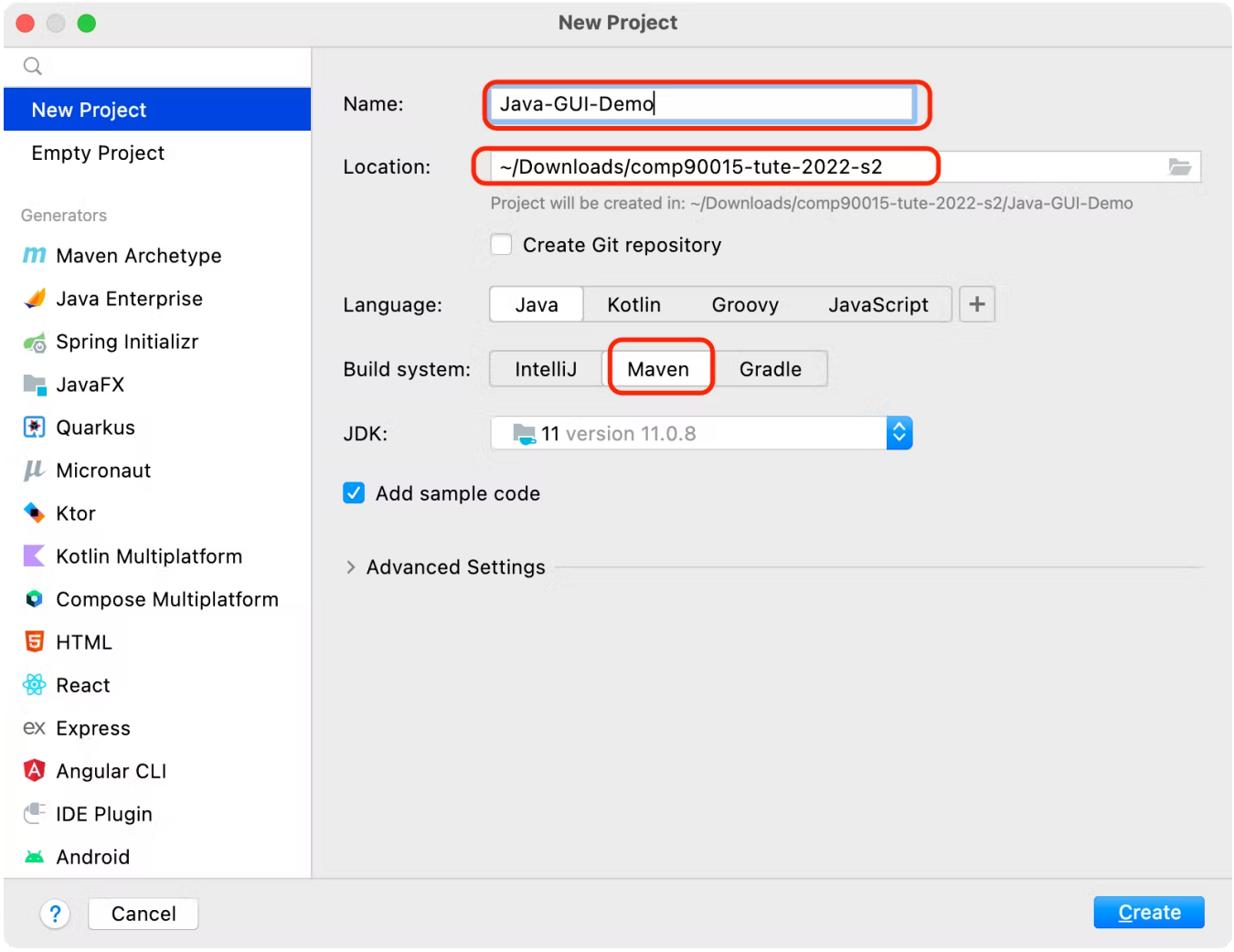
