**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA QUỐC TẾ**

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


# Báo cáo bài tập lớn môn học
# Công nghệ Java

*Đề tài: Lập trình game rắn săm mồi*

Giảng viên hướng dẫn: Vũ Huấn

Lớp : Công nghệ thông tin Việt Anh I

Thành viên nhóm:


- Vũ Minh Thiện - 212606017
- Nguyễn Tài Anh Quân – 212611136

# MỤC LỤC

# I. Tổng quan

## 1. Giới thiệu

Bắt nguồn từ trò chơi điện tử arcade hai người chơi năm 1976 Blockade from Gremlin Industries trong đó mục tiêu là sinh tồn lâu hơn người chơi khác. Khái niệm này đã phát triển thành một biến thể chơi đơn trong đó một con rắn dài hơn với mỗi miếng thức ăn được ăn — thường là táo hoặc trứng.

## 2. Mô tả đề tài

Trong "Snake", người chơi điều khiển đầu của một đường mở rộng theo chủ đề rắn. Nó trở nên khó khăn hơn khi con rắn dài ra để ngăn nó va vào chính nó cũng như các chướng ngại vật khác. Điểm số của người chơi phụ thuộc vào số táo ăn được và sau mỗi số táo nhất định, 1 màn chơi mới sẽ được mở. Khi đó tốc độ của rắn sẽ tăng lên sau mỗi level, người chơi chỉ thua khi để đầu rắn chạm vào thân rắn.

# II. Xây dựng đề tài

## 1. Lớp GameScreen

```
package Screen;

import java.awt.Color;

public class GameScreen extends JPanel implements Runnable {
    SnakeRepo repo = new SnakeRepo();
    ArrayList<User> list = (ArrayList<User>) repo.getAll();

    static int[][] bg = new int[20][20];
    Thread thread;
    Snake s;
    static int padding = 10;
    static int WIDTH = 400;
    static int HEIGHT = 400;
    static boolean isPlaying = false;
    static boolean textStartGame = true;
    static boolean isGameOver = false;
    static int currentLevel = 1;
    static int gamePoint = 0;
    static boolean pause = false;

    public GameScreen() {
        s = new Snake();
        DataSnake.loadImage();
        DataSnake.loadAnimation();
        bg[10][7] = 2;
        thread = new Thread(this);
        thread.start();
    }
```

- Chứa các phương thức để tạo game screen bằng 1 ma trận 20x20, bắt đầu, kết thúc game và lưu thông tin người chơi.

1.1. Hàm PaintBG, Vien

```
public void PaintBg(Graphics g) {
    g.setColor(Color.black);
    g.fillRect(0, 0, WIDTH + padding * 2 + 370, HEIGHT + padding * 2);
    for (int i = 0; i < 20; i++) {
        for (int j = 0; j < 20; j++) {
            if (bg[i][j] == 2) {
                s.viTriMoi();
                g.drawImage(DataSnake.tao2, i * 20 - 6 + padding, j * 20 - 6 + padding, null);
            }
        }

    }
}

public void Vien(Graphics g) {
    g.setColor(Color.white);
    g.drawRect(0, 0, WIDTH + padding * 2, HEIGHT + padding * 2);
    g.drawRect(1, 1, WIDTH + padding * 2 - 2, HEIGHT + padding * 2 - 2);
    g.drawRect(2, 2, WIDTH + padding * 2 - 4, HEIGHT + padding * 2 - 4);

    g.drawRect(0, 0, WIDTH + padding * 2 + 366, HEIGHT + padding * 2);
    g.drawRect(1, 1, WIDTH + padding * 2 - 2 + 366, HEIGHT + padding * 2 - 2);
    g.drawRect(2, 2, WIDTH + padding * 2 - 4 + 366, HEIGHT + padding * 2 - 4);
}
```

- Hai phương thức trên được dùng để vẽ nền cho game, background màu đen và phần khung viền để chia màn hình game .

1.2. Hàm run

```
public void run() {
    long t = 0;
    long t1 = 0;
    while (true) {

        if (System.currentTimeMillis() - t1 > 500) {
            textStartGame = !textStartGame;
            t1 = System.currentTimeMillis();
        }

        if (isPlaying) {
            s.Update();
            s.BgMusic.loop();
        }
        repaint();
        try {
            Thread.sleep(20);
        } catch (InterruptedException e) {
        }

    }
```

- Khi game được chạy sẽ chạy nhạc đồng thời vẽ lên màn hình, tốc độ chạy của rắn.

