

MEMBERS



วรดร น้อยนงเยาว์



วัชรวิร์ ศรีพุกษ์



จิรวัฒน์ นิมิตรวงศ์



นนทพัทร์ จิวน้อต



เบญจลูกา สุบินดี

ROLES

ช่วยพัฒนาระบบและ
แก้ไขข้อผิดพลาดต่าง ๆ

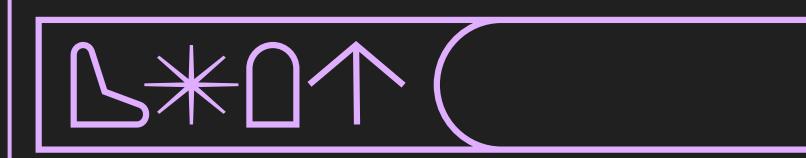
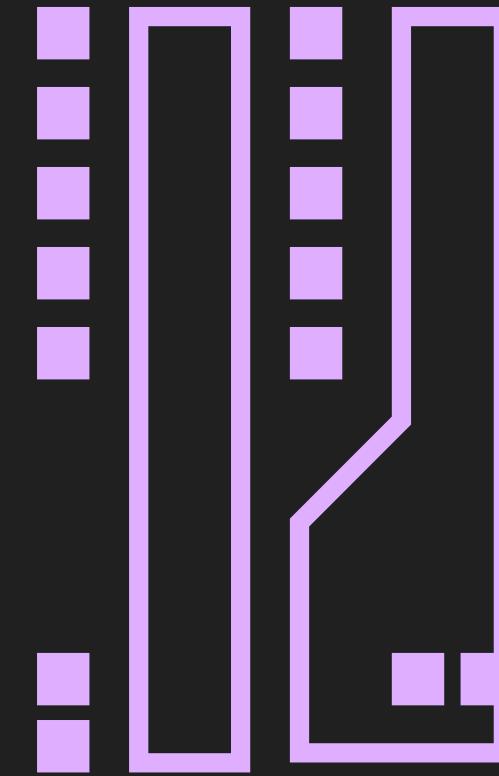
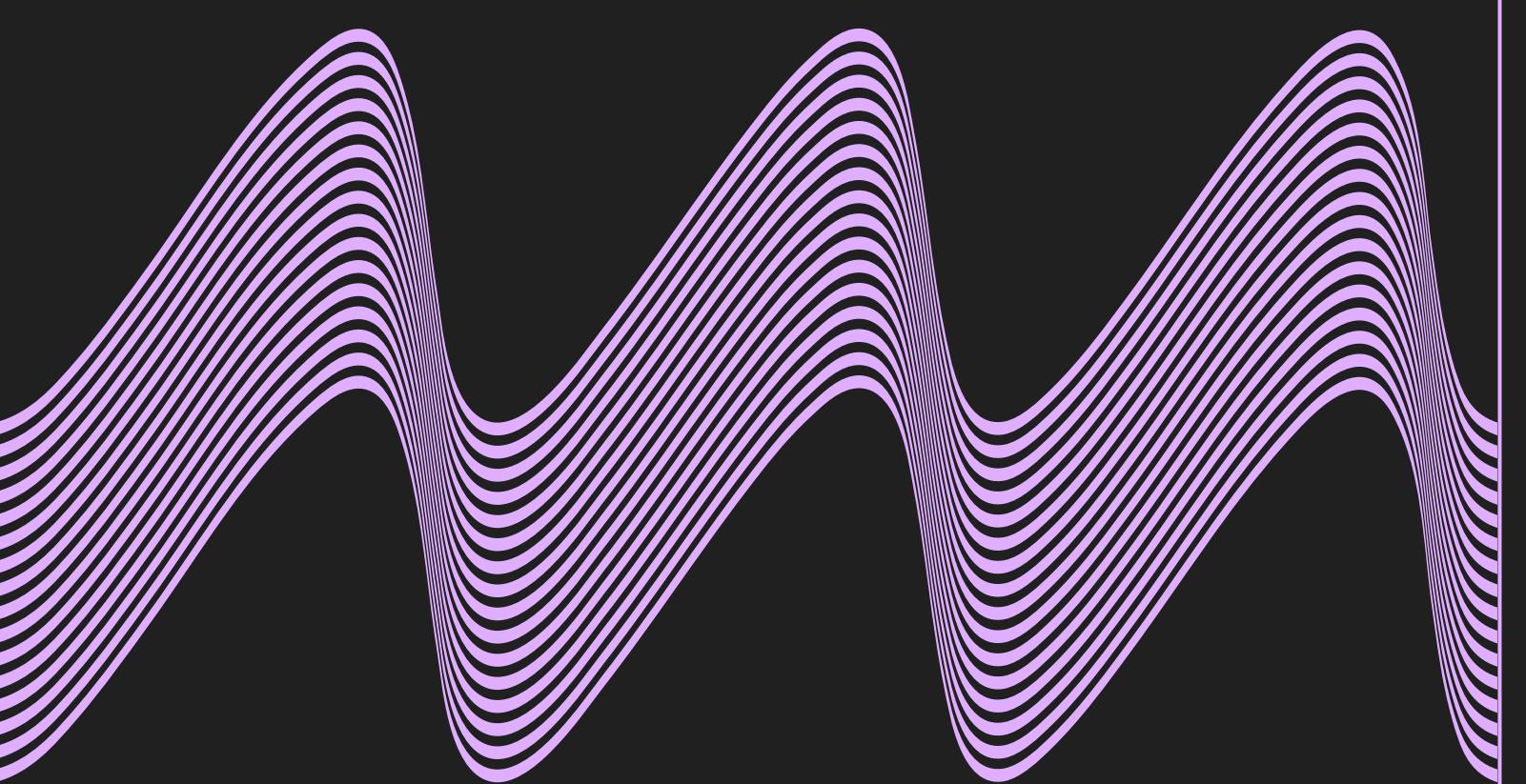
ออกแบบระบบ ดูภาพรวม
พัฒนาระบบเบื้องต้น

