TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA ĐÀO TẠO QUỐC TẾ

----------o0o----------

Báo cáo bài tập lớn Java

ĐỀ TÀI : GAME DINOSAUR

Giảng viên hướng dẫn	**:**	Vũ Huấn	
Sinh viên thực hiện	**:**	Đỗ Thị Thu Hương	- 212630868
		Đỗ Minh Giang	- 212600745
Lớp	**:**	CNTT V-A1 - K62	

Hà Nội, tháng 04 năm 2023

Mục Lục

Danh mục hình ảnh

Lời mở đầu

Nhóm em xin gửi đến thầy Vũ Huấn bản báo cáo của bài tập lớn môn công nghệ Java của nhóm em. Chúng em cám ơn thầy đã nhiệt tình giúp đỡ nhóm em và cho chúng em nhiều kiến thức và tài liệu quý giá giúp chúng em hoàn thành đề tài một cách tốt nhất. Đây là thành quả của nhóm 15 chúng em. Trò chơi mà chúng em thiết kế là một trò đơn giản, được lấy ý tưởng từ trò game khủng long mất mạng. Đây là game đầu tiên của chúng em tạo ra, tuy đã được chúng em kiểm tra và chỉnh sửa hoàn thiện nhiều lần nhưng nó vẫn chưa thể hoàn thiện hết và sẽ còn nhiều thiếu xót. Chúng em mong nhận được nhiều lời góp ý đến từ thầy giúp chúng em hoàn thiện game. Nhóm em xin cảm ơn ạ!

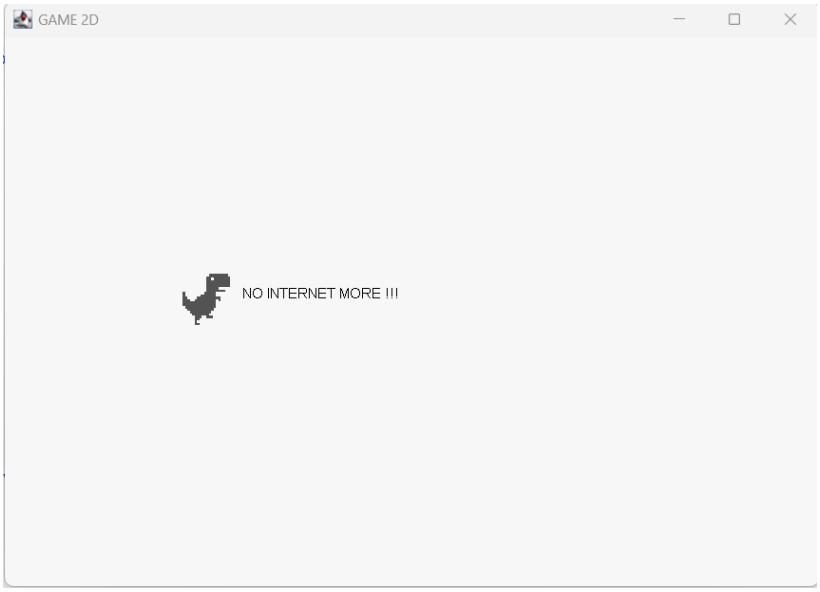
I. GIỚI THIỆU TỔNG QUAN

Game Dinosaur là một game đơn giản được thiết kế bằng ngôn ngữ java. Trong game người chơi sẽ điều khiển một chú khủng long. Người chơi chỉ cần vượt qua các chướng ngại vật như xương rồng, chim bằng cách nhảy lên sao cho không va chạm vào các chướng ngại vật và sẽ đạt được điểm sau mỗi lần vượt qua. Game sẽ kết thúc khi người chơi va chạm với bất kì chướng ngại vật nào của game.