1.3. Hàm paint

```
    public void paint(Graphics g) {
        PaintBg(g);
        s.Draw(g);
        Vien(g);
        if (!isPlaying) {
            if (textStartGame) {
                g.setColor(Color.yellow);
                g.setFont(getFont().deriveFont(20.0f));
                g.drawString("Press Space To Start Game!", 80, 230);
            }

        }
        if (pause) {
            g.setColor(Color.yellow);
            g.setFont(getFont().deriveFont(20.0f));
            g.drawString("Press Enter To Continue Game!", 80, 230);

        }

        if (isGameOver) {
            g.setColor(Color.yellow);
            g.setFont(getFont().deriveFont(40.0f));
            g.drawString("Game Over!", 100, 200);
        }
        g.setColor(Color.yellow);
        g.setFont(getFont().deriveFont(28.0f));
        g.drawString("LEVEL " + currentLevel, 450, 40);
        g.setColor(Color.yellow);
        g.setFont(getFont().deriveFont(24.0f));
        g.drawString("SCORES: " + gamePoint, 450, 80);

        g.setColor(Color.yellow);
        g.setFont(getFont().deriveFont(20.0f));
        g.drawString("PLAYER LIST", 540, 120);

        for (int i = 0; i < Frame.users2.size(); i++) {
            g.setColor(Color.yellow);
            g.setFont(getFont().deriveFont(20.0f));
            g.drawString(Frame.users2.get(i).toString(), 450, i * 30 + 150);

        }

    }
```

- Vẽ lên màn hình nhưng thông tin cần thiết như thông báo bắt đầu game, kết thúc, điểm số, level và danh sách những người đã chơi.

## 2. Lớp Frame

### 2.1. Hàm Frame

```
import java.awt.event.KeyEvent;
public class Frame extends JFrame {
    GameScreen gameScreen;
    SnakeRepo ran;
    public static ArrayList<User> users;
    public static ArrayList<User2> users2;


    public Frame() {
        users = new ArrayList<>();
        users2 = new ArrayList<>();
        gameScreen = new GameScreen();
        ran = new SnakeRepo();

        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(800, 458);
        add(gameScreen);
        this.addKeyListener(new KeyHandler());
        this.addWindowListener(new WindowAdapter() {

            @Override
            public void windowClosing(WindowEvent e) {
                updateData();
            }

        });
        setVisible(true);
        setLocationRelativeTo(null);

    }
```

- Tạo game screen trên màn hình bao gồm độ rộng, độ dài, tắt window khi ấn X. Bên cạnh đó còn có 2 list người chơi để lưu thông tin, 1 để lưu vào file và 1 để lưu vào cơ sở dữ liệu SQL.

### 2.2. Hàm điều khiển

```
    private class KeyHandler implements KeyListener {

        @Override
        public void keyTyped(KeyEvent e) {

        }

        @Override
        public void keyPressed(KeyEvent e) {
            if (e.getKeyCode() == KeyEvent.VK_SPACE) {
                GameScreen.isPlaying = !gameScreen.isPlaying;
                if (!gameScreen.isPlaying) {
                    gameScreen.s.BgMusic.stop();
                }
                if (gameScreen.isGameOver) {
                    gameScreen.isGameOver = false;
                    gameScreen.s.resetGame();
                }
            }

//          if(e.getKeyCode() == KeyEvent.VK_R) {
//
//                  gameScreen.s.daoNguoc();
//
//          }


            if (e.getKeyCode() == KeyEvent.VK_UP) {
                gameScreen.s.setVector(Snake.GO_UP);
            }
            if (e.getKeyCode() == KeyEvent.VK_DOWN) {
                gameScreen.s.setVector(Snake.GO_DOWN);
            }
            if (e.getKeyCode() == KeyEvent.VK_LEFT) {
                gameScreen.s.setVector(Snake.GO_LEFT);
            }
            if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
                gameScreen.s.setVector(Snake.GO_RIGHT);
            }

        }
```

- Thêm các chức năng di chuyển cho người chơi bao gồm lên, xuống, trái , phải. Bấm Space để bắt đầu game đồng thời cho nhạc chạy, nếu như Game Over sẽ reset game và chạy lại bằng cách nhấn Space.

