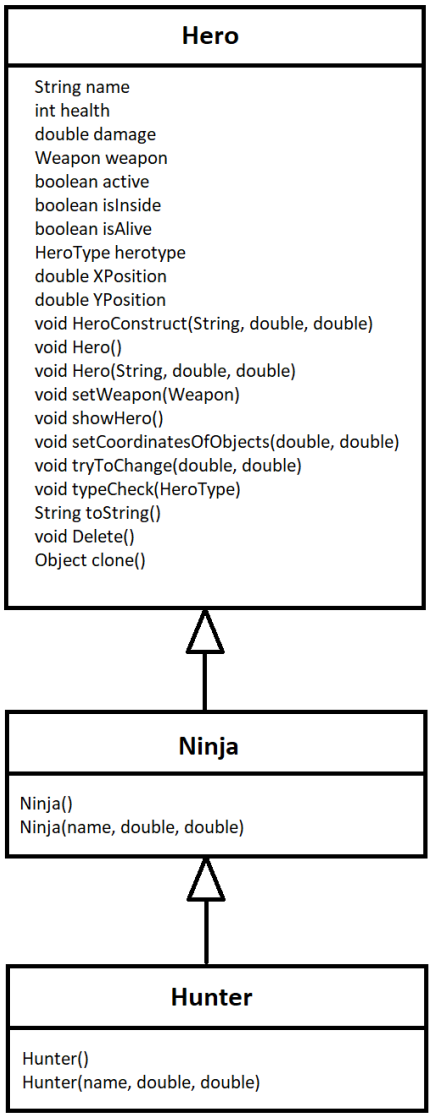

Рисунок 4.1 – Діаграма наслідування

Змінні класу:

- Name – ім’я героя
- Health – здоров’я героя
- Damage – шкода зброї
- Active – активність героя

- isInside – чи герой всередині макрооб’єкта
- process – чи герой зайнятий
- isAlive – чи герой живий
- HeroType – тип героя
- XPosition – позиція героя по X
- YPosition – позиція героя по Y

Віртуальні методи класу Hero:

- HeroConstruct()
- Hero()
- Hero(String name, double x, double y)
- setWeapon(weapon)
- showHero()
- setCoordinatesOfObjects(double x, double y)
- tryToChange(double x, double y)
- typoCheck(HeroType herotype)
- toStrigh()
- Delete()
- clone()

4.1.2 Діаграми агрегації

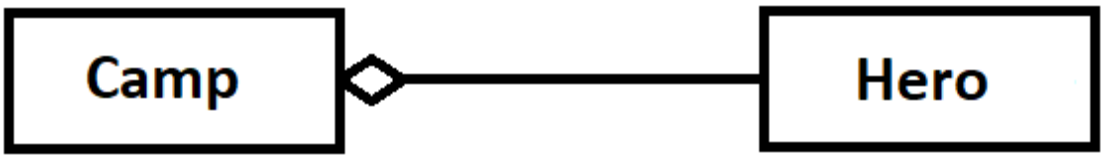

Рисунок 4.2 – Діаграма агрегації (барак - герой)

Різниця між композицією і агрегацією полягає в тому, що в разі композиції ціле явно контролює час життя своєї складової частини (частина не існує без цілого), а в разі агрегації ціле хоч і містить свою складову частину, але їх життя не пов'язане (наприклад, складова частина передається через параметри конструктора).

4.1.3 Діаграми композиції

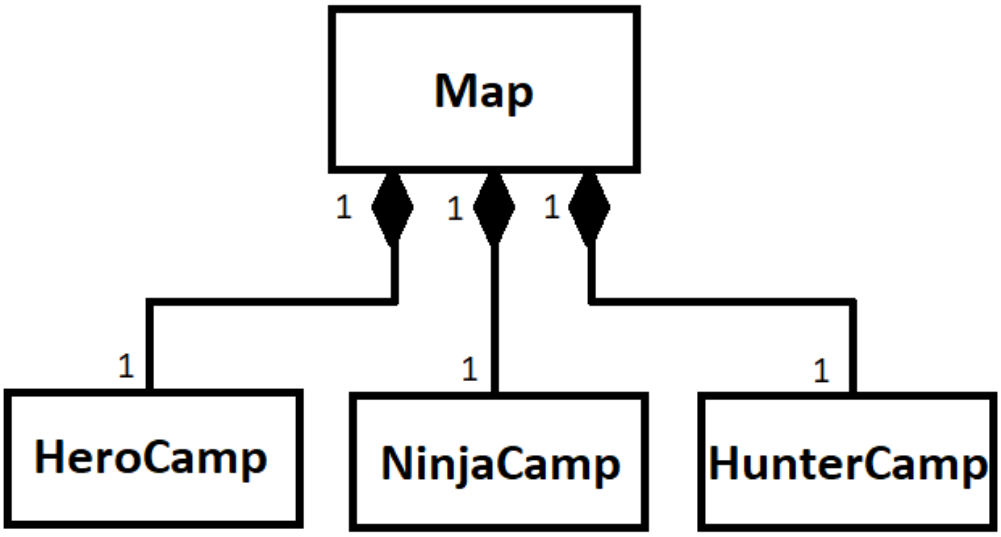

Рисунок 4.3 – Діаграма композиції більшого та меншого макрооб’єктів

4.1.4 Діаграми асоціації

Рисунок 4.4 – Діаграма асоціації

4.2 Діаграми кооперації

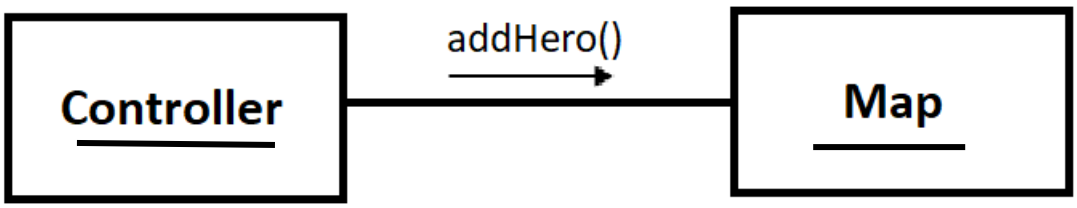

Рисунок 4.5 – Діаграма кооперації (створення нового об’єкту)

