***Икономически университет***

***Варна***

***Курсов***

***проект***

***По дисциплината:***

***ВИЗУАЛНА СРЕДА ЗА ПРОГРАМИРАНЕ НА***

***На тема :***

*Играта „Междупланетни войни“*

*Изготвили: Проверили:*

*Йордан Йорданов доц.д-р Павел Петров*

*Фак.номер: 102878 АС. СТОЙЧО СТОЕВ*

*Стилиан Димитров*

*Фак.номер: 102985*

*Мирослав Калчев*

*Фак.номер:103345*

*Генади Аврамов*

*Фак.номер: 103046*

Курсовият проект по дисциплината е една скромна игра.Целта на играта е танкът да достигне червения квадрат, докато защитава синия от злите човечета. С бутона space танкът може да стреля и да олесни нелеката си задача. Играта приключва успешно, ако танка е достигнал червения квадрат или обратно , ако някой „враг“ е завзел синия.

Имаме и „приятели“, които разсейват лошите герои от тяхната мисия.

Играта е разделена на няколко папки с множество класове.

Кодът на играта е следния:

**🡪папка Display**

Клас Display

**package** display;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
  
**public class** Display **extends** Canvas{  
 *//Creating our Window frame* **private** JFrame **frame**;  
 *//Creating a field on which we draw* **private** Canvas **canvas**;  
  
 **private** String **title**;  
 **private int width**, **height**;  
  
 **public** Display(String title, **int** width, **int** height) {  
 **this**.**title** = title;  
 **this**.**width** = width;  
 **this**.**height** = height;  
  
 *//Creating the JFrame* createDisplay();  
 }  
  
 **private void** createDisplay() {  
 *//Creates a new Frame with a title* **frame** = **new** JFrame(**this**.**title**);  
 *//Sets the frame's size* **frame**.setSize(**this**.**width**, **this**.**height**);  
 *//Ensures that when the exit button is clicked the whole app stops* **frame**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 *//Frame cannot be resizable* **frame**.setResizable(**false**);  
 *//Set the default position of the frame in the center of the monitor* **frame**.setLocationRelativeTo(**null**);  
 *//Frame is not visible by default, so we enable the visibility* **frame**.setVisible(**true**);  
 *//Enabling the frame to be focusable* **frame**.setFocusable(**true**);  
  
 *//Initializing Canvas in the window* **canvas** = **new** Canvas();  
 *//Setting the size of the Canvas. It works only with Dimension argument* **canvas**.setPreferredSize(**new** Dimension(**width**, **height**));  
 *//Making sure the Canvas will stay with the given width and height* **canvas**.setMaximumSize(**new** Dimension(**width**, **height**));  
 **canvas**.setMinimumSize(**new** Dimension(**width**, **height**));  
  
 *//Linking the canvas onto the frame* **frame**.add(**canvas**);  
 *//Resizing a little bit to ensure that everything on the canvas is visible* **frame**.pack();  
 }  
  
 *//Creating a getter to access the Canvas object from other classes* **public** Canvas getCanvas() {  
 **return this**.**canvas**;  
 }  
}

* **Папка game**

**Клас Bullet**

**package** game;  
  
**import** gfx.Assets;  
  
**import** java.awt.\*;  
**import** java.awt.image.BufferedImage;  
  
**public class** Bullet {  
 **private int x**;  
 **private int y**;  
 **private int width**;  
 **private int height**;  
  
 **private** Rectangle **boundingBoxBullet**;  
 BufferedImage **image**;  
  
 **public** Bullet(**int** x, **int** y) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**image** = Assets.*bullet*;  
 **this**.**width** = 26;  
 **this**.**height** = 35;  
 **this**.**boundingBoxBullet** = **new** Rectangle(**this**.**width**,**this**.**height**);  
  
 }  
 **public int** getX(){**return x**;}  
 **public int** getY(){**return y**;}  
  
 **public boolean** Intersects(Rectangle r) {  
 **if** (**this**.**boundingBoxBullet**.contains(r) || r.contains(**this**.**boundingBoxBullet**)) {  
 **return true**;  
 }  
 **return false**;  
 }  
 **public void** tick(){  
 **y** -=15;  
 }  
 **public void** render(Graphics g){  
 g.drawImage(**image**,**x**,**y**,**null**);  
 **this**.**boundingBoxBullet**.setBounds(**x**,**y**,**this**.**width**,**this**.**width**);  
  