พัฒนา Graphic

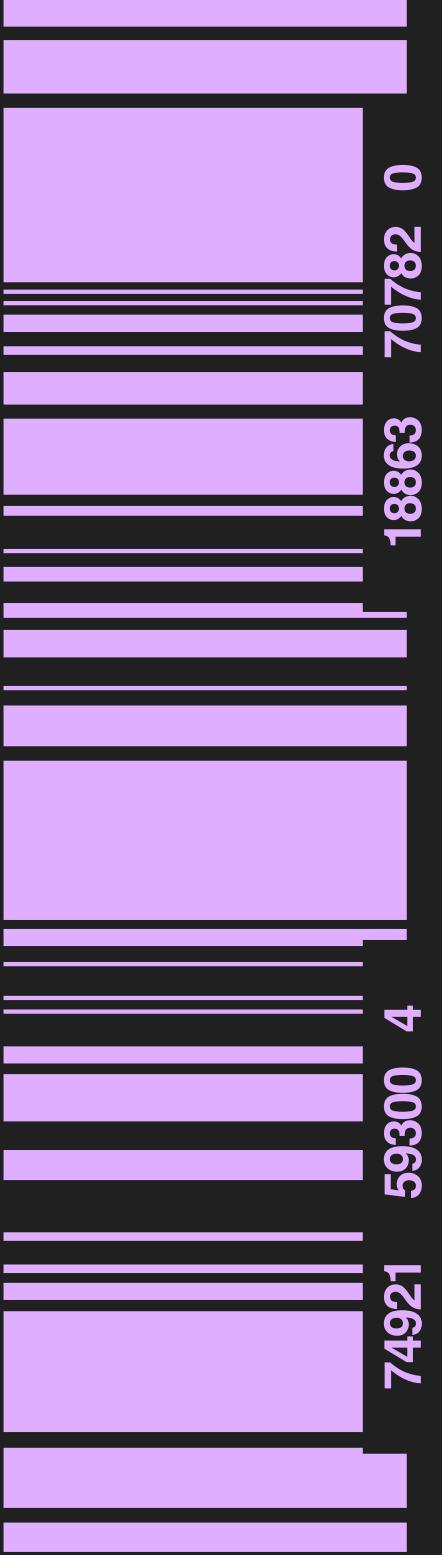
พัฒนา Graphic และเสียง

ออกแบบ Graphic และจัดวางภาพ

PLANET RUN PRESENTATION



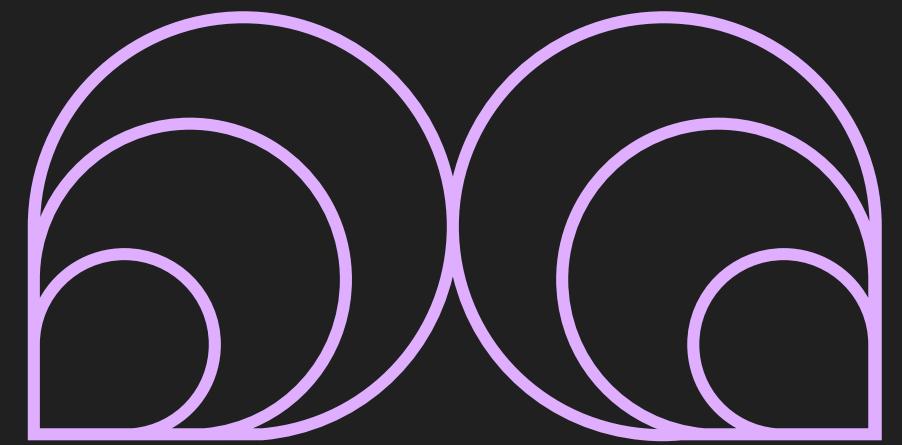
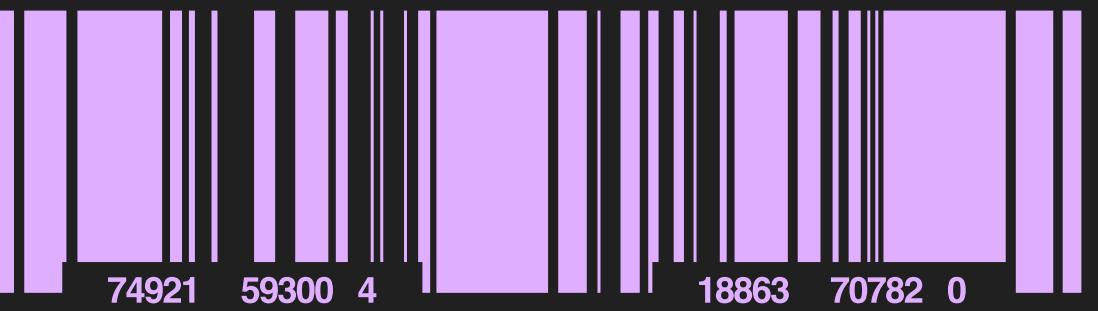
OBJECT -
ORIENTED
PROGRAMMING
(OOP)





บทคัดย่อ

โปรเจคนี้ถูกจัดทำขึ้นเพื่อนำการ
เขียนโปรแกรม JAVA เชิงวัตถุของ
วิชา OBJECT-ORIENTED
PROGRAMMING มาประยุกต์ใช้ใน
รูปแบบของวีดีโองame

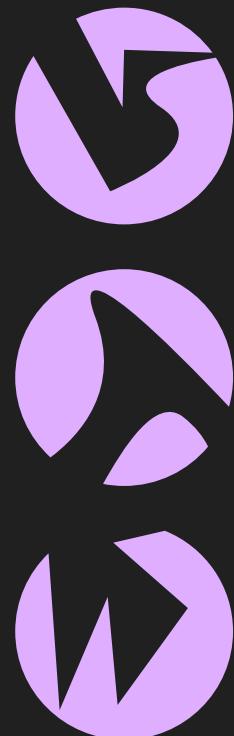


โดยเกม PLANET RUN นั้นดัดแปลง
มาจากการบอร์ดเกมที่มีอยู่แล้ว เป็นเกม
แนววางแผนเอาชีวิตรอดในอวกาศ
ผู้เล่นจะต้องใช้เวลาแต่ละวันในการหา
อาหาร เก็บทรัพยากร เพื่อนำไปซ้อม
ยานกลับดาวโลกภายในเวลา 15 วัน
เมื่อครบ 15 วันก็จะมีการคิดคะแนนว่า
ผู้เล่นเล่นได้ดีเท่าไร

