Acknowledgement

We would like to thank our course instructors Syed Muhammad Hassan & Ali Mobin for all their help and advice with this project. We would also like to thank people & the internet for the brilliant ideas/references,whom without these this project would have not been possible. We also appreciate all the support and platform to be creative that we received from the university.

We are highly indebted to our instructors for their guidance and constant supervision as well as for providing necessary information regarding the project.

And our special gratitude towards all member for their kind co-operationand encouragement which help us in completion of this project.

Table of Content:

Project Title:

Wizard Shooting Game.

Overview:

This project is based on only java programming language with the concepts of object oriented programming, the platform and tools we've used is Eclipse IDE, GUI, Paint to showcase everything as decent as possible.

Description:

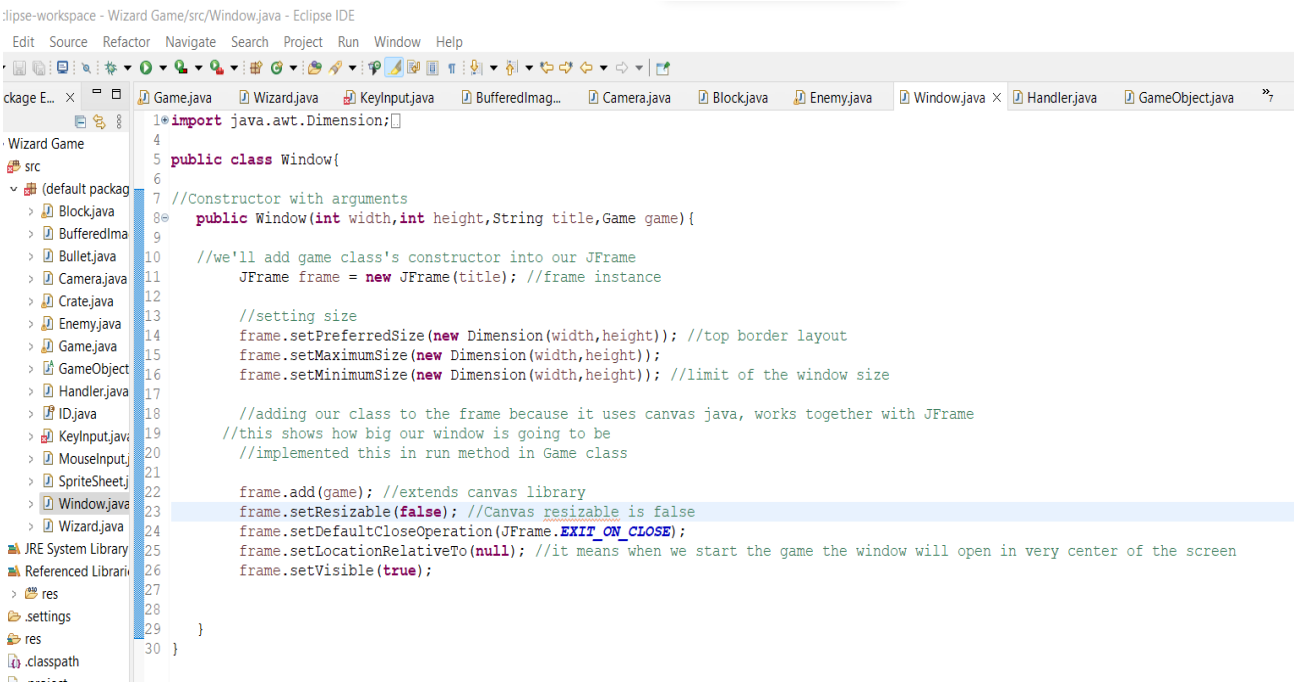
As the main concept of our project revolves around game development. We have make a single player shooting game with a multiple enemies. We've designed the map, complexities, characters to make it more fun. The gamer has to keep track of the enemy, (where they're going and how fast they're moving), focus the crosshairs on the enemy, and make sure gunfire is hitting the target while keeping an eye on their health and ammo.

SOURCE CODE:

Classes:

1. Window Class (Controls GUI):

The window class is going to be just a class that we call right off the bat to create our game window on the GUI.



2. Game Class (Main Class):

Game class this is going to be our ***main class*** this is where everything gets called this is where our game loop is going to be, this is where everything is going to happen so from here from the game class, the game class might it's going to handle all like the updating for X & Y positions for our characters in the game, it's going to handle all the rendering so all of the you know drawing the graphics improves like that, this is game class is going to be handling all that.

```java
//Main Class
import java.awt.Canvas;

public class Game extends Canvas implements Runnable{

   private static final long serialVersionUID = 1L; //so it doesn't give as warning

    private boolean isRunning=false;
    private Thread thread; //runs simultaneously with our actual program so when we start up this thread what it's going to do is cal
    private Handler handler;
    private Camera camera;
    private SpriteSheet ss;

   private BufferedImage level = null;
   private BufferedImage sprite_sheet = null;
   private BufferedImage floor = null;

   public int ammo = 100;
   public int hp = 100;

   ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

   //Constructor where we call our window, the main method constructor
    public Game(){
        //in this constructor we are going call our window class
        new Window(1000,563,"Wizard Game", this); //setting dimensions
        start();

        handler = new Handler();
        camera = new Camera(0, 0);
        this.addKeyListener(new KeyInput(handler));

        BufferedImageLoader loader = new BufferedImageLoader();
        level = loader.loadImage("/wizard_level.png");
        sprite_sheet = loader.loadImage("/sprite_sheet.png");

        ss = new SpriteSheet(sprite_sheet);

        floor = ss.grabImage(4, 2, 32, 32);

        this.addMouseListener(new MouseInput(handler, camera, this, ss));

        loadLevel(level);
```

```java
        loadLevel(level);
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    //Start our thread
   private void start(){
       isRunning = true;
       thread = new Thread(this); //we use "this" because we are calling run method
       thread.start();
    }
   //stop our thread
   private void stop(){
       isRunning = false;
       //try & catch method in case it fails
       try {
           thread.join();
       }
       catch(InterruptedException e) {
           e.printStackTrace();

   }
}

   ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//run method for window class, so it'll not show errors
   //upgrades the game 60 times a second
   //this is where our game loop is going to be stored
   public void run(){
       //Actual game loop of our program
       //making the instance and whenever we make an instance we'll call window constructor
       this.requestFocus();
       long lastTime = System.nanoTime();
       double amountOfTicks = 60.0;
       double ns = 1000000000/amountOfTicks;
       double delta = 0;
       long timer = System.currentTimeMillis();
       int frames = 0;
       while(isRunning){
           long now = System.nanoTime();
           delta +=(now - lastTime)/ns;
           lastTime = now;
          while(delta>=1){
```

Wizard Game/src/Game.java - Eclipse IDE

Refactor Navigate Search Project Run Window Help

*Game.java

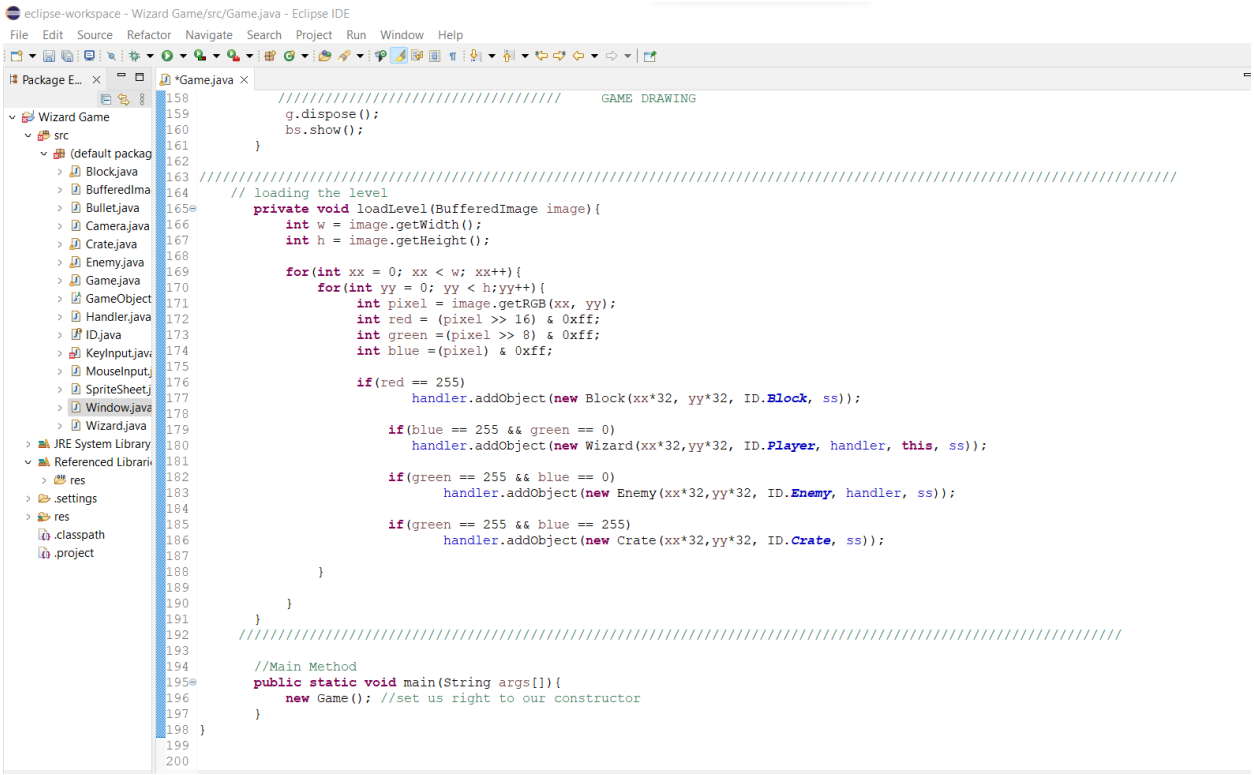
```
            long now = System.nanoTime();
            delta +=(now - lastTime)/ns;
            lastTime = now;
           while(delta>=1){
                tick();

                delta--;
           }
            render();
            frames++;
            if(System.currentTimeMillis() - timer>1000){
                timer+=1000;
                frames=0;

            }
     }
        stop();
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    //this method will update everything in the game
    public void tick() {

        for(int i=0; i<handler.object.size();i++){
             if(handler.object.get(i).getId() == ID.Player){
                 camera.tick(handler.object.get(i));
             }
        }
        handler.tick();
        }
    //to handle graphics and images
        public void render(){

            BufferStrategy bs = this.getBufferStrategy();
            if(bs == null){
                this.createBufferStrategy(3); //pre-loading frames behind the actual window
               return;
            }

     ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

            //drawings
            Graphics g = bs.getDrawGraphics();
```