II. THIẾT KẾ VÀ GIAO DIỆN

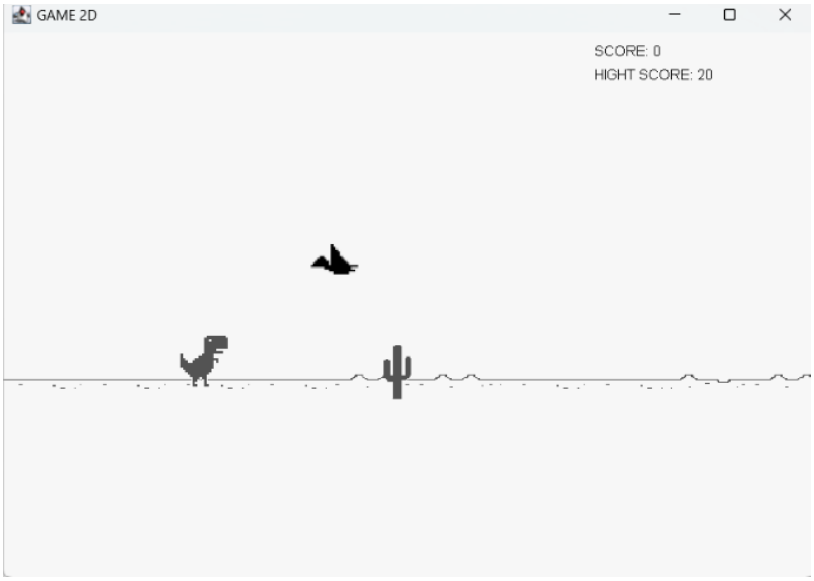
1.1 Start game

- Khi vào game mới, cửa sổ hiện ra với hình 1 chú khủng long và dòng chữ "No internet more !!!":



Hình 1: Start game

- Sau khi ấn nút Space game sẽ bắt đầu: các chướng ngại vật sẽ hiện ra, đồng thời score và high score sẽ hiện ra ở trên góc phải của màn hình.



Hình 2: Run game

1.2 Nội dung

- Game sử dụng nút Space để nhảy sao cho vượt qua các cây xương rồng và không chạm vào chú chim thì mỗi lần qua 1 chướng ngại vật sẽ được cộng 20 điểm. Điểm sẽ được hiển thị ở góc phải trên màn hình, sau mỗi game nếu bạn vượt qua hight score thì hight score sẽ cập nhật điểm mới.
- Game có 3 tốc độ: thấp, trung bình và cao. Khi bạn có điểm số từ 0 – 200 bạn sẽ ở tốc độ thấp, từ 220 – 400 sẽ là tốc độ trung bình và trên 400 sẽ là tốc độ cao.

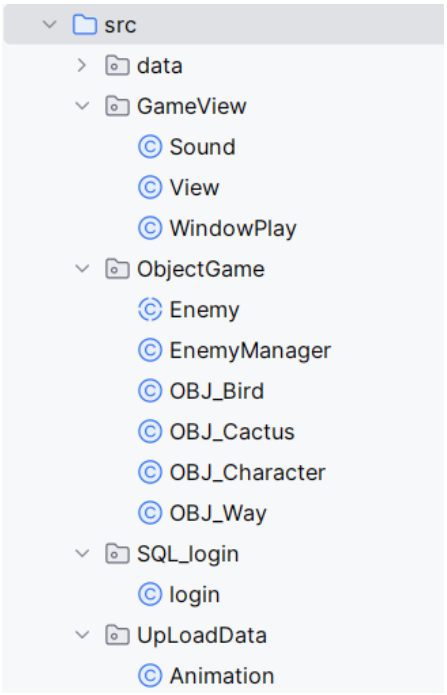
1.3 End game và restart

- Khi va chạm với chướng ngại vật như chim hoặc xương rồng, game sẽ kết thúc:

Hình 3: Game over

- Nếu muốn chơi lại ấn Enter, game sẽ restart và trở lại trạng thái new game.

III. MÔ TẢ CÁC LỚP



Hình 4: Các package và các class

3.1 Package data

Package này chứa các hình ảnh, âm thanh của các thành phần trong game như các hình ảnh main character, cactus, land…

3.2 Package GameView

3.2.1 Class WindowPlay

Class này là class main dùng để bắt đầu game, khởi tạo 1 cửa sổ với title là “DINOSAUR” có kích thước 700x500 px đặt ở giữa màn hình.

```
3 usages
public class WindowPlay extends JFrame {
    4 usages
    private static View view;
    //    private int gameState;
    1 usage
    public WindowPlay()
    {
        super( title: "DINOSAUR");
        setSize( width: 700, height: 500);
        setLocation( x: 400, y: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        view = new View();
        add(view);

        addKeyListener(view);
    }

    1 usage
    public void startGame()
    {
        view.startGame();
    }
    no usages
    public static void main(String args[]) {
        WindowPlay wp = new WindowPlay();
        wp.setVisible(true);
        wp.startGame();
    }

}
```

