一、引言

1.1 背景与目的

本项目旨在创建一个基于Java的图形化界面（GUI）和网络编程的多人游戏——**猜数字游戏**。该游戏通过客户端和服务器之间的网络通信，允许多个玩家连接到同一个服务器进行游戏。玩家需要在规定范围内猜测一个由服务器随机生成的数字，服务器会返回相应的提示信息，告知玩家数字猜测是“太高了”、“太低了”还是“正确”。

通过这个游戏项目，我们可以实践以下几个方面的技术：

- **图形用户界面（GUI）**：使用Java Swing实现游戏的前端界面。
- **网络编程**：通过Java Socket技术实现客户端和服务器之间的网络通信。
- **多线程编程**：服务器端支持多个客户端同时连接，并且客户端能够异步接收服务器的反馈信息。

1.2 目标

本游戏的目标是：

- 为客户端提供简单的界面，让玩家能够输入猜测的数字并查看提示信息。
- 服务器能够处理来自多个客户端的请求，并判断玩家的猜测是否正确。
- 游戏支持多人连线，客户端与服务器之间的通信基于Socket协议。

二、总体设计

2.1 系统架构

本系统基于客户端-服务器架构，主要分为**客户端**和**服务器端**两个部分：

1. **服务器端**：
   - 监听客户端连接请求。
   - 为每个连接的客户端生成一个目标数字，并接收玩家的猜测。
   - 判断玩家的猜测是否正确，返回相应的提示。
   - 可以同时处理多个客户端的连接请求，确保游戏能够多玩家进行。
2. **客户端**：
   - 启动后与服务器建立网络连接。
   - 提供图形化界面，允许玩家输入猜测的数字，并显示来自服务器的反馈。
   - 将玩家的猜测数字通过网络发送到服务器，并显示返回的结果。
3. **网络通信**：
   - 基于 **TCP/IP Socket** 协议进行客户端与服务器之间的数据传输。
   - 客户端与服务器之间的通信是同步的，服务器通过多线程处理每个客户端的连接请求。

2.2 技术栈

- **编程语言**：Java

- **GUI**：Java Swing（用于构建游戏的图形界面）
- **网络通信**：Java Socket编程
- **多线程**：Java线程（`Thread` 类）用于实现服务器同时处理多个客户端连接。