Wizard Game/src/Game.java - Eclipse IDE

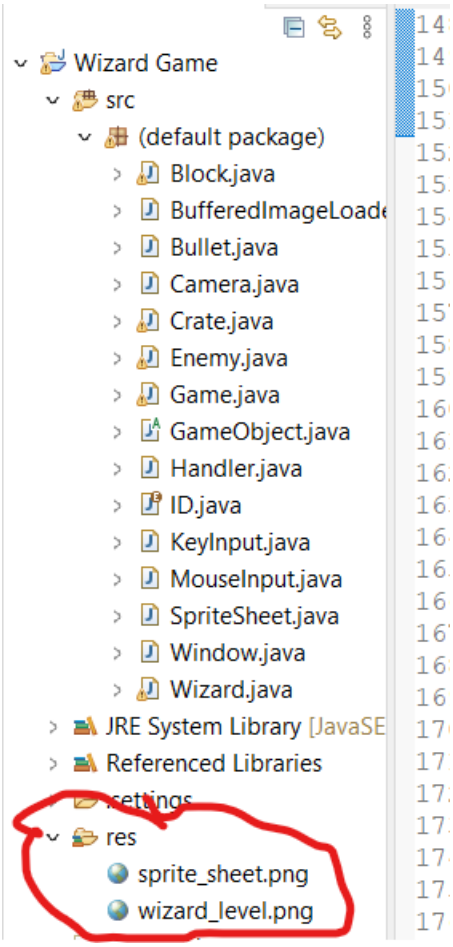
Refactor Navigate Search Project Run Window Help

*Game.java

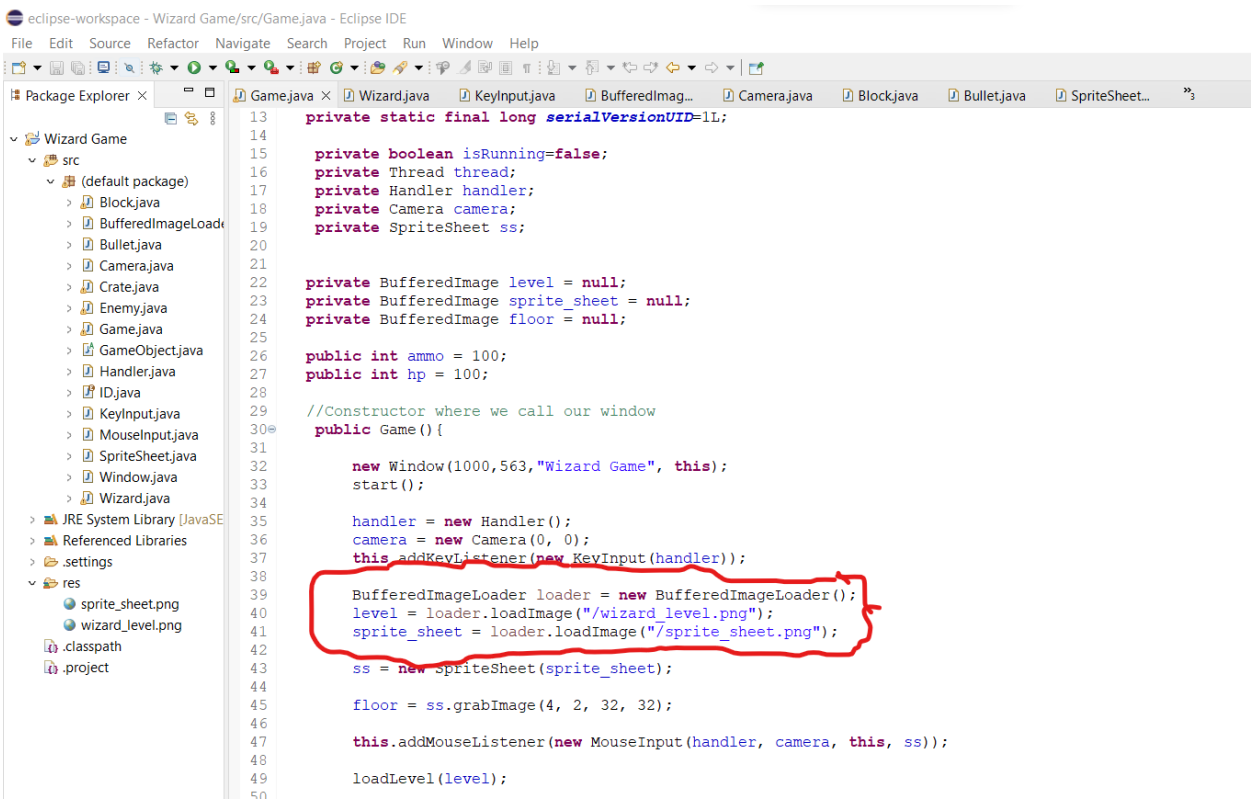
```
            Graphics g = bs.getDrawGraphics();
            Graphics2D g2d = (Graphics2D) g;
            ////////////////////////////////////     GAME DRAWING

            g2d.translate(-camera.getX(), -camera.getY());

            for(int xx = 0; xx < 30*72; xx+=32){
                 for(int yy = 0;yy < 30*72; yy+=32){
                     g.drawImage(floor, xx, yy, null);
                 }
            }

            handler.render(g);

            g2d.translate(camera.getX(), camera.getY());

            g.setColor(Color.gray);
            g.fillRect(5, 5, 900, 32);
            g.setColor(Color.green);
            g.fillRect(5, 5, hp*9, 32);
            g.setColor(Color.black);
            g.drawRect(5, 5, 900, 32);

            g.setColor(Color.white);
            g.drawString("Ammo: "  +ammo, 5, 50);


           ////////////////////////////////////     GAME DRAWING
            g.dispose();
            bs.show();
        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    // loading the level
       private void loadLevel(BufferedImage image){
            int w = image.getWidth();
            int h = image.getHeight();

            for(int xx = 0; xx < w; xx++){
                for(int yy = 0; yy < h;yy++){
                     int pixel = image.getRGB(xx, yy);
                     int red = (pixel >> 16) & 0xff;
                     int green =(pixel >> 8) & 0xff;
```



Res Folder (where paint files are added):



Implemented in main class:

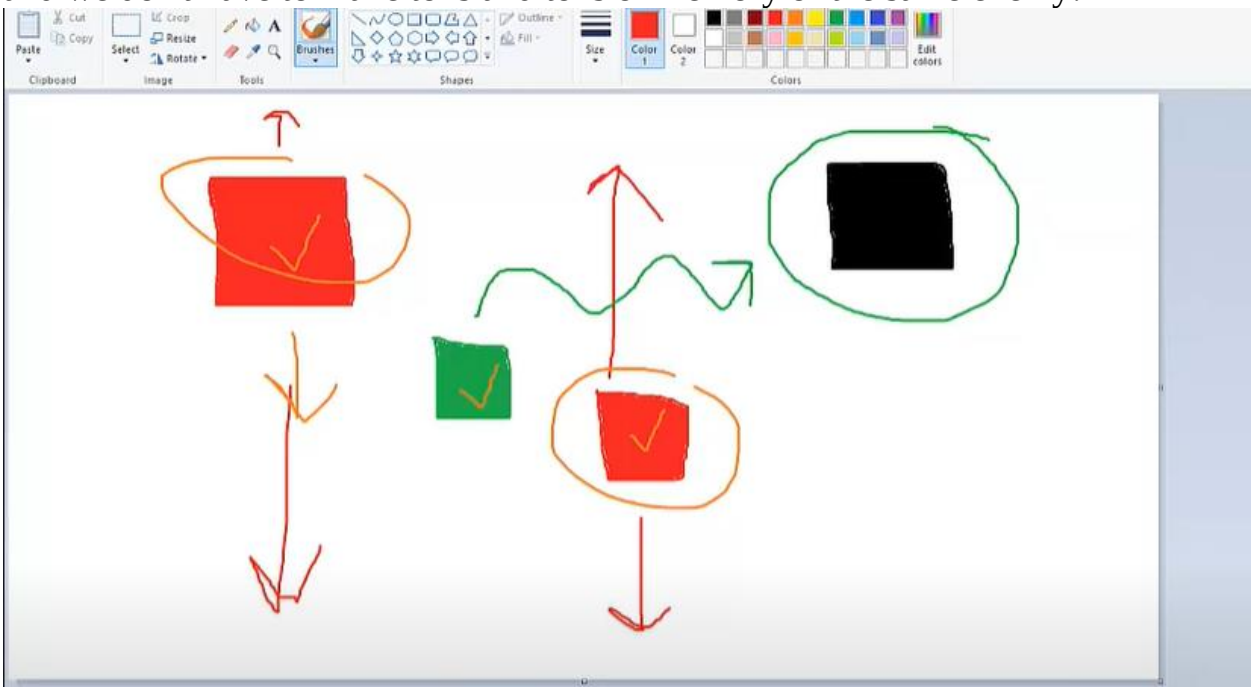


3. Game Object Class (For the objects):

Concept:

What we're going to do is look at creating game objects and our handler class to handle all of the game objects enter game, what a game object really is? So, in paint and let's say this is our window right this is our actual game and let's say we want to create, you know a box right and this is going to be like a box that maybe you can move around and do different things right so generally speaking then if we wanted to create this box, we can create a class, a key input everything like that let's say, we wanted to make an enemy now all right so there's our enemy let's say this enemy just moves up and down. We wanted to make another enemy though the same thing just moves up and down all right up and down, we wanted to make another enemy that maybe moves towards our player, so if we actually keep going with this and this is just one level of our game we're now looking at creating four different class files for each one of these guys because we can't even if we have an enemy one an enemy two that does the same thing we can't make two different instances of the same class, we can't say new enemy and then like update our enemy and they're like destroy it because if we destroy this class then it's going to destroy this class like the health goes down, so it gets really confusing and what if we had over 50 enemies in our

game, we need 50 different classes and we wanted to change just one argument in one of these to make this enemy move up and down faster right now, we got to go into 50 different class files and change that it's just so unofficial so what we need to do is create one object that controls everything. One object that controls a block, one object that controls our create, one object that controls our player, one object that controls our enemies different enemies that shoot, that run from you, enemies that explode all controlled with one object and that is our game object and this is what ***object-oriented programming*** is so essentially now exactly just going to be just one object but instead of having 50 different instances of just our same enemy, we can have one instance of that enemy and then from that we can create multiple ones and then if we want to change something we don't have to go into 50 different classes and we don't have to make tons and tons of memory of the same enemy.