## 2.3. Hàm lưu và đọc dữ liệu

```
public void updateData() {
    BufferedWriter bw = null;
    try {
        FileWriter fw = new FileWriter("Data/data.txt");
        bw = new BufferedWriter(fw);
        for(User2 u : users2) {
             bw.write(u.getName2()+" "+u.getLevel2()+" "+u.getScores2());
             bw.newLine();
        }

        bw.close();
    } catch (IOException e) {}
    finally {
        try {
            bw.close();
        } catch (Exception e2) {}
    }
}
```

- Đặt 1 biến làm con chạy để cập nhật dữ liệu người chơi trong file Data.txt bao gồm Tên + Level + Số điểm.

```
public void readData() {
    try {
        FileReader fr= new FileReader("Data/data.txt");
        BufferedReader br= new BufferedReader(fr);
        String line = null;
        while((line= br.readLine())!= null) {
            String[] str = line.split(" ");
            users2.add(new User2(str[0], str[1], str[2]));

        }
        br.close();
    } catch (IOException e) {
```

- Hàm đọc dữ liệu từ file Data.txt sau đó thêm tên người chơi mới nhập vào bảng điểm.

## 3. Lớp Snake

### 3.1. Hàm vẽ rắn

```
public class Snake {
    int doDai = 3;
    int[] x;
    int[] y;
    long t = 0;
    boolean unChange = false;
    boolean save = false;
    public SoundPlayer eatApple, snakeOver, levelUp, BgMusic;
    SnakeRepo repo = new SnakeRepo();
    //ArrayList<User> list = new ArrayList<>();
    public static int GO_UP = 1;
    public static int GO_DOWN = -1;
    public static int GO_LEFT = 2;
    public static int GO_RIGHT = -2;

    int vector = Snake.GO_DOWN;

    int speed = 300;
    int LVMax = 10;

    public Snake() {
        x = new int[100];
        y = new int[100];
        x[0] = 5;
        y[0] = 4;
        x[1] = 5;
        y[1] = 3;
        x[2] = 5;
        y[2] = 2;
```

- Cho độ dài rắn là 3 ô vuông trong ma trận 20 ô, gắn mỗi hướng đi của rắn là 1 giá trị để dễ kiểm soát hướng di chuyển. Speed lúc đầu sẽ được đặt là 300ms để rắn đi được 1 ô. Tiếp đến là đặt rắn ở 1 vị trí cụ thể.

#### *3.1.1. Hàm resetGame*

- Đơn giản là vẽ lại rắn ở 1 vị trí cho sẵn sau khi gameOver

```
public void resetGame() {
    x = new int[100];
    y = new int[100];
    x[0] = 5;
    y[0] = 4;

    x[1] = 5;
    y[1] = 3;

    x[2] = 5;
    y[2] = 2;
    doDai=3;
    vector = GO_DOWN;
```

### *3.1.2. Hàm kiểm tra táo, reset táo, tăng tốc, vector*

```
public void setVector(int v) {
    if (vector != -v && unChange)
        vector = v;
    unChange = false;
}

public Point viTriMoi() {
    Random r = new Random();
    int x, y;
    do {
        x = r.nextInt(19);
        y = r.nextInt(19);
    } while (kiemTraMoi(x, y));
    return new Point(x, y);
}

public boolean kiemTraMoi(int x1, int y1) {
    for (int i = 0; i < doDai; i++) {
        if (x[i] == x1 && y[i] == y1)
            return true;

    }
    return false;
}
public int getCurrentLevelSpeed() {
    for (int i = 0; i < GameScreen.currentLevel; i++) {
        speed *= 0.8;
    }
    return speed;
}
```

- Hàm vector sử dụng các biến được khai báo ở trên để hoạt động tránh trường hợp rắn đi lên nhưng vẫn có thể đi xuống. Giới hạn di chuyển của rắn từ 180 độ xuống còn 90 độ.

- Hàm viTriMoi sẽ chạy khi ngay sau khi rắn ăn 1 táo trước đó, và để tránh táo chui vào người rắn thì hàm kiemtraMoi sẽ xử lí vấn đề này.