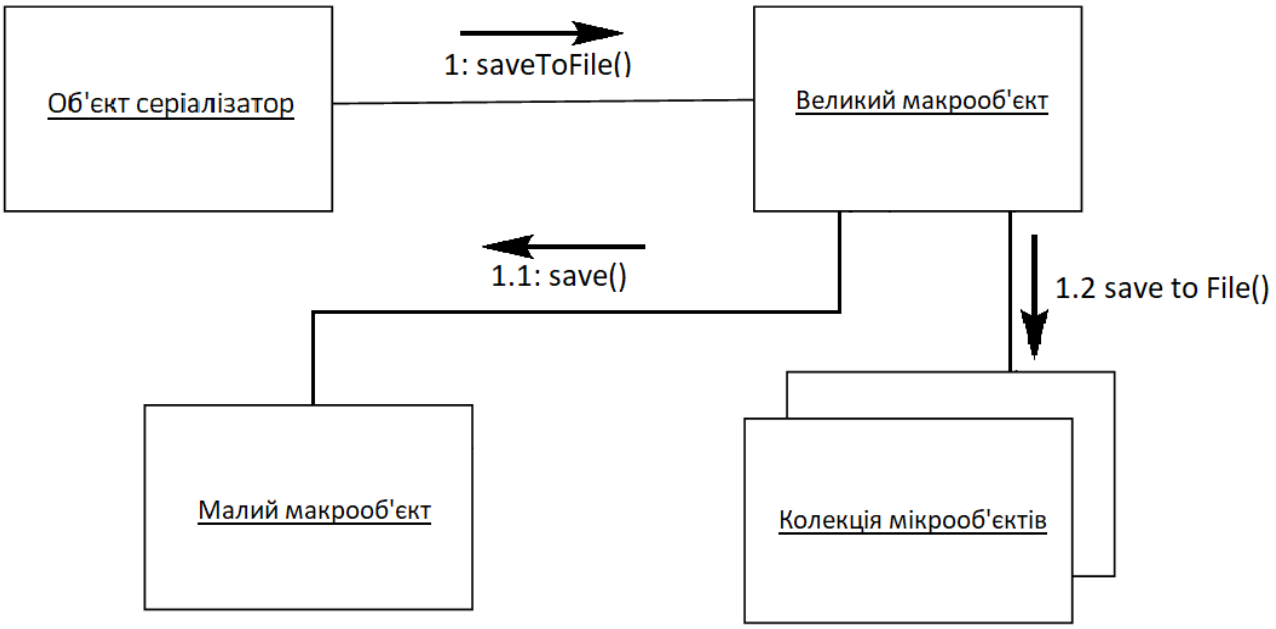

Рисунок 4.6 – Ієрархічна операція

4.3 Діаграми послідовності

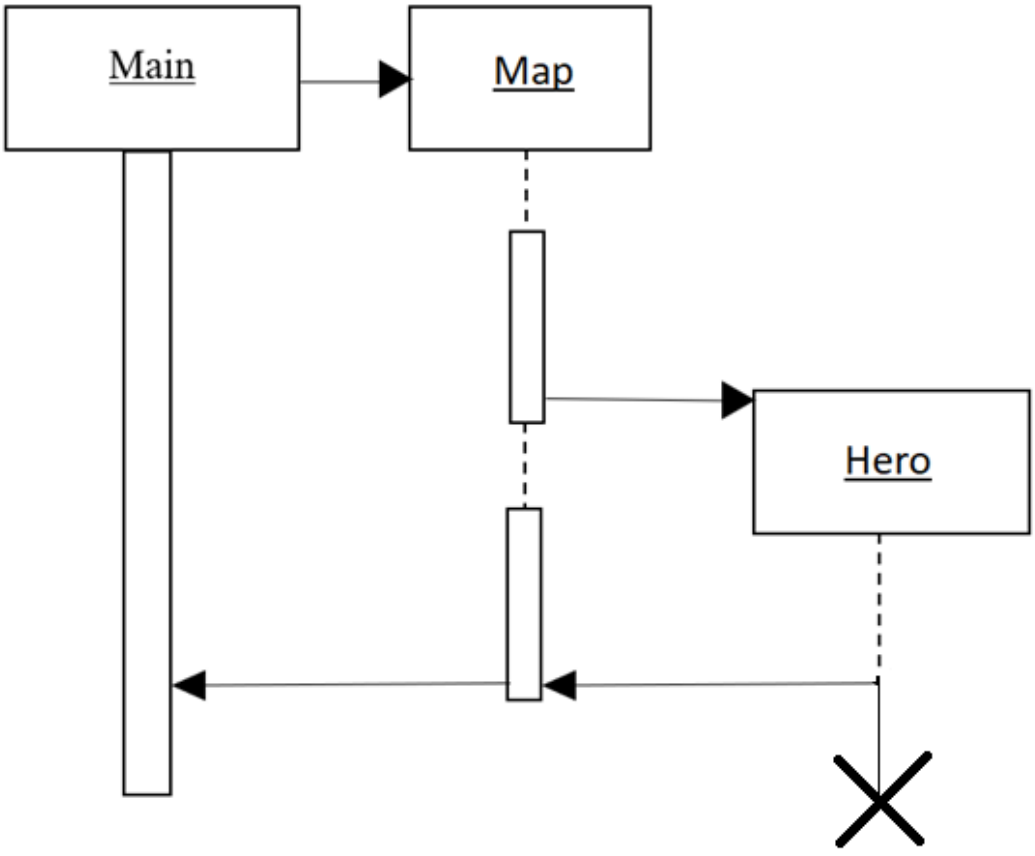

Рисунок 4.7 – Діаграма послідовності

4.4 Діаграми стану

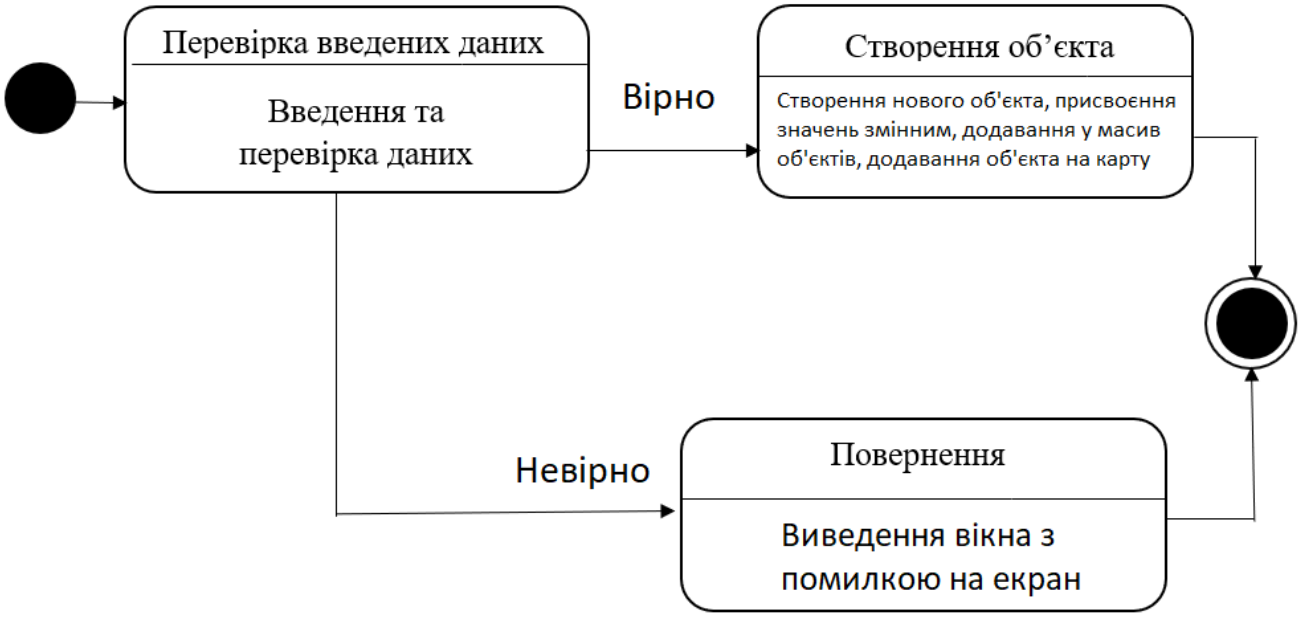

Рисунок 4.8 – Діаграма стану для створення об’єкта

5 ВИКОРИСТАННЯ ЗАСОБІВ ПРОГРАМУВАННЯ JAVAFX

Обираючи засоби програмування графічних програм, було прийнято рішення використовувати саме графічну бібліотеку JavaFX. У курсовій роботі використано такі методи, як пересування героя по карті, обробка клавіш, обробка натискувань миші, перевірка на активність героя, видалення героя зі сцени та ін.

5.1 Пересування героя по карті – moveHero()

Цей метод дозволяє переміщувати усі графічні примітиви одночасно відносно один одного. Приймає координати, які встановлюються за допомогою метода handle(), та корегує положення графічних об’єктів відносно параметрів. Тут використано звичайне наслідування класів бібліотек JavaFX, а зокрема оператори методи setX(), setY(), getX() та getY(), та такі стандартні класи бібліотек JavaFX, як:

- javafx.scene.image.Image,
- javafx.scene.image.ImageView,
- javafx.scene.shape.Rectangle,
- javafx.scene.text.Text.

Код методу представлений на рисунку 5.1.

```
public void moveHero(double XPosition, double YPosition) {
    if ((rectangle.getX() + XPosition) >= 0 && (rectangle.getY() + YPosition >= 0) &&
(rectangle.getX() + XPosition + 80) <= 1200 && (rectangle.getY() + YPosition + 200) < 700)
{

        heroNameText.setX(heroNameText.getX() + XPosition);
        heroNameText.setY(heroNameText.getY() + YPosition);

        imageViewOfHero.setX(imageViewOfHero.getX() + XPosition);
        imageViewOfHero.setY(imageViewOfHero.getY() + YPosition);

        heroHealthText.setX(heroHealthText.getX() + XPosition);
        heroHealthText.setY(heroHealthText.getY() + YPosition);

        imageViewOfActiveIcon.setX(imageViewOfActiveIcon.getX() + XPosition);
        imageViewOfActiveIcon.setY(imageViewOfActiveIcon.getY() + YPosition);

        rectangle.setX(rectangle.getX() + XPosition);
        rectangle.setY(rectangle.getY() + YPosition);
    }}
```

Рисунок 5.1 - Графічні примітиви мікрооб’єкта

5.2 Обробка клавіш – handle(KeyEvent event);

Цей метод використовується для обробки натискування клавіш клавіатури та подальшого керування програмою. Функція отримує код натиснення клавіші та вирішує, що потрібно зробити в даний момент. Наприклад, натиснувши клавішу, UP мікрооб'єкт почне рухати вгору, а якщо ж Insert, то відкриється діалогове вікно, де можна створити новий мікрооб'єкт. Використанні такі стандартні класи бібліотек JavaFX, як javafx.event.EventHandler та javafx.scene.input.KeyEvent. Код методу представлений на рисунку 5.2.

```
scene.setOnKeyPressed(new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent event) {
        double x = 0;
        double y = 0;

        double delta = 10;

        switch (event.getCode()) {
            case UP:
                y -= delta;
                for (int i = 0; i < heroList.size(); i++) {
                    if (heroList.get(i).isActive()) {
                        heroList.get(i).moveHero(x, y);
                    }}
                break;
            case DOWN:
                y += delta;
                for (int i = 0; i < heroList.size(); i++) {
                    if (heroList.get(i).isActive()) {
                        heroList.get(i).moveHero(x, y);
                    }}
                break;
            case RIGHT:
                x += delta;
                for (int i = 0; i < heroList.size(); i++) {
                    if (heroList.get(i).isActive()) {
                        heroList.get(i).imageViewOfHero.setScaleX(1.0);
                        heroList.get(i).moveHero(x, y);
                    } }
                break;
            case LEFT:
                x -= delta;
                for (int i = 0; i < heroList.size(); i++) {
                    if (heroList.get(i).isActive()) {
                        heroList.get(i).imageViewOfHero.setScaleX(-1.0);
                        heroList.get(i).moveHero(x, y);
                    }}
                break;
```

Рисунок 5.2 - метод handle()

```
case INSERT:
                try {
                    DialogWindow.display();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;

            case ESCAPE:
                for (int i = 0; i < heroList.size(); i++) {
                    if (heroList.get(i).isActive()) {
                        heroList.get(i).setActive(false);
                        Main.group.getChildren().remove(heroList.get(i).getRectActive());
                        Main.group.getChildren().remove(heroList.get(i).getActiveIcon());
                    }
                }
                break;
            case DELETE:
                for (int i = 0; i < heroList.size(); i++) {
                    if (heroList.get(i).isActive()) {
                        heroList.get(i).removeColonist();
                    }
                }
                break;
        }}});
```

Рисунок 5.2 (продовження)

5.3 Обробка натискувань миші – handle(MouseEvent event)

Цей метод використовується для обробки натискування миші та подальшого керування програмою. Функція отримує координати натиснення миші та вирішує, що потрібно зробити в даний момент. У програмі реалізовано активування/деактивування мікрооб’єкта за допомогою натиснення клавіш миші. Це дає змогу керувати одразу кількома об’єктами, тільки одним, або ж навіть жодним. Код методу представлений на рисунку 5.3.

```
scene.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        System.out.printf("Mouse event %f %f scene  %f %f screen %f %f\n",
                event.getX(), event.getY(),
                event.getSceneX(), event.getSceneY(),
                event.getScreenX(), event.getScreenY());
        for (int i = 0; i < heroList.size(); i++) {
            heroList.get(i).tryToActivate(event.getX(), event.getY());
        }}});
```

Рисунок 5.3 - Метод handle()(обробка миші)

6 РОЗРОБКА ПІДСИСТЕМИ СЕРІАЛІЗАЦІЇ/ДЕСЕРІАЛІЗАЦІЇ ДАНИХ

6.1 Розробка формату файлу

Проблема серіалізації та десеріалізації у сучасному світі є досить актуальним питанням, тому що вона використовується в кожній комп’ютерній грі. Для початку, розберемось із цими поняттями.