3.2.2 Class View

Class này để điều khiển game và kết nối cơ sở dữ liệu.

```
12 usages
public class View extends JPanel implements Runnable, KeyListener{
    3 usages
    public static final int GAME_FIRST = 0;
    5 usages
    public static final int GAME_PLAY = 1;
    4 usages
    public static final int GAME_OVER = 2;

    5 usages
    public static final float limitY = 300;

    1 usage
    public static final float speedDrop = 1f;
    2 usages
    public static int diem;

    9 usages
    private int gameState = GAME_FIRST;
    2 usages
    private BufferedImage textOver, button;

    13 usages
    private OBJ_Character character;
    6 usages
    private OBJ_Way way;
    1 usage
    private OBJ_Bird bird;
    5 usages
    private EnemyManager enemyManager;
    7 usages
    public static int score = 0;

    7 usages
    private int hightScore;
    3 usages
    private Sound SE;

    3 usages
    public Thread thread;
```

```
1 usage
public View()  {

    character = new OBJ_Character();
    bird = new OBJ_Bird(character);
    character.setPosX(150);
    character.setPosY(200);
    SE = new Sound();
    way = new OBJ_Way( v: this);
    enemyManager = new EnemyManager(character,  view: this);
    thread = new Thread( target: this);
    try {
        textOver = ImageIO.read(getClass().getResourceAsStream( name: "/data/gameover_text.png"));
        button = ImageIO.read(getClass().getResourceAsStream( name: "/data/replay_button.png"));
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void startGame()
{
    thread.start();
}

@Override
public void paint(Graphics g)
{
    super.paint(g);
    g.setColor(Color.decode( nm: "#f7f7f7"));
    g.fillRect( x: 0, y: 0,getWidth(),getHeight());
    switch (gameState){
        case GAME_FIRST:
            character.draw(g);
            g.drawString( str: "NO INTERNET MORE !!!", x: 200, y: 220);
            break;
        case GAME_PLAY:
            way.draw(g);
            enemyManager.draw(g);
            character.draw(g);
            g.drawString( str: "SCORE: "+String.valueOf(score),  x: 500, y: 20);
            g.drawString( str: "HIGHT SCORE: "+String.valueOf(hightScore),  x: 500, y: 40);
            break;
        case GAME_OVER:
            way.draw(g);
            enemyManager.draw(g);
            character.drawDead(g);
            g.drawImage(textOver, x: 250, y: 200, observer: null);
            g.drawImage(button, x: 325, y: 230, observer: null);
            g.drawString( str: "SCORE: "+String.valueOf(score),  x: 500, y: 20);
            g.drawString( str: "HIGHT SCORE: "+String.valueOf(hightScore),  x: 500, y: 40);
            break;
    }
}
```

```java
@Override
public void run() {
    while (true){
        try {
            updateGame();
            repaint();
            thread.sleep( millis: 18);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
}
```

```java
1 usage
public void updateGame()throws SQLException, ClassNotFoundException
{
    switch (gameState){
        case GAME_PLAY:
            character.update();
            if (this.getScore()>-1 && this.getScore()< 200){
                way.updateWay( i: 1);
            } else if (this.getScore()>=200 && this.getScore()< 400) {
                way.updateWay( i: 1.5);
            } else if (this.getScore()>=400) {
                way.updateWay( i: 2);
            }
            enemyManager.update();
            if (!character.getLive()){
                Connection conn = null;
                Statement stmt = null;
                try{
                    Class.forName(login.DRIVER_CLASS);
                    conn = DriverManager.getConnection(login.DB_URL, login.USER, login.PASS);
                    stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery( sql: "SELECT Diem FROM diem");
                    String sql="Update diem Set Diem=? where Diem <?";
                    PreparedStatement pstmt = conn.prepareStatement(sql);
                    pstmt.setInt( parameterIndex: 1, hightScore);
                    pstmt.setInt( parameterIndex: 2, hightScore);
                    int rowAffected = pstmt.executeUpdate();
                    while (rs.next()) {
```

```
                    diem = rs.getInt( columnLabel: "Diem");
                    System.out.print("Diem: " + diem+"\n");
                }
                rs.close();

            }catch (Exception e){
                e.printStackTrace();
            }finally {
                if (stmt != null)
                    stmt.close();
                if (conn != null)
                    conn.close();
            }
            gameState =GAME_OVER;
        }
        break;
    }
}
1 usage
public void resetGame()
{
    character.setLive(true);
    character.setPosX(150);
    enemyManager.reset();
    resetScore();

}
```

```
2 usages
public void setScore(int score){
    this.score += score;
    if (this.score > hightScore){
        hightScore = this.score;
    }


}
1 usage
public void resetScore() { this.score = 0; }

@Override
public void keyTyped(KeyEvent e) {

}

@Override
public void keyPressed(KeyEvent e) {
    switch (e.getKeyCode()){
        case KeyEvent.VK_SPACE:
            character.jump();

            break;
        case KeyEvent.VK_ESCAPE :
            if (gameState == GAME_PLAY){
                gameState = GAME_OVER;
                playSE( i: 1);
            }

    }

}
```

```
    @Override
    public void keyReleased(KeyEvent e) {
        switch (e.getKeyCode()){
            case KeyEvent.VK_SPACE:
                if (gameState == GAME_FIRST){
                    gameState = GAME_PLAY;
                }
                else {
                    playSE( i: 0);
                }
                break;
            case KeyEvent.VK_ENTER:
                if (gameState == GAME_OVER) {
                    resetGame();
                    gameState = GAME_PLAY;
                }
                break;

        }

    }
    4 usages
    public void playSE(int i)
    {
        this.SE.setUpFile(i);
        this.SE.play();

    }


    10 usages
    public int getScore() { return score; }
    no usages
    public int getHightScore() {
        {
            return hightScore;
        }

    }
}
```