 }  
}

**Клас Enemy**

**package** game;  
  
  
**import** gfx.Assets;  
  
**import** java.awt.\*;  
**import** java.awt.image.BufferedImage;  
**import** java.util.LinkedList;  
  
**public class** Enemy {  
 **private int x**;  
 **private int y**;  
 **private int width**;  
 **private int height**;  
 **private int dX**;  
 **private int dY**;  
**private** Rectangle **boundingBoxEnemy**;  
  
  
 BufferedImage **image**;  
  
 **public** Enemy(**int** x, **int** y) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**width** = 50;  
 **this**.**height** = 50;  
 **this**.**image** = Assets.*enemy*;  
 **this**.**dX** = 8;  
 **this**.**dY** = 8;  
 **this**.**boundingBoxEnemy** = **new** Rectangle(**this**.**width**,**this**.**height**);  
  
  
 }  
  
 **public boolean** Intersects(Rectangle r) {  
 **if** (**this**.**boundingBoxEnemy**.contains(r) || r.contains(**this**.**boundingBoxEnemy**)) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
  
 **private** LinkedList<Enemy> **e** = **new** LinkedList<Enemy>();  
  
 Enemy **TempEnemy**;  
  
 **public void** tick() {  
  
  
 **if** (**x**<=0||(**x**+50)>= 800){  
 **dX** = -1\***dX**;  
 }  
 **if** (**y**<=0||(**y**+50)>= 600){  
 **dY**= -1\***dY**;  
 }  
 **x** +=**dX**;  
 **y** +=**dY**;  
  
 **for** (**int** i = 0; i < **e**.size(); i++) {  
 **TempEnemy** = **e**.get(i);  
 **TempEnemy**.tick();  
  
 }  
 *//y -= 5;* **this**.**boundingBoxEnemy**.setBounds(**this**.**x**, **this**.**y**, **this**.**width**, **this**.**height**);  
  
  
 }  
  
 **public void** render(Graphics g) {  
 **for** (**int** i = 0; i < **e**.size(); i++) {  
 **TempEnemy** = **e**.get(i);  
 **TempEnemy**.render(g);  
 }  
 g.drawImage(**image**, **x**, **y**, **null**);  
  
  
 }  
  
 **public void** addEnemy(Enemy block) {  
 **e**.add(block);  
 }  
  
 **public void** removeEnemy(Enemy block) {  
 **e**.remove(block);  
 }  
  
  
}

**Клас Game**

**package** game;  
  
**import** display.Display;  
**import** gfx.Assets;  
**import** gfx.ImageLoader;  
**import** gfx.SpriteSheet;  
**import** states.\*;  
  
**import** java.awt.\*;  
**import** java.awt.image.BufferStrategy;  
**import** java.awt.image.BufferedImage;  
  
**public class** Game **implements** Runnable {  
 **private** Display **display**;  
 **public int width**, **height**;  
 **public** String **title**;  
  
 **private boolean running** = **false**;  
 **private** Thread **thread**;  
  
 **private** InputHandler **inputHandler**;  
 **private** BufferStrategy **bs**;  
 **private** Graphics **g**;  
  
 **private** BufferedImage **img**;  
 **private** SpriteSheet **sh**;  
  
  
 **private** State **gameState**;  
 **private** State **menuState**;  
 **private** State **settingsState**;  
  
  
 **public static** Player *player*;  
 **public static** Rectangle *enemyStatic*;  
 **public static** Rectangle *goal*;  
 **public static** Enemy *enemyDown*, *enemyUp*, *enemyRight*, *enemyLeft*;  
 **public static** noEnemy *enemyDownOther*, *enemyUpOther*, *enemyRightOther*, *enemyLeftOther*;  
 **public static** Bullet *bullet*;  
  
  
 **public** Game(String title, **int** width, **int** height) {  
 **this**.**width** = width;  
 **this**.**height** = height;  
 **this**.**title** = title;  
 }  
  