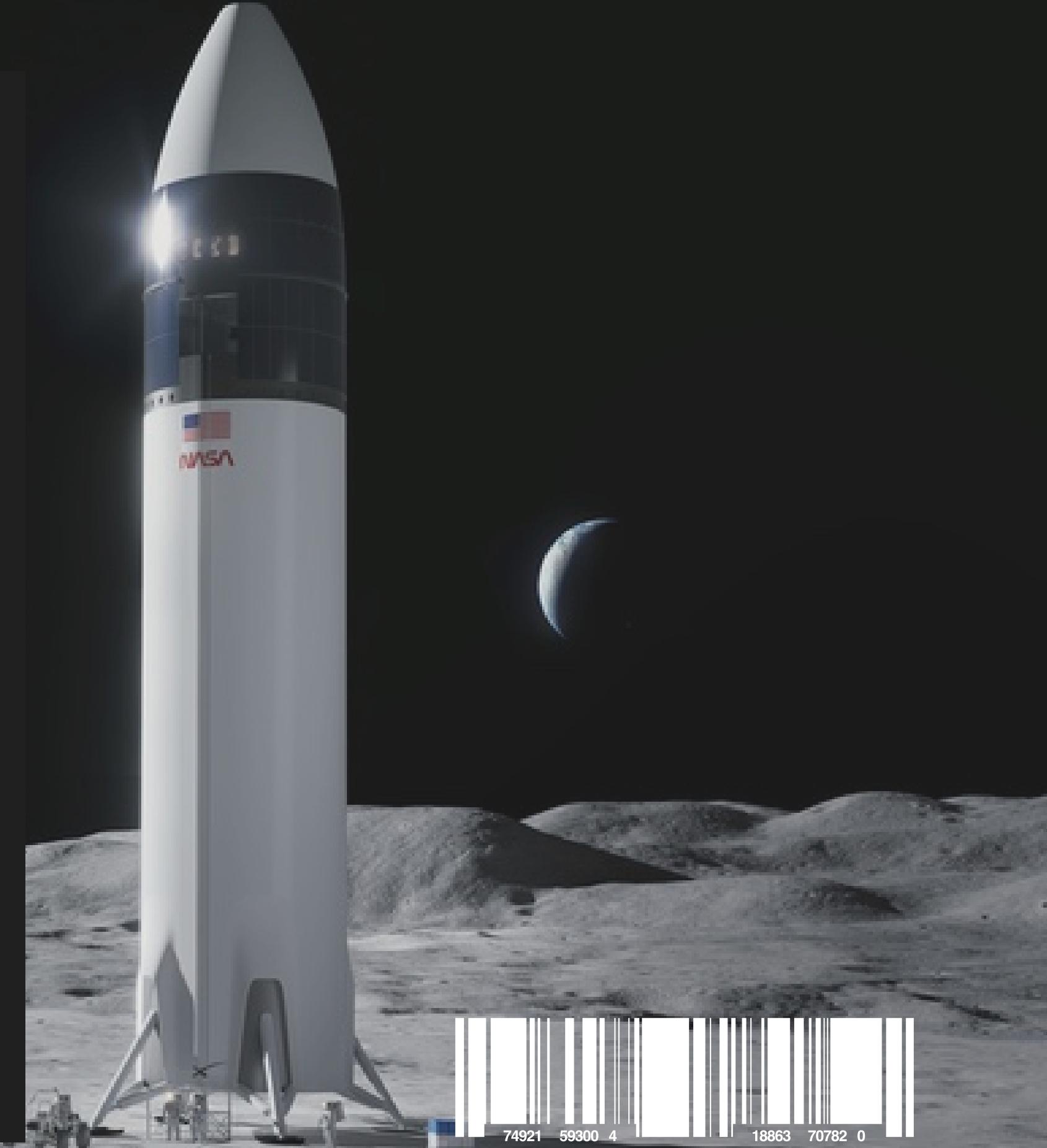


PURPOSE

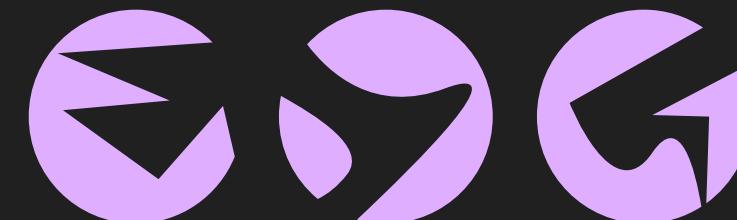
วัตถุประสงค์



- เพื่อศึกษาและเข้าใจในการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming)
- เพื่อศึกษาการนำภาษา Java ไปใช้ในการพัฒนาวิดีโอเกม