eclipse-workspace - Wizard Game/src/GameObject.java - Eclipse IDE

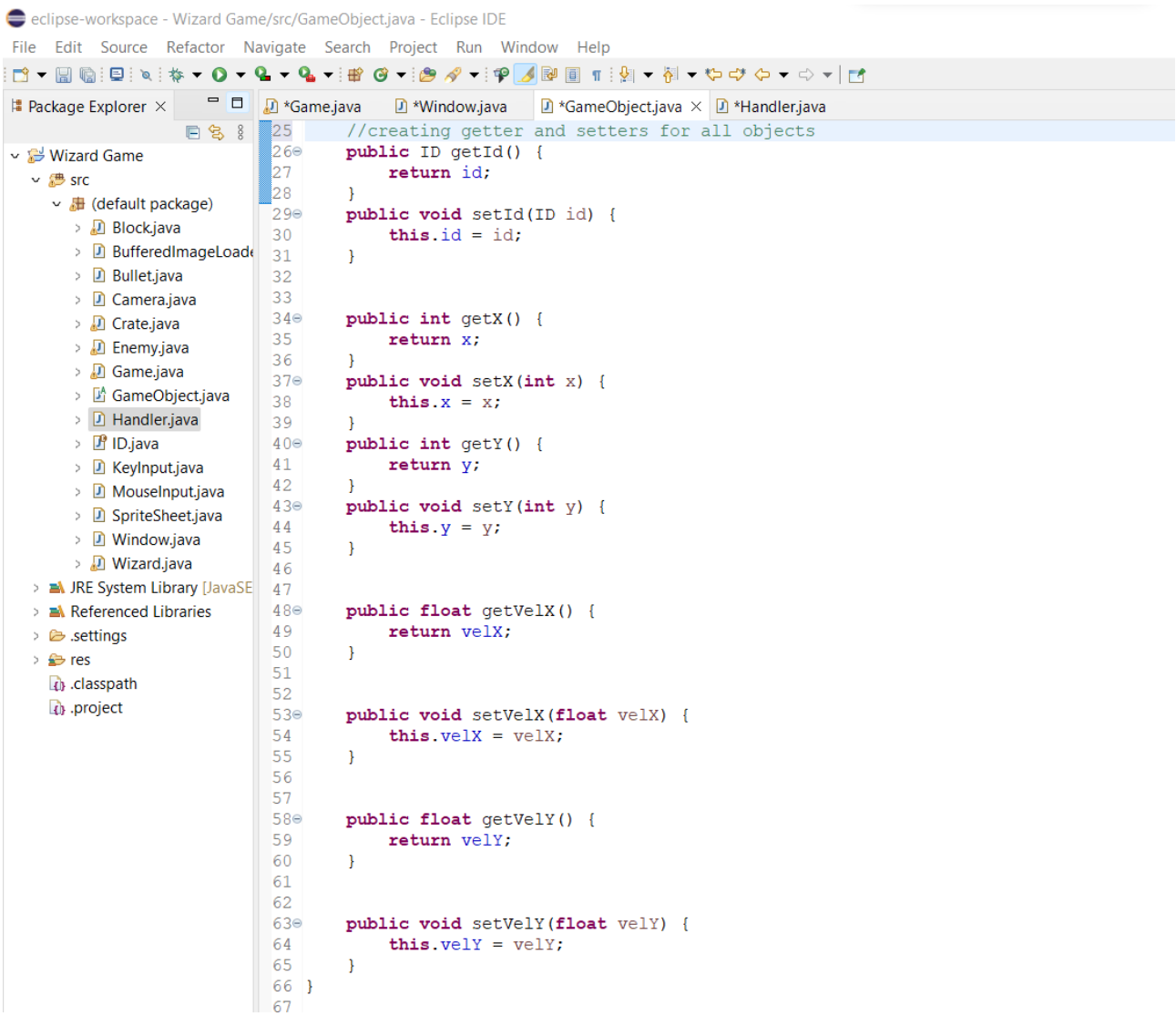
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Wizard Game
  - src
    - (default package)
      - Block.java
      - BufferedImageLoad
      - Bullet.java
      - Camera.java
      - Crate.java
      - Enemy.java
      - Game.java
      - GameObject.java
      - Handler.java
      - ID.java
      - KeyInput.java
      - MouseInput.java
      - SpriteSheet.java
      - Window.java
      - Wizard.java
  - JRE System Library [JavaSE
  - Referenced Libraries
  - .settings
  - res
  - .classpath
  - .project

*Game.java | *Window.java | *GameObject.java | *Handler.java

```java
import java.awt.Graphics;
//Abstract class
public abstract class GameObject{

    protected int x, y;
    protected float velX = 0,velY = 0; //set velocity, speed of our objects going
    protected ID id;
    protected SpriteSheet ss;

    //Constructor
    public GameObject(int x,int y, ID id, SpriteSheet ss){
        this.x = x;
        this.y = y;
        this.id = id;
        this.ss = ss;
    }


    public abstract void tick(); //every object needs upgrade
    public abstract void render(Graphics g);//every objects needs to draw something/appear to be something
    public abstract Rectangle getBounds(); //for collision detection, every object will be rectangle


    //creating getter and setters for all objects
    public ID getId() {
        return id;
    }
    public void setId(ID id) {
        this.id = id;
    }


    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
```




```
    //creating getter and setters for all objects
    public ID getId() {
        return id;
    }
    public void setId(ID id) {
        this.id = id;
    }


    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }


    public float getVelX() {
        return velX;
    }


    public void setVelX(float velX) {
        this.velX = velX;
    }


    public float getVelY() {
        return velY;
    }


    public void setVelY(float velY) {
        this.velY = velY;
    }
}
```

4. Handler Class (To handle all objects):

We make handler class because this is going to handle all of our objects and here we going to create a new linked list game object, we call it object and equal to a new linked list game object, now linked list is essentially an array of objects, so it gets a bunch of other different information that we need we can also use the linked list functions to get a certain ID of one object over another.

eclipse-workspace - Wizard Game/src/Handler.java - Eclipse IDE

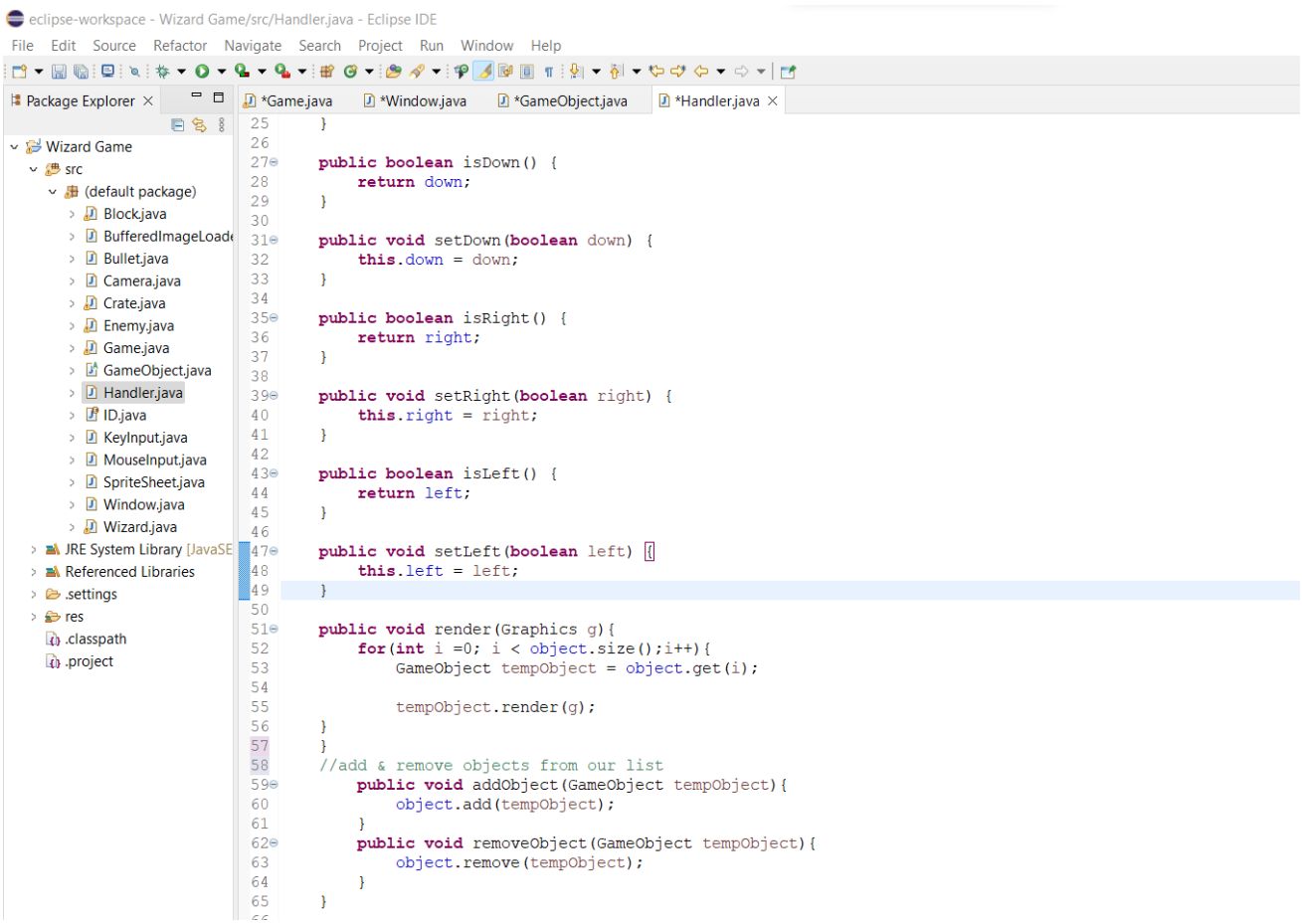
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Wizard Game
  - src
    - (default package)
      - Block.java
      - BufferedImageLoad
      - Bullet.java
      - Camera.java
      - Crate.java
      - Enemy.java
      - Game.java
      - GameObject.java
      - Handler.java
      - ID.java
      - KeyInput.java
      - MouseInput.java
      - SpriteSheet.java
      - Window.java
      - Wizard.java
  - JRE System Library [JavaSE
  - Referenced Libraries
  - .settings
  - res
  - .classpath
  - .project

*Game.java *Window.java *GameObject.java *Handler.java

```java
import java.awt.Graphics;
//handle all of our objects
public class Handler{


    LinkedList <GameObject> object = new LinkedList<GameObject>();

    private boolean up = false, down = false, right = false, left = false;

    public void tick(){ //update all our game objects
        for(int i=0; i < object.size();i++){ //this loop will run in our all game objects
            GameObject tempObject = object.get(i); //temporary game object

            tempObject.tick();
        }
    }

    public boolean isUp() {
        return up;
    }

    public void setUp(boolean up) {
        this.up = up;
    }

    public boolean isDown() {
        return down;
    }

    public void setDown(boolean down) {
        this.down = down;
    }

    public boolean isRight() {
        return right;
    }

    public void setRight(boolean right) {
        this.right = right;
    }

    public boolean isLeft() {
        return left;
```