  
 **private void** init() {  
  
 **display** = **new** Display(**this**.**title**, **this**.**width**, **this**.**height**);  
 **img** = ImageLoader.*loadImage*(**"/textures/bckg.jpg"**);  
 **sh** = **new** SpriteSheet(ImageLoader.*loadImage*(**"/textures/player.png"**));  
  
  
 **this**.**inputHandler** = **new** InputHandler(**this**.**display**);  
  
 Assets.*init*();  
  
  
 **gameState** = **new** GameState();  
 **menuState** = **new** MenuState();  
 **settingsState** = **new** SettingsState();  
  
 StateManager.*setState*(**gameState**);  
  
 *player* = **new** Player();  
 *enemyStatic* = **new** Rectangle(*player*.getX(), *player*.getY(), 20, 20);  
 *goal* = **new** Rectangle(0, 0, 60, 60);  
 *enemyDown* = **new** Enemy(400, 10);  
 *enemyUp* = **new** Enemy(10, 400);  
 *enemyRight* = **new** Enemy(500, 150);  
 *enemyLeft* = **new** Enemy(200, 400);  
 *enemyDownOther* = **new** noEnemy(400, 10);  
 *enemyUpOther* = **new** noEnemy(10, 400);  
 *enemyRightOther* = **new** noEnemy(500, 150);  
 *enemyLeftOther* = **new** noEnemy(200, 400);  
  
  
  
 }  
  
  
  
 **private void** tick() {  
  
 **if** (StateManager.*getState*() != **null**) {  
 StateManager.*getState*().tick();  
 }  
 *player*.tick();  
 **if** (*enemyDown*.Intersects(*enemyStatic*) || *enemyRight*.Intersects(*enemyStatic*) ||  
 *enemyUp*.Intersects(*enemyStatic*) || *enemyLeft*.Intersects(*enemyStatic*)) {  
 System.***out***.print(**"You died"**);  
 stop();  
 }  
 **if** (*enemyUpOther*.Intersects(*enemyStatic*)) {  
 *goal*.setLocation(600, 0);  
 }  
 **if** (*enemyRightOther*.Intersects(*enemyStatic*)) {  
 *goal*.setLocation(0, 600);  
 }  
 **if** (*enemyDownOther*.Intersects(*enemyStatic*)) {  
 *goal*.setLocation(300, 50);  
 }  
 **if** (*enemyLeftOther*.Intersects(*enemyStatic*)){  
 *goal*.setLocation(50,350);  
 }  
  
  
 **if** (*player*.Intersects(*goal*)) {  
 System.***out***.println(**"You WIN"**);  
 stop();  
 }  
  
  
  
 **if** (*player*.*goingDown*) {  
 *enemyStatic*.setLocation(*player*.getX() + 50, *player*.getY() - 50);  
 }  
 **if** (*player*.*goingUp*) {  
 *enemyStatic*.setLocation(*player*.getX() + 50, *player*.getY() + 150);  
 }  
 **if** (*player*.*goingRight*) {  
  
 *enemyStatic*.setLocation(*player*.getX() - 50, *player*.getY() + 50);  
  
 }  
 **if** (*player*.*goingLeft*) {  
 *enemyStatic*.setLocation(*player*.getX() + 150, *player*.getY() + 50);  
 }  
  
 **if** (*player*.*isShooting*) {  
 }  
 *enemyDown*.tick();  
 *enemyUp*.tick();  
 *enemyRight*.tick();  
 *enemyLeft*.tick();  
 *enemyDownOther*.tick();  
 *enemyUpOther*.tick();  
 *enemyRightOther*.tick();  
 *enemyLeftOther*.tick();  
 }  
  
  
 **private void** render() {  
  
 **this**.**bs** = **display**.getCanvas().getBufferStrategy();  
  
 **if** (**bs** == **null**) {  
  
 **display**.getCanvas().createBufferStrategy(2);  
  
 **return**;  
 }  
  
 **g** = **bs**.getDrawGraphics();  
  
 **g**.clearRect(0, 0, **this**.**width**, **this**.**height**);  
  
  
 **g**.drawImage(**img**, 0, 0, **this**.**width**, **this**.**height**, **null**);  
  
 *player*.render(**g**);  
 **g**.setColor(Color.***BLUE***);  
 **g**.fillRect(**this**.*enemyStatic*.**x**, **this**.*enemyStatic*.**y**, **this**.*enemyStatic*.**width**, **this**.*enemyStatic*.**height**);  
  