3.2.3 Class Sound

Class này dùng để thêm âm thanh vào game

```
2 usages
public class Sound {
    4 usages
    Clip clip;
    3 usages
    URL sound[] = new URL[5];

    1 usage
    public Sound()
    {
        sound[0] = getClass().getResource( name: "/data/jump1.wav");

        sound[1] = getClass().getResource( name: "/data/dead.wav");



    }
    1 usage
    public void setUpFile(int i)
    {
        try {
            AudioInputStream soundInp = AudioSystem.getAudioInputStream(sound[i]);
            clip = AudioSystem.getClip();
            clip.open(soundInp);
        } catch (Exception e) {

        }
    }

    1 usage
    public void play()
    {
        clip.start();

    }
    no usages
    public void stop() { clip.stop(); }
}
```

3.3 Package ObjectGame

3.3.1 Class OBJ_Character

Class này dùng để khởi tạo nhân vật main khủng long với các chắc năng. Vẽ khủng long lên cửa sổ window và thực hiện xử lý va chạm.

```
8 usages
public class OBJ_Character{

    5 usages
    private float posX = 0;
    9 usages
    private float posY = 0;
    5 usages
    private float speedY = 0;
    11 usages
    private Animation animation_Run;
    no usages
    private BufferedImage chaImg;
    6 usages
    private Rectangle rect;
    2 usages
    private boolean live = true;
    2 usages
    private BufferedImage imgDead;
```

```
    1 usage
    public OBJ_Character(){
        try {
            animation_Run = new Animation( time: 100);
            animation_Run.addF(ImageIO.read(getClass().getResourceAsStream( name: "/data/main-character1.png")));
            animation_Run.addF(ImageIO.read(getClass().getResourceAsStream( name: "/data/main-character2.png")));
            animation_Run.addF(ImageIO.read(getClass().getResourceAsStream( name: "/data/main-character3.png")));

                imgDead = ImageIO.read(getClass().getResourceAsStream( name: "/data/main-character4.png"));

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        rect = new Rectangle();

    }

    2 usages
    public Rectangle Colision() { return rect; }
```

```
public void update()
{
    animation_Run.update();
    if (posY >= limitY-animation_Run.getFrame().getHeight()){
        speedY = 0;
        posY = limitY-animation_Run.getFrame().getHeight();
    }else {
        speedY += speedDrop;
        posY += speedY;
}
    rect.x = (int) posX;
    rect.y = (int) posY;
    rect.width = animation_Run.getFrame().getWidth()-10;
    rect.height = animation_Run.getFrame().getHeight()-10;
}

2 usages
public void draw(Graphics g){
    g.setColor(Color.BLACK);
//      g.drawRect((int)posX,(int)posY,chacRun.getFrame().getWidth(),chacRun.getFrame().getHeight());
    g.drawImage(animation_Run.getFrame(), (int)posX, (int)posY, observer: null);
}
1 usage
public  void drawDead(Graphics g)
{
    g.setColor(Color.BLACK);
    g.drawImage(imgDead,(int)posX, (int)posY, observer: null);
}
```

```
    public  void drawDead(Graphics g)
    {
        g.setColor(Color.BLACK);
        g.drawImage(imgDead,(int)posX, (int)posY, observer: null);
    }


    1 usage
    public void jump()
    {
        if (posY == limitY-animation_Run.getFrame().getHeight()) {
            speedY = -12;
            posY += speedY;



       }
    }

    2 usages
    public float getPosX() { return posX; }
    2 usages
    public void setPosX(float posX) { this.posX = posX; }

    1 usage
    public void setPosY(float posY) { this.posY = posY; }

    3 usages
    public void setLive(boolean live) { this.live = live; }
    1 usage
    public boolean getLive() { return this.live; }
}
```