三、详细设计

3.1 服务器端设计

服务器端的主要功能是接收客户端的连接请求，生成目标数字，并接收客户端猜测的数字，判断结果后返回反馈信息。为了支持多个客户端同时连接，服务器端采用多线程设计。

3.1.1 服务器主类 `GuessNumberServer`

服务器端启动时会创建一个 `ServerSocket`，监听来自客户端的连接请求，并为每个连接的客户端创建一个新的线程进行处理。

JAVA

```java
import java.io.*;
import java.net.*;
import java.util.*;

public class GuessNumberServer {
    private static final int PORT = 12345;
    private static final int MAX_GUESS = 100;
    private static Random random = new Random();

    public static void main(String[] args) {
        System.out.println("Server started...");
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            while (true) {
                // 等待客户端连接
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected!");

                // 为每个客户端创建一个处理线程
                new ClientHandler(clientSocket).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static class ClientHandler extends Thread {
        private Socket socket;
        private int targetNumber;
        private PrintWriter out;
        private BufferedReader in;

        public ClientHandler(Socket socket) {
            this.socket = socket;
            this.targetNumber = random.nextInt(MAX_GUESS) + 1;
        }

        @Override
        public void run() {
            try {
                in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                out = new PrintWriter(socket.getOutputStream(), true);

                out.println("Welcome to Guess the Number Game! The number is between
1 and 100.");
                String guess;
```

```
                while ((guess = in.readLine()) != null) {
                    try {
                        int userGuess = Integer.parseInt(guess);
                        if (userGuess < targetNumber) {
                            out.println("Too low! Try again.");
                        } else if (userGuess > targetNumber) {
                            out.println("Too high! Try again.");
                        } else {
                            out.println("Congratulations! You guessed the right
number!");
                            break;
                        }
                    } catch (NumberFormatException e) {
                        out.println("Invalid input. Please enter a number.");
                    }
                }
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

主要功能：

- 使用 `ServerSocket` 监听端口，等待客户端连接。
- 每当客户端连接时，为每个客户端创建一个新线程来处理该客户端的猜测。
- 每个客户端有一个随机生成的目标数字，接收玩家猜测并给予反馈。

3.2 客户端设计

客户端提供了一个图形用户界面（GUI），让玩家能够输入猜测的数字，并显示来自服务器的反馈。客户端与服务器通过 Socket 进行通信。

3.2.1 客户端主类 `GuessNumberClient`

客户端通过 `Socket` 与服务器建立连接，通过输入框获取玩家输入的数字，并在点击按钮后将数字发送给服务器。

JAVA

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class GuessNumberClient {
    private static final String SERVER_ADDRESS = "localhost";
    private static final int SERVER_PORT = 12345;

    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;

    private JTextField guessField;
    private JTextArea resultArea;

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            try {
                new GuessNumberClient().start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
    }

    public void start() throws IOException {
        socket = new Socket(SERVER_ADDRESS, SERVER_PORT);
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        out = new PrintWriter(socket.getOutputStream(), true);

        JFrame frame = new JFrame("Guess the Number Game");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout());

        JLabel guessLabel = new JLabel("Enter your guess (1-100): ");
        panel.add(guessLabel);

        guessField = new JTextField(10);
        panel.add(guessField);

        JButton guessButton = new JButton("Guess");
```

```
        panel.add(guessButton);

        resultArea = new JTextArea(10, 30);
        resultArea.setEditable(false);
        panel.add(new JScrollPane(resultArea));

        guessButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    sendGuess();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
        });

        frame.add(panel);
        frame.setVisible(true);

        // 接收服务器消息
        new Thread(() -> {
            try {
                String message;
                while ((message = in.readLine()) != null) {
                    resultArea.append(message + "\n");
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }).start();
    }

    private void sendGuess() throws IOException {
        String guess = guessField.getText();
        out.println(guess);
        guessField.setText(""); // 清空输入框
    }
}
```

主要功能：

- 使用Swing组件创建GUI：包括标签、文本框和按钮。
- 通过Socket与服务器通信，将玩家输入的猜测数字发送给服务器，并显示服务器的反馈信息。

四、测试与运行

4.1 测试目标

本系统的测试目标是验证客户端和服务器的功能是否正常，确保游戏能够顺利进行。但本项目尚未debug完成。

4.2 测试内容

1. **客户端功能测试**：
   - 输入数字，查看是否能够正确发送给服务器。
   - 显示服务器返回的结果（例如“太高了”、“太低了”或者“恭喜你猜对了”）。
2. **服务器功能测试**：
   - 服务器能否正确接收并处理多个客户端的请求。
   - 服务器能够生成正确的随机数字，并根据玩家猜测的结果返回正确的提示。
3. **网络连接测试**：
   - 客户端能否正确连接到服务器，并维持稳定的通信。

4.3 运行步骤

1. 启动服务器：运行 `GuessNumberServer` 类启动服务器。
2. 启动客户端：运行 `GuessNumberClient` 类启动客户端，多个客户端可以同时启动进行测试。

五、总结

本项目实现了一个简单的网络游戏，涵盖了**图形用户界面设计**、**Socket编程**、**多线程技术**等多个方面。游戏提供了基本的功能：玩家输入数字并与服务器进行交互，服务器返回相应的提示。

未来改进方向：

1. **增强功能**：
   - 支持更多类型的游戏模式，如限时挑战、排行榜等。
   - 加入玩家昵称、聊天功能等社交功能。
2. **性能优化**：
   - 服务器端可以改为线程池形式，以提高对多个玩家的支持能力。
3. **界面优化**：
   - 美化游戏界面，增加动画效果、音效等提升用户体验。