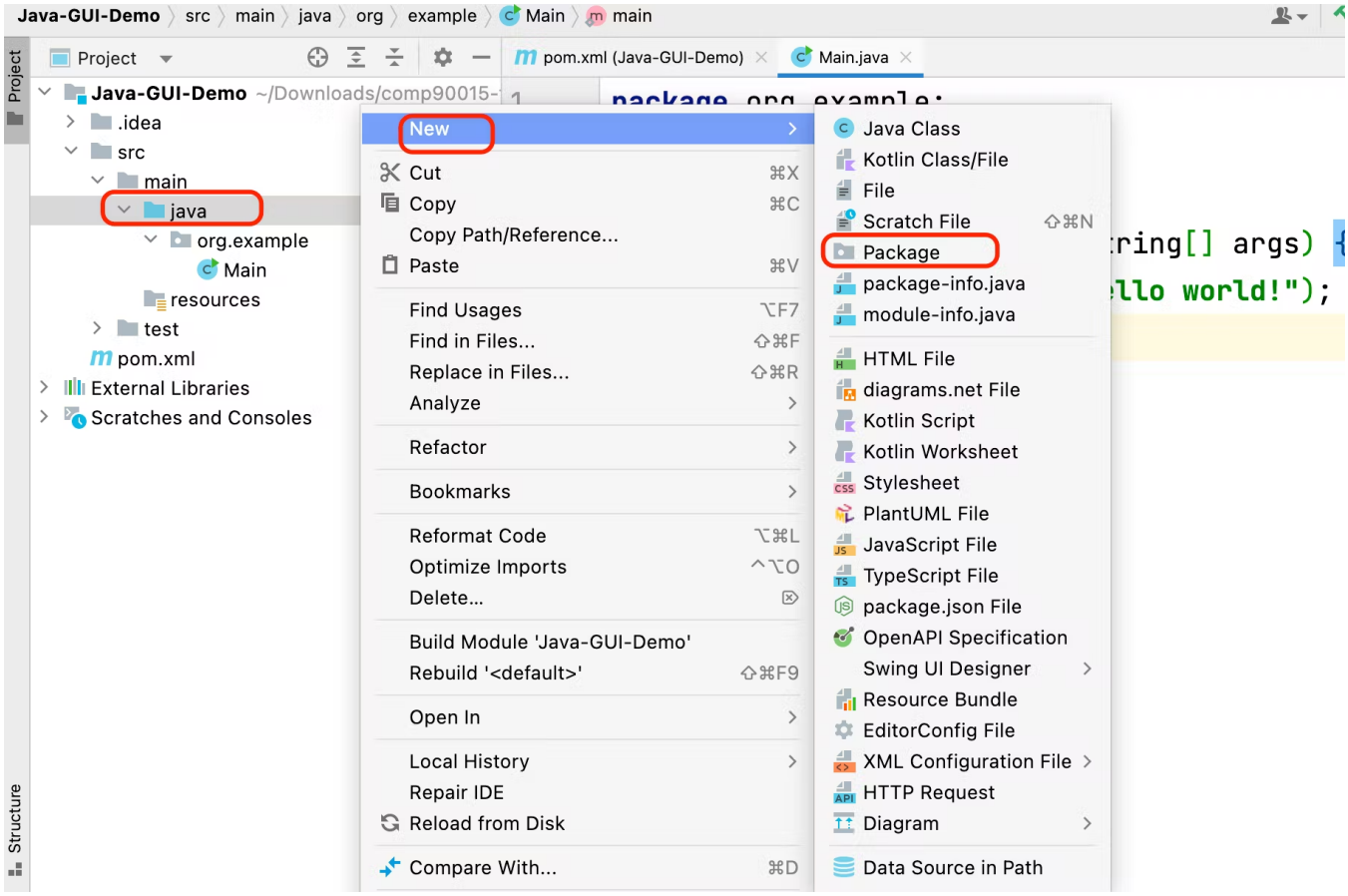
GUI Demo Instruction

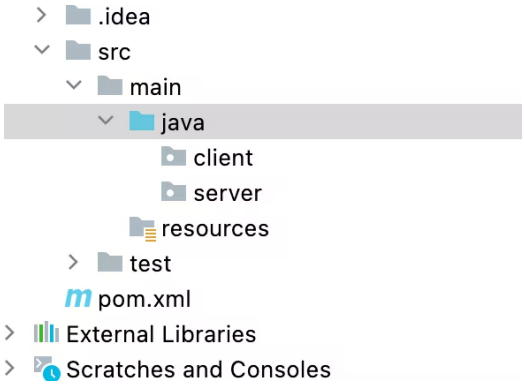
Start a GUI project via IDEA

- New a **Maven** project

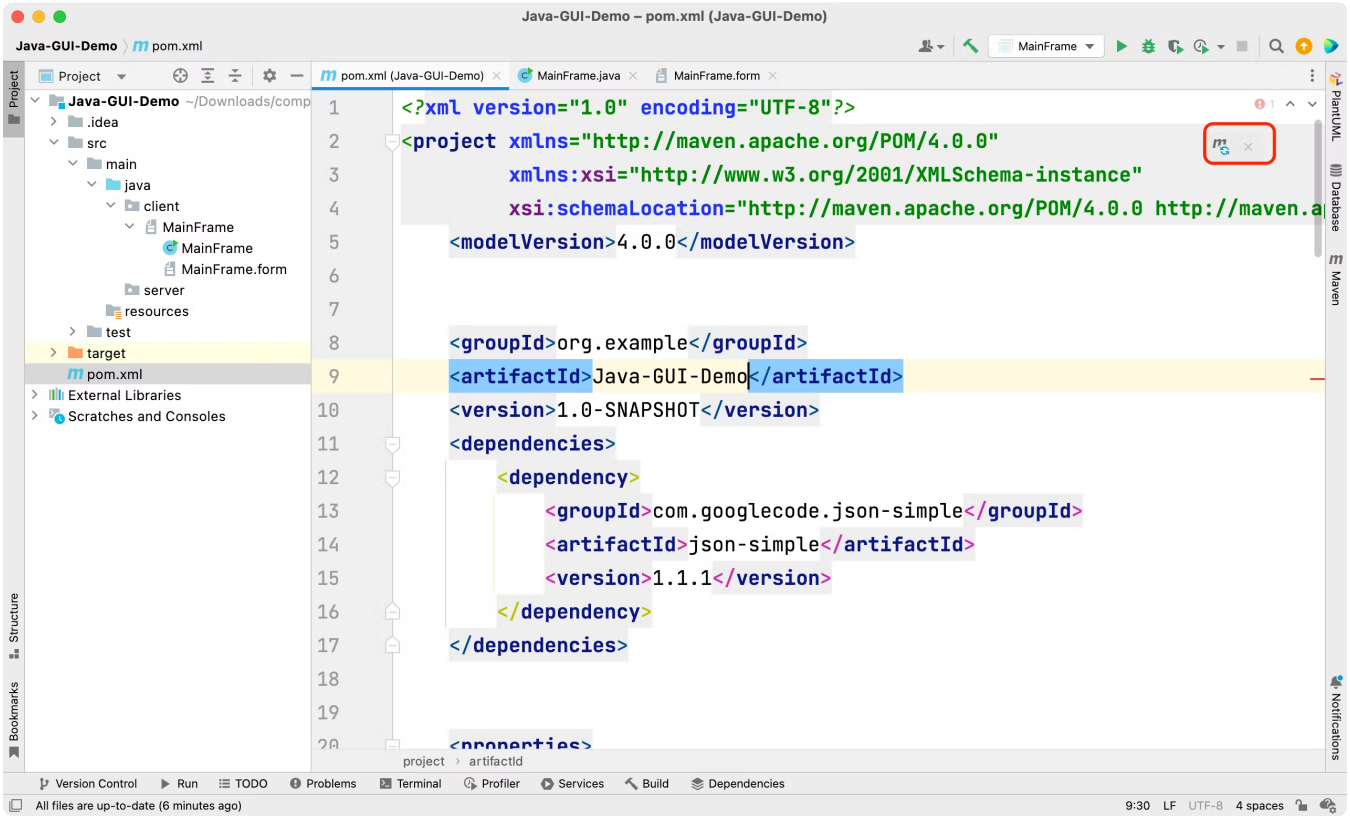


- Right click **java** folder. New a **Client** and **Server** package, delete **org.example** package





- Update **pom.xml** file, click apply change

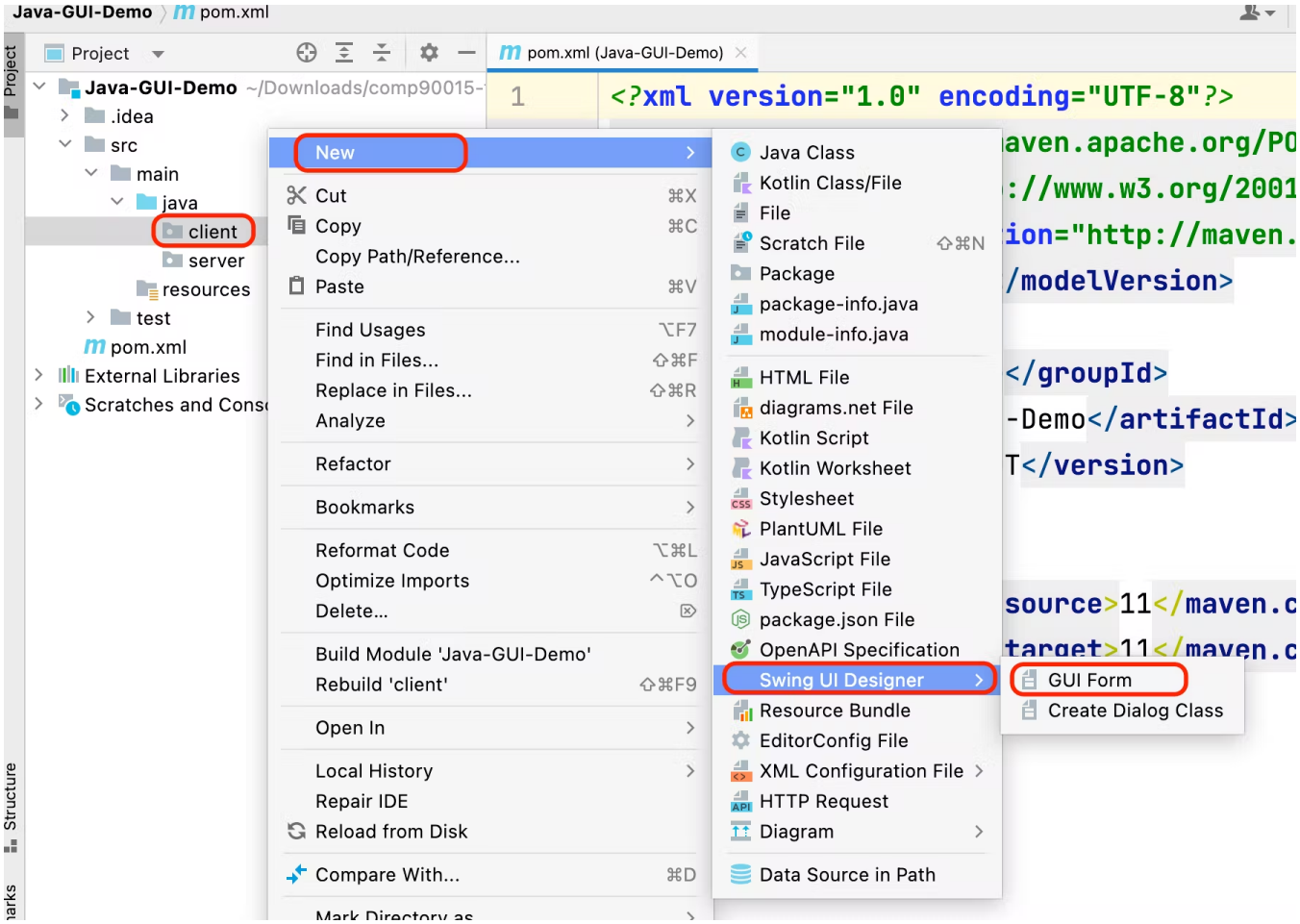


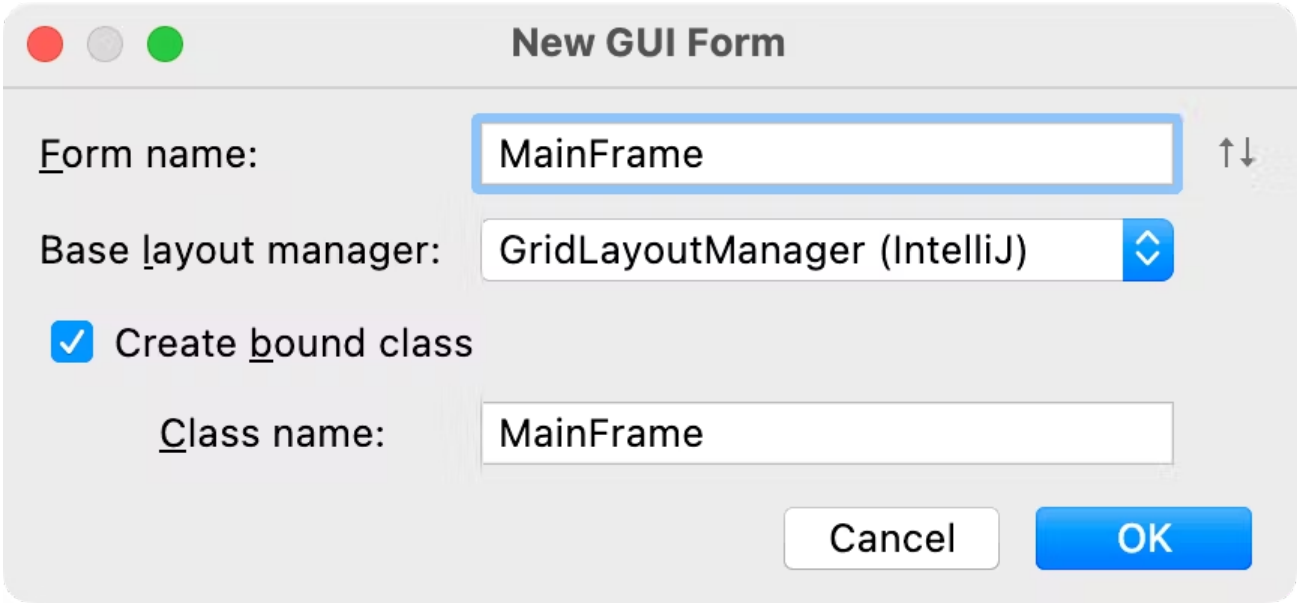

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>


    <groupId>org.example</groupId>
    <artifactId>Java-GUI-Demo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>com.googlecode.json-simple</groupId>
            <artifactId>json-simple</artifactId>
            <version>1.1.1</version>
        </dependency>
        <dependency>
            <groupId>com.intellij</groupId>
            <artifactId>forms_rt</artifactId>
            <version>7.0.3</version>
        </dependency><dependency>
        <groupId>com.intellij</groupId>
        <artifactId>forms_rt</artifactId>
        <version>7.0.3</version>
    </dependency>
```

```
    </dependencies>


    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <executions>
                    <execution>
                        <id>build-a</id>
                        <configuration>
                            <archive>
                                <manifest>
                                    <mainClass>client.MainFrame</mainClass>
                                </manifest>
                            </archive>
                            <descriptorRefs>
                                <descriptorRef>jar-with-dependencies</descriptorRef>
                            </descriptorRefs>
                            <!-- <appendAssemblyId>false</appendAssemblyId> -->
                            <finalName>client</finalName>
                        </configuration>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>build-b</id>
                        <configuration>
                            <archive>
                                <manifest>
                                    <mainClass>server.Server</mainClass>
                                </manifest>
                            </archive>
                            <descriptorRefs>
```

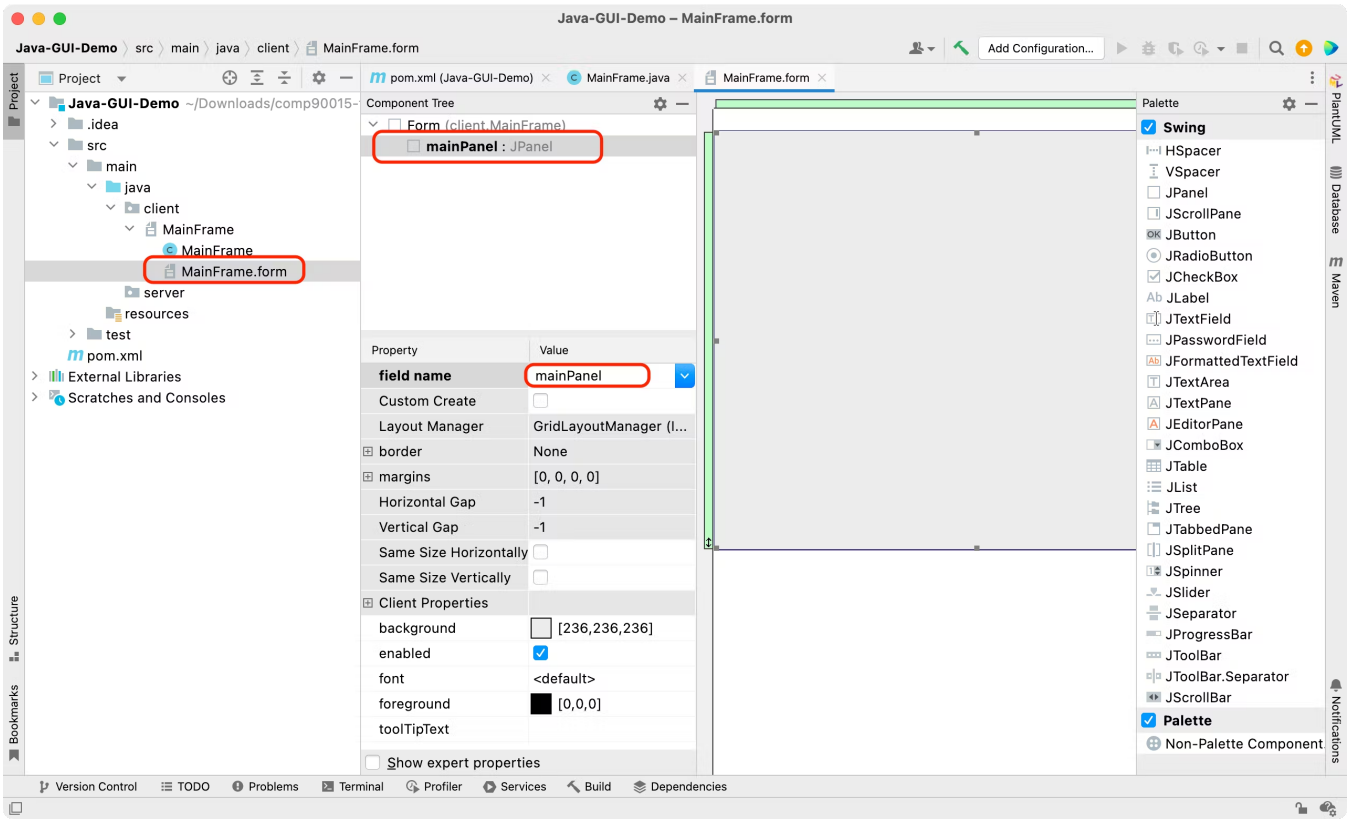
```
                            <descriptorRef>jar-with-dependencies</descriptorRef>
                        </descriptorRefs>
                        <!-- <appendAssemblyId>false</appendAssemblyId> -->
                        <finalName>server</finalName>
                    </configuration>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>

</project>
```

- Right click **client** package, new a **GUI form,** name **MainFrame**