3.3.2 Class OBJ_Way

Class này dùng để vẽ đường và sinh ngẫu nhiên đường.

```
2 usages
public class OBJ_Way {
    6 usages
    private BufferedImage imgWay1, imgWay2, imgWay3;
    11 usages
    private List<ImgWay> listImg;
    2 usages
    private Random rd;
    1 usage
    public OBJ_Way(View v)  {
        try {
            rd = new Random();
            imgWay1 = ImageIO.read(getClass().getResourceAsStream( name: "/data/land1.png"));
            imgWay2 = ImageIO.read(getClass().getResourceAsStream( name: "/data/land2.png"));
            imgWay3 = ImageIO.read(getClass().getResourceAsStream( name: "/data/land3.png"));
            listImg = new ArrayList<ImgWay>();
            int sizeOfWay = 700 / imgWay1.getWidth() +5;
            for (int i = 0; i< sizeOfWay; i++)
            {
                ImgWay imgWay = new ImgWay();
                imgWay.posX = (int)(i * imgWay1.getWidth());
                imgWay.img = getImgWay();
                listImg.add(imgWay);
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }
```

```
    1 usage
    private BufferedImage getImgWay()
    {
        int i = rd.nextInt( bound: 6);
        switch (i)
        {
            case 0:
                return imgWay1;
            case 2:
                return imgWay3;
            default:
                return imgWay2;
        }
    }
```

```
    3 usages
    public void updateWay(double i)
    {
        for (ImgWay imgWay: listImg) {
            imgWay.posX -= 6*i;
        }
        ImgWay firstEml = listImg.get(0);
        if (listImg.get(0).posX + imgWay1.getWidth() < 0)
        {
            firstEml.posX = listImg.get(listImg.size()-1).posX + imgWay1.getWidth();
            listImg.add(listImg.get(0));
            listImg.remove( index: 0);
        }
    }
    2 usages
    public void draw(Graphics g)
    {
        for (ImgWay imgWay:listImg) {
            g.drawImage(imgWay.img,imgWay.posX, y: (int)limitY - 20, observer: null);
        }

    }
    7 usages
    private class ImgWay{
        6 usages
        int posX;
        2 usages
        BufferedImage img;

    }
}
```

3.3.3 Class Enemy

Class này là class trừu tượng bao gồm các phương thức trừu tượng cần thiết để thiết kế một chướng ngại vật. Tạo class này giúp cho nhà sáng tạo dễ dàng tạo ra các chướng ngại vật khác bằng cách kế thừa lại class này.

```
11 usages  2 inheritors
public abstract class Enemy {
    2 usages  2 implementations
    public abstract Rectangle Colision();
    2 usages  2 implementations
    public abstract void draw(Graphics g);
    4 usages  2 implementations
    public abstract void update(double i);
    2 usages  2 implementations
    public abstract boolean Out();
    2 usages  2 implementations
    public abstract boolean isOver();
    2 usages  2 implementations
    public abstract boolean isScore();
    2 usages  2 implementations
    public abstract void setScore(boolean isScore);


}
```