5. ID Class (Objects proper ID)

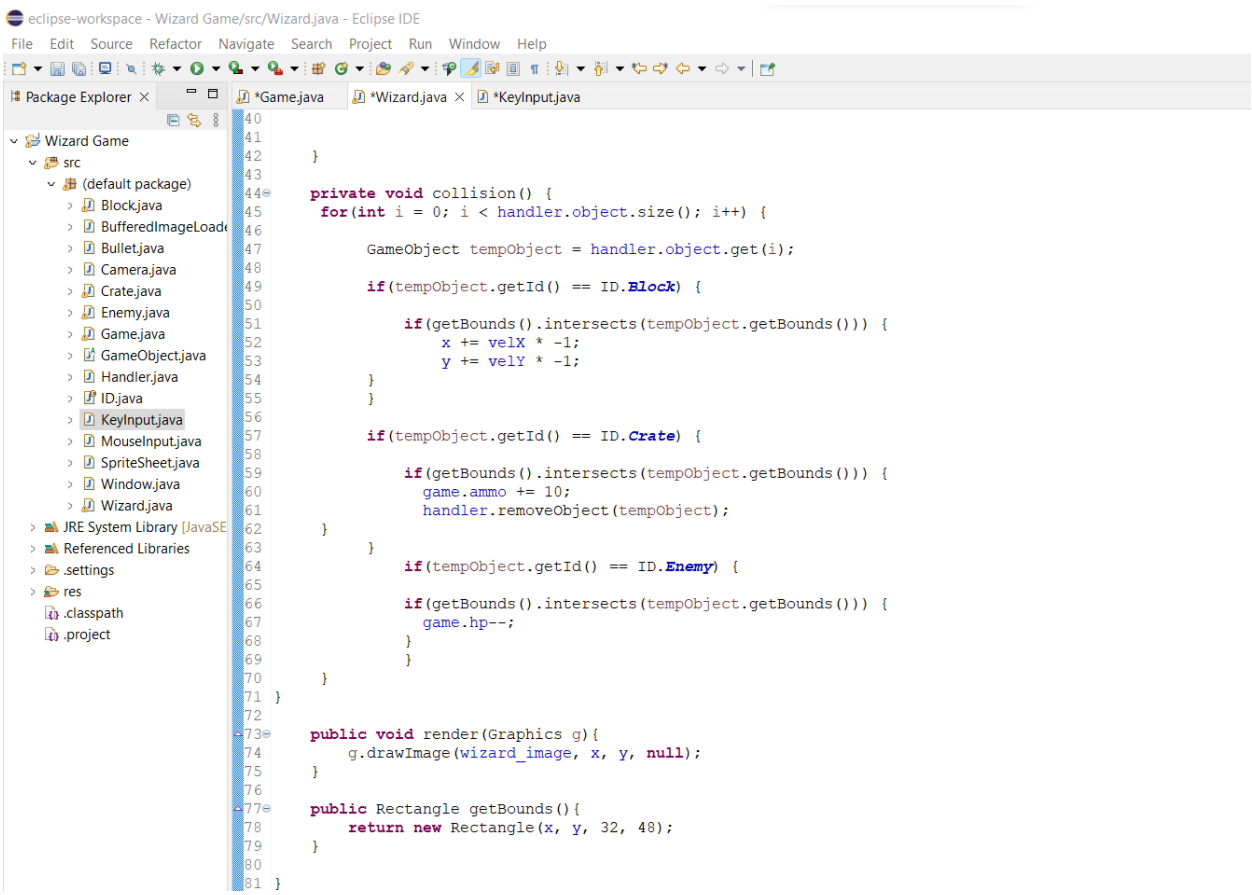
We just going to create a new enum and going to call it ID and all capitals there and essentially, what an enum is our enumeration is the concept of okay you can basically put an ID on something but you can just call it whatever you'd like so like instead of like maybe say having an integer of like, if, int ID equals one then it's a player, if ID two equals or if the ID equals two then, it's a box something like that instead, we can just in this ID enumeration here put let's say player and if we do that then now we have this player ID that we can call “going to go ahead” and put everything in this ID. Now we're going to have in our game so what do we have, we have by ID, we have our block that we're going to have in the game, we have what's there crate so we can pick up ammo, we have our bullet that we're going to have eventually and let's say we have our enemy We think that's their anything so we're going to have five separate objects in this game they're all going to be game objects every one you see so we have four player block, create bullet man, our enemy all going to be game objects and now they have their proper IDs.

```java
public enum ID {      // enum class to represent a group of constants means unchangeable variable like final variables
    Player(),
    Block(),
    Bullet(),
    Enemy(),
    Crate(),
}
```

6. Wizard Class (Main Character Functions):

Here we have described our main characters features and how it'll deal with the enemies, collect crate and the movements of the character.

```java
import java.awt.Color;
//Our main Character
public class Wizard extends GameObject{

    Handler handler;
    Game game;

    private BufferedImage wizard_image ;

    public Wizard(int x,int y,ID id, Handler handler, Game game, SpriteSheet ss){
        super(x, y, id, ss);
        this.handler = handler;
        this.game = game;

        wizard_image = ss.grabImage(1, 1, 32, 48);

    }
    public void tick(){
        x += velX;
        y += velY;

        collision();


        // movement
        if(handler.isUp())velY = -5;
        else if(!handler.isDown())velY=0;

        if(handler.isDown())velY=5;
        else if(!handler.isUp())velY=0;

        if(handler.isRight())velX=5;
        else if(!handler.isLeft())velX=0;

        if(handler.isLeft())velX=-5;
        else if(!handler.isRight())velX=0;


    }

    private void collision() {
     for(int i = 0; i < handler.object.size(); i++) {
```



7. KeyInput Class (Initializing the keys for movements):

The wizards X position like goes to the right 3 pixels, if it's being held but what we want to have happen is that it goes to the right three pixels but if you immediately like go to the left then you're going to have no lag whatsoever. All right now it shouldn't make a lot of sense to you right now, especially if you've never done any sort of key input with Java but essentially, originally here or you know if you press the key then it goes to the right this amount of pixels, you have lag between the keys and it really just plays for it's just an awful play experience if you're hitting the left key and it's not going left right away you really want to have that nice stable control.

Wizard Game/src/KeyInput.java - Eclipse IDE

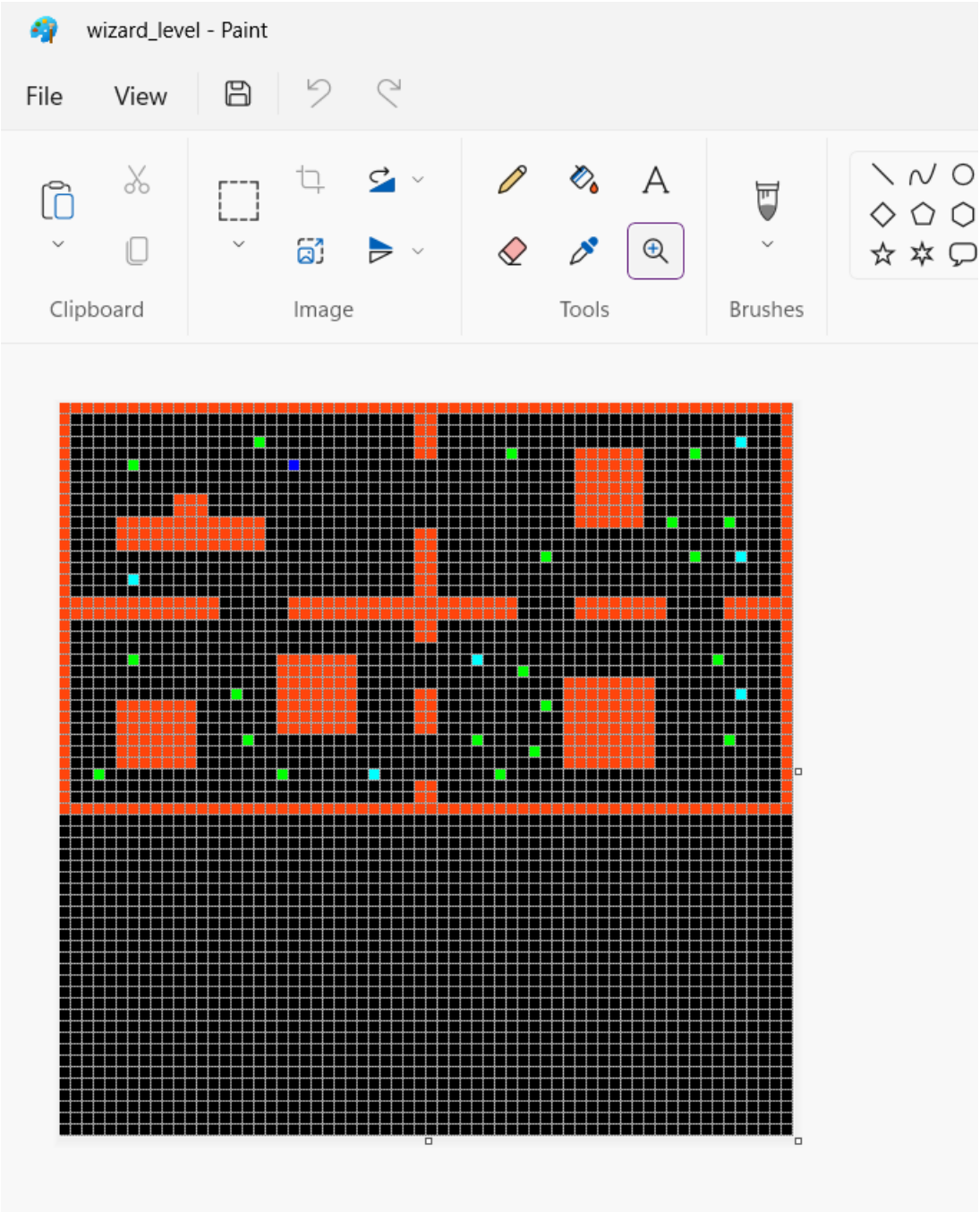
Refactor Navigate Search Project Run Window Help

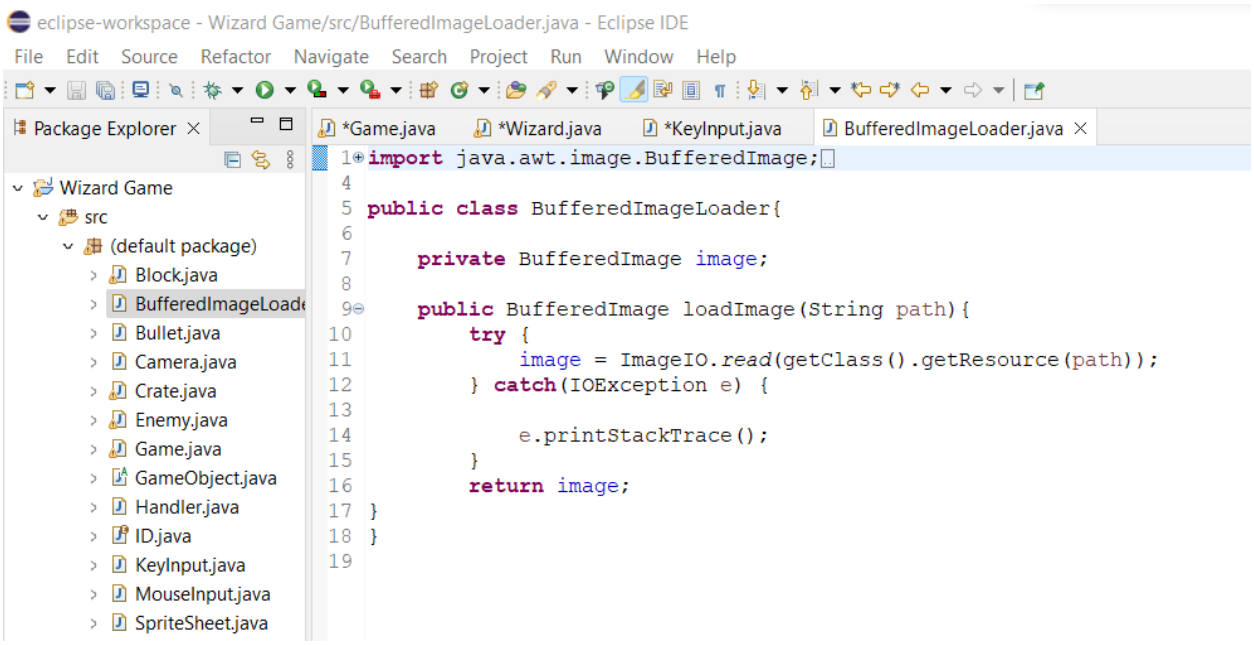
*Game.java *KeyInput.java

```java
//initializes the keys for movements

import java.awt.event.KeyAdapter;

public class KeyInput extends KeyAdapter{

    Handler handler;

    public KeyInput(Handler handler) {
        this.handler = handler;
    }
//initialized & handled in Handler class
    public void keyPressed(KeyEvent e){
        int key = e.getKeyCode();

        for(int i=0;i<handler.object.size();i++){
            GameObject tempObject=handler.object.get(i);

            //setting up the keys for the movement of player when pressed
            if(tempObject.getId() == ID.Player){
                if(key == KeyEvent.VK_W)handler.setUp(true);
                if(key == KeyEvent.VK_S)handler.setDown(true);
                if(key == KeyEvent.VK_A)handler.setLeft(true);
                if(key == KeyEvent.VK_D)handler.setRight(true);
            }
        }
        }
    //method when keys are released
   public void keyReleased(KeyEvent e){
            int key = e.getKeyCode();
            for(int i=0;i<handler.object.size();i++){
                GameObject tempObject=handler.object.get(i);

                if(tempObject.getId() == ID.Player){
                    if(key == KeyEvent.VK_W)handler.setUp(false);
                    if(key == KeyEvent.VK_S)handler.setDown(false);
                    if(key == KeyEvent.VK_A)handler.setLeft(false);
                    if(key == KeyEvent.VK_D)handler.setRight(false);
                }
            }
   }
   }
```