Серіалізація представляє процес запису стану об'єкта в потік, відповідно процес вилучення або відновлення стану об'єкта з потоку називається **десеріалізацією.** Серіалізація використовується для збереження даних програми в окремий файл, щоб в подальшому відновити роботу продукту в конкретній встановленій точці. [6]

Якщо ж згадати 90-ті роки, то тоді проблема збереження даних стояла дуже гостро, так як будь-яку гру приходилось проходити з самого початку кожного разу. Тому й було вирішено використовувати серіалізацію для подальшого відновлення даних програми. Уся інформація записується в тестовий файл та зберігається в пам’яті комп’ютера.

Формат файлу

10	Кількість героїв
Ruslan	Ім’я героя
1000	Здоров’я
20	Шкода
false	Чи зайнятий
false	Чи активний
false	Чи всередині
250	Координата x
500	Координата y

Рисунок 6.1 – Формат файлу

Чому ж саме текстовий файл?

У процесі розробки було прийнято рішення зберігати дані саме в текстовому файлі для максимальної синхронізації із усіма ПК. Цей формат займає мало пам’яті на диску та може бути відкритий на будь-якому пристрої, навіть мобільному. До того ж, це дозволяє створювати стільки точок збереження, скільки потрібно. Саме текстовий файл дозволяє просто і швидко відновити програму та продовжити її роботу.

6.2 Механізми введення виведення мови Java

За допомогою FileWriter ми можемо створювати файли. (*рис. 6.2*) Клас FileWriter є похідним від класу Writer. Він використовується для запису текстових файлів. Так, в конструктор передається або шлях до файлу у вигляді рядка, або об'єкт File, який посилається на конкретний текстовий файл.

Рисунок 6.2 – FileWriter

За допомогою FileReader ми можемо зчитувати файли. (*рис. 6.3*)

Рисунок 6.3 - FileReader

7 КЕРІВНИЦТВО КОРИСТУВАЧА

Керівництво користувача є одним з найважливіших пунктів цієї курсової роботи. Завдяки цьому користувач одразу може дізнатись, як правильно встановити програму та як нею користуватись.

Встановлення та перший запуск

Перед початком користування програмою потрібно становити усі необхідні компоненти для ОС Windows. Так як розробка проходила в середовищі InteliJ IDEA, то цю програму потрібно скачати з офіційного сайту [7] та інсталювати на свій ПК (*рис. 7.1*).

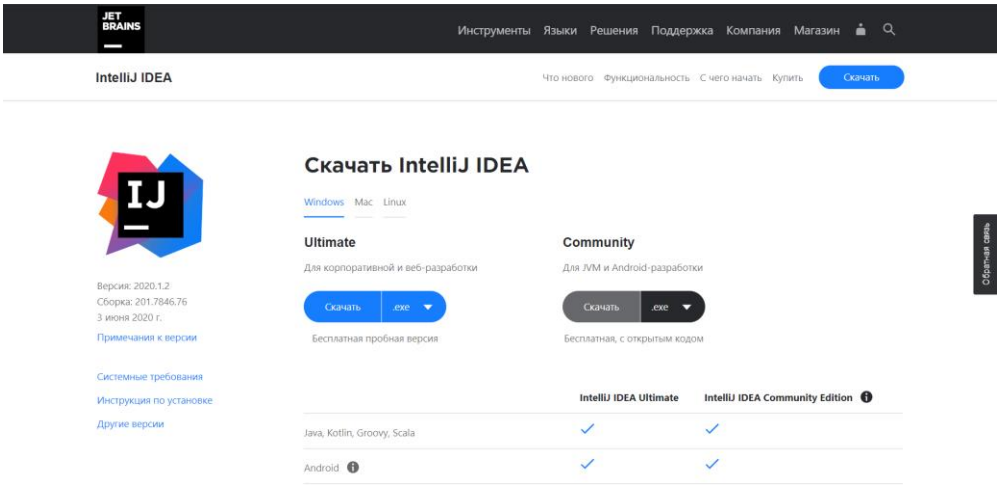

Рисунок 7.1 – Скачування IntelliJ IDEA

Після цього потрібно встановити усі компоненти JDK 11, які також можна знайти на офіційному сайті. [8]

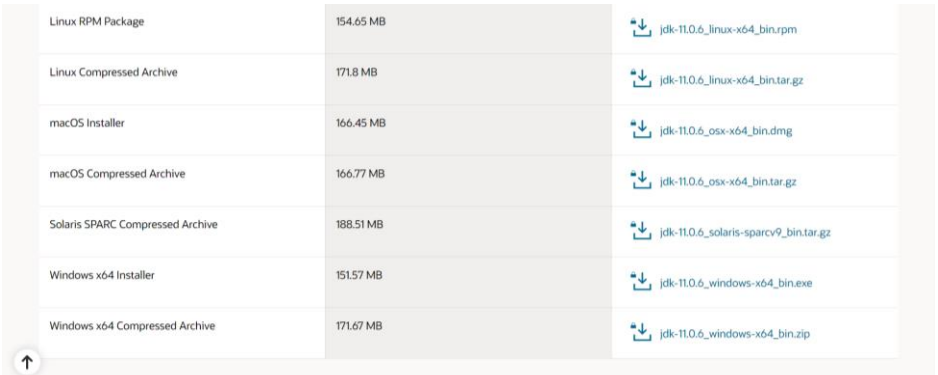

Рисунок 7.2 – Скачування SDK

Також для роботи програми необхідне інсталювання SDK. [9] Після успішного встановлення усіх компонентів потрібно розпакувати архів з проектом гри CourseWork.zip. Та відкрити його за допомогою IntelliJ IDEA. Запустити програму.

Клавіші управління:

- UP / DOWN / LEFT / RIGHT – переміщення героїв та камери
- Insert – додавання нового героя (*рис. 7.3*)

Рисунок 7.3 – Створення нового персонажа

- Delete – видалення виділених героїв
- Esc – зняття виділення усіх героїв
- T – зібрання героїв (*рис. 7.4*)

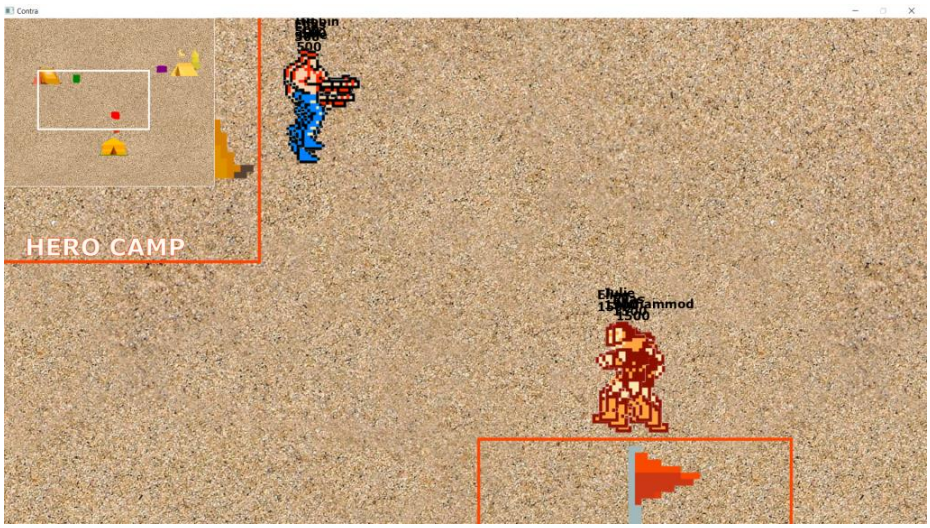

Рисунок 7.4 – Збір героїв

- S – збереження гри
- L – завантаження збереженої гри

Після завершення гри, її потрібно зберегти у файл та закрити за допомогою хрестика у верхньому правому куртку та закрити середу розробки IntelliJ IDEA. Будьте обережним, щоб випадково не пошкодити код та не зламати правильне функціонування програми.

ВИСНОВКИ

В процесі роботи над курсовою роботою було створено готовий програмний продукт, яким може скористатись кожен, а саме гру під назвою «Contra». Програма була написана в середовищі розробки IntelliJ IDEA від Jet Brains, яка дозволяє пришвидшити розробку ПЗ в кілька разів та полегшити її. Продукт був створений на базі мови програмування Java, а зокрема графічних бібліотек JavaFX.

Ця гра дає змогу побувати у світі героїв, які ворогують між собою та приєднатись до битв. Тут кожен зможе відчути себе справжнім воїном та побувати у цікавому світі постійних битв.

Під час написання курсової роботи були покращенні знання мови програмування Java, вивчені такі механізми, як віконний інтерфейс програм, зчитування та обробка натискувань клавіатурі та миші, інтерфейси Cloneable та Comparable, серіалізація та ін. Було ознайомлено і вивчено значну частину графічних бібліотек JavaFX та успішно використано їх в процесі розробки програми.

ПЕРЕЛІК ПОСИЛАНЬ

1. Contra

URL: https://uk.wikipedia.org/wiki/Contra (дата звернення 29.03.2020).

2. Java для игр

URL: https://senior.ua/articles/sozdanie-javaigry-s-chego-nachat (дата звернення 29.03.2020).

3. C# против Java

URL: https://itvdn.com/ru/blog/article/csharp_vs_java (дата звернення 29.03.2020).

4. Руководство JavaFX Scene

URL: https://metanit.com/java/javafx/1.5.php (дата звернення 18.04.2020).

5. Руководство JavaFX Group

URL: https://o7planning.org/ru/11103/javafx-group-tutorial (дата звернення 18.04.2020).