3.3.4 Class EnemyManager

Class này dùng để khởi tạo, random và quản lý các chướng ngại vật, xử lý va chạm.

```
2 usages
public class EnemyManager {
    9 usages
    private List<Enemy> ens, ens1;
    3 usages
    private Random random;
    2 usages
    private BufferedImage imgCac1,imgCac2,imgCacb1,imgCacb2;
    7 usages
    private OBJ_Character character;
    1 usage
    private OBJ_Bird bird;
    10 usages
    private View view;
```

```
1 usage
public EnemyManager(OBJ_Character character, View view)
{
    this.view = view;
    this.character = character;
    bird = new OBJ_Bird(character);
    ens = new ArrayList<Enemy>();
    ens1 = new ArrayList<Enemy>();

    OBJ_Cactus cactus = new OBJ_Cactus(character);
    try {
        imgCac1 = ImageIO.read(getClass().getResourceAsStream( name: "/data/cactus1.png"));
        imgCac2 = ImageIO.read(getClass().getResourceAsStream( name: "/data/cactus2.png"));

    } catch (IOException e) {
        e.printStackTrace();
    }
    OBJ_Bird bird = new OBJ_Bird(character);
    try {
        imgCacb1 = ImageIO.read(getClass().getResourceAsStream( name: "/data/0.png"));
        imgCacb2 = ImageIO.read(getClass().getResourceAsStream( name: "/data/0.png"));

    } catch (IOException e) {
        e.printStackTrace();
    }
    random = new Random();
    ens.add(RanDom());
    ens1.add(RanDom1());
}
```

```
1 usage
public void update(){
    for (Enemy e1: ens1) {
        e1.update( i: 1);
        if (e1.isOver() && !e1.isScore()){
            view.setScore(20);
            e1.setScore(true);
        }
        if (e1.Colision().intersects(character.Colision())){
            character.setLive(false);
            view.playSE( i: 1);


        }}
```

```
        for (Enemy e: ens) {
            if (view.getScore() > -1 && view.getScore()<200){
                e.update( i: 1);
            }else if (view.getScore() >= 200 && view.getScore()< 400){
                e.update( i: 1.5);
            }else if (view.getScore() >= 400){
                e.update( i: 2);
            }
            if (e.isOver() && !e.isScore()){
                view.setScore(20);
                e.setScore(true);
            }
            if (e.Colision().intersects(character.Colision())){
                character.setLive(false);
                view.playSE( i: 1);


            }
        }
        Enemy firstE = ens.get(0);
        Enemy firstE1 = ens1.get(0);
        if (firstE.Out()){
            ens.remove(firstE);
            ens.add(RanDom());
        }
        if (firstE1.Out()){
            ens1.remove(firstE1);
            ens1.add(RanDom1());
        }

    }
```

```
2 usages
public void draw(Graphics g){
    for (Enemy e: ens) {
        e.draw(g);
    }
    for (Enemy e1: ens1
         ) {
        e1.draw(g);

    }
}
```

```
1 usage
public void reset(){
    ens.clear();
    ens.add(RanDom());
    ens1.clear();
    ens1.add(RanDom1());
}
3 usages
private OBJ_Cactus RanDom(){
    OBJ_Cactus cactus;

    cactus = new OBJ_Cactus(character);

    cactus.setX(700);

    if (random.nextBoolean()){

        cactus.setImg(imgCac1);
    }else {

        cactus.setImg(imgCac2);
    }

    return cactus;

}
```

```
    3 usages
    private OBJ_Bird RanDom1(){
        OBJ_Bird bird;

        bird = new OBJ_Bird(character);

        bird.setX(800);

        if (random.nextBoolean()){

            bird.setImg(imgCacb1);
        }else {

            bird.setImg(imgCacb2);
        }

        return bird;

    }

}
```