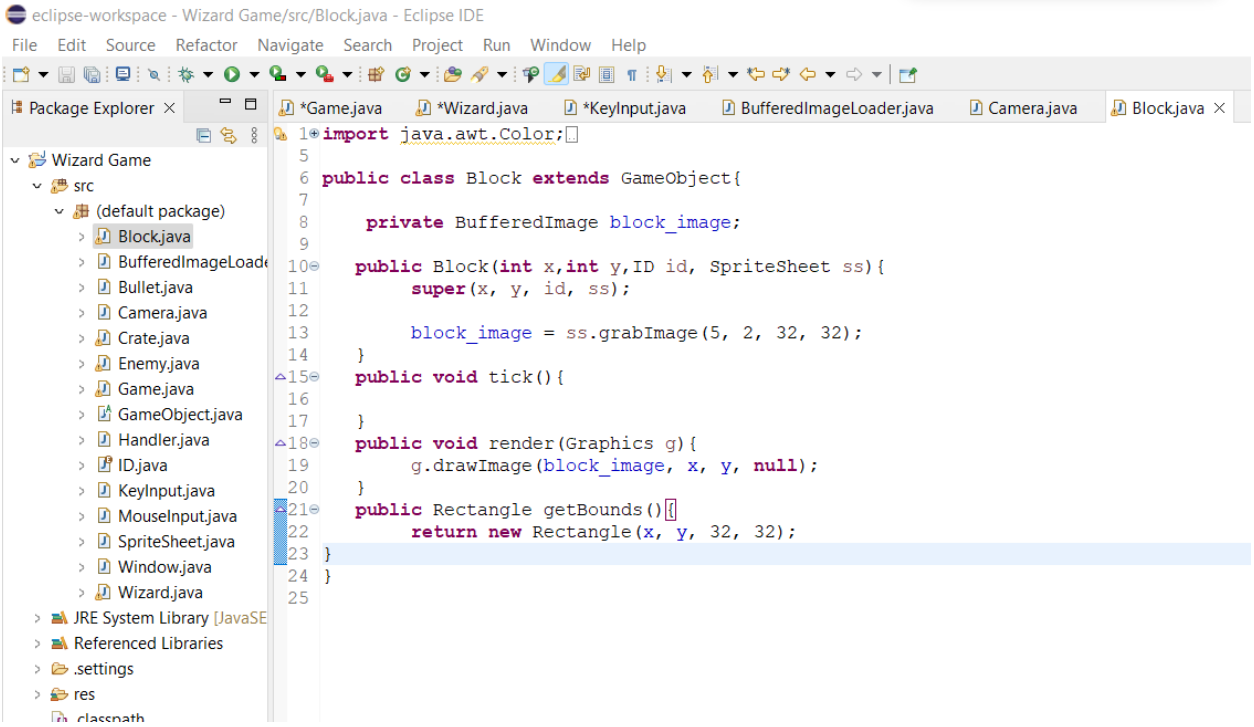
8. BufferedImageLoader Class (for implementation of map design):

Now we have draw the raw map using the paint and gonna implement that in the code. We are actually gonna make a folder under the name “res” and copy paste our paint files there and copy the URL of that file in the main class “Game class”.


wizard_level - Paint
File
View
Clipboard
Image
Tools
Brushes

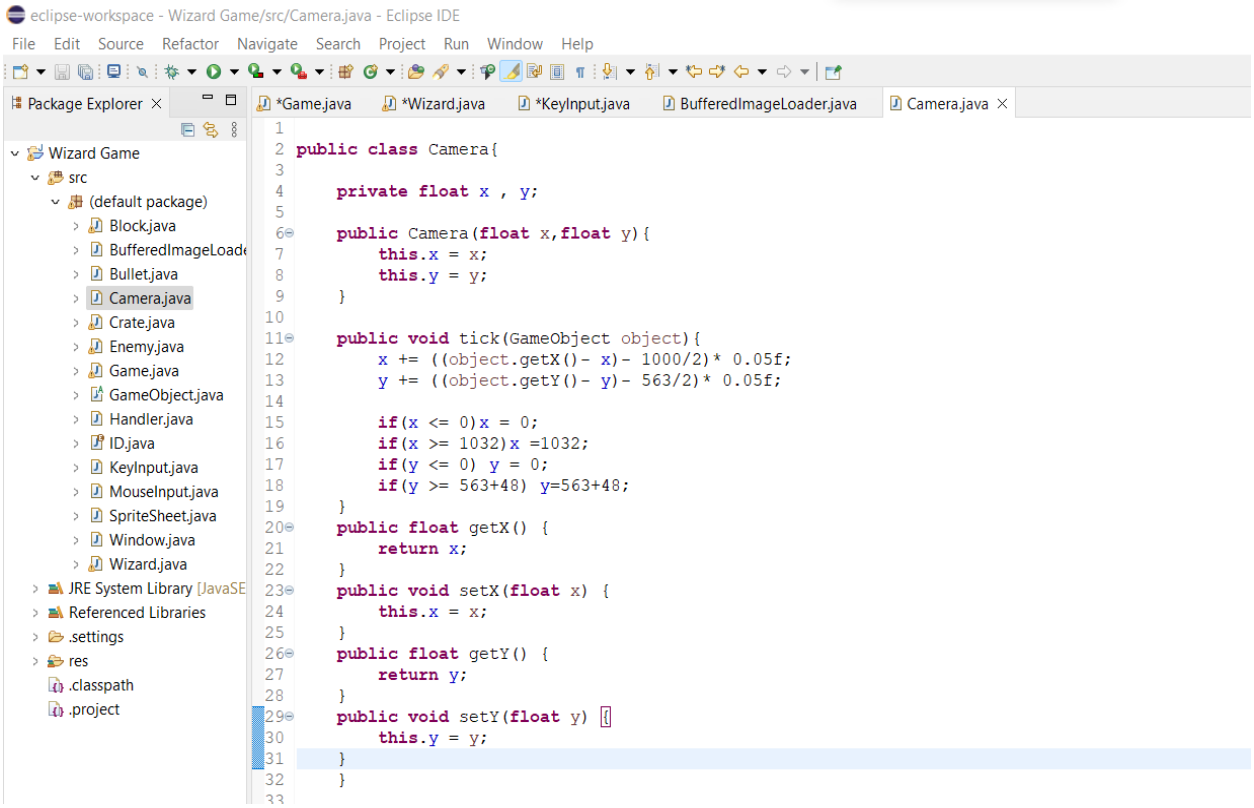


9. Block Class (For the player to not get through the map blocks):



10. Camera Class (To move the view angle with the player):

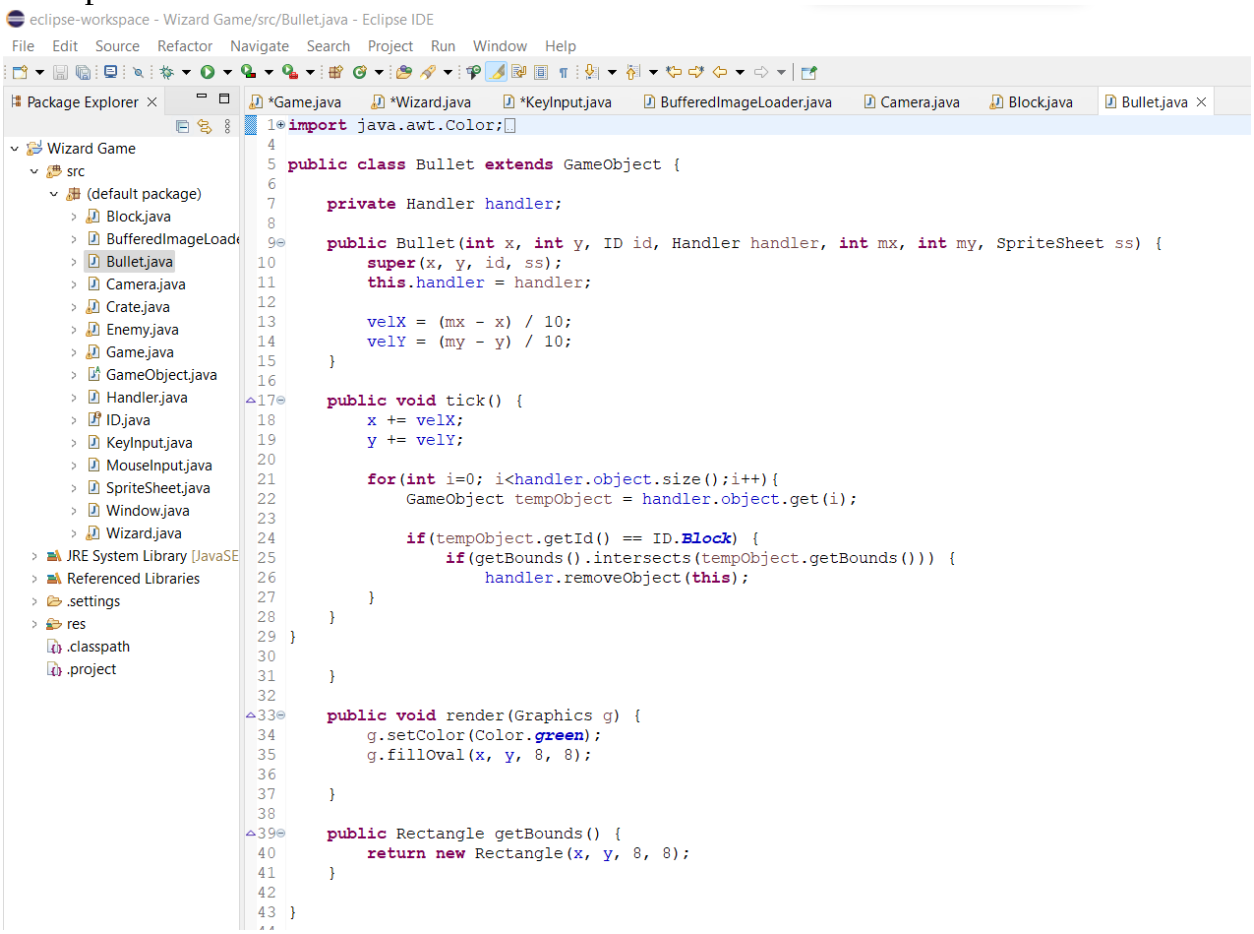
Now we have this level here that we've created, there's more to explore here but we physically can't in the game why because we need a camera that's going to follow our character so that's what we're going to make.




```
public class Camera{

    private float x , y;

    public Camera(float x,float y){
        this.x = x;
        this.y = y;
    }

    public void tick(GameObject object){
        x += ((object.getX()- x)- 1000/2)* 0.05f;
        y += ((object.getY()- y)- 563/2)* 0.05f;

        if(x <= 0)x = 0;
        if(x >= 1032)x =1032;
        if(y <= 0) y = 0;
        if(y >= 563+48) y=563+48;
    }
    public float getX() {
        return x;
    }
    public void setX(float x) {
        this.x = x;
    }
    public float getY() {
        return y;
    }
    public void setY(float y) {
        this.y = y;
    }
    }
```