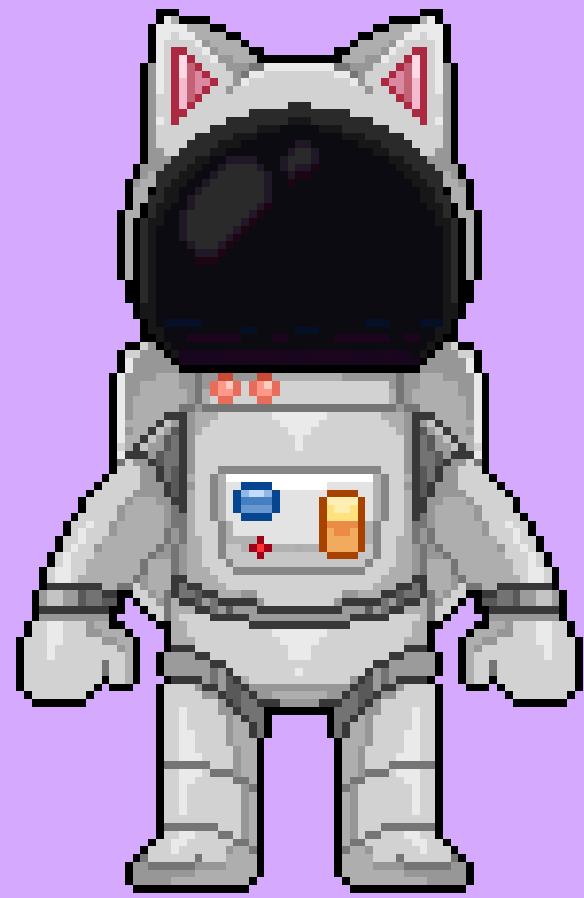
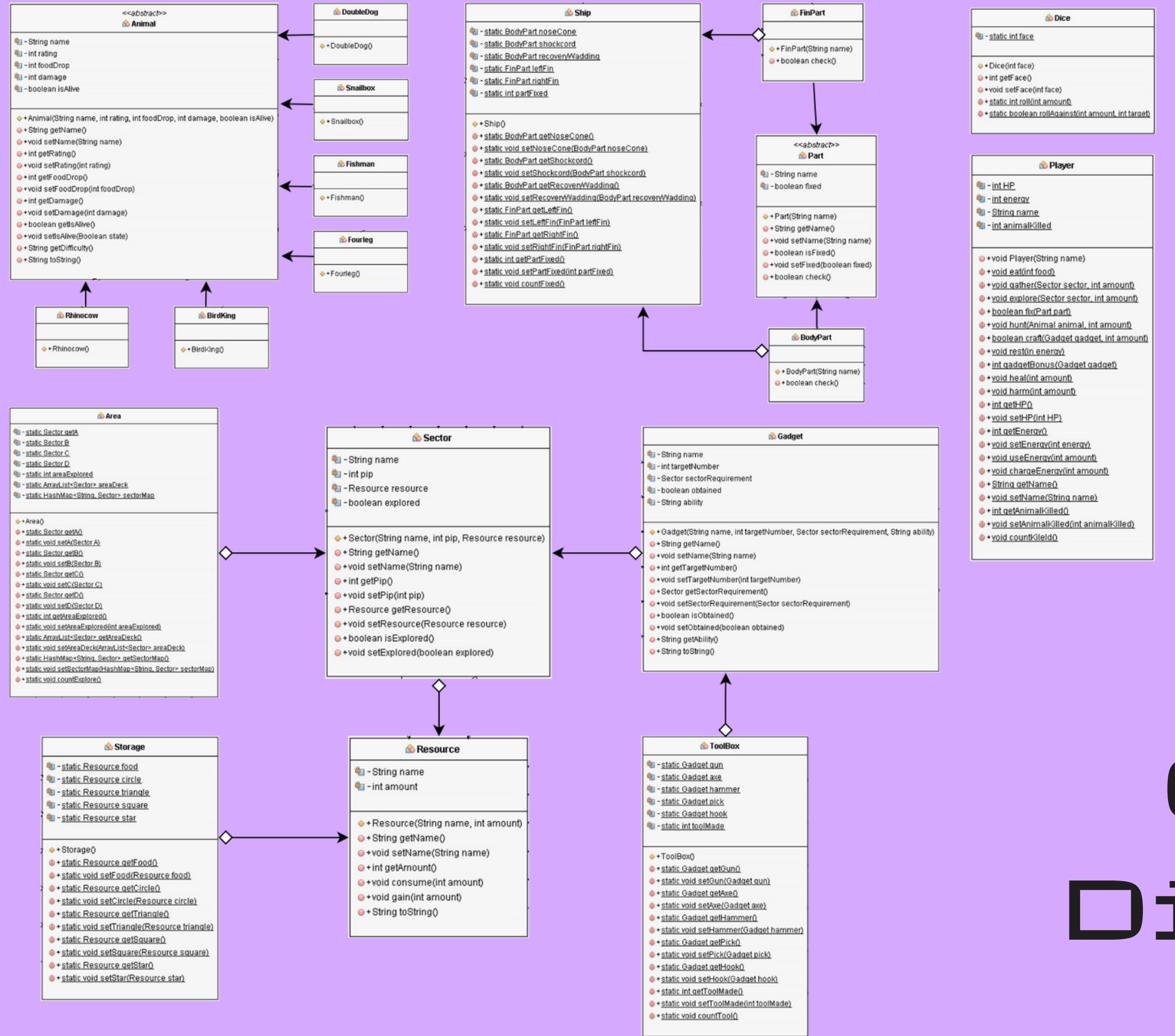
ประโยชน์ ที่ได้รับ