3.3.5 Class OBJ_Cactus

Class này kế thừa class Enemy dùng để khởi tạo ra đối tượng xương rồng với tất cả các phương thức kế thừa từ class enemy.

```
5 usages
public class OBJ_Cactus extends Enemy {
    6 usages
    private BufferedImage img;
    7 usages
    private int posX, posY;
    6 usages
    private Rectangle rect;
    2 usages
    private OBJ_Character character;
    2 usages
    private boolean isScore = false;


    2 usages
    public OBJ_Cactus(OBJ_Character character)
    {
        this.character = character;
        posX = 400;
        posY = 265;
        try {
          img = ImageIO.read(getClass().getResourceAsStream( name: "/data/cactus1.png"));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        rect = new Rectangle();

    }
    4 usages
    public void update(double i)
    {

        posX-=6*i;
        rect.x = posX;
        rect.y = posY;
        rect.width = img.getWidth()-10;
        rect.height = img.getHeight()-10;
    }
```

```
@Override
    public Rectangle Colision() { return rect; }
    2 usages
    @Override
    public void draw(Graphics g)
    {
        g.drawImage(img,posX,posY, observer: null);

    }
    1 usage
    public void setX(int x) { posX = x; }
    2 usages
    public void setImg(BufferedImage img) { this.img =img; }

    2 usages
    @Override
    public boolean Out() { return (posX + img.getWidth() < 0); }
    2 usages
    @Override
    public boolean isOver(){
        return  character.getPosX() > posX;

    }
    2 usages
    @Override
    public boolean isScore() { return isScore; }
    2 usages
    public void setScore(boolean score) { this.isScore = score; }
}
```

3.3.6 Class OBJ_bird

Class này kế thừa class Enemy dùng để khởi tạo ra đối tượng chú chim với tất cả các phương thức kế thừa từ class enemy.

```
9 usages
public class OBJ_Bird extends Enemy{
    6 usages
    private BufferedImage img;
    7 usages
    private int posX, posY;
    6 usages
    private Rectangle rect;
    2 usages
    private OBJ_Character character;
    2 usages
    private boolean isScore = false;


    4 usages
    public OBJ_Bird(OBJ_Character character)
    {
        this.character = character;
        posX = 600;
        posY = 180;
        try {
            img = ImageIO.read(getClass().getResourceAsStream( name: "/data/0.png"));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        rect = new Rectangle();

    }

    2 usages
    @Override
    public Rectangle Colision() { return rect; }

    2 usages
    @Override
    public void draw(Graphics g) {
        g.drawImage(img, posX,posY,  observer: null);

    }
```

```
    4 usages
    public void update(double i)
    {
        posX-=10*i;
        rect.x = posX;
        rect.y = posY;
        rect.width = img.getWidth()-10;
        rect.height = img.getHeight()-10;
    }

    1 usage
    public void setX(int x) { posX = x; }
    2 usages
    public void setImg(BufferedImage img) { this.img =img; }


    no usages
    public void setScore()
    {
    }
    2 usages
    public boolean Out() { return (posX + img.getWidth() < 0); }
    2 usages
    @Override
    public boolean isOver(){
        return  character.getPosX() > posX;
    }
    2 usages
    @Override
    public boolean isScore() { return isScore; }
    2 usages
    public void setScore(boolean score) { this.isScore = score; }
}
```

3.4 Package UpLoadData

Class Animation: dùng để tạo ra hoạt ảnh bằng cách duyệt lặp đi lặp lại các hình ảnh trong 1 list và tạo ra sự chuyển động.

```
3 usages
public class Animation {
    5 usages
    private List frames;
    4 usages
    private int frameIn = 0;
    2 usages
    private int time;
    2 usages
    private long previousTime = 0;
    1 usage
    public Animation(int time)
    {
        this.time = time;
        frames = new ArrayList<BufferedImage>();
    }
    1 usage
    public void update()
    {
       if (System.currentTimeMillis() - previousTime > time){
            frameIn++;
            if (frameIn>=frames.size()){
                frameIn = 0;
           }
            previousTime = System.currentTimeMillis();
       }

    }
```

```
    3 usages
    public void addF(BufferedImage frame){
        frames.add(frame);
    }
    6 usages
    public BufferedImage getFrame()
    {
        if (frames.size()>0)
        {
            return (BufferedImage) frames.get(frameIn);
        }
        return null;
    }
}
```

3.5 Package SQL_login

Class login: dùng để kết nối game với cơ sở dữ liệu.

```
5 usages
public class login {
    1 usage
    public static final String DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";
    1 usage
    public static final String DB_URL = "jdbc:mysql://localhost:3306/HH";
    1 usage
    public static final String USER = "root";
    1 usage
    public static final String PASS ="" ;
//    static final String PASS = "";

}
```

IV. TÀI LIỆU THAM KHẢO

https://www.youtube.com/@amigoscode

https://www.youtube.com/@TITVvn

https://www.youtube.com/watch?v=3Jh-jV71in8

https://practice.geeksforgeeks.org/