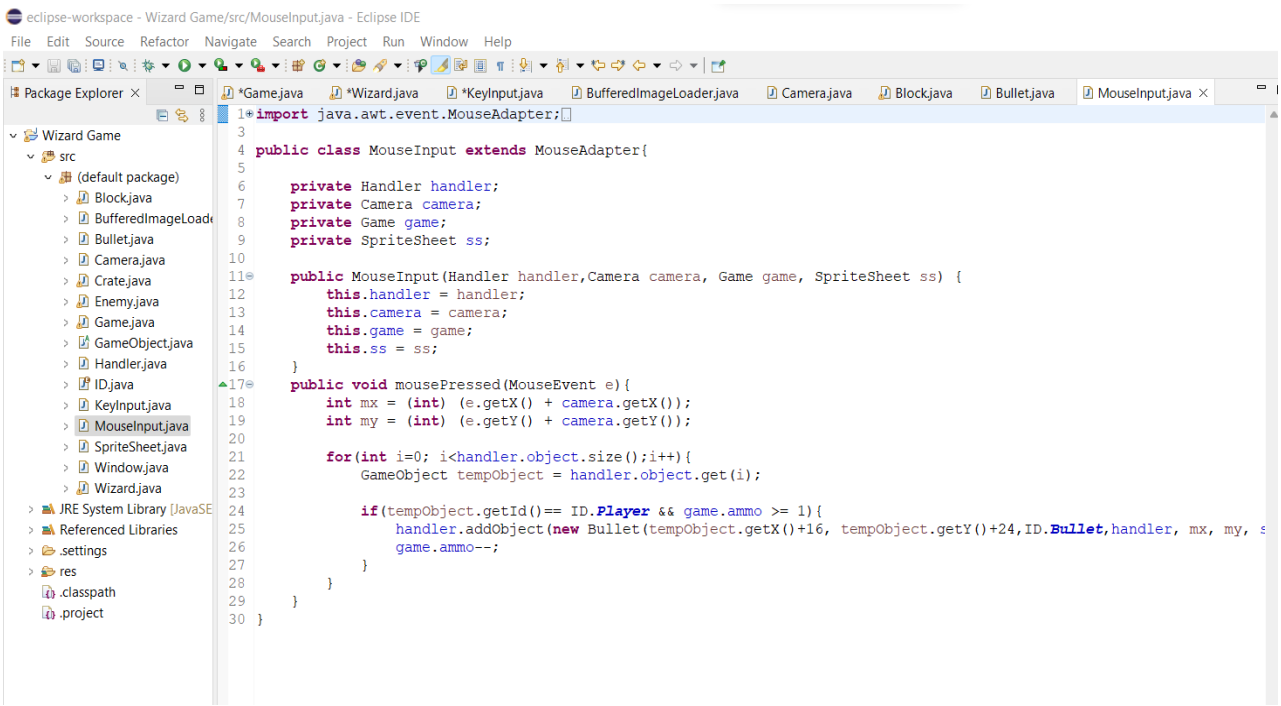
11. Bullet Class (To shoot the enemies):

In this what we're going to do is create a shooting system for our game so basically, we can run around our world here and we can use the mouse anywhere we'd like to click point and shoot a bullet.



12. MouseInput Class (For the control to shoot):

So, just like your key input we need a mouse input and we need to extend our mouse adapter, ctrl shift o to import, it so it's not the wrong all right and in our mouth adapter we want to save it.



13. Enemy Class (To create multiple enemies and their functions):

Just like Wizard class here we are going to implements all the functions designs and features of our enemies.

eclipse-workspace - Wizard Game/src/Enemy.java - Eclipse IDE

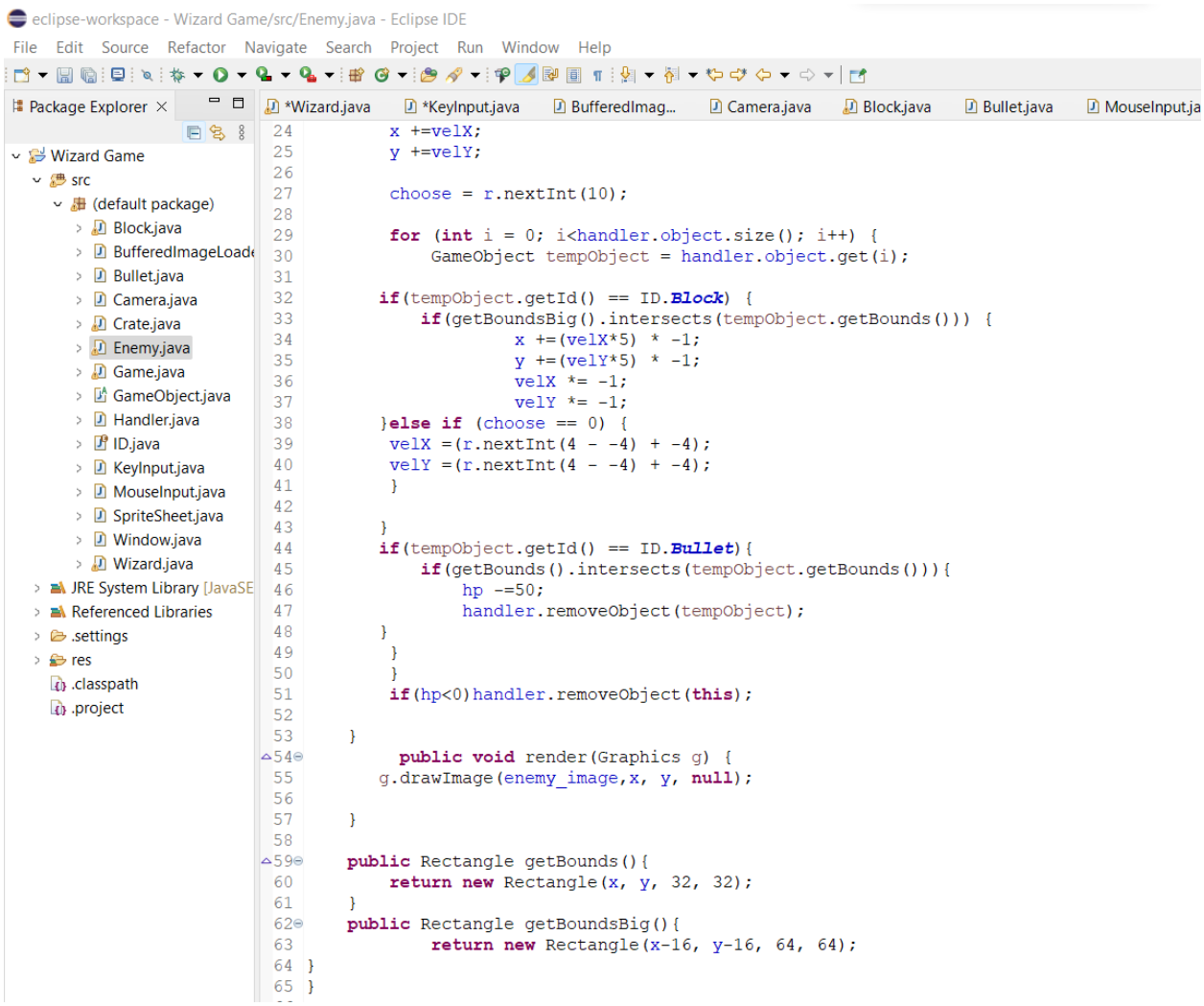
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Wizard Game
  - src
    - (default package)
      - Block.java
      - BufferedImageLoad
      - Bullet.java
      - Camera.java
      - Crate.java
      - Enemy.java
      - Game.java
      - GameObject.java
      - Handler.java
      - ID.java
      - KeyInput.java
      - MouseInput.java
      - SpriteSheet.java
      - Window.java
      - Wizard.java
  - JRE System Library [JavaSE
  - Referenced Libraries
  - .settings
  - res
  - .classpath
  - .project

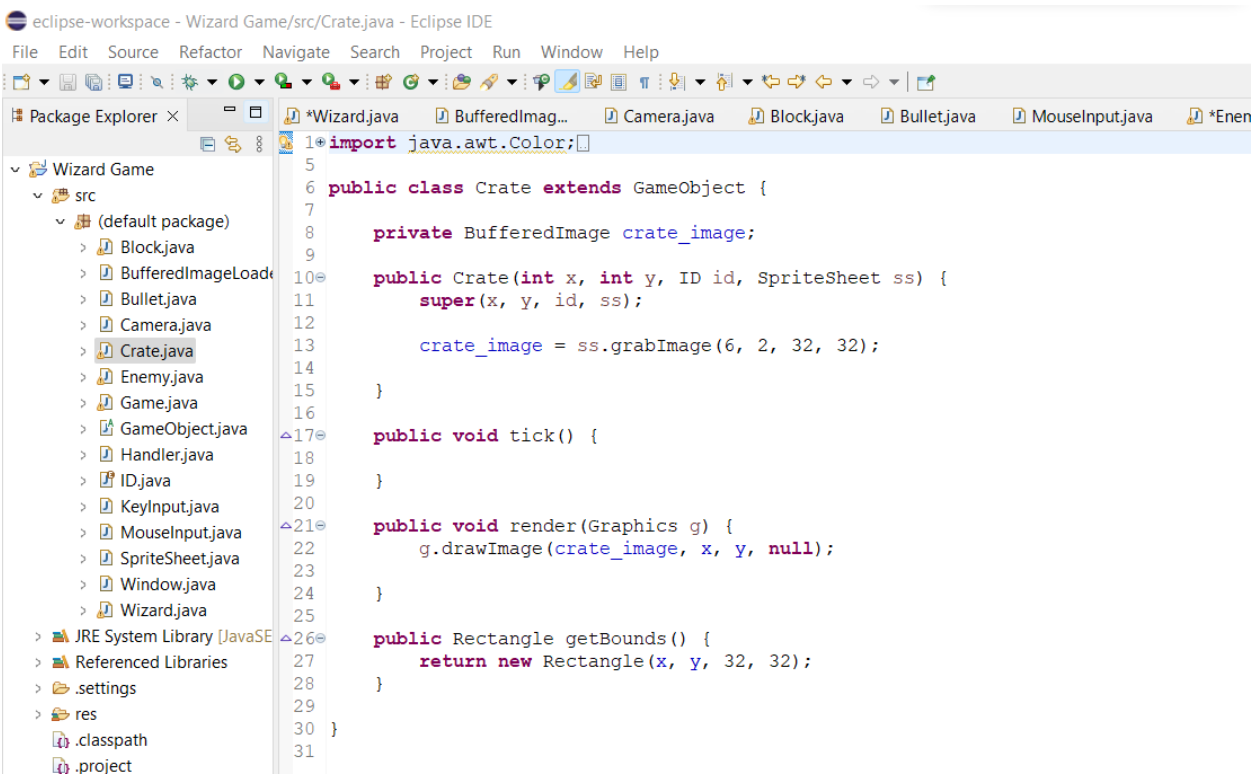
*Wizard.java | *KeyInput.java | BufferedImag... | Camera.java | Block.java | Bullet.java | MouseInput.java

```
1  import java.awt.Color;
7
8  public class Enemy extends GameObject{
9
10     private Handler handler;
11     Random r = new Random(); // for the movements controls of the enemies
12     int choose = 0;
13     int hp = 100; //health points
14
15     private BufferedImage enemy_image;
16
17     public Enemy(int x,int y,ID id,Handler handler, SpriteSheet ss){
18         super(x,y,id, ss);
19         this.handler=handler;
20
21         enemy_image = ss.grabImage(4, 1, 32, 32);
22     }
23     public void tick(){
24         x +=velX;
25         y +=velY;
26
27         choose = r.nextInt(10);
28
29         for (int i = 0; i<handler.object.size(); i++) {
30             GameObject tempObject = handler.object.get(i);
31
32        if(tempObject.getId() == ID.Block) {
33            if(getBoundsBig().intersects(tempObject.getBounds())) {
34                     x +=(velX*5) * -1;
35                     y +=(velY*5) * -1;
36                     velX *= -1;
37                     velY *= -1;
38        }else if (choose == 0) {
39         velX =(r.nextInt(4 - -4) + -4);
40         velY =(r.nextInt(4 - -4) + -4);
41         }
42
43        }
44        if(tempObject.getId() == ID.Bullet){
45            if(getBounds().intersects(tempObject.getBounds())){
46                hp -=50;
47                handler.removeObject(tempObject);
48        }
```