 **g**.setColor(Color.***RED***);  
 **g**.fillRect(**this**.*goal*.**x**, **this**.*goal*.**y**, **this**.*goal*.**width**, **this**.*goal*.**height**);  
 *enemyDown*.render(**g**);  
 *enemyUp*.render(**g**);  
 *enemyRight*.render(**g**);  
 *enemyLeft*.render(**g**);  
 *enemyLeftOther*.render(**g**);  
 *enemyRightOther*.render(**g**);  
 *enemyUpOther*.render(**g**);  
 *enemyDownOther*.render(**g**);  
  
  
 **if** (StateManager.*getState*() != **null**) {  
 StateManager.*getState*().render(**this**.**g**);  
 }  
  
  
 **bs**.show();  
  
 **g**.dispose();  
 }  
  
  
 @Override  
 **public void** run() {  
 init();  
  
 *//Sets the frames per seconds* **int** fps = 30;  
 *//1 000 000 000 nanoseconds in a second. Thus we measure time in nanoseconds  
 //to be more specific. Maximum allowed time to run the tick() and render() methods* **double** timePerTick = 1\_000\_000\_000.0 / fps;  
 *//How much time we have until we need to call our tick() and render() methods* **double** delta = 0;  
 *//The current time in nanoseconds* **long** now;  
 *//Returns the amount of time in nanoseconds that our computer runs.* **long** lastTime = System.*nanoTime*();  
 **long** timer = 0;  
 **int** ticks = 0;  
  
 **while** (**running**) {  
 *//Sets the now variable to the current time in nanoseconds* now = System.*nanoTime*();  
 *//Amount of time passed divided by the max amount of time allowed.* delta += (now - lastTime) / timePerTick;  
 *//Adding to the timer the time passed* timer += now - lastTime;  
 *//Setting the lastTime with the values of now time after we have calculated the delta* lastTime = now;  
  
 *//If enough time has passed we need to tick() and render() to achieve 60 fps* **if** (delta >= 1) {  
 tick();  
 render();  
 *//Reset the delta* ticks++;  
 delta--;  
 }  
  
 **if** (timer >= 1\_000\_000\_000) {  
 System.***out***.println(**"Ticks and Frames: "** + ticks);  
 ticks = 0;  
 timer = 0;  
 }  
 }  
  
 *//Calls the stop method to ensure everything has been stopped* stop();  
 }  
  
 *//Creating a start method for the Thread to start our game  
 //Synchronized is used because our method is working with threads  
 //so we ensure ourselves that nothing will go bad* **public synchronized void** start() {  
 *//If the game is running exit the method  
 //This is done in order to prevent the game to initialize  
 //more than enough threads* **if** (**running**) {  
 **return**;  
 }  
 *//Setting the while-game-loop to run* **running** = **true**;  
 *//Initialize the thread that will work with "this" class (game.Game)* **thread** = **new** Thread(**this**);  
 *//The start method will call start the new thread and it will call  
 //the run method in our class* **thread**.start();  
 }  
  
 *//Creating a stop method for the Thread to stop our game* **public synchronized void** stop() {  
 *//If the game is not running exit the method  
 //This is done to prevent the game from stopping a  
 //non-existing thread and cause errors* **if** (!**running**) {  
 **return**;  
 }  
 **running** = **false**;  
 *//The join method stops the current method from executing and it  
 //must be surrounded in try-catch in order to work* **try** {  
 **thread**.join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

**клас InputHandler**

**package** game;  
  
 **import** display.Display;  
  
 **import** java.awt.event.KeyEvent;  
 **import** java.awt.event.KeyListener;  
  
 **public class** InputHandler **implements** KeyListener {  
  
 **public** InputHandler(Display display) {  
 display.getCanvas().addKeyListener(**this**);  
 }  
  
 @Override  
 **public void** keyPressed(KeyEvent e) {  
 **int** keyCode = e.getKeyCode();  
  