- Hàm tốc độ điều chỉnh tốc độ sau mỗi level từ 300ms sau đó cứ mỗi level sẽ nhân với 0.8 .

## 3.2. Hàm Update

### 3.2.1. Update và lấy dữ liệu

```
public void Update() {
    if(doDai == LVMax) {
        resetGame();
        GameScreen.currentLevel++;
        levelUp.play();
        LVMax+=5;
        speed  = getCurrentLevelSpeed();
    }

    for (int i = 1; i <doDai; i++) {
        if(x[0]==x[i] && y[0]==y[i]) {
            ArrayList<User> list ;
            String name = JOptionPane.showInputDialog("Moi Nhap Ten Player:");
            Frame.users2.add(new User2(name, String.valueOf(GameScreen.currentLevel), String.valueOf(GameScreen.gamePoint)
            User user = new User(name, String.valueOf(GameScreen.currentLevel), String.valueOf(GameScreen.gamePoint));
            repo.add(user);
            <User>) replist = (Arrayll();ListoetA.g
            loadData(list);
            GameScreen.isPlaying= false;
            GameScreen.isGameOver = true;
            snakeOver.play();
            GameScreen.gamePoint =0;
            GameScreen.currentLevel =1;
            speed = 300;
            LVMax =10;
```

- Với mỗi số táo cố định ăn được người chơi sẽ được lên 1 level cao hơn, rắn sẽ ngắn lại và speed sẽ tăng dần lên sau mỗi lần.

- Trong vòng lặp for, khi người chơi cho rắn cắn vào chính nó sẽ hiện lên thông báo nhập tên, sau đó lấy tên vừa nhập + số điểm + level vừa đạt được để lưu thông tin.

### 3.2.2. Animation phần đầu và tăng điểm

```
if (System.currentTimeMillis() - t > speed) {
    unChange = true;
    DataSnake.headGoUp.update();
    DataSnake.headGoDown.update();
    DataSnake.headGoLeft.update();
    DataSnake.headGoRight.update();
    if (GameScreen.bg[x[0]][y[0]] == 2) {
        doDai++;
        GameScreen.gamePoint+= 10;
        eatApple.play();
        GameScreen.bg[x[0]][y[0]] = 0;
        GameScreen.bg[viTriMoi().x][viTriMoi().y] = 2;
    }
```

- Sử dụng animation trong phần DataSnake để vẽ từng hướng đi của rắn

- Biến táo được đặt bằng 2 nên khi x[0], y[0] = 2 (ăn táo) thì độ dài của rắn sẽ được cộng thêm 1 ô đồng thời cộng 10 điểm cho người chơi, sau đó tạo 1 táo mới ở vị trí khác.

### 3.2.3. Hướng di chuyển và cơ chế đi xuyên tường

```
        for (int i = doDai - 1; i > 0; i--) {
            x[i] = x[i - 1];
            y[i] = y[i - 1];
        }
        if (vector == Snake.GO_UP) {
            y[0]--;
        }
        if (vector == Snake.GO_DOWN) {
            y[0]++;
        }
        if (vector == Snake.GO_RIGHT) {
            x[0]++;
        }
        if (vector == Snake.GO_LEFT) {
            x[0]--;
        }
        if (x[0] < 0) {
            x[0] = 19;
        }
        if (x[0] > 19) {
            x[0] = 0;
        }
        if (y[0] < 0) {
            y[0] = 19;
        }
        if (y[0] > 19) {
            y[0] = 0;
        }
        t = System.currentTimeMillis();
    }
```

- Với mỗi hướng đi sẽ tương ứng với 1 vector, gốc tọa độ được đặt ở góc trên bên trái màn nên mỗi hướng đi sẽ ảnh hưởng bởi x và y.