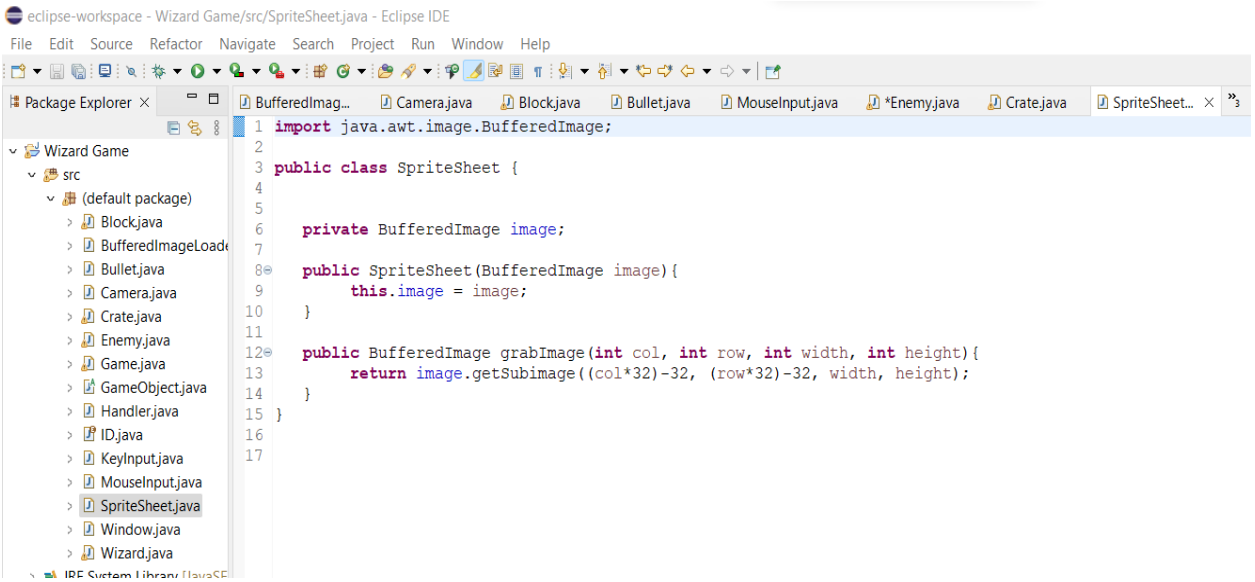


14. Crate (for the ammo refill):

Going to add design (box) to the map and implement that, so if the player ran out of ammo it can collect and refill.



15. SpriteSheet Class (Design of overall game):

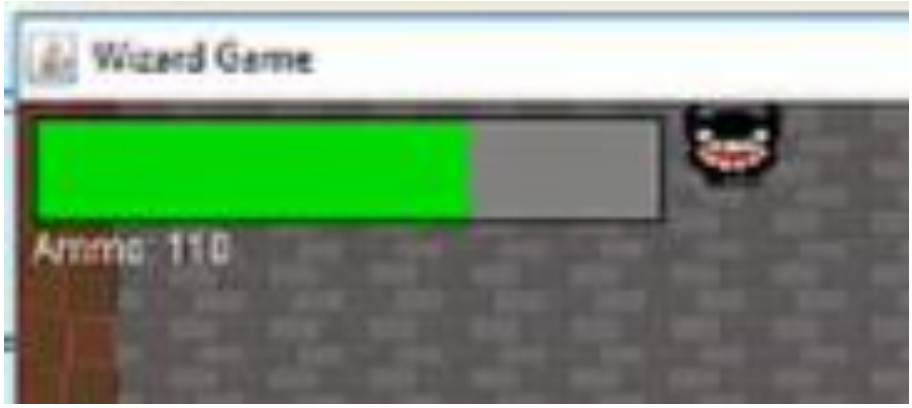


OUTPUT:

- **Raw Output:**

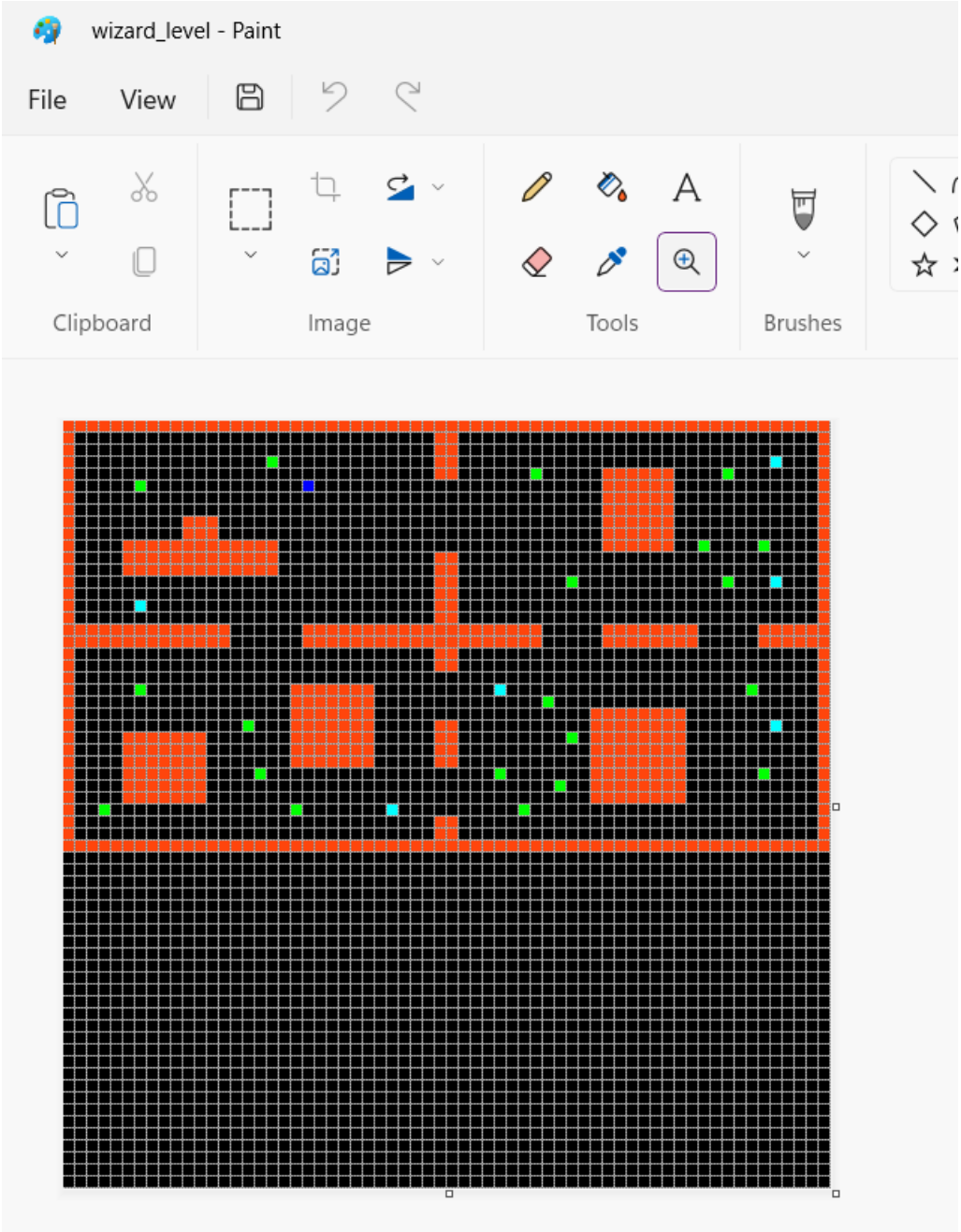
- **Overall:**

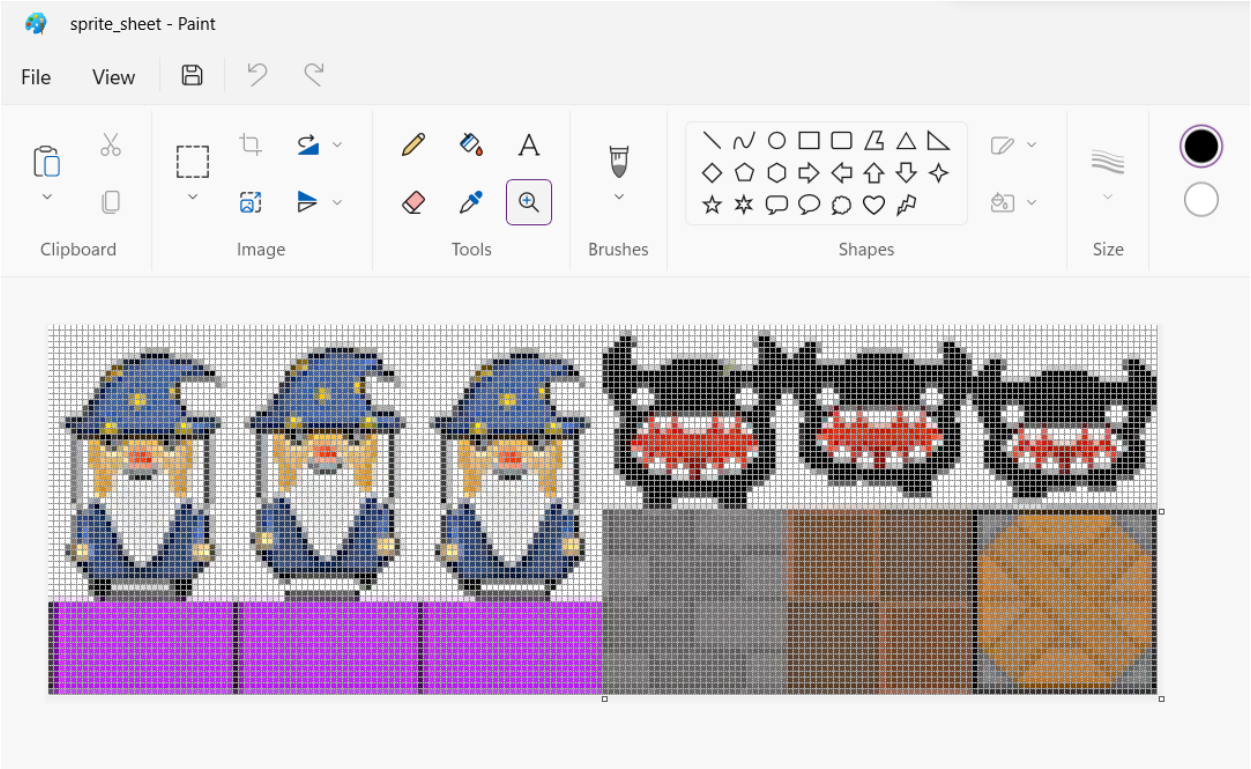
- **Health & Ammo:**



- **Shooting Part:**

DESIGN:


wizard_level - Paint
File
View
Clipboard
Image
Tools
Brushes



SCOPE:

This report outlines the said project's details, which have great potential and benefits in the long term as it can be played by any age group people also it was a great & new experience for us as well. Although shooting games are violent, they can be very efficient in teaching hand-eye coordination, motor, and spatial skills.

CONCLUSION:

Thinking of the game as a part of a bigger educational process is really in the core mind-set that this project wants to promote. Games can do many things very well, but they certainly cannot do everything at once. Especially not without solid supporting structures around them. Throughout the project and the case studies we built this was true.

CORRECTIONS THAT WERE ASKED TO DO IN AFTER VIVA:

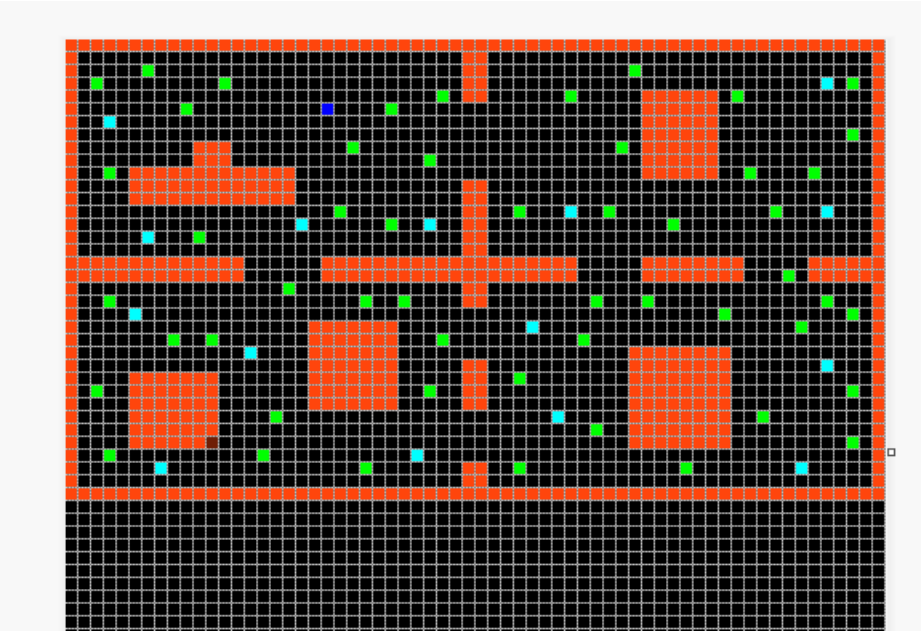
- **Game exit after player health has been over:**

```
        }
        }

      //intersect library is introduced which determines the intersection between player and crate
        if(tempObject.getId() == ID.Crate) {

            if(getBounds().intersects(tempObject.getBounds())) {
              game.ammo += 20;
              handler.removeObject(tempObject);
  }
        }

      //intersect library is introduced which determines the intersection between player and enemy
            if(tempObject.getId() == ID.Enemy) {
            if(getBounds().intersects(tempObject.getBounds())) {
              game.hp--;
            }
            else if(game.hp==0) { //condition that when health is empty the game will exit
               System.exit(0);
            }
            }
  }
}

//render method is created which converts the image into 2D and 3D
public void render(Graphics g){
    g.drawImage(wizard_image, x, y, null);
}


//this constructor helps create a rectangle
public Rectangle getBounds(){
    return new Rectangle(x, y, 32, 48);
}
```

Writable | Smart Insert | 76 : 16 : 232

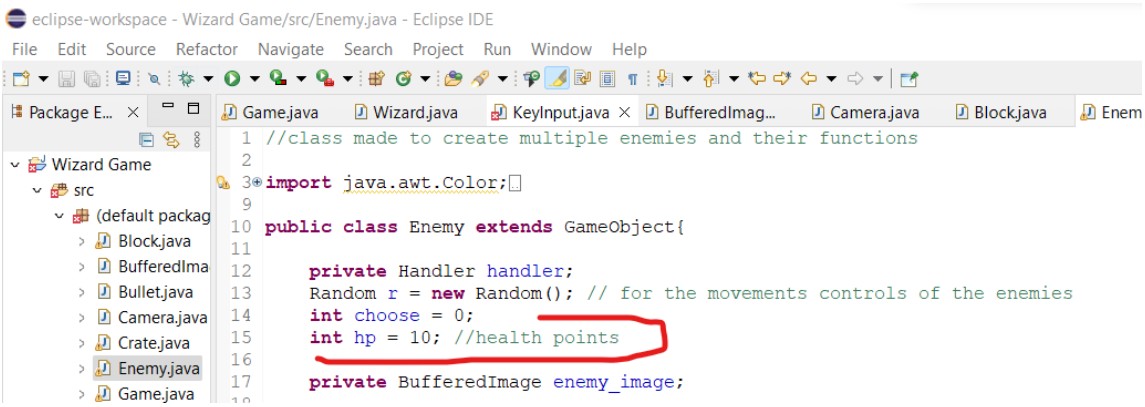
In wizard class we have implemented the condition that the game will stop as soon as the health point (hp) get to 0.

- **Added more enemies as per directions:**



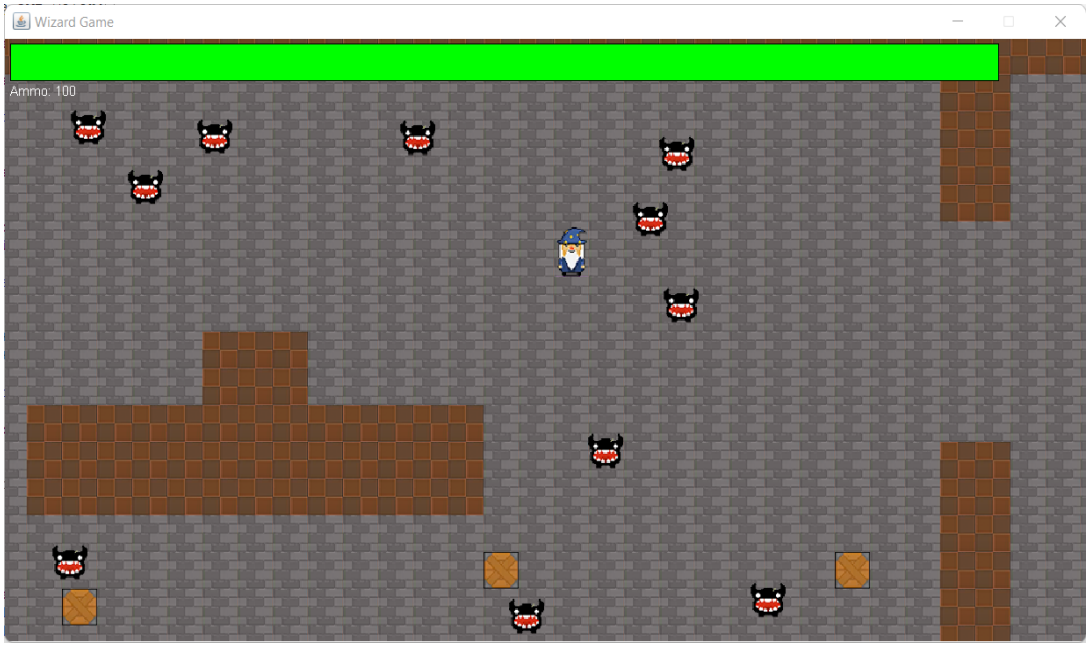
The green dots represent enemies.

- **A single shoot will kill the enemy:**

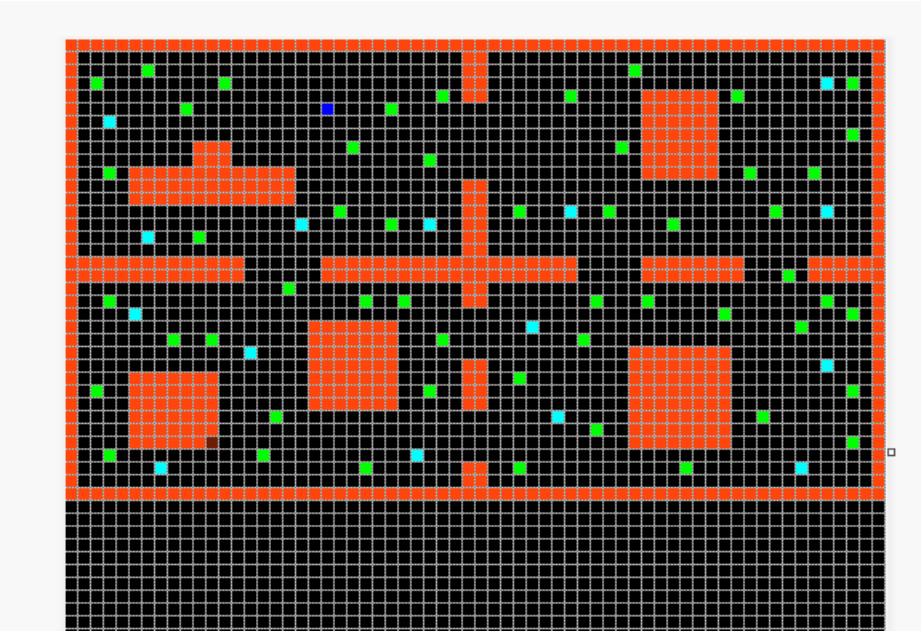


Decreased the enemy health so it would be killed by 1 bullet.

- **Increase the health level bar:**



- **Increase the ammo crate:**



Cyan is the ammo crate.

- **More detailed comments added to code.**