 **if** (keyCode == KeyEvent.***VK\_UP***) {  
 Game.*player*.*goingUp* = **true**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_DOWN***) {  
 Game.*player*.*goingDown* = **true**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_LEFT***) {  
 Game.*player*.*goingLeft* = **true**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_RIGHT***) {  
 Game.*player*.*goingRight* = **true**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_ESCAPE***) {  
 System.*exit*(0);  
 }  
 **if**(keyCode == KeyEvent.***VK\_SPACE***){  
 Game.*player*.*isShooting* = **true**;  
  
 }  
 }  
  
 @Override  
 **public void** keyTyped(KeyEvent e) {  
  
 }  
  
  
 @Override  
 **public void** keyReleased(KeyEvent e) {  
 **int** keyCode = e.getKeyCode();  
  
 **if** (keyCode == KeyEvent.***VK\_UP***) {  
 Game.*player*.*goingUp* = **false**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_DOWN***) {  
 Game.*player*.*goingDown* = **false**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_LEFT***) {  
 Game.*player*.*goingLeft* = **false**;  
 }  
 **if** (keyCode == KeyEvent.***VK\_RIGHT***) {  
 Game.*player*.*goingRight* = **false**;  
 }  
 **if**(keyCode == KeyEvent.***VK\_SPACE***){  
 Game.*player*.*isShooting* = **false**;  
  
 }  
 }  
}

**клас noEnemy**

**package** game;  
  
  
**import** gfx.Assets;  
  
**import** java.awt.\*;  
**import** java.awt.image.BufferedImage;  
**import** java.util.LinkedList;  
  
**public class** noEnemy {  
 **private int x**;  
 **private int y**;  
 **private int width**;  
 **private int height**;  
 **private int** dX;  
 **private int dY**;  
 **private** Rectangle **boundingBoxEnemy**;  
  
  
 BufferedImage **image**;  
  
 **public** noEnemy(**int** x, **int** y) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**width** = 64;  
 **this**.**height** = 64;  
 **this**.**image** = Assets.*noEnemy*;  
 **this**.dX = 8;  
 **this**.**dY** = 8;  
 **this**.**boundingBoxEnemy** = **new** Rectangle(**this**.**width**,**this**.**height**);  
  
  
 }  
  
 **public boolean** Intersects(Rectangle r) {  
 **if** (**this**.**boundingBoxEnemy**.contains(r) || r.contains(**this**.**boundingBoxEnemy**)) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
  
 **private** LinkedList<Enemy> **e** = **new** LinkedList<Enemy>();  
  
 Enemy **TempEnemy**;  
  
 **public void** tick() {  
  
  
 **if** (**x**<=0||(**x**+64)>= 800){  
 dX = -1\*dX;  
 }  
 **if** (**y**<=0||(**y**+64 )>= 600){  
 **dY**= -1\***dY**;  
 }  
 **x** -=dX;  
 **y** -=**dY**;  
  
 **for** (**int** i = 0; i < **e**.size(); i++) {  
 **TempEnemy** = **e**.get(i);  
 **TempEnemy**.tick();  
  
 }  
 *// y -= 5;* **this**.**boundingBoxEnemy**.setBounds(**this**.**x**, **this**.**y**, **this**.**width**, **this**.**height**);  
  
  
 }  
  
 **public void** render(Graphics g) {  
 **for** (**int** i = 0; i < **e**.size(); i++) {  
 **TempEnemy** = **e**.get(i);  
 **TempEnemy**.render(g);  
 }  
 g.drawImage(**image**, **x**, **y**, **null**);  
  
  
 }  
  
 **public void** addEnemy(Enemy block) {  
 **e**.add(block);  
 }  
  
 **public void** removeEnemy(Enemy block) {  
 **e**.remove(block);  
 }  
  
  
}

**клас Player**

**package** game;  
  
**import** gfx.Assets;  
**import** javafx.geometry.BoundingBox;  
  
**import** java.awt.\*;  
**import** java.util.LinkedList;  
  
  
**public class** Player {  
 **private int x**, **y**,**dX**,**dY**;  
 **private int velocity**;  
 **private int width**, **height**;  
 **private int health**;  
  