6. Сериализация

URL: https://metanit.com/java/tutorial/6.10.php (дата звернення 05.06.2020).

7. InteliJ IDEA

URL: https://www.jetbrains.com/ru-ru/idea/(дата звернення 10.06.2020).

8. JDK 11

URL: https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html (дата звернення: 10.06.2020).

9. SDK

URL: https://www.oracle.com/java/technologies/install-javafx-sdk.html (дата звернення: 10.06.2020).

ДОДАТОК А

Лістинг програми

Модуль Main.java

```
package sample;

import javafx.animation.AnimationTimer;
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;



import java.io.IOException;

public class Main extends Application {

    public double XPointer = 0;
    public double YPointer = 0;
    private static double screenMove = 50.0;
    private boolean createStatus = false;

    public static Group group;
    public static Group minimap;
    public static Scene scene;
    public static Map mapObject;

    @Override
    public void start(Stage primaryStage) throws Exception {
```

```
        group = new Group();
        minimap = new Group();

        Image imageHeroCamp = new Image("images/camp4.png", 500, 500, false, false);
        Image imageNinjaCamp = new Image("images/camp5.png", 500, 500, false, false);
        Image imageHunterCamp = new Image("images/camp6.png", 500, 500, false, false);
        Image imageMiniHeroCamp = new Image("images/camp4.png", 50, 50, false, false);
        Image imageMiniNinjaCamp = new Image("images/camp5.png", 50, 50, false, false);
        Image imageMiniHunterCamp = new Image("images/camp6.png", 50, 50, false, false);
        Hero.heroImage = new Image("images/hero1.png", 120, 168, false, false);
        Ninja.NinjaImage = new Image("images/hero3.png", 120, 168, false, false);
        Hunter.HunterImage = new Image("images/hero2.png", 120, 168, false, false);

        mapObject = new Map(imageHeroCamp, imageNinjaCamp, imageHunterCamp);

        primaryStage.setTitle("Contra");
        primaryStage.setMaximized(true);
        primaryStage.setResizable(false);
        scene = new Scene(group, 1280, 720);
        primaryStage.setScene(scene);

        scene.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                Map.isPressed(event.getX() + XPointer, event.getY() + YPointer);
                System.out.println("x = " + event.getX() + XPointer + " y = " + event.getY() + YPointer);

                double XPositionOfLocation = 0;
                double YPositionOfLocation = 0;

                if (event.getX() >= 0 && event.getX() < 350 && event.getY() >= 0 && event.getY() <=
280) {
                    XPositionOfLocation = -(event.getX()) / 0.126;
                    YPositionOfLocation = -(event.getY()) / 0.135;
                    if (-XPositionOfLocation + 1460 >= 2800) {
```

```
                XPositionOfLocation = -1300;
            }
            if (-YPositionOfLocation + 700 >= 2050) {
                YPositionOfLocation = -1350;
            }

            YPointer = -YPositionOfLocation;
            XPointer = -XPositionOfLocation;
            mapObject.setCoordinates(-XPositionOfLocation, -YPositionOfLocation);
            Main.group.setLayoutX(XPositionOfLocation);
            Main.group.setLayoutY(YPositionOfLocation);
        }

    }
});

scene.setOnKeyPressed(new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent event) {
        double x = Main.group.getLayoutX();
        double y = Main.group.getLayoutY();

        if (event.getCode() == KeyCode.ESCAPE) {
            Map.cancelPressed();
        } else if (event.getCode() == KeyCode.G){
            Map.fightMode = !Map.fightMode;
        }
        else if (event.getCode() == KeyCode.DELETE) {
            Map.delete();
        } else if (event.getCode() == KeyCode.INSERT) {
            try {
                DialogWindow.display();
            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
            } else if (event.getCode().equals(KeyCode.LEFT)) {
                if (XPointer - Main.screenMove >= 0 && XPointer - Main.screenMove <= 1300) {
                    XPointer -= Main.screenMove;
                    x += Main.screenMove;
                }
                Map.heroKeyMove(-Main.screenMove, 0);
            } else if (event.getCode().equals(KeyCode.RIGHT)) {
                if (XPointer + Main.screenMove <= 1300 && XPointer + Main.screenMove >= 0) {
                    XPointer += Main.screenMove;
                    x -= Main.screenMove;
                }
                Map.heroKeyMove(Main.screenMove, 0);
            } else if (event.getCode().equals(KeyCode.UP)) {
                if (YPointer - Main.screenMove >= 0 && YPointer - Main.screenMove <= 1350) {
                    YPointer -= Main.screenMove;
                    y += Main.screenMove;
                }
                Map.heroKeyMove(0, -Main.screenMove);
            } else if (event.getCode().equals(KeyCode.DOWN)) {
                if (YPointer + Main.screenMove <= 1350 && YPointer + Main.screenMove >= 0) {
                    YPointer += Main.screenMove;
                    y -= Main.screenMove;
                }
                Map.heroKeyMove(0, Main.screenMove);
            }

            Main.group.setLayoutX(x);
            Main.group.setLayoutY(y);
            mapObject.setCoordinates(-x, -y);
        }
    });

    mapObject.setHeroObjects(imageMiniHeroCamp, imageMiniNinjaCamp,
imageMiniHunterCamp, Map.mapImage);
```

```
    AnimationTimer timer = new AnimationTimer() {
        @Override
        public void handle(long l) {
            mapObject.AutomaticMoveOfHeroes();
            if(!createStatus){
                mapObject.heroesSpawn(HeroType.Hero);
                mapObject.heroesSpawn(HeroType.Hero);
                mapObject.heroesSpawn(HeroType.Hero);
                mapObject.heroesSpawn(HeroType.Hero);
                mapObject.heroesSpawn(HeroType.Ninja);
                mapObject.heroesSpawn(HeroType.Ninja);
                mapObject.heroesSpawn(HeroType.Ninja);
                mapObject.heroesSpawn(HeroType.Ninja);
                mapObject.heroesSpawn(HeroType.Hunter);
                mapObject.heroesSpawn(HeroType.Hunter);
                mapObject.heroesSpawn(HeroType.Hunter);
                mapObject.heroesSpawn(HeroType.Hunter);
            }
                createStatus = true;
                mapObject.startTheLifeOfTheObjects();

        }
    };

    timer.start();

    primaryStage.show();
}

public static void addHero(String name, double XPosition, double YPosition) {
    mapObject.addNewHero(new Hero(name, XPosition, YPosition));
}

public static void addNinja(String name, double XPosition, double YPosition) {
    mapObject.addNewHero(new Ninja(name, XPosition, YPosition));
```

```
    }

    public static void addHunter(String name, double XPosition, double YPosition) {
        mapObject.addNewHero(new Hunter(name, XPosition, YPosition));
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Модуль Hero.java

```
// клас героя

package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

import java.util.Random;

import static javafx.scene.paint.Color.RED;
import static javafx.scene.paint.Color.TRANSPARENT;

public class Hero implements Cloneable {
    private String name;
    private int health;
    private double damage;
    private Weapon weapon;
```

```
private boolean active;
private boolean isInside = false;
private boolean process;
protected boolean isAlive;
protected HeroType heroType;
Random random = new Random();

protected int time_delay = 50;
protected int time_current;

protected double XPosition;
protected double YPosition;
public double aimx;
public double aimy;

public static Image heroImage;
public static Image heroActiveIconImage;
protected ImageView heroImageView;
protected ImageView heroActiveIconImageView = new ImageView(heroActiveIconImage);
public Text heroHealthText;
public Text heroNameText;
public Rectangle rectangle;

public void HeroConstruct(String name, double x, double y) { // конструктор героя
    this.name = name;
    this.XPosition = x;
    this.YPosition = y;
    this.isAlive = true;

    if (typeCheck(HeroType.Hero)) {
        this.health = 500;
        this.damage = 10;
    } else if (typeCheck(HeroType.Ninja)) {
        this.health = 1000;
        this.damage = 20;
```

```
    }
    else if (typeCheck(HeroType.Hunter)) {
        this.health = 1500;
        this.damage = 50;
    }
    this.weapon = new Weapon(damage);
}

public void setWeapon(Weapon weapon) {
    this.weapon = weapon;
}

public Hero(String name, double XPosition, double YPosition) {
    Weapon weapon = new Weapon(damage);
    setWeapon(weapon);

    this.heroType = HeroType.Hero;
    heroImageView = new ImageView(heroImage);
    this.HeroConstruct(name, XPosition, YPosition);
    heroImageView.setX(XPosition);
    heroImageView.setY(YPosition);

    Text textName = new Text(name);
    textName.setFont(new Font( 20));
    textName.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
    textName.setX(heroImageView.getX() + 20);
    textName.setY(heroImageView.getY() - 40);


    Text textHealth = new Text(Integer.toString(health));
    textHealth.setFont(new Font(20));
    textHealth.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
    textHealth.setX(heroImageView.getX() + 20);
    textHealth.setY(heroImageView.getY() - 20);
```

```
    rectangle = new Rectangle(130, 175);
    rectangle.setFill(Color.TRANSPARENT);
    rectangle.setStrokeWidth(5);
    rectangle.setStroke(TRANSPARENT);
    rectangle.setX(heroImageView.getX() - 5);
    rectangle.setY(heroImageView.getY() - 10);

    heroActiveIconImageView.setX(heroImageView.getX());
    heroActiveIconImageView.setY(heroImageView.getY() + 177);

    this.heroNameText = textName;
    this.heroHealthText = textHealth;

    Main.group.getChildren().addAll(heroImageView, heroNameText, heroHealthText, rectangle);
    this.showHero();
}

public Hero(){};

//public Hero(){this("Ruslan", 1000, 1000);}

public void showHero() {
    System.out.println(toString());
}

public void setCoordinatesOfObjects(double x, double y){
    if(heroImageView !=null && heroNameText!=null) {
        heroImageView.setX(x);
        heroImageView.setY(y);

        heroNameText.setX(heroImageView.getX() + 20.0);
        heroNameText.setY(heroImageView.getY() - 40.0);

        heroHealthText.setX(heroImageView.getX() + 20.0);
        heroHealthText.setY(heroImageView.getY() - 20.0);
```

```
            rectangle.setX(heroImageView.getX()-5);
            rectangle.setY(heroImageView.getY()-10);
        }
    }

    public boolean heroIntersects (ImageView hero, ImageView a, Image b){
        double height,width;
        height = 175;
        width = 130;


if((hero.getX()>a.getX())&&(hero.getX()<a.getX()+b.getWidth())&&(hero.getY()>a.getY())&&(hero.
getY()<a.getY()+b.getHeight())) {
            return true;
        }
        else
if((hero.getX()+width>a.getX())&&(hero.getX()+width<a.getX()+b.getWidth())&&(hero.getY()>a.get
Y())&&(hero.getY()<a.getY()+b.getHeight())) {
            return true;
        }
        else
if((hero.getX()>a.getX())&&(hero.getX()<a.getX()+b.getWidth())&&(hero.getY()+height>a.getY())&
&(hero.getY()+height<a.getY()+b.getHeight())) {
            return true;
        }
        else
if((hero.getX()+width>a.getX())&&(hero.getX()+width<a.getX()+b.getWidth())&&(hero.getY()+heig
ht>a.getY())&&(hero.getY()+height<a.getY()+b.getHeight())){
            return true;
        }

        return false;
    }
```

```
public void move()
{
   if( isActive() ) return;

   if( !isEmptyAim() && !isProcess()) {

      double x = getHeroImageView().getX();
      double y = getHeroImageView().getY();

      if ((Math.abs(x - aimx) + Math.abs(y - aimy)) < 1.0) {
         clearAim();
      } else {

         double signdx = Math.signum(aimx - x);
         double dx = Math.abs(aimx - x);
         dx = ((dx < 2) ? dx : 2);
         dx = signdx * dx;

         double signdy = Math.signum(aimy - y);
         double dy = Math.abs(aimy - y);
         dy = ((dy < 2) ? dy : 2);
         dy = signdy * dy;

         x += dx;
         y += dy;

         setCoordinatesOfObjects(x,y);
      }

   }
}

public void clearAim() { aimx = -1000; aimy = -1000; }

private boolean isEmptyAim() {
```

```
    if( (aimx==-1000) || (aimy==-1000) ){
        return true;
    }
    return false;
}

public void tryToChange(double mx, double my) {
    if( rectangle.boundsInParentProperty().get().contains(mx,my) ) {

        active = !active;

        if (!active) {
            rectangle.setStroke(TRANSPARENT);
        }
        else
        {
            rectangle.setStroke(RED);
        }
    }
}

public boolean typeCheck(HeroType heroType) {
    if (heroType == this.heroType) return true;
    return false;
}

@Override
public String toString() {
    return "Hero{" +
            "name='" + name + '\'' +
            ", health=" + health +
            ", spawnActive=" + active +
            ", isAlive=" + isAlive +
            ", damageOfWeapon=" + weapon.damageOfWeapon+
            '}';
```

```
}

public void lifeCycle() {
    time_current++;
    if (time_current < time_delay) return;
    time_current = 0;
    if (!this.process) {
        double setX, setY;
        do{
            setX = 100 + (3840-100)*random.nextDouble();
            setY = 100 + (2160-100)*random.nextDouble();
        }while (Main.mapObject.freePlace(setX,setY));
        this.setAim(setX, setY);



    }
}
public void setAim(double x,double y)
{
    aimx=x;
    aimy=y;
}

public void Delete () {
    active = false;
    heroImageView.setImage(null);
    heroNameText.setText(null);
    heroHealthText.setText(null);
}

public boolean cross( ImageView heroImageView, ImageView a, Image b) {
    double height,width;
    height = width =200;
```

```
if((heroImageView.getX()>a.getX())&&(heroImageView.getX()<a.getX()+b.getWidth())&&(heroImageView.getY()>a.getY())&&(heroImageView.getY()<a.getY()+b.getHeight())) {
            return true;
        }
        else
if((heroImageView.getX()+width>a.getX())&&(heroImageView.getX()+width<a.getX()+b.getWidth())&&(heroImageView.getY()>a.getY())&&(heroImageView.getY()<a.getY()+b.getHeight())) {
            return true;
        }
        else
if((heroImageView.getX()>a.getX())&&(heroImageView.getX()<a.getX()+b.getWidth())&&(heroImageView.getY()+height>a.getY())&&(heroImageView.getY()+height<a.getY()+b.getHeight())) {
            return true;
        }
        else
if((heroImageView.getX()+width>a.getX())&&(heroImageView.getX()+width<a.getX()+b.getWidth())&&(heroImageView.getY()+height>a.getY())&&(heroImageView.getY()+height<a.getY()+b.getHeight())){
            return true;
        }

        return false;

    }

    @Override
    public Hero clone() throws CloneNotSupportedException { // оверрайд функції клонування
        Hero heroCopy = (Hero) super.clone();
        heroCopy.weapon = weapon.clone();
        return heroCopy;
    }

    public static Hero cloneTheHero(Hero hero){ // клонування
        Hero hero1 = null;
```

```
        try {
            hero1 = (Hero) hero.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
        return hero1;
    }

    public boolean isProcess() {
        return process;
    }

    public void isInside(boolean a) {this.isInside = a;}
    public boolean inside(){return this.isInside;}
    public void isProcessing(boolean a){this.process = a;}
    public void setInside(boolean inside) { isInside = inside; }
    public boolean isAlive() { return isAlive; }
    public void setAlive(boolean alive) { isAlive = alive; }
    public HeroType getHeroType() { return heroType; }
    public void setHeroType(HeroType heroType) { this.heroType = heroType; }
    public static Image getHeroImage() { return heroImage; }
    public static void setHeroImage(Image heroImage) { Hero.heroImage = heroImage; }
    public static Image getHeroActiveIconImage() { return heroActiveIconImage; }
    public static void setHeroActiveIconImage(Image heroActiveIconImage) {
Hero.heroActiveIconImage = heroActiveIconImage; }
    public ImageView getHeroImageView() { return heroImageView; }
    public void setHeroImageView(ImageView heroImageView) { this.heroImageView =
heroImageView; }
    public ImageView getHeroActiveIconImageView() { return heroActiveIconImageView; }
    public void setHeroActiveIconImageView(ImageView heroActiveIconImageView) {
this.heroActiveIconImageView = heroActiveIconImageView; }
    public Text getHeroHealthText() { return heroHealthText; }
    public void setHeroHealthText(Text heroHealthText) { this.heroHealthText = heroHealthText; }
    public Text getHeroNameText() { return heroNameText; }
    public void setHeroNameText(Text heroNameText) { this.heroNameText = heroNameText; }
```

```
    public Rectangle getRectangle() { return rectangle; }
    public void setRectangle(Rectangle rectangle) { this.rectangle = rectangle; }
    public void setActive(boolean active) { this.active = active; }
    public Rectangle getRectActive() { return rectangle; }
    public boolean isActive() { return active; }
    public void setProcess(){this.process = true;}
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getHealth() { return health; }
    public void setHealth(int health) { this.health = health; }
    public double getXPosition() { return XPosition; }
    public void setXPosition(double XPosition) { this.XPosition = XPosition; }
    public double getDamage(){return this.weapon.damageOfWeapon;}
    public double getYPosition() { return YPosition; }
    public void setYPosition(double YPosition) { this.YPosition = YPosition; }
    public double getXCoordOfImage(){ return heroImageView.getX(); }
    public double getYCoordOfImage(){ return heroImageView.getY(); }
    public Image getImg() { return heroImage; }
    public void scaleImageLeft(){ heroImageView.setScaleX(-1); }
    public void scaleImageRight(){ heroImageView.setScaleX(1); }

}
```

Модуль Ninja.java

```
package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

public class Ninja extends Hero{
```

```
public static Image NinjaImage;

public Ninja(){};

//public Ninja(){this("Ruslan", 1000, 1000);};

public Ninja(String name, double XPosition, double YPosition) {
   super();

   this.setName(name);
   this.XPosition = XPosition;
   this.YPosition = YPosition;
   this.isAlive = true;

   double damage = 20;
   Weapon weapon = new Weapon(damage);
   this.setWeapon(weapon);
   this.heroType = HeroType.Ninja;
   heroImageView = new ImageView(NinjaImage);
   heroImageView.setX(XPosition);
   heroImageView.setY(YPosition);

   Text textName = new Text(name);
   textName.setFont(new Font( 20));
   textName.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
   textName.setX(heroImageView.getX()+20);
   textName.setY(heroImageView.getY()-40);

   int health = 1000;
   this.setHealth(1000);
   Text textHealth = new Text(Integer.toString(health));
   textHealth.setFont(new Font(20));
   textHealth.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
   textHealth.setX(heroImageView.getX() + 20);
   textHealth.setY(heroImageView.getY() - 20);
```

```
        rectangle = new Rectangle(130, 175);
        rectangle.setFill(Color.TRANSPARENT);
        rectangle.setStrokeWidth(5);
        rectangle.setStroke(Color.TRANSPARENT);
        rectangle.setX(heroImageView.getX() - 5);
        rectangle.setY(heroImageView.getY()- 10);

        heroActiveIconImageView.setX(heroImageView.getX());
        heroActiveIconImageView.setY(heroImageView.getY() + 177);

        this.heroNameText = textName;
        this.heroHealthText = textHealth;

        Main.group.getChildren().addAll(heroImageView, heroNameText, heroHealthText, rectangle);
        this.showHero();
    }
}
```

Модуль Hunter.java

```
package sample;

import com.sun.media.jfxmediaimpl.HostUtils;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

public class Hunter extends Ninja{
    public static Image HunterImage;

    public Hunter(String name, double XPosition, double YPosition) {
```

```
super();
this.setName(name);
this.XPosition = XPosition;
this.YPosition = YPosition;
this.isAlive = true;

double damage = 50;
Weapon weapon = new Weapon(damage);
setWeapon(weapon);

this.heroType = HeroType.Hunter;
heroImageView = new ImageView(HunterImage);
heroImageView.setX(XPosition);
heroImageView.setY(YPosition);

Text textName = new Text(name);
textName.setFont(new Font( 20));
textName.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
textName.setX(heroImageView.getX()+20);
textName.setY(heroImageView.getY()-40);

int health = 1500;
this.setHealth(1500);
Text textHealth = new Text(Integer.toString(health));
textHealth.setFont(new Font(20));
textHealth.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));
textHealth.setX(heroImageView.getX() + 20);
textHealth.setY(heroImageView.getY() - 20);

rectangle = new Rectangle(130, 175);
rectangle.setFill(Color.TRANSPARENT);
rectangle.setStrokeWidth(5);
rectangle.setStroke(Color.TRANSPARENT);
rectangle.setX(heroImageView.getX() - 5);
rectangle.setY(heroImageView.getY() - 10);
```

```
        heroActiveIconImageView.setX(heroImageView.getX());
        heroActiveIconImageView.setY(heroImageView.getY() + 177);

        this.heroNameText = textName;
        this.heroHealthText = textHealth;

        Main.group.getChildren().addAll(heroImageView, heroNameText, heroHealthText, rectangle);
        this.showHero();
    }
}
```

Модуль Map.java

```
package sample;

import com.sun.security.auth.module.LdapLoginModule;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import org.w3c.dom.ls.LSOutput;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

public class Map extends Minimap {

    private String[] names = {"Robin", "Alaw", "Dorian", "Safaa", "Elias", "Ira", "Cayson",
"Mohammod", "Julie", "Xena"};
    public static Image mapImage = new Image("images/map2.png");
    public static Image logoImage = new Image("images/contra.png");
    public static Image fight = new Image("images/boom.gif");

    private ImageView mapImageView;
    private ImageView logoImageView;
```

```
    private Rectangle rectangle;
    public static boolean fightMode = true;
    private Random random = new Random();

    public static ArrayList<Hero> heroesList = new ArrayList<Hero>();

    private HeroCamp heroCamp;
    private NinjaCamp ninjaCamp;
    private HunterCamp hunterCamp;

    protected int time_delay=100;
    protected int time_current=0;

    public static int mapX = 3000;
    public static int mapY = 2250;



    public static void heroKeyMove(double x, double y) {
        for (Hero hero : heroesList) {
            if (hero.isActive()) {
                if (hero.getXCoordOfImage() + x >= 0 && hero.getYCoordOfImage() + y - 40 > 0 &&
hero.getXCoordOfImage() + x + hero.getImg().getWidth() < 2850 && hero.getYCoordOfImage() + y
+ hero.getImg().getHeight() < 2210) {
                    if (x < 0) {
                        hero.scaleImageLeft();
                    } else if (x > 0) {
                        hero.scaleImageRight();
                    }
                    hero.setCoordinatesOfObjects(hero.getXCoordOfImage() + x,
hero.getYCoordOfImage() + y);
                }
            }
        }
    }
```

```
    public Map(Image imgHeroCamp, Image imgNinjaCamp, Image ingHunterCamp) {
        super();
        mapImageView = new ImageView(mapImage);
        logoImageView = new ImageView(logoImage);
        logoImageView.setFitHeight(250);
        logoImageView.setFitWidth(500);
        logoImageView.setX(1250);
        logoImageView.setY(50);
        rectangle = new Rectangle(2830, 2180);
        rectangle.setFill(Color.TRANSPARENT);
        rectangle.setStroke(Color.ORANGERED);
        rectangle.setStrokeWidth(20);


        Main.group.getChildren().addAll(mapImageView, rectangle, logoImageView);

        heroCamp = new HeroCamp(imgHeroCamp, 365, 525);
        ninjaCamp = new NinjaCamp(imgNinjaCamp, 2200, 360);
        hunterCamp = new HunterCamp(ingHunterCamp, 1250, 1340);

        locationsOfObjects = new Placement[]{heroCamp, ninjaCamp, hunterCamp};
    }

    public static void isPressed(double x, double y) {
        for (Hero hero : heroesList) {
            hero.tryToChange(x, y);
        }
    }

    public static void cancelPressed() {
        for (Hero colonist : heroesList) {
            if (colonist.isActive())
                colonist.tryToChange(colonist.heroImageView.getX() + 20,
colonist.heroImageView.getY() + 20);
```

```
    }
}

public void addNewHero(Hero hero) {
    heroesList.add(hero);
    miniHero(hero);
}

public void AutomaticMoveOfHeroes() {
    MiniHeroesSet();
}

public static void delete() {
    for (int i = 0; i < heroesList.size(); i++) {
        if (heroesList.get(i).isActive()) {
            delete(heroesList.get(i));
            i--;
        }
    }
}

public static void delete(Hero colonist) {
    for (int i = 0; i < heroesList.size(); i++) {
        if (heroesList.get(i) == colonist) {
            Main.group.getChildren().remove(heroesList.get(i));
            Main.group.getChildren().remove(heroesList.get(i).rectangle);
            heroesList.get(i).Delete();
            Minimap.miniHero.remove(i);
            heroesList.remove(heroesList.get(i));
        }
    }
}

public void heroesSpawn(HeroType heroType) {
```

```
    Hero hero = new Hero();
    String name;
    double x;
    double y;

    name = names[random.nextInt(9)];

    do{
        x = random.nextInt(mapX - 300);
        y = random.nextInt(mapY - 300);
    } while ((!isRight(x, y)));

    switch (heroType){
        case Hero:
            hero = new Hero(name, x, y);
            break;
        case Ninja:
            hero = new Ninja(name, x, y);
            break;
        case Hunter:
            hero = new Hunter(name, x, y);
            break;
    }
    addNewHero(hero);
}


public boolean freePlace(double x,double y){
    if(heroCamp.cross(heroCamp.imageView, heroCamp.image, x, y)){
        return false;
    }
    else if(ninjaCamp.cross(ninjaCamp.imageView, ninjaCamp.image, x, y)){
        return false;
    }
    else if(hunterCamp.cross(hunterCamp.imageView, hunterCamp.image, x, y)){
```

```
        return false;
    }
    return true;
}

public boolean isRight(double x, double y) {
    if (heroCamp.cross(heroCamp.imageView, heroCamp.image, x, y))
        return false;
    if (ninjaCamp.cross(ninjaCamp.imageView, ninjaCamp.image, x, y))
        return false;
    if (hunterCamp.cross(hunterCamp.imageView, hunterCamp.image, x, y))
        return false;
    return true;

}
public void startTheLifeOfTheObjects(){
    MiniHeroesSet();
    heroCamp.life();
    ninjaCamp.life();
    hunterCamp.life();

    for(Hero hero: heroesList){
        hero.lifeCycle();
        hero.move();
    }

    time_current ++;
    if ( time_current < time_delay) {
        return;
    }
    time_current = 0;

   for(Hero hero : heroesList){
      for (Hero hero1 : heroesList){
```

```
if (hero.heroIntersects(hero.getHeroImageView(), hero1.getHeroImageView(),
hero1.getImg()) && hero.heroType != hero1.heroType && hero.getHealth() >= 0 &&
hero1.getHealth() >= 0 && fightMode){

                hero.isProcessing(true);
                hero1.isProcessing(true);

                 hero.heroImageView.setImage(fight);
                 hero1.heroImageView.setImage(fight);

                    hero.setHealth((int) (hero.getHealth() - hero1.getDamage()));
                    System.out.println(hero.getHealth());
                    hero1.setHealth((int) (hero1.getHealth() - hero.getDamage()));
                    System.out.println(hero1.getHealth());



                hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
                 hero1.heroHealthText.setText(Integer.toString(hero1.getHealth()));

                 if(hero.getHealth() <= 0 || hero1.getHealth() <= 0){
                    hero.isProcessing(false);
                    hero1.isProcessing(false);
                    if(hero.heroType == HeroType.Hero){
                       hero.heroImageView.setImage(Hero.heroImage);
                       hero.aimx = 365;
                       hero.aimy = 525;
                    }
                    else if(hero.heroType == HeroType.Ninja){
                       hero.heroImageView.setImage(Ninja.NinjaImage);
                       hero.aimx = 2200;
                       hero.aimy = 360;
                    }
                    else if(hero.heroType == HeroType.Hunter){
                       hero.heroImageView.setImage(Hunter.HunterImage);
                       hero.aimx = 1250;
```

```
                hero.aimy = 1340;
            }
            if(hero1.heroType == HeroType.Hero){
                hero1.heroImageView.setImage(Hero.heroImage);
                hero1.aimx = 365;
                hero1.aimy = 525;
            }
            else if(hero1.heroType == HeroType.Ninja){
                hero1.heroImageView.setImage(Ninja.NinjaImage);
                hero1.aimx = 2200;
                hero1.aimy = 360;
            }
            else if(hero1.heroType == HeroType.Hunter){
                hero1.heroImageView.setImage(Hunter.HunterImage);
                hero1.aimx = 1250;
                hero1.aimy = 1340;
            }
        }
    }
  }
}
public static void arraysMethods(ArrayList arrayList) { // функція реалізація методів
    Object[] secondArray = arrayList.toArray(); // конвертувати ArrayList до Array

    System.out.println("This is converted array: ");
    for (int i = 0; i < secondArray.length; i++) { // вивести масив на екран за допомогою циклу
        System.out.println(secondArray[i]);
    }
    System.out.println();

    System.out.println("This array is showed by method \"toString\": ");
    System.out.println(Arrays.toString(secondArray)); // виведення масиву за допомогою
методу "toString"
    System.out.println();
```

```
        System.out.println("This is the copy of array: ");
        Object[] copyOfSecondArray = Arrays.copyOf(secondArray, secondArray.length); //
копіювання масиву
        for (int i = 0; i < copyOfSecondArray.length; i++) { // вивід скопійованого масиву на екран
            System.out.println(copyOfSecondArray[i]);
        }
        System.out.println();

        System.out.println("Are this two arrays the same?");
        System.out.println(Arrays.equals(secondArray, copyOfSecondArray)); // порівняння двох
масивів
        System.out.println();
    }
}
```

Модуль Placement.java

```
package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;

import java.util.ArrayList;

public class Placement {
    protected Image image;
    protected ImageView imageView;
    protected int quantityOfHeroes = 0;
    protected ArrayList<Hero> heroIn = new ArrayList<>();

    protected int time_delay;
    protected int time_current;

    public Placement(){}
```

```
    public Placement(int quantityOfHeroes){
        this.quantityOfHeroes = quantityOfHeroes;
    };
}
```

Модуль HeroCamp.java

```
package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

import javax.lang.model.util.SimpleAnnotationValueVisitor6;
import java.util.Iterator;

public class HeroCamp extends Placement {
    private Rectangle rectangle;
    private Text text;

    public HeroCamp (Image image, double x, double y){
        super();
        imageView = new ImageView(image);

        this.image = image;

        imageView.setY(y);
        imageView.setX(x);

        rectangle = new Rectangle(520, 520);
        rectangle.setX(imageView.getX() - 10);
        rectangle.setY(imageView.getY() + 10);
```

```
        rectangle.setStrokeWidth(5);
        rectangle.setStroke(Color.ORANGERED);
        rectangle.setFill(Color.TRANSPARENT);


        text = new Text("HERO CAMP");
        text.setFont(Font.font("Verdana", FontWeight.BOLD, 40));
        text.setFill(Color.WHITE);
        text.setStroke(Color.ORANGERED);
        text.setStrokeWidth(1);
        text.setX(imageView.getX() + 120);
        text.setY(imageView.getY() + 520);

        Main.group.getChildren().addAll(rectangle, imageView, text);
    }

    public boolean cross(ImageView imageView, Image image, double x, double y){
        double height = 500;
        double width = 500;


if((x>imageView.getX())&&(x<imageView.getX()+image.getWidth())&&(y>imageView.getY())&&(
y<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x+width>imageView.getX())&&(x+width<imageView.getX()+image.getWidth())&&(y>imageVie
w.getY())&&(y<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x>imageView.getX())&&(x<imageView.getX()+image.getWidth())&&(y+height>imageView.get
Y())&&(y+height<imageView.getY()+image.getHeight())) {
            return true;
        }
```

```
        else
if((x+width>imageView.getX())&&(x+width<imageView.getX()+image.getWidth())&&(y+height>im
ageView.getY())&&(y+height<imageView.getY()+image.getHeight())){
            return true;
        }
        return false;
    }

    public void life() {
        time_current++;
        if (time_current < time_delay) {
            return;
        }
        time_current = 0;

        for (Hero hero : Map.heroesList) {
            if (hero != null) {
                if (hero.isActive() && hero.cross(hero.getHeroImageView(), this.imageView,
this.image)) {
                    if (hero.heroType == HeroType.Hero) {
                        this.heroIn.add(hero);
                        hero.setHealth(500);
                        hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
                        System.out.println("Intersect!");
                        // функція збільшення хп
                        hero.isInside(true);
                        hero.setProcess();
                        continue;
                    }
                }
            }

            if (hero.cross(hero.getHeroImageView(), this.imageView, this.image) && hero.heroType
== HeroType.Hero) {
                this.heroIn.add(hero);
```

```
            hero.setHealth(500);
            hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
            hero.isInside(true);
            hero.setProcess();
        }
    }
    for(Iterator<Hero> it = heroIn.iterator(); it.hasNext();){
        Hero hero = it.next();
        if(hero.getHealth() == 500){
            for(Hero hero1: Map.heroesList){
                if(hero == hero1 && hero.inside()){
                    hero.isInside(false);
                    hero.isProcessing(false);
                    // запуск плана
                }
            }
        }
    }
}
```

Модуль NinjaCamp.java

```
package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

import java.util.Iterator;

public class NinjaCamp extends Placement {
    private Rectangle rectangle;
```

```
private Text text;

public NinjaCamp (Image image, double x, double y){
    super();

    imageView = new ImageView(image);
    this.image = image;

    imageView.setY(y);
    imageView.setX(x);

    rectangle = new Rectangle(520, 520);
    rectangle.setX(imageView.getX() - 10);
    rectangle.setY(imageView.getY() + 10);
    rectangle.setStrokeWidth(5);
    rectangle.setStroke(Color.ORANGERED);
    rectangle.setFill(Color.TRANSPARENT);

    text = new Text("NINJA CAMP");
    text.setFont(Font.font("Verdana", FontWeight.BOLD, 40));
    text.setFill(Color.WHITE);
    text.setStroke(Color.ORANGERED);
    text.setStrokeWidth(1);
    text.setX(imageView.getX() + 120);
    text.setY(imageView.getY() + 520);

    rectangle = new Rectangle(520, 520);
    rectangle.setX(imageView.getX() - 10);
    rectangle.setY(imageView.getY() + 10);
    rectangle.setStrokeWidth(5);
    rectangle.setStroke(Color.ORANGERED);
    rectangle.setFill(Color.TRANSPARENT);

    Main.group.getChildren().addAll(rectangle, imageView, text);
}
```

```
    public boolean cross(ImageView imageView, Image image, double x, double y){
        double height = 500;
        double width = 500;


if((x>imageView.getX())&&(x<imageView.getX()+image.getWidth())&&(y>imageView.getY())&&(
y<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x+width>imageView.getX())&&(x+width<imageView.getX()+image.getWidth())&&(y>imageVie
w.getY())&&(y<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x>imageView.getX())&&(x<imageView.getX()+image.getWidth())&&(y+height>imageView.get
Y())&&(y+height<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x+width>imageView.getX())&&(x+width<imageView.getX()+image.getWidth())&&(y+height>im
ageView.getY())&&(y+height<imageView.getY()+image.getHeight())){
            return true;
        }
        return false;
    }
    public void life() {
        time_current++;
        if (time_current < time_delay) {
            return;
        }
        time_current = 0;

        for (Hero hero : Map.heroesList) {
            if (hero != null) {
```

```
            if (hero.isActive() && hero.cross(hero.getHeroImageView(), this.imageView,
this.image)) {
                if (hero.heroType == HeroType.Ninja) {
                    this.heroIn.add(hero);
                    hero.setHealth(1000);
                    hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
                    System.out.println("Intersect!");
                    // функція збільшення хп
                    hero.isInside(true);
                    hero.setProcess();
                    continue;
                }
            }
        }
        if (hero.cross(hero.getHeroImageView(), this.imageView, this.image) && hero.heroType
== HeroType.Ninja) {
            this.heroIn.add(hero);
            hero.setHealth(1000);
            hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
            // функція збільшення хп
            hero.isInside(true);
            hero.setProcess();
        }
    }
    for(Iterator<Hero> it = heroIn.iterator(); it.hasNext();){
        Hero hero = it.next();
        if(hero.getHealth() == 1000){
            for(Hero hero1: Map.heroesList){
                if(hero == hero1 && hero.inside()){
                    hero.isInside(false);
                    hero.isProcessing(false);
                    // запуск плана
                }
            }
        }
```

```
        }
    }
}
```

Модуль HunterCamp.java

```
package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

import java.util.Iterator;

public class HunterCamp extends Placement {
    private double XPosition, YPosition;
    private Rectangle rectangle;
    private Text text;

    public HunterCamp (Image image, double x, double y){
        super();
        imageView = new ImageView(image);

        this.image = image;

        imageView.setY(y);
        imageView.setX(x);

        rectangle = new Rectangle(520, 520);
        rectangle.setX(imageView.getX() - 10);
        rectangle.setY(imageView.getY() + 10);
        rectangle.setStrokeWidth(5);
        rectangle.setStroke(Color.ORANGERED);
```

```
        rectangle.setFill(Color.TRANSPARENT);


        text = new Text("HUNTER CAMP");
        text.setFont(Font.font("Verdana", FontWeight.BOLD, 40));
        text.setFill(Color.WHITE);
        text.setStroke(Color.ORANGERED);
        text.setStrokeWidth(1);
        text.setX(imageView.getX() + 95);
        text.setY(imageView.getY() + 520);

        Main.group.getChildren().addAll(rectangle, imageView, text);

    }

    public boolean cross(ImageView imageView, Image image, double x, double y){
        double height = 500;
        double width = 500;


if((x>imageView.getX())&&(x<imageView.getX()+image.getWidth())&&(y>imageView.getY())&&(
y<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x+width>imageView.getX())&&(x+width<imageView.getX()+image.getWidth())&&(y>imageVie
w.getY())&&(y<imageView.getY()+image.getHeight())) {
            return true;
        }
        else
if((x>imageView.getX())&&(x<imageView.getX()+image.getWidth())&&(y+height>imageView.get
Y())&&(y+height<imageView.getY()+image.getHeight())) {
            return true;
        }
```

```
        else
if((x+width>imageView.getX())&&(x+width<imageView.getX()+image.getWidth())&&(y+height>im
ageView.getY())&&(y+height<imageView.getY()+image.getHeight())){
            return true;
        }
        return false;
    }
    public void life() {
        time_current++;
        if (time_current < time_delay) {
            return;
        }
        time_current = 0;

        for (Hero hero : Map.heroesList) {
            if (hero != null) {
                if (hero.isActive() && hero.cross(hero.getHeroImageView(), this.imageView,
this.image)) {
                    if (hero.heroType == HeroType.Hunter) {
                        this.heroIn.add(hero);
                        hero.setHealth(1500);
                        hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
                        System.out.println("Intersect!");
                        // функція збільшення хп
                        hero.isInside(true);
                        hero.setProcess();
                        continue;
                    }
                }
            }
            if (hero.cross(hero.getHeroImageView(), this.imageView, this.image) && hero.heroType
== HeroType.Ninja) {
                this.heroIn.add(hero);
                hero.setHealth(1500);
                hero.heroHealthText.setText(Integer.toString(hero.getHealth()));
```

```
            // функція збільшення хп
            hero.isInside(true);
            hero.setProcess();
        }
    }
    for(Iterator<Hero> it = heroIn.iterator(); it.hasNext();){
        Hero hero = it.next();
        if(hero.getHealth() == 1500){
            for(Hero hero1: Map.heroesList){
                if(hero == hero1 && hero.inside()){
                    hero.isInside(false);
                    hero.isProcessing(false);
                    // запуск плана
                }
            }
        }
    }
}
```

Модуль Weapon.java

```
// клас зброї

package sample;

public class Weapon implements Cloneable{
    public double damageOfWeapon;

    public Weapon(){} // констуруктор по замовчуванню

    public Weapon(double damageOfWeapon) { // конструктор зброї

        this.damageOfWeapon = damageOfWeapon;
    }
```

```
    @Override
    public String toString() {
        return "Weapon{" +
                "damageOfWeapon=" + damageOfWeapon +
                '}';
    }

    @Override
    public Weapon clone() throws CloneNotSupportedException { // оверрайд функції клонування
        return (Weapon) super.clone();
    }

    public void showInfoAboutWeapon(){
        System.out.println(toString());
    } // вивести інфо про зброю

    // геттери та сеттери

    public double getDamageOfWeapon() {
        return damageOfWeapon;
    }
    public void setDamageOfWeapon(double damageOfWeapon) {
        this.damageOfWeapon = damageOfWeapon;
    }
}
```

Модуль Minimap.java

```
package sample;

import javafx.scene.Group;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.shape.Rectangle;

import java.util.ArrayList;
```

```
import static javafx.scene.paint.Color.*;


public class Minimap {
    public Group group;
    protected Placement locationsOfObjects[];
    public static ArrayList<Hero> miniHero = new ArrayList<Hero>();
    private ArrayList<Rectangle> rectangles = new ArrayList<Rectangle>();
    private Rectangle bigRectangle = new Rectangle();
    private Rectangle miniRectangle = new Rectangle();
    private Rectangle rect;

    protected double x=0;
    protected double y=0;
    private double mainX = 0;
    private double mainY = 0;
    private double XResize = 0.126; // відношення карти до мінікарти
    private double YResize = 0.135;
    private boolean active;

    public double getXResize() { return XResize; }
    public double getYResize() { return YResize; }

    ImageView miniMapImageView;
    ImageView miniHeroCampImageView;
    ImageView miniNinjaCampImageView;
    ImageView miniHunterCampImageView;
    public void miniHero(Hero hero){
        miniHero.add(hero);
    }

    public void setHeroObjects(Image miniHeroCampImage, Image miniNinjaCampImage, Image
miniHunterCampImage, Image miniMapImage){
        group = new Group();
        active = true;
```

```
        miniMapImageView = new ImageView(miniMapImage);
        miniMapImageView.setFitWidth(350);
        miniMapImageView.setFitHeight(280);

        bigRectangle.setX(mainX+x* XResize);
        bigRectangle.setY(mainY+y* YResize);
        bigRectangle.setHeight(280);
        bigRectangle.setWidth(350);

        miniRectangle.setWidth(1465*XResize);
        miniRectangle.setHeight(720*YResize);
        miniRectangle.setX(mainX);
        miniRectangle.setY(mainY);
        miniRectangle.setFill(TRANSPARENT);
        miniRectangle.setStrokeWidth(3);
        miniRectangle.setStroke(WHITE);

        miniHeroCampImageView = new ImageView(miniHeroCampImage);
        miniNinjaCampImageView = new ImageView(miniNinjaCampImage);
        miniHunterCampImageView = new ImageView(miniHunterCampImage);
    }
    public void MiniHeroesSet(){
        if(!active)return;

        Main.group.getChildren().remove(group);
        Main.group.getChildren().remove(miniRectangle);
        group.getChildren().removeAll(miniMapImageView,
bigRectangle,miniHeroCampImageView, miniNinjaCampImageView, miniHunterCampImageView);


        for(int i = 0; i < rectangles.size(); i++) {
            Main.group.getChildren().remove(rectangles.get(i));
            rectangles.remove(rectangles.get(i));
            i--;
```

```
        }
        bigRectangle.setX(mainX+x);
        bigRectangle.setY(mainY+y);
        bigRectangle.setFill(TRANSPARENT);
        bigRectangle.setStroke(WHITE);

        miniRectangle.setX(mainX+x* XResize +x);
        miniRectangle.setY(mainY+y* YResize +y);

        miniHeroCampImageView.setX(mainX+ locationsOfObjects[0].imageView.getX()* XResize
+x);
        miniHeroCampImageView.setY(mainY+ locationsOfObjects[0].imageView.getY()* YResize
+y);

        miniNinjaCampImageView.setX(mainX+ locationsOfObjects[1].imageView.getX()* XResize
+x);
        miniNinjaCampImageView.setY(mainY+ locationsOfObjects[1].imageView.getY()* YResize
+y);

        miniHunterCampImageView.setX(mainX+ locationsOfObjects[2].imageView.getX()*
XResize +x);
        miniHunterCampImageView.setY(mainY+ locationsOfObjects[2].imageView.getY()*
YResize +y);

        miniMapImageView.setX(mainX+x);
        miniMapImageView.setY(mainY+y);

        group.getChildren().addAll(miniMapImageView, bigRectangle,miniHeroCampImageView,
miniNinjaCampImageView, miniHunterCampImageView);

        Main.group.getChildren().addAll(group);
        for(int i = 0;i < miniHero.size();i++) {
           rect = new Rectangle();
           rect.setX(mainX + miniHero.get(i).getXCoordOfImage() * XResize + x);
           rect.setY(mainY + miniHero.get(i).getYCoordOfImage() * YResize + y);
```

```
            rect.setHeight(10);
            rect.setWidth(10);
            if (miniHero.get(i).heroType == HeroType.Hero) {
                rect.setFill(GREEN);
            } else if (miniHero.get(i).heroType == HeroType.Ninja) {
                rect.setFill(PURPLE);
            } else {
                rect.setFill(RED);
            }
                Main.group.getChildren().add(rect);
                rectangles.add(rect);
        }
        Main.group.getChildren().addAll(miniRectangle);
    }
    public void setCoordinates(double x_, double y_){
        this.x=x_;
        this.y=y_;
    }}
```

Модуль Controller.java

```
package sample;

import javafx.event.ActionEvent;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

import java.net.URL;
import java.util.Random;
import java.util.ResourceBundle;
public class Controller implements Initializable {
    public Button okButton;
    public ComboBox<String> heroComboBox = new ComboBox<>();
    public Label healthLabel;
    public TextField nameTextField;
```

```
public ImageView heroImageView2;
public Image heroImage2;
Random random = new Random();

public void choise() {
   DialogWindow.group2.getChildren().removeAll(heroImageView2, healthLabel);
   if (heroComboBox.getValue().equals("8-bit")) {
      heroImage2 = new Image("/images/hero1.png", 150, 210, false, false);
      healthLabel.setText("Health = 500");
   }
   else if (heroComboBox.getValue().equals("Ninja")) {
         heroImage2 = new Image("/images/hero3.png", 150, 210, false, false);
         healthLabel.setText("Health = 1000");
   } else if (heroComboBox.getValue().equals("Hunter")) {
      heroImage2 = new Image("/images/hero2.png", 150, 210, false, false);
      healthLabel.setText("Health = 1500");
   }
   heroImageView2 = new ImageView(heroImage2);
   heroImageView2.setY(35);
   heroImageView2.setX(60);
   DialogWindow.group2.getChildren().addAll(heroImageView2, healthLabel);
}

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
   heroComboBox.getItems().addAll("8-bit", "Ninja", "Hunter");
}
public void pressCancel(javafx.event.ActionEvent actionEvent) {
   DialogWindow.group2.getChildren().removeAll(heroImageView2, healthLabel);
   DialogWindow.window.close();
}
public void pressOK(ActionEvent actionEvent) {
   DialogWindow.group2.getChildren().removeAll(heroImageView2, healthLabel);
   try {
      String nameOfHero = nameTextField.getText();
```

```
            if (heroComboBox.getValue() == null) {
                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Error!");
                alert.setHeaderText("Hero type is not chosen!");
                alert.showAndWait();
                return;
            }
            if(nameOfHero == null && nameOfHero.length() == 0){
                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Error!");
                alert.setHeaderText("Hero name is not entered!");
                alert.showAndWait();
                return;
            }
            String heroTypeString = heroComboBox.getValue();
            if(heroTypeString.equals("8-bit")){
                Main.addHero(nameOfHero, random.nextInt(2700), random.nextInt(2000));
            }else if(heroTypeString.equals("Ninja")){
                Main.addNinja(nameOfHero, random.nextInt(2700), random.nextInt(2000));
            }else if(heroTypeString.equals("Hunter")){
                Main.addHunter(nameOfHero, random.nextInt(2700), random.nextInt(2000));
            }
            //Main.primaryStage.setScene(Main.scene);
            DialogWindow.window.close();
        } catch (Exception e) {
            Alert alert = new Alert(Alert.AlertType.INFORMATION);

            alert.setTitle("Error!");
            alert.setHeaderText("Something is wrong.");
            alert.showAndWait();
            return;
        }   }}
```

Модуль DialogWindow.java

```
package sample;
```

```
import javafx.fxml.FXMLLoader;
import javafx.scene.Group;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.io.IOException;

public class DialogWindow {
    public static Group group2;
    public static Stage window = null;
    public static Scene scene;

    public static void display() throws IOException {
        group2 = new Group();
        Parent root = FXMLLoader.load(Main.class.getResource("dialogWindow.fxml"));
        group2.getChildren().add(root);
        window = new Stage();
        window.initModality(Modality.APPLICATION_MODAL);
        window.setTitle("Create New Hero");

        scene = new Scene(group2, 500, 300);
        window.setScene(scene);
        window.show();
    }
}
```

Модуль HeroType.java

```
package sample;

public enum HeroType {
    Hero,
    Ninja,
    Hunter
}
```