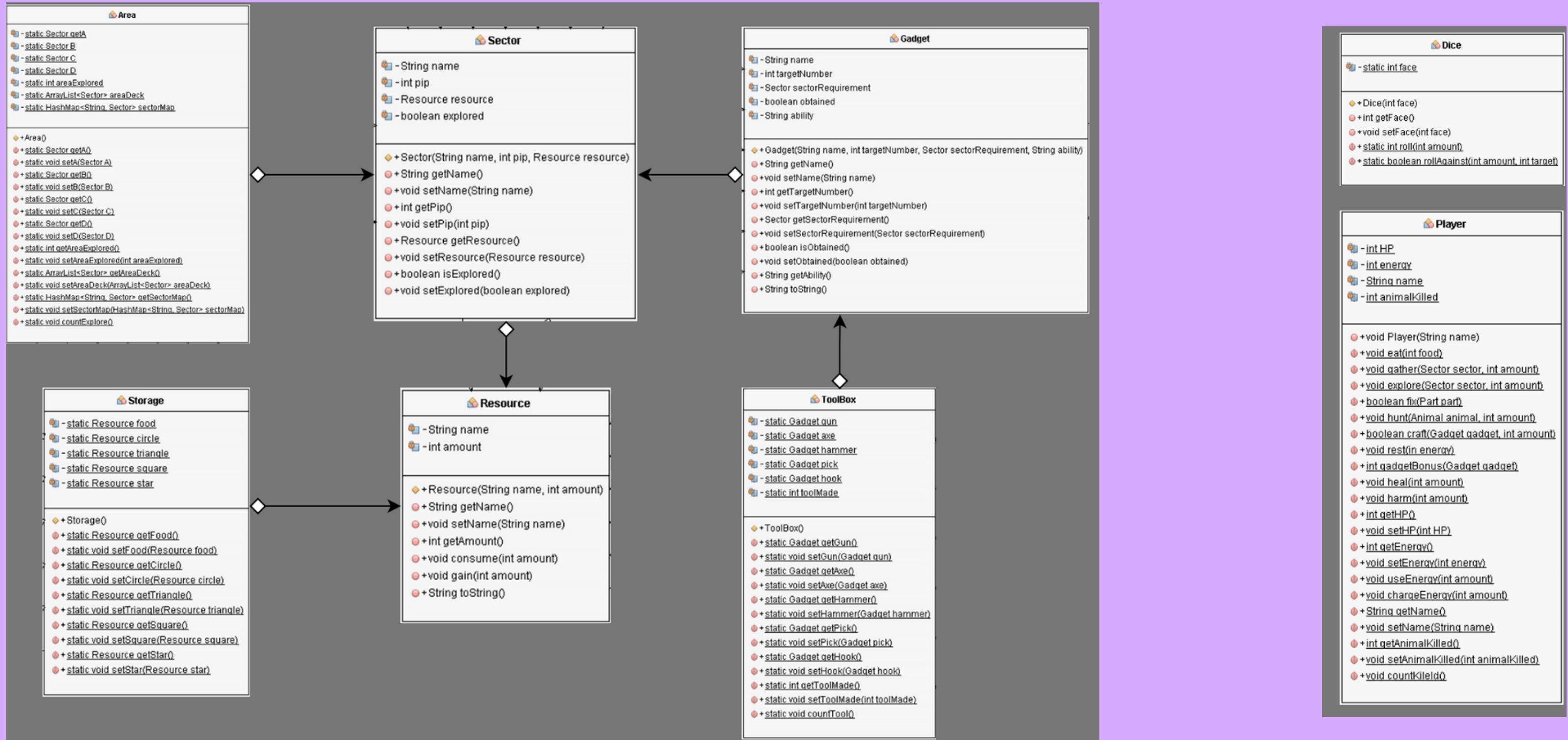
ได้นำการเขียน Java ที่ได้เรียนรู้มา
ประยุกต์ใช้จริงในการทำ Project