 **private** Rectangle **boundingBox**;  
  
 **public static boolean** *goingUp*;  
 **public static boolean** *goingDown*;  
 **public static boolean** *goingLeft*;  
 **public static boolean** *goingRight*;  
 **public static boolean** *isShooting*;  
  
 **private** LinkedList<Bullet> **b** = **new** LinkedList<Bullet>();  
  
 Bullet **TempBullet**;  
  
  
 **public** Player() {  
  
 **this**.**x** = 200;  
 **this**.**y** = 500;  
 **this**.**dX** = 1;  
 **this**.**dY** = 1;  
 **this**.**width** = 112;  
 **this**.**height** = 124;  
 **this**.**health** = 50;  
 **this**.**velocity** = 2;  
 **this**.**boundingBox** = **new** Rectangle(**this**.**width**,**this**.**height**);  
  
 *goingUp* = **false**;  
 *goingDown* = **false**;  
 *goingLeft* = **false**;  
 *goingRight* = **false**;  
 }  
  
 **public int** getHealth() {  
 **return this**.**health**;  
 }  
 **public int** getX(){**return this**.**x**;}  
 **public int** getY(){**return this**.**y**;}  
  
 *//Checks if the player intersects with something* **public boolean** Intersects(Rectangle r) {  
 **if** (**this**.**boundingBox**.contains(r) || r.contains(**this**.**boundingBox**)) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
  
 *//Update the movement of the player* **public void** tick() {  
  
  
 **if** (**x**<=0||(**x**+112)>= 800) {  
  
 **dX** =-1\***dY**;  
 }  
 **if** (**y**<=0||(**y**+124)>= 600){  
  
 **dY** = -1\***dY**;  
 }  
 *//x+=dX;  
 //y+=dY;* **for** (**int** i = 0; i < **b**.size(); i++) {  
 **TempBullet** = **b**.get(i);  
 **TempBullet**.tick();  
 }  
 *//Update the bounding box's position* **this**.**boundingBox**.setBounds(**this**.**x**, **this**.**y**, **this**.**width**, **this**.**height**);  
  
 **if** (*goingUp*) {  
 **this**.**y** -= **this**.**velocity**;  
  
  
 }  
 **if** (*goingDown*) {  
 **this**.**y** += **this**.**velocity**;  
  
 }  
 **if** (*goingLeft*) {  
 **this**.**x** -= **this**.**velocity**;  
 }  
 **if** (*goingRight*) {  
 **this**.**x** += **this**.**velocity**;  
 }  
 **if** (*isShooting*) {  
 addBullet(**new** Bullet(**this**.**x**, **this**.**y**));  
 }  
 }  
  
 *//Draws the player* **public void** render(Graphics g) {  
 **for** (**int** i = 0; i < **b**.size(); i++) {  
 **TempBullet** = **b**.get(i);  
 **TempBullet**.render(g);  
 }  
  
 g.drawImage(Assets.*player1*, **this**.**x**, **this**.**y**, **null**);  
 }  
  
 **public void** addBullet(Bullet block) {  
 **b**.add(block);  
 }  
  
 **public void** removeBullet(Bullet block) {  
 **b**.remove(block);  
 }  
}

**Launcher**

**package** game;  
  
**public class** Launcher {  
 **public static void** main(String[] args) {  
 Game game = **new** Game(**"Междупланетни войни"**, 800, 600);  
 game.start();  
 }  
}

* ***Папка gfx***

***Клас Assets***

**package** gfx;  
  
**import** java.awt.image.BufferedImage;  
  
**public class** Assets {  
  
 **private static final int *width*** = 95, ***height*** = 130;  
 **public static** BufferedImage *bullet*,*enemy*,*noEnemy*;  
 **public static** BufferedImage *player1*, *player2*;  
 *//Loads every resource needed for the game* **public static void** init() {  
  
 SpriteSheet sheet = **new** SpriteSheet(ImageLoader.*loadImage*(**"/textures/player.png"**));  
  