- Khi đầu rắn (x[0]) đi vào tường thì sẽ có thể đi qua đó và đi tiếp ở phía đối diện, khi x[0] chạm 0 trong ma trận có 20 thì sẽ cho x[0] xuất hiện lại ở ô cuối cùng trong cùng hàng, ở đây là ô 19 và kéo theo phần thân của rắn. Áp dụng tương tự với 3 bức tường còn lại chỉ khác là thay đổi giữa x[0] và y[0]

### 3.2.4. Hàm Draw

```
public void Draw(Graphics g) {
    g.setColor(Color.red);
    for (int i = 1; i < doDai; i++) {
        g.drawImage(DataSnake.imagebody, x[i] * 20 + GameScreen.padding, y[i] * 20 + GameScreen.padding, null);

        if(vector == GO_UP)
            g.drawImage(DataSnake.headGoUp.getCurrentImage(), x[0]*20-6+ GameScreen.padding, y[0]*20-6+ GameScreen.padding, null);
        else if (vector == GO_LEFT)
            g.drawImage(DataSnake.headGoLeft.getCurrentImage(), x[0]*20-6+ GameScreen.padding, y[0]*20-6+ GameScreen.padding, null);
        else if (vector == GO_RIGHT)
            g.drawImage(DataSnake.headGoRight.getCurrentImage(), x[0]*20-6+ GameScreen.padding, y[0]*20-6+ GameScreen.padding, null);
        else if (vector == GO_DOWN)
            g.drawImage(DataSnake.headGoDown.getCurrentImage(), x[0]*20-6+ GameScreen.padding, y[0]*20-6+ GameScreen.padding, null);
    }
```

- Cụ thể, vòng lặp for sẽ lặp qua tất cả các phần tử của mảng x và y, đại diện cho tọa độ của từng khối của con rắn. Với mỗi khối, phương thức sẽ vẽ một hình ảnh của khối đó lên màn hình với vị trí được tính dựa trên tọa độ của khối đó.

- Ở phần sau, phương thức sử dụng câu lệnh if-else để vẽ đầu của con rắn. Đầu của con rắn được vẽ bằng cách sử dụng hình ảnh của đầu con rắn, tùy thuộc vào hướng di chuyển hiện tại của con rắn.

- Cuối cùng, padding được sử dụng để giữ khoảng cách giữa con rắn và cạnh của màn hình.

4. **Lớp Animation**

```
package Screen;

import java.awt.Image;

public class Animation {
    public Image[] images;
    int n;
    public int currentImage ;
    public Animation(){
         n=0;
         currentImage = 0;
    }

    public void addImage(Image image) {
        Image[] arr = images;
        images = new Image[n+1];
        for(int i=0; i<n;i++) images[i] =  arr[i];
        images[n] = image;
        n++;
    }
    public void update() {
        currentImage++;
        if(currentImage >=n) currentImage =0;
    }
    public Image getCurrentImage() {
        return images[currentImage];
    }
}
```

- Tạo hoạt ảnh của đầu rắn khi di chuyển, phương thức addImage để thêm một hình ảnh mới vào mảng. Trong phương thức này, hình ảnh được thêm vào cuối mảng images bằng cách tạo một mảng mới có kích thước lớn hơn một đơn vị so với mảng cũ, sau đó sao chép các phần tử từ mảng cũ sang mảng mới và thêm hình ảnh mới vào cuối mảng. Biến n được tăng thêm một đơn vị để thể hiện số lượng hình ảnh hiện tại trong mảng. Sau khi hiển thị ra ảnh mới thì sẽ lấy lại ảnh từ lúc đầu.

5. **Lớp SoundPlayer**

```java
package Screen;

import java.io.File;

public class SoundPlayer {

    private Clip clip;

    public SoundPlayer(File path){
        try{
            AudioInputStream ais;
            ais = AudioSystem.getAudioInputStream(path);
            AudioFormat baseFormat = ais.getFormat();
            AudioFormat decodeFormat = new AudioFormat(
                    AudioFormat.Encoding.PCM_SIGNED,
                    baseFormat.getSampleRate(),
                    16,
                    baseFormat.getChannels(),
                    baseFormat.getChannels()*2,
                    baseFormat.getSampleRate(),
                    false
            );
            AudioInputStream dais = AudioSystem.getAudioInputStream(decodeFormat, ais);
            clip = AudioSystem.getClip();
            clip.open(dais);
        }catch(Exception e){}
    }
```

- Sử dụng lớp SoundPlayer để phát các tệp âm thanh khi game chạy, khai báo 1 biến clip để giải mã các tệp âm thanh và phát ra trên màn hình.

```java
    public void play(){
        if(clip !=null){
            stop();
            clip.setFramePosition(0);
            clip.start();
        }
    }
    public void stop(){
        if(clip.isRunning()) clip.stop();
    }

    public void close(){
        clip.close();
    }
    public void loop() {
        clip.loop(Clip.LOOP_CONTINUOUSLY)
        ;
        clip.start();
    }
```