ได้รับความรู้เพิ่มเติมจากการค้นคว้าหา
ข้อมูลเพื่อนำมาใช้ในงาน Project

ได้พัฒนาการทำงานเป็นทีมร่วมกับผู้อื่น

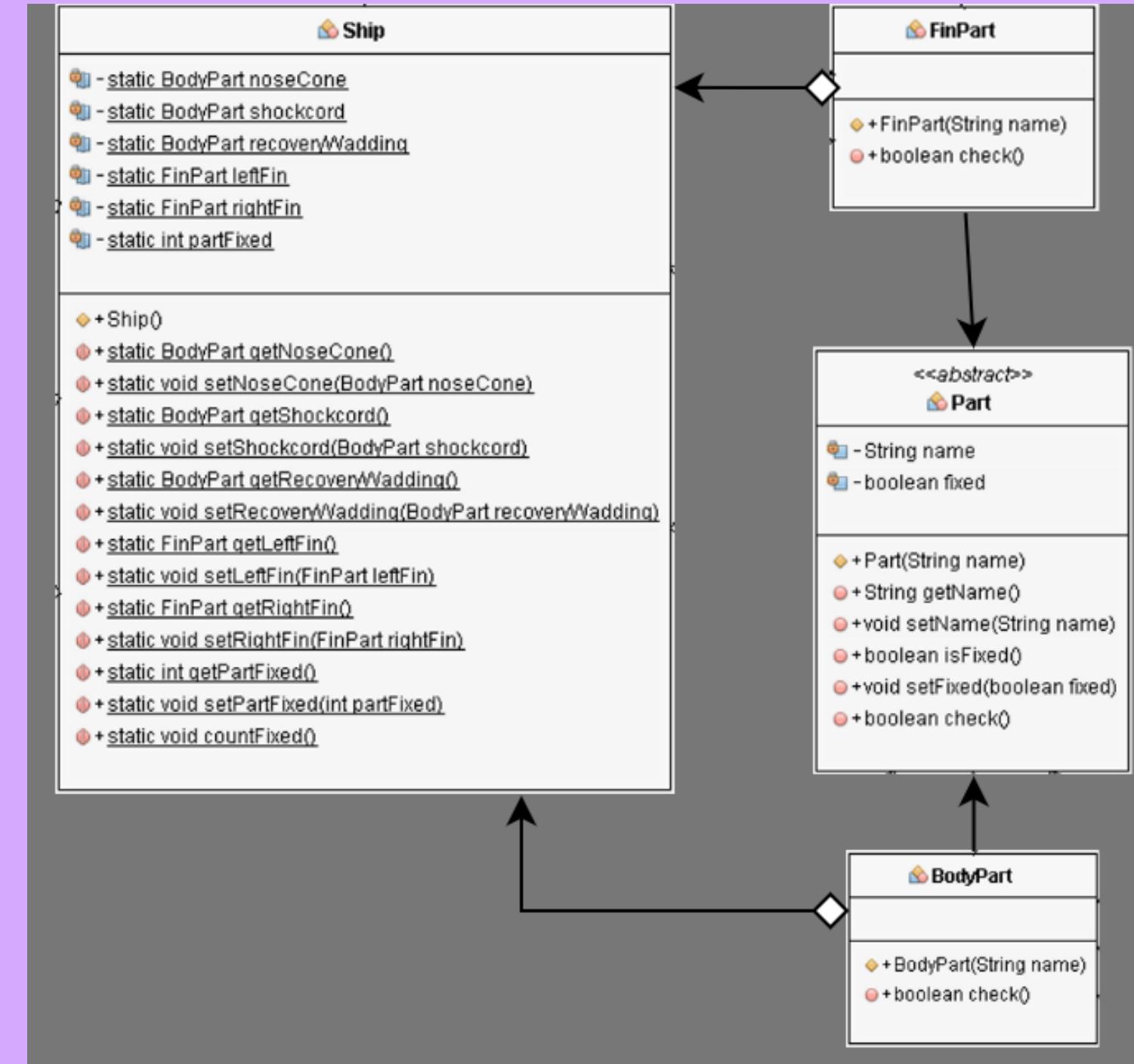
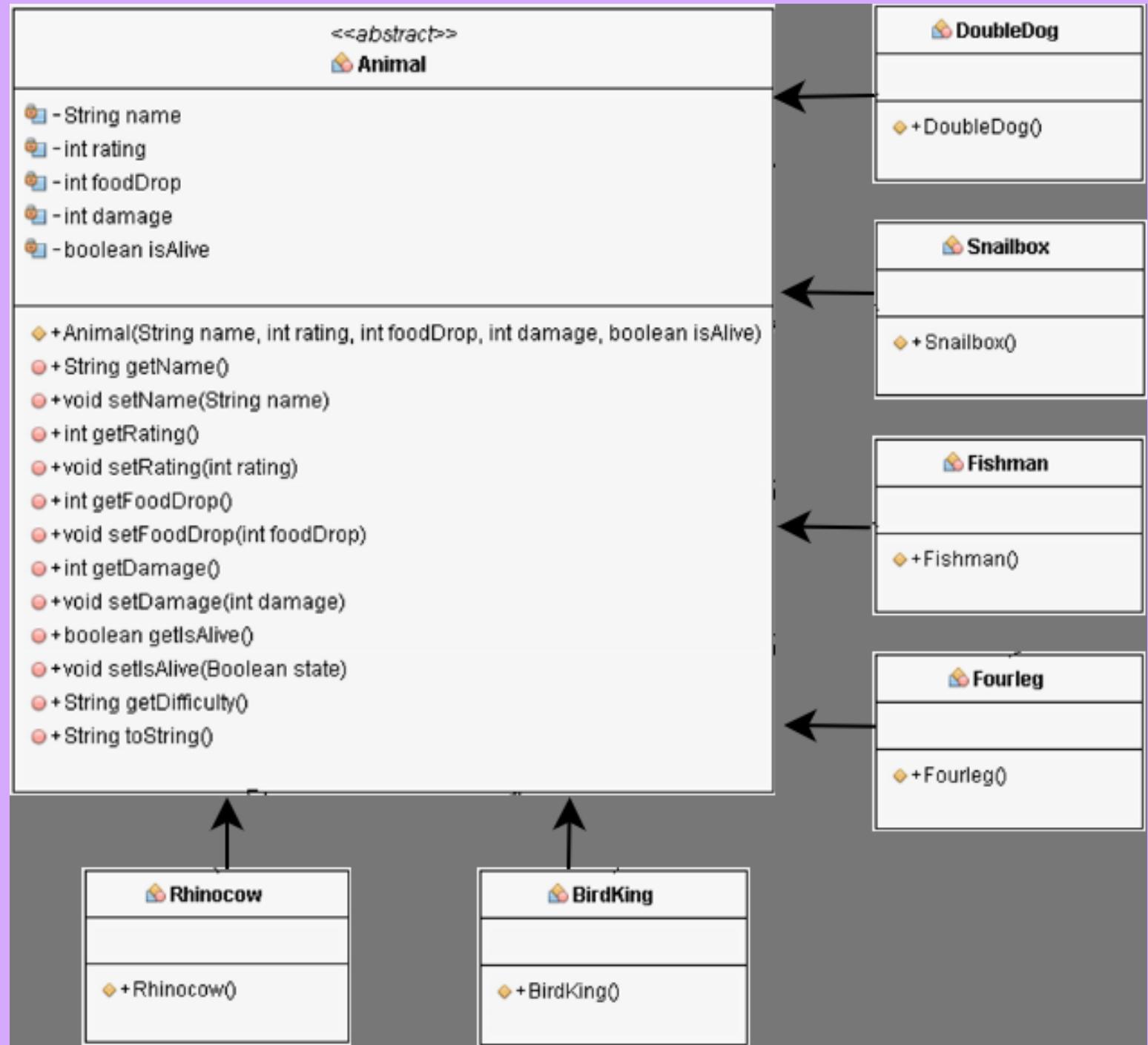


Class Diagram



Sectors, Gadgets & Resources

Player & Dice



Animals

Ship Parts

Player

```
import javax.swing.*;  
  
public class Player {  
    private static int HP;  
    private static int energy;  
    private static String name;  
    private static int animalKilled;  
  
    public Player(String name) {  
        Player.name = name;  
        HP = 4;  
        energy = 3;  
        animalKilled = 0;  
    }  
  
    public static void eat(int food){  
        switch (food) {  
            case 0 -> {  
                setEnergy(2);  
                harm(1);  
            }  
            case 1 -> setEnergy(2);  
            case 2 -> setEnergy(3);  
            case 3 -> setEnergy(4);  
            case 4 -> setEnergy(5);  
            case 5 -> setEnergy(6);  
            default -> {}  
        }  
    }  
}
```

```
public static void gather(Sector sector, int amount){  
    Gadget gadget = null;  
    switch (sector.getName()) {  
        case "Forest":  
            gadget = ToolBox.getAxe();  
            break;  
        case "Mountain":  
            gadget = ToolBox.getPick();  
            break;  
        case "Desert":  
            gadget = ToolBox.getHammer();  
            break;  
        default:  
            gadget = null;  
            break;  
    }  
    if (Dice.rollAgainst(amount+gadgetBonus(gadget), sector.getPip())){  
        sector.getResource().gain(1);  
        JOptionPane.showMessageDialog(null, "Success to gathering. Gain 1 "+sector.getResource().getName()+".", "", JOptionPane.PLAIN_MESSAGE);  
    }  
    else{  
        JOptionPane.showMessageDialog(null, "Fail to gathering resource from "+sector.getName()+".", "", JOptionPane.ERROR_MESSAGE);  
    }  
    useEnergy(amount);  
}
```