 *player1* = sheet.crop(0, 0, ***width***, ***height***);  
 *player2* = sheet.crop(***width***, 0, ***width***, ***height***);  
 *bullet* = ImageLoader.*loadImage*(**"/textures/bullet.png"**);  
 *enemy* = ImageLoader.*loadImage*(**"/textures/enemy.png"**);  
 *noEnemy* = ImageLoader.*loadImage*(**"/textures/noEnemy.png"**);  
  
 }  
}

**клас ImageLoader**

**package** gfx;  
  
**import** javax.imageio.ImageIO;  
**import** java.awt.image.BufferedImage;  
**import** java.io.IOException;  
  
**public class** ImageLoader {  
  
 *//Method that returns an image* **public static** BufferedImage loadImage(String path) {  
 **try** {  
 **return** ImageIO.*read*(ImageLoader.**class**.getResource(path));  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 *//Exits the application* System.*exit*(1);  
 }  
  
 *//Ensures that no errors are passed* **return null**;  
 }  
}

***клас SpriteSheet***

**package** gfx;  
  
**import** java.awt.image.BufferedImage;  
  
**public class** SpriteSheet {  
  
 **private** BufferedImage **sheet**;  
  
 **public** SpriteSheet(BufferedImage sheet) {  
 **this**.**sheet** = sheet;  
 }  
  
 *//Method that crops the image from x to width and from y to height* **public** BufferedImage crop(**int** x, **int** y, **int** width, **int** height) {  
 *//Returns a new image in the area we specified* **return sheet**.getSubimage(x, y, width, height);  
 }  
}

* ***Папка states***

***Клас GameState***

**package** states;  
  
  
**import** gfx.Assets;  
  
**import** java.awt.\*;  
  
**public class** GameState **extends** State{  
  
 **public** GameState() {  
  
 }  
  
 @Override  
 **public void** tick() {  
  
 }  
  
 @Override  
 **public void** render(Graphics g) {  
 }  
}

***Клас MenuState***

**package** states;  
  
**import** java.awt.\*;  
  
**public class** MenuState **extends** State{  
  
 **public** MenuState() {  
  
 }  
  
 @Override  
 **public void** tick() {  
  
 }  
  
 @Override  
 **public void** render(Graphics g) {  
  
 }  
}

**Клас SettingsState**

**package** states;  
  
**import** java.awt.\*;  
  
**public class** SettingsState **extends** State {  
  
 **public** SettingsState() {  
  
 }  
  
 @Override  
 **public void** tick() {  
  
 }  
  
 @Override  
 **public void** render(Graphics g) {  
  
 }  
}

**Клас State**

**package** states;  
  
  
**import** java.awt.\*;  
  
**public abstract class** State {  
  
 *//Every state has it's own tick() method* **public abstract void** tick();  
  
 *//Every state has it's own render() method* **public abstract void** render(Graphics g);  
  
}

**Клас StateManager**

**package** states;  
  
  
**public class** StateManager {  
 *//It will hold what state we currently want to tick() and render()* **private static** State *currentState* = **null**;  
  
 *//Changes the current state of the game* **public static void** setState(State state) {  
 *currentState* = state;  
 }  
  
 *//Gets the current state of the game* **public static** State getState() {  
 **return** *currentState*;  
 }

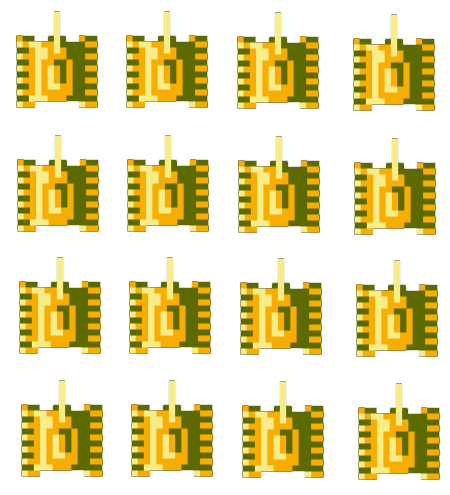
}

**🡪папка RES**

**Папка TEXTURES**



bckg.jpg



player.png



enemy



bullet.png enemy.png noEnemy.png