- Các phương thức chạy nhạc, dừng nhạc, loop để sử dụng cho các class khác

### 6. Lớp User, User2

```java
package Screen;

public class User {
      private String name;
      private String level;
      private String scores;

    public User() {
    }

    public User(String name, String level, String scores) {
        this.name = name;
        this.level = level;
        this.scores = scores;
    }

    public String getName() {
        return name;
    }

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

    public String getLevel() {
        return level;
    }

    public void setLevel(String level) {
        this.level = level;
    }

    public String getScores() {
        return scores;
    }

    public void setScores(String scores) {
        this.scores = scores;
    }

    @Override
    public String toString() {
        return  "Name:"+name +" "+"LV:"+ level+" "+"Diem:" + scores;
    }
}
```

- ‘private String name’; ‘private String level’; ‘private String scores’ : Các biến thành viên của lớp, chứa thông tin về tên, cấp độ và điểm số của người dùng.

- Các phương thức getter và setter để truy xuất hoặc thiết lập thông tin của người dùng, sau đó ghép các thông tin thành 1 chuỗi

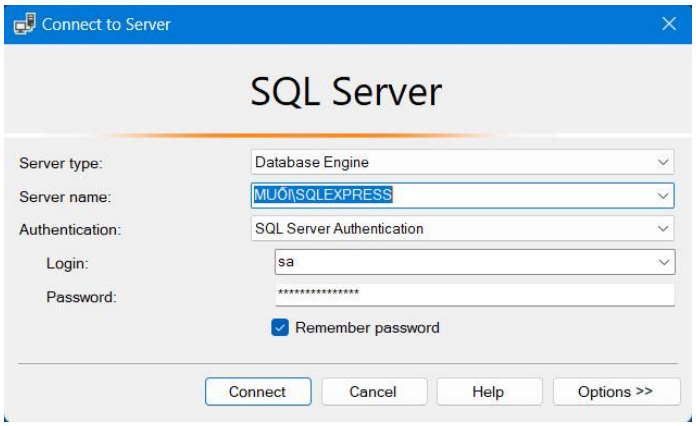
## 7. Lớp lưu trữ sang SQL(sqlConnection, SnakeRepo)

### 7.1. Kết nối Database

```java
package Screen;

import java.sql.*;

/**
 *
 * @author vha74
 */
public class sqlConnection {
    public static final String HOSTNAME = "localhost";
    public static final String PORT = "1433";
    public static final String DBNAME = "duLieuGame";
    public static final String USERNAME = "sa";
    public static final String PASSWORD = "123456789";
```




- Tạo class liên kết dữ liệu người chơi với database ở sql gồm host, port, database, username và password

```java
/**
 * Get connection to MSSQL Server
 *
 * @return Connection
 */
public static Connection getConnection() {

    // Create a variable for the connection string.
    String connectionUrl = "jdbc:sqlserver://" + HOSTNAME + ":" + PORT + ";"
            + "databaseName=" + DBNAME + ";encrypt=true;trustServerCertificate=true;";

    try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        return DriverManager.getConnection(connectionUrl, USERNAME, PASSWORD);
    } // Handle any errors that may have occurred.
    catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace(System.out);
    }
    return null;
}

public static void main(String[] args) {
    System.out.println(getConnection());
}
```

- ‘public static Connection getConnection() { ... }’: Phương thức này sử dụng JDBC (Java Database Connectivity) để tạo kết nối đến cơ sở dữ liệu SQL Server. Đầu tiên, nó tạo một chuỗi kết nối (connectionUrl) bằng cách sử dụng các biến HOSTNAME, PORT, DBNAME, USERNAME và PASSWORD. Sau đó, phương thức tải lớp trình điều khiển JDBC cho SQL Server bằng cách sử dụng phương

thức Class.forName(). Cuối cùng, phương thức trả về một đối tượng Connection để đại diện cho kết nối đến cơ sở dữ liệu SQL Server.