กำหนดคุณลักษณะและสิ่งที่ Player ทำได้

Gadget

กำหนดคุณลักษณะของ gadget
ซึ่งความสามารถ
พื้นที่ที่ต้องสำรวจก่อน

```
public class Gadget {  
    private String name;  
    private int targetNumber;  
    private Sector sectorRequirement;  
    private boolean obtained;  
    private final String ability;  
  
    public Gadget(String name, int targetNumber, Sector sectorRequirement, String ability) {  
        this.name = name;  
        this.targetNumber = targetNumber;  
        this.sectorRequirement = sectorRequirement;  
        this.obtained = false;  
        this.ability = ability;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getTargetNumber() {  
        return targetNumber;  
    }  
  
    public void setTargetNumber(int targetNumber) {  
        this.targetNumber = targetNumber;  
    }  
  
    public Sector getSectorRequirement() {  
        return sectorRequirement;  
    }  
  
    public void setSectorRequirement(Sector sectorRequirement) {  
        this.sectorRequirement = sectorRequirement;  
    }  
  
    public boolean isObtained() {  
        return obtained;  
    }  
  
    public void setObtained(boolean obtained) {  
        this.obtained = obtained;  
    }  
  
    public String getAbility() {  
        return ability;  
    }  
  
    @Override  
    public String toString() {  
        return name + ", Target Number: " + targetNumber + ", Sector: " + sectorRequirement + ", Obtained: " + obtained + ", Ability: " + ability;  
    }  
}
```

Part

กำหนดคุณลักษณะของ Part
แยกส่วนเป็น FinPart และ BodyPart

```
/*
public class FinPart extends Part{

    public FinPart(String name) {
        super(name);
    }

    @Override
    public boolean check() {
        if (Storage.getStar().getAmount() >= 4) {
            Storage.getStar().consume(4);
            return true;
        }
        else{
            return false;
        }
    }
}

/*
public class BodyPart extends Part{

    public BodyPart(String name) {
        super(name);
    }

    @Override
    public boolean check() {
        if (Storage.getTriangle().getAmount() >= 1 && Storage.getCircle().getAmount() >= 1 && Storage.getSquare().getAmount()
            Storage.getTriangle().consume(1);
            Storage.getCircle().consume(1);
            Storage.getSquare().consume(1);
            return true;
        }
        else{
            return false;
        }
    }
}
```



```
public abstract class Part {
    private String name;
    private boolean fixed;

    public Part(String name) {
        this.name = name;
        fixed = false;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isFixed() {
        return fixed;
    }

    public void setFixed(boolean fixed) {
        this.fixed = fixed;
    }

    // private Resource requirement;
    public abstract boolean check();
}
```

Sector

กำหนดคุณลักษณะของ Sector
ชื่อ ความยากในการสำรวจ
ทรัพยากรที่มี และสถานะการสำรวจ

```
public class Sector {  
    private String name;  
    private int pip;  
    private Resource resource;  
    private boolean explored;  
  
    public Sector(String name, int pip, Resource resource) {  
        this.name = name;  
        this.pip = pip;  
        this.resource = resource;  
        this.explored = false;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getPip() {  
        return pip;  
    }  
  
    public void setPip(int pip) {  
        this.pip = pip;  
    }  
  
    public Resource getResource() {  
        return resource;  
    }  
  
    public void setResource(Resource resource) {  
        this.resource = resource;  
    }  
  
    public boolean isExplored() {  
        return explored;  
    }  
  
    public void setExplored(boolean explored) {  
        this.explored = explored;  
    }  
}
```

```

public abstract class Animal {
    private String name;
    private int rating;
    private int foodDrop;
    private int damage;
    private boolean isAlive;

    public Animal(String name, int rating, int foodDrop, int damage , boolean isAlive) {
        this.name = name;
        this.rating = rating;
        this.foodDrop = foodDrop;
        this.damage = damage;
        this.isAlive = true;
    }

    // getter setter no one cares
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getRating() {
        return rating;
    }
    public void setRating(int rating) {
        this.rating = rating;
    }
    public int getFoodDrop() {
        return foodDrop;
    }
    public void setFoodDrop(int foodDrop) {
        this.foodDrop = foodDrop;
    }
    public int getDamage() {
        return damage;
    }
    public void setDamage(int damage) {
        this.damage = damage;
    }

    public boolean getIsAlive() {
        return isAlive;
    }
    public void setIsAlive(Boolean state) {
        this.isAlive = state;
    }
}

```

Animal

คุณลักษณะของสัตว์ พลังโฉมตี
ความยากในการโฉมตี อาหารที่ได้รับ

```

public String getDifficulty(){
    switch (rating){
        case 4:
            return "(Easy+)";
        case 5:
            return "(Normal)";
        case 6:
            return "(Normal+)";
        case 7:
            return "(Hard)";
        case 8:
            return "(Hard+)";
        case 9:
            return "(Very Hard)";
        default:
            return "0";
    }
}

@Override
public String toString() {
    return "Animal{" + "name=" + name + ", rating=" + rating + ", foodDrop=" + foodDrop + ", damage=" + damage + '}';
}

```

```

public class MainUI extends JPanel implements ActionListener, Runnable {

    public static JFrame fr;
    private static JPanel pAll, pShip, pHuntingGround, pSectors, pGadget, pRest,
    pRocket, pHuntAll, pSec, pInmain, pInRocket, showclock;
    private static JButton noseCone, shockcord, recoveryWadding, leftFin, rightF
    private static JButton sector1, sector2, sector3, sector4, rest, maketool;
    public static JButton huntingGround0, huntingGround1;
    private static JButton exit, howToPlay;
    private static JTextField title1;
    public static ImageIcon rockcone, rockleft, rockright, rockbase, rockbody, s
    private static JLabel name, clockshow;

    private static JPanel pInput, pEnergy, pEat;
    private static ButtonGroup group;
    private static JRadioButton e1, e2, e3, e4, e5, e6;
    private static JLabel howMany;
    private static JSlider slider;

    public MainUI() {
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int screenDPI = Toolkit.getDefaultToolkit().getScreenResolution();
        double scaleTest = (double) 96 / screenDPI;
        day = 0;
        sec = 0;

        huntingGround = new Animal[2];
        animalDeck = new Animal[6];

        animalDeck[0] = new BirdKing();
        animalDeck[1] = new DoubleDog();
        animalDeck[2] = new Fourleg();
        animalDeck[3] = new Fishman();
        animalDeck[4] = new Snailbox();
        animalDeck[5] = new Rhinocow();

        ani1 = new ImageIcon("src/images/Ani1.png");
        ani2 = new ImageIcon("src/images/Ani2.png");
        ani3 = new ImageIcon("src/images/Ani3.png");
        ani4 = new ImageIcon("src/images/Ani4.png");
        ani5 = new ImageIcon("src/images/Ani5.png");
        ani6 = new ImageIcon("src/images/Ani6.png");
    }
}

```

MainUI

ควบคุมคุณลักษณะและการกระทำ
ของ GUI ภายในเกม
การจัดการเกี่ยวกับ Graphic
การตอบสนองต่อผู้ใช้

```

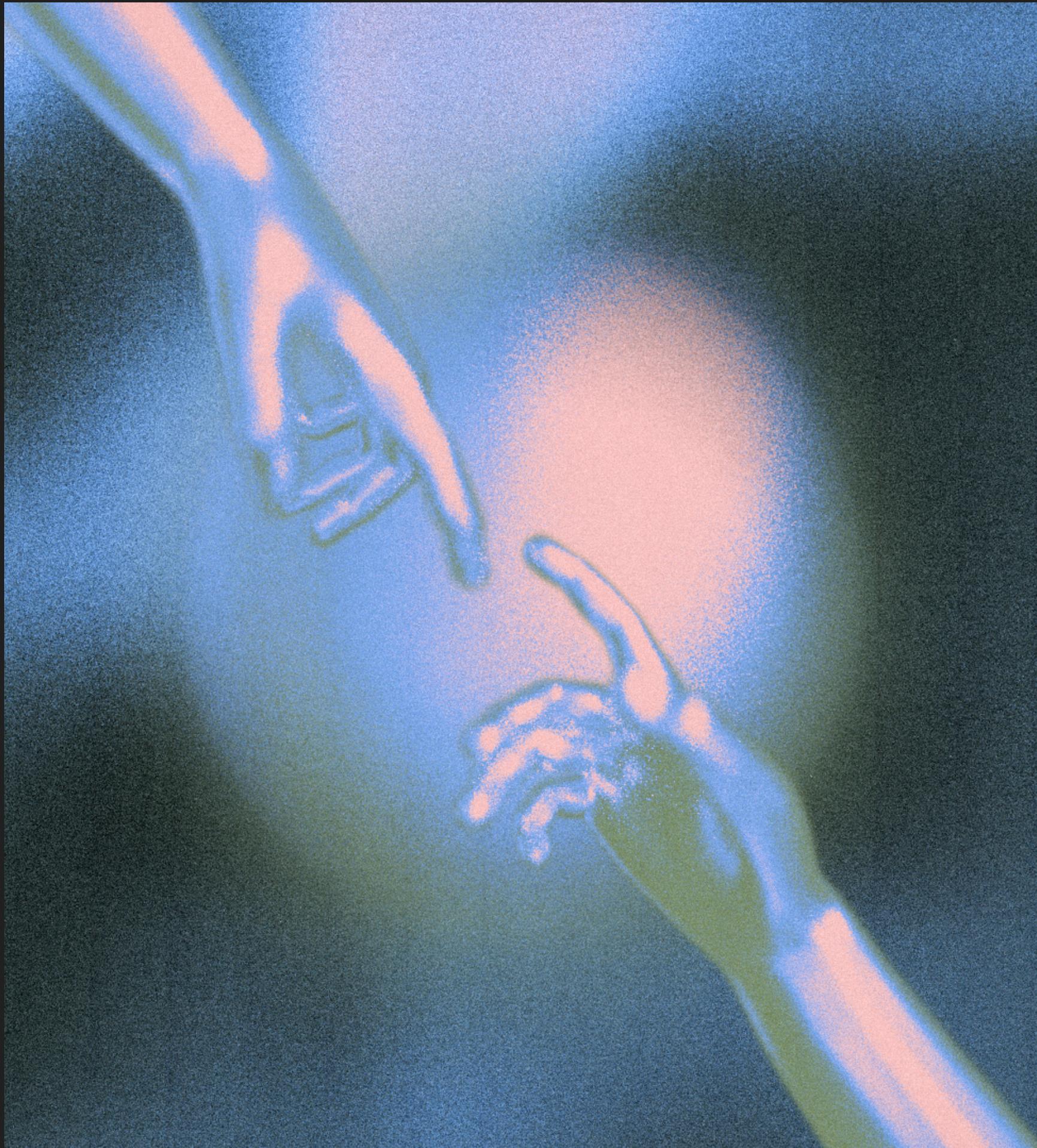
public static int screenScale(double pixel) {
    int screenDPI = Toolkit.getDefaultToolkit().getScreenResolution();
    double scaleTest = (double) 96 / screenDPI;
    return (int) (pixel * scaleTest);
}

```

MainHome

กำหนดคุณลักษณะ
และการกระทำของหน้าเริ่มเกม
แสดงหน้าตารางคะแนน
แสดงหน้าวิธีเล่น

```
public class MainHome extends JPanel implements ActionListener {  
  
    public static JFrame fr, fri;  
    private JPanel p1, p2, pName;  
    private JLabel titleL, nameLbl;  
    private JButton bStart, bHighscore, bHow, bExit;  
    private String name;  
    private JTextField nameField;  
    public static MainUI frame;  
  
    private static String str = "";  
    public static String titleMusic = "title.wav", playingMusic = "playing.wav";  
    private static MusicStuff musicplay = new MusicStuff();  
    private static HashMap<String, Integer> highScore;  
    private static LinkedHashMap<String, Integer> sortedScore;  
  
    Image backgroundimg;  
    Image ani1, ani2, ani3, ani4;  
  
    public int px = 0, py = 0;  
  
    public MainHome(JFrame fr) {  
        sortedScore = new LinkedHashMap<>();  
        highScore = new HashMap<String, Integer>();  
    }  
}
```



THANK
YOU!

