Politechnika Świętokrzyska w Kielcach
Wydział Elektroniki, Automatyki i Informatyki

Projekt: **Programowanie obiektowe (Java)**

Temat projektu: **Space Invaders**	Autorzy: **Adrian Chmielowiec** **Mateusz Kubas** **Tomasz Kubik**
	Rok studiów i rodzaj: **Stacjonarne** **2 rok**
Ocena:	Grupa: **2ID12B**

Spis treści

1. Ogólny opis projektu

Space Invaders to gra, która pierwotnie została zaprojektowana na automaty przez Tomohirę Nishikadę, wydana została w 1978r. Jest to jedna z najwcześniejszy gier typu shoot'em up. Wykorzystuje ona grafikę dwuwymiarową. Gra polega na niszczeniu kolejnych fal kosmitów przy użyciu działka i na zbieraniu jak największej liczby punktów. Gra ta była prekursorem współczesnych gier komputerowych i przyczyniła się do ich przemiany z ciekawostki w światowy pomysł. Space Inavders była bardzo popularna. Została zapisana do księgi rekordów Guinnessa jako top arcade game. Gra ta stanowiła inspirację dla innych gier wideo, była wydawana na kolejne platformy oraz tworzono jej sequele.

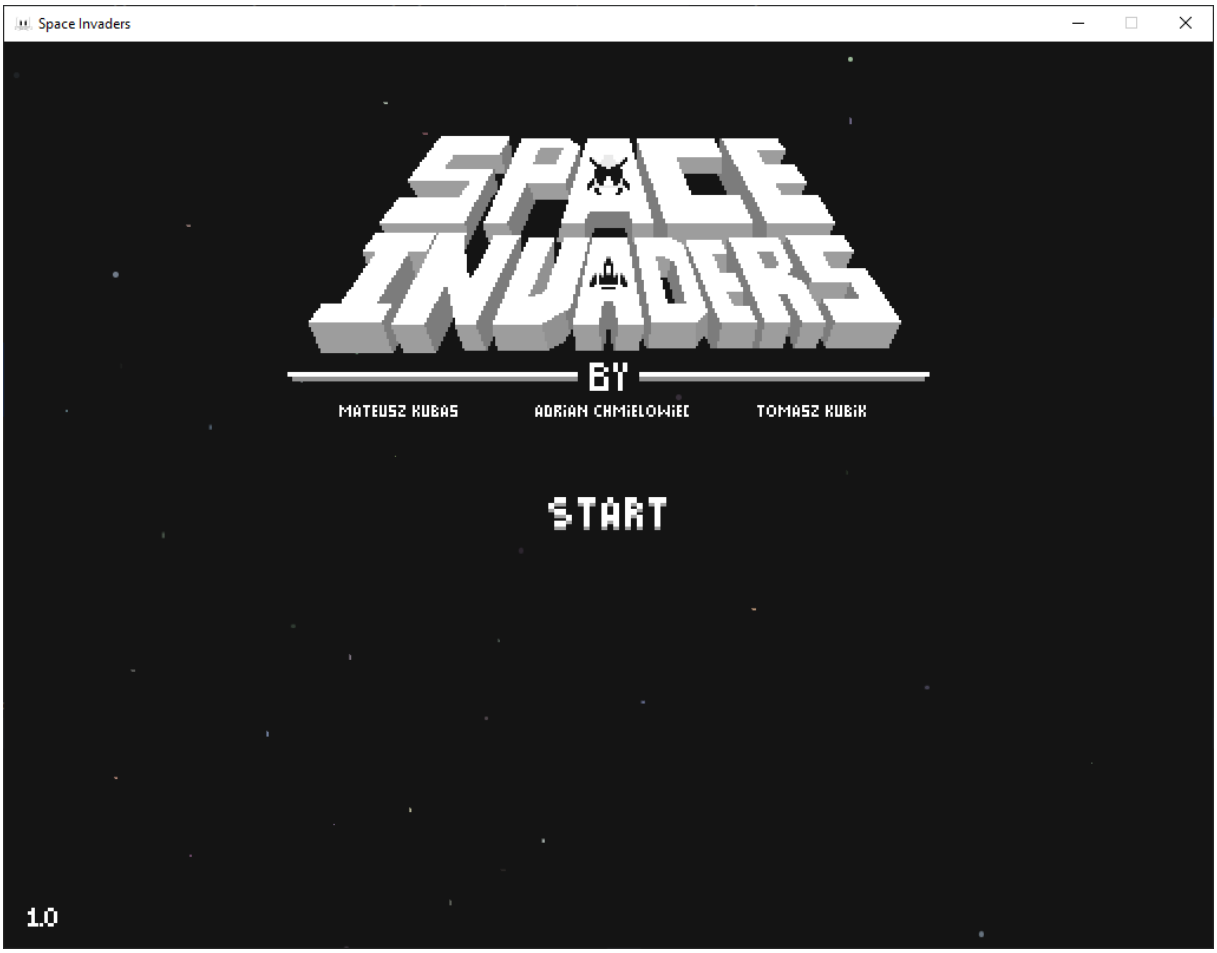
Rozgrywka jest bardzo prosta. Cała plansza widoczna jest cały czas na ekranie a gracz kontroluje swój samolot (działko), przemieszczając je na dole ekranu i strzelając do wrogich statków, kosmitów. Celem gry z założenia jest zniszczenie pięciu rzędów kosmitów, po jedenaście przeciwników w każdym (chociaż, może się to różnić i mogą obowiązywać inne zasady), które poruszają się poziomo tam i z powrotem oraz opadają na dół ekranu. Gdy gracz pokona kosmitę, zyskuje pewną liczbę punktów. Im więcej kosmitów gracz zniszczył, tym szybsze są ich ruchy oraz muzyka. Pokonanie jednej fali kosmitów powoduje nadejście kolejnej, trudniejszej i tak w nieskończoność. Niektóre wersje gry pozwalają kosmitą również do nas strzelać by nas zniszczyć jednocześnie zbliżając się ku dołowi planszy. Gdy tam dotrą i stracimy wszystkie życia (w zależności od wersji może ich nie być) gra się kończy. Możliwe jest też spotkanie specjalnego „tajemniczego statku", który przemiesza się u góry ekranu, za jego zniszczenie gracz otrzymuje dodatkowe punkty. Działko chronione jest przez kilka stacjonarnych stanowisk obronnych (ich liczba i występowanie zależy od wersji gry), które są stopniowo niszczone przez wrogi ostrzał

Nasza wersja gry oparta jest o bibliotekę graficzną javafx i prawidłowe działanie wymaga dołączenia właśnie tej że biblioteki. Umieszczona jest ona w katalogu plugins i jest dodana do projektu. Gra tworzona była w środowisku programistycznym Inteliji jako projekt maven.

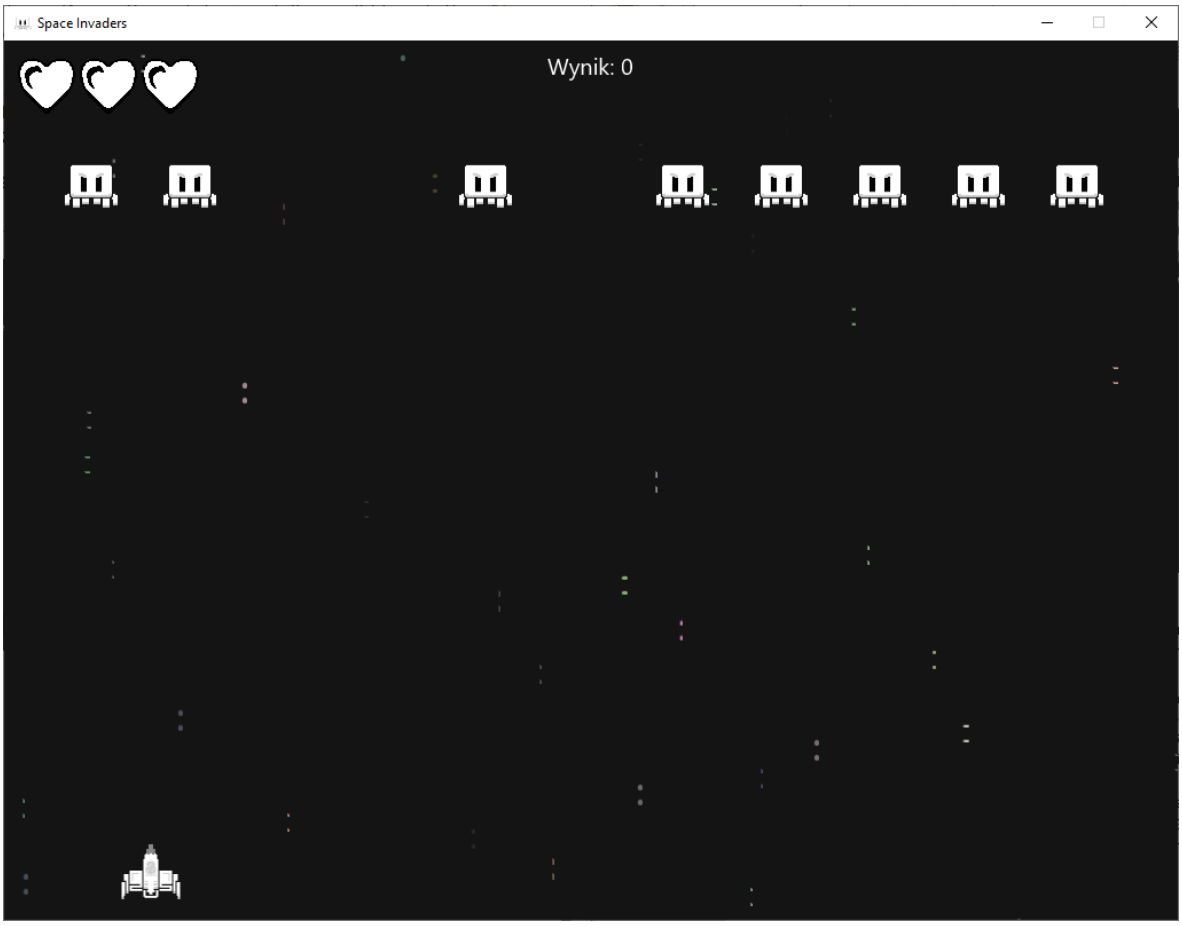
Założenia naszej gry są następujące: gracz może posiadać życia (max. 5), zdobywa punkty, które przyspieszają grę, nie ma stanowisk obronnych oraz tajemniczych statków, kosmici nie mogą do nas strzelać, gra trwa w nieskończoność aż do utraty wszystkich żyć.

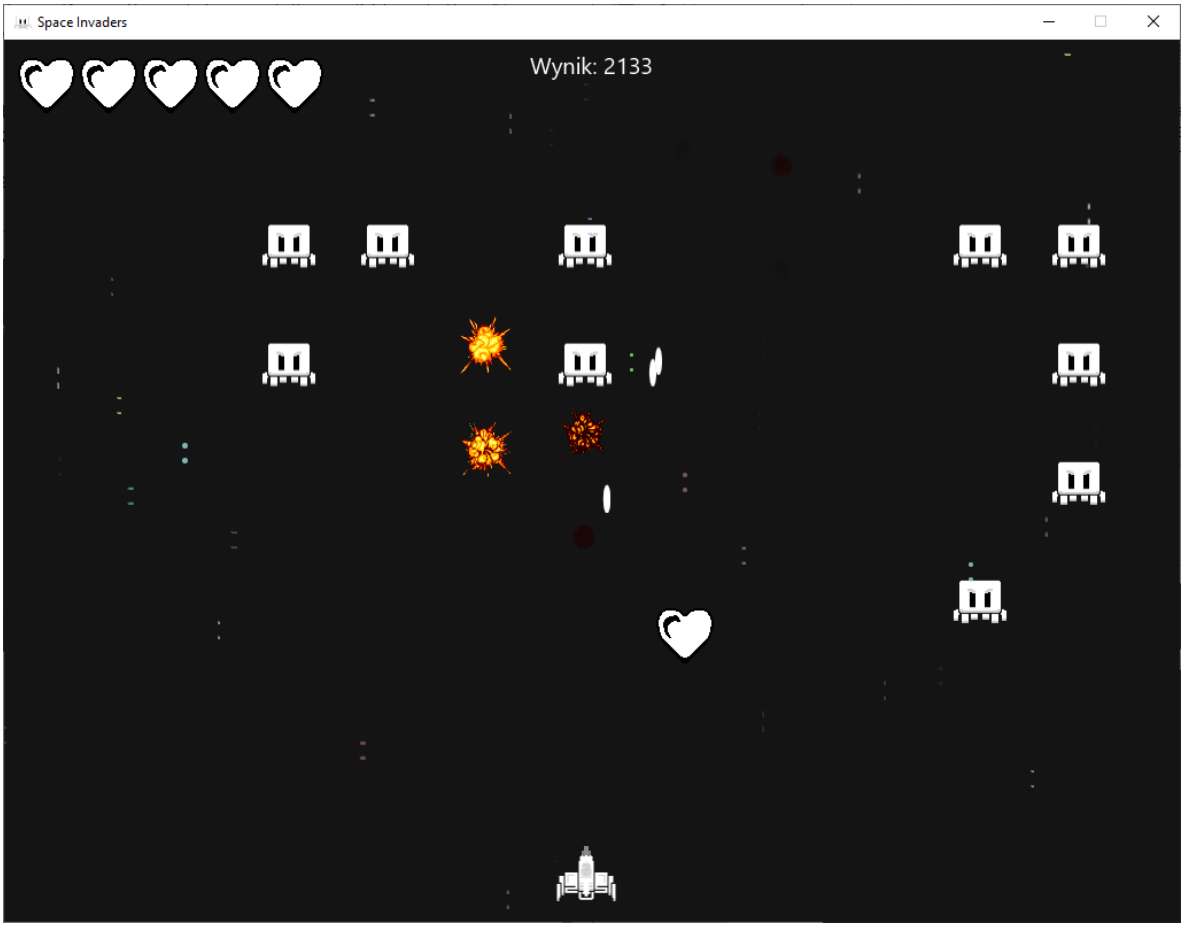
2. Informacje na temat funkcjonalności

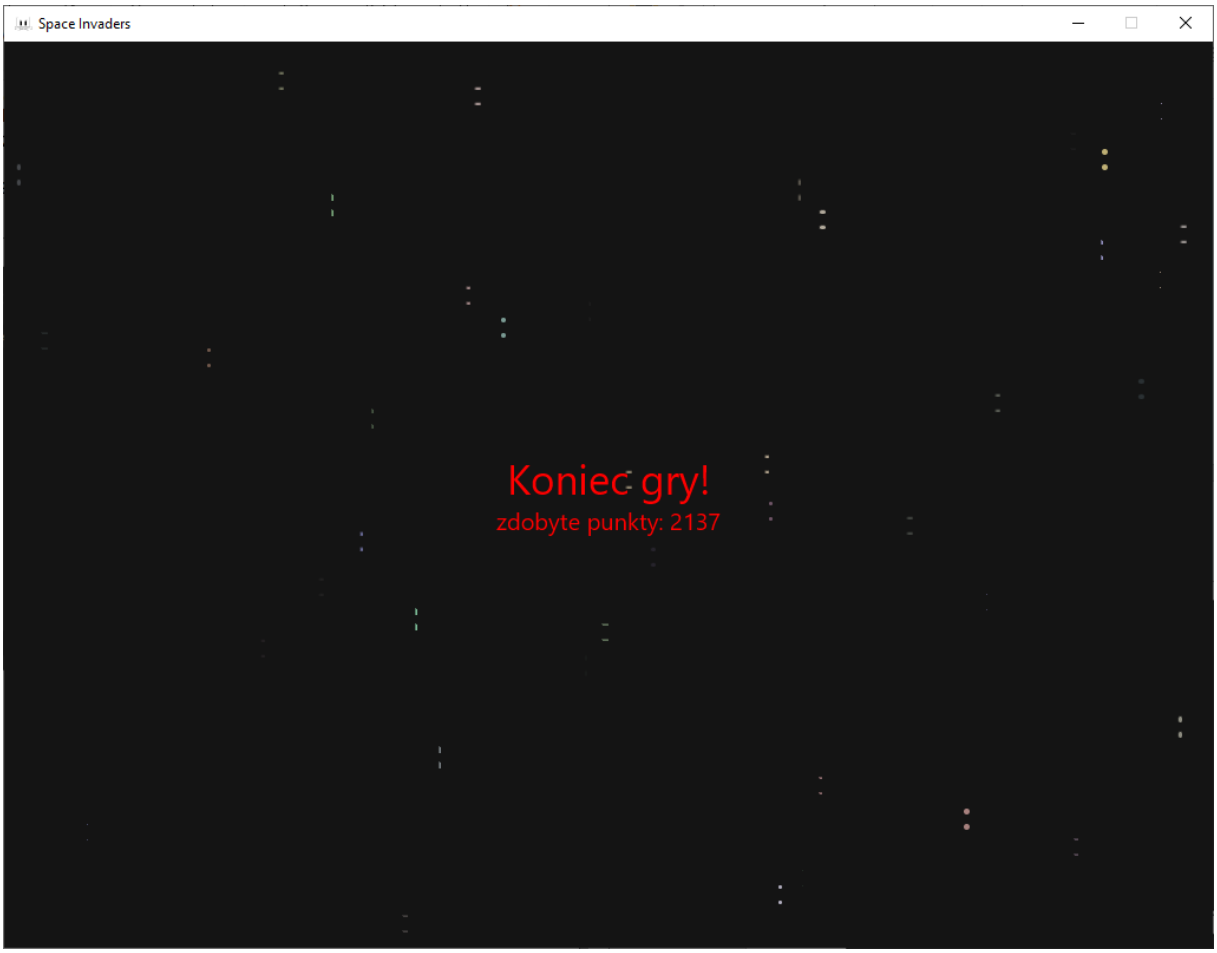
Po uruchomieniu naszej gry wita nas menu główne z logiem, informacjami na temat autorów oraz przyciskiem start, który gdy naciśniemy rozpocznie się gra. W tle możemy usłyszeć muzykę, która ma wprowadzić nas w stylistykę gry oraz widzimy animowane tło. W lewym dolnym rogu ekranu znajduje się informacja na temat wersji gry.



Po kliknięciu przycisku start przechodzimy do samej gry. Pojawia się na dole nasz statek, którym możemy sterować poziomo po ekranie za pomocą myszki. Ma on możliwość strzelania za pomocą przycisków myszy, im szybciej klikamy, tym szybciej strzelamy. Pojawiają się też nasi przeciwnicy czyli kosmici, którzy poruszają się poziomo tam i z powrotem i opadają. Każde zestrzelenie jest punktowane a ponad to mamy szanse na to by z kosmity wyleciał medkit, który dodaje nam życie, maksymalna ilość jaką możemy posiadać to 5. Na start posiadamy 3 życia (ilość posiadanych aktualnie wyświetlana jest w lewym górnym rogu), możemy je stracić, gdy kosmita uderzy nas lub przekroczy naszą linie. Po zdobyciu określonej liczby punktów (informacja o ilości znajduje się na środku na górze okna) dostajemy także bonus w postaci szybszego strzelania, kosmici również zaczynają spadać coraz szybciej. Gra trwa w nieskończoność aż stracimy wszystkie życia. Na końcu zostaje wyświetlona nam liczba zdobytych punktów. Gdy klikniemy ponownie klawisz myszy grę rozpoczniemy od początku. Aby zamknąć ją klikamy po prostu „X” w prawym górnym rogu okna gry. Gra posiada liczne animacje i dźwięki. Każdy nasz strzał posiada dźwięk, tak jak niszczenia kosmitów czy też zbierania medkit’ów, zniszczenie naszego statku również posiada dźwięk. Każde zniszczenie kosmity, oprócz sygnalizacji dźwiękiem jest również widoczne poprzez animacje eksplozji. Wszystkie grafiki zostały stworzone własnoręcznie przez nas.


Space Invaders
Wynik: 0


Space Invaders
Wynik: 2133



3. Informacje na temat sposobu uruchomienia oraz obsługi projektu

Naszą grę można uruchomić poprzez otwarcie projektu w środowisku programistycznym np. Inteliji i dokonania kompilacji. Możliwe jest również uruchomienie gry za pomocą wiersza poleceń. W tym celu otwieramy wiersz polecenia w folderze projektu i piszemy: mvn clean javafx:run. Po wydaniu tej komendy nasza gra się uruchomi. Ostatnią metodą uruchomienia jest użycie pliku jar znajdującego się w out\artifacts\SpaceInvaders_jar, jednak ta metoda nie zawsze zadziała.

Obsługa gry jest bardzo prosta, w menu poruszamy się myszką, klikniecie start rozpoczyna grę. W trakcie gry również poruszamy się za pomocą myszy w boki i strzelamy lewym lub prawym przyciskiem myszy. Grę zamykamy „x” w prawym górnym rogu ekranu.

4. Informacje na temat stworzonych klas i metod

Klasy:

public class Enemy - Klasa określające prędkość spadania kosmitów oraz ich pozycje

public class Medkit - Klasa służy do określanie pozycji medkita i jego ruch

public class Music - Klasa służąca do przechowywania muzyki, dźwięków gry, ustawiania oraz ich puszczania

public class Ship - Klasa która odpowiada za tworzenie statku gracza o danym obrazku na wskazanej pozycji oraz umożliwienie poruszania się i strzelania

public class Shot - Klasa, która ma za zadanie określenie pozycji oddanego strzału, prędkości i sprawdzenia trafienia

public class Star - Klasa która odpowiada za tworzenie spadających "gwiazd" w tle o różnych kolorach i w różnych miejscach

public class Button - Klasa odpowiadająca za przycisk, tworzy go w określonym miejscu i pozycji oraz sprawdza czy został kliknięty

public class GameObject - Klasa która ma za zadanie ustawianie obiektów gry na określonej pozycji, określania ich wymiarów oraz przypisywania wskazanych obrazków

public class Main - Klasa główna, która odpowiada za przechowywanie plików graficznych, wymiarów, scen, wartości oraz odpowiada za uruchomienie gry

Metody:

public Enemy(float posX, float posY, Image image, float speed) - metoda służąca do przypisywania prędkości, pozycji oraz obrazka kosmitą

public void move(double speed, boolean up) - metoda odpowiadająca za ruch w dół o określonej prędkości

public boolean isCollidingY - Sprawdź, czy obiekt nie znajduje się w strefie podanego punktu na osi y

public Medkit(double posX, double posY, Image img) - konstruktor, który tworzy medkit o danym obrazku na pozycji określonej przez parametr

public void move() - Ruch w dół

public Music() - konstruktor, który przypisuje do zmiennych ścieżki plików muzycznych, ustawia ich głośność oraz zapętlenie

public void playMainmenu() - metoda odpowiadająca za włączenie muzyki w menu gry

public void stopMainmenu() - metoda odpowiadająca za wyłączenie muzyki w menu gry

public void playBackground() - metoda odpowiadająca za włączenie muzyki podczas gry

public void playExplosion() - metoda odpowiadająca za odtworzenie dźwięku eksplozji podczas gry

public void playShot() - metoda odpowiadająca za odtworzenie dźwięku strzału podczas gry

public void playMedkit() - metoda odpowiadająca za odtworzenie dźwięku zebrania medkitu podczas gry

public Ship(double posX, double posY, Image image) - konstruktor, który tworzy statek o wskazanym obrakzu na podanej pozycji

public void updatePosition(double posX, double posY) - metoda, która ma za zadanie aktualizacji naszego statku gdy się ruszamy

public Shot Shoot() - metoda, która umożliwia strzelnie statkowi

public Shot(double posX, double posY, float speed) - konstruktor, który tworzy strzał w danej pozycji o określonej prędkości

public Star(double posX, double posY, int WIDTH) - konstruktor, który tworzy gwiazde o losowym kolorze, rozmiarze i umieszcza ją w określonej pozycji

public Color getColor() - metoda która ma za zadanie zwrócić kolor

public void move() - metoda, która odpowiada za ruch gwiazd

public Button(double posX, double posY, Image img, Image hoveringImg) - konstruktor który tworzy przycisk o określonym obrazku i na określonej pozycji

public boolean buttonColliding(double posX, double posY) - metoda sprawdzająca czy przycisk został wciśnięty

public void acion() - metoda pozwalająca zasygnalizować wciśnięcie

public GameObject(double posX, double posY, Image img) - konstruktor, który ustawia obiekt na odpowiedniej pozycji, przypisuje jej grafikę i ustawia wymiary na jej podstawie

public GameObject(double posX, double posY) - konstruktor, który służy do aktualizacji pozycji obiektu

public void setDimentions(double width, double height) - metoda, który służy do ustawienia wymiarów obiektu na podstawie obrazka

public void updatePosition(double posX, double posY) - metoda, która określona nową pozycje obiektu

public boolean killY(int heightBoundary) - metoda, która sprawdza czy obiekt przekroczył podaną granice Y

public boolean isColliding (GameObject o) - metoda, która sprawdza czy obiekty ze sobą kolidują

public void updateExplosion() - metoda, która ma za zadanie tworzyć animacje eksplozji

public void move(double speed, boolean up) - metoda, która ma za zadanie poruszać obiekt w dół

public void start(Stage stage) - metoda start, która odpowiada za tworzenie płótna gry i menu oraz licznika dla gry i wykonywania zdarzeń

private void switchToGameplay(Stage stage, Canvas canvas) - metoda, która ma za zadanie ustawić wszystko co potrzeba do rozgrywki

private void setup() - metoda, która ma za zadanie utworzyć gracza, przeciwników na okeślonych pozycjach

private void run(GraphicsContext gc) - metoda, która odpowiada za rysowanie i gameplay np. sprawdzanie czy pocisk trafił

public void drawPlayer() - metoda odpowiadająca za rysowanie gracza

public void drawEnemies() - metoda odpowiadająca za rysowanie przeciwników

public void drawShots() - metoda odpowiadająca za rysowanie strzałów

public void drawLives() - metoda odpowiadająca za rysowanie ilości żyć

public void drawMedkits() - metoda odpowiadająca za rysowanie apteczek

public void drawExplosions(boolean update) - metoda odpowiadająca za rysowanie eksplozji

private void dex(GameObject o, boolean u) - metoda odpowiadająca za tworzenie animacji eksplozji

public void drawStars() - metoda odpowiadająca za rysowanie gwiazd

public void speedUp() - metoda odpowiadająca za przyspieszenie gry

public void spawnEnemies(int emptySpaces) - metoda odpowiadająca za rysowanie przeciwników

int[] DivideEqual(int value, int by, int originOffset, int endOffset) - metoda odpowiadająca za podzielenie planszy na równe części

void clearAllObjects() - metoda odpowiadająca za czyszczenie przeciwników, strzałów i apteczek

public static void main(String[] args) - metoda odpowiadająca za uruchomienie gry