- Sử dụng phương thức getConnection() để lấy đối tượng kết nối đến cơ sở dữ liệu SQL Server

7.2. Lấy dữ liệu từ game và truyền vào Database

```
import java.sql.Connection;

public class SnakeRepo {
    public List<User> getAll() {
        String query = "Select *from snakeUser";
        try (Connection con = sqlConnection.getConnection(); PreparedStatement ps = con.prepareStatement(query);) {
            ResultSet rs = ps.executeQuery();
            List<User> listkh = new ArrayList<>();
            while (rs.next()) {
                User kh = new User(rs.getString(1), rs.getString(2), rs.getString(3));
                listkh.add(kh);
            }
            return listkh;

        } catch (Exception e) {
            e.printStackTrace(System.out);
        }
        return null;
    }
```

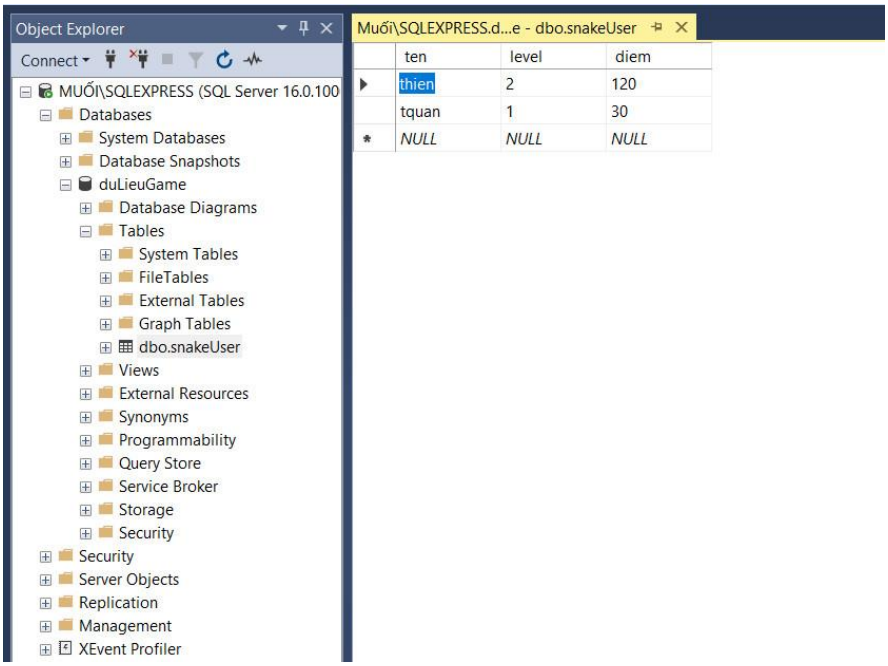
- Phương thức getAll() trong lớp SnakeRepo, được sử dụng để truy xuất dữ liệu từ cơ sở dữ liệu.

- Đầu tiên, câu lệnh truy vấn SQL được xây dựng: "Select *from snakeUser". Sau đó, phương thức sử dụng getConnection() của đối tượng sqlConnection để thiết lập kết nối đến cơ sở dữ liệu. Một PreparedStatement được tạo ra để thực thi câu lệnh truy vấn SQL trên kết nối được thiết lập. Kết quả trả về từ truy vấn được lưu trữ trong đối tượng ResultSet.

- Phương thức này được sử dụng để lấy tất cả các bản ghi trong bảng snakeUser ở Database và trả về chúng dưới dạng danh sách các đối tượng User.

```
    public boolean add(User kh) {
        String query = "insert into snakeUser(ten,level,diem) values (?,?,?)";
        int check = 0;
        try (Connection con = sqlConnection.getConnection(); PreparedStatement ps = con.prepareStatement(query);) {
            ps.setObject(1, kh.getName());
            ps.setObject(2, kh.getLevel());
            ps.setObject(3, kh.getScores());
            check = ps.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace(System.out);
        }
        return check > 0;
    }

    public static void main(String[] args) {
        System.out.println(new SnakeRepo().getAll());
    }
}
```




- Phương thức trên được dùng để thêm dữ liệu vào SQL và lưu vào bảng SnakeUser với mỗi thông tin in trong 1 cột.

# TÀI LIỆU THAM KHẢO


https://zetcode.com/javagames/snake/

https://viettuts.vn/java-jdbc/ket-noi-java-voi-sqlserver

https://www.w3schools.com/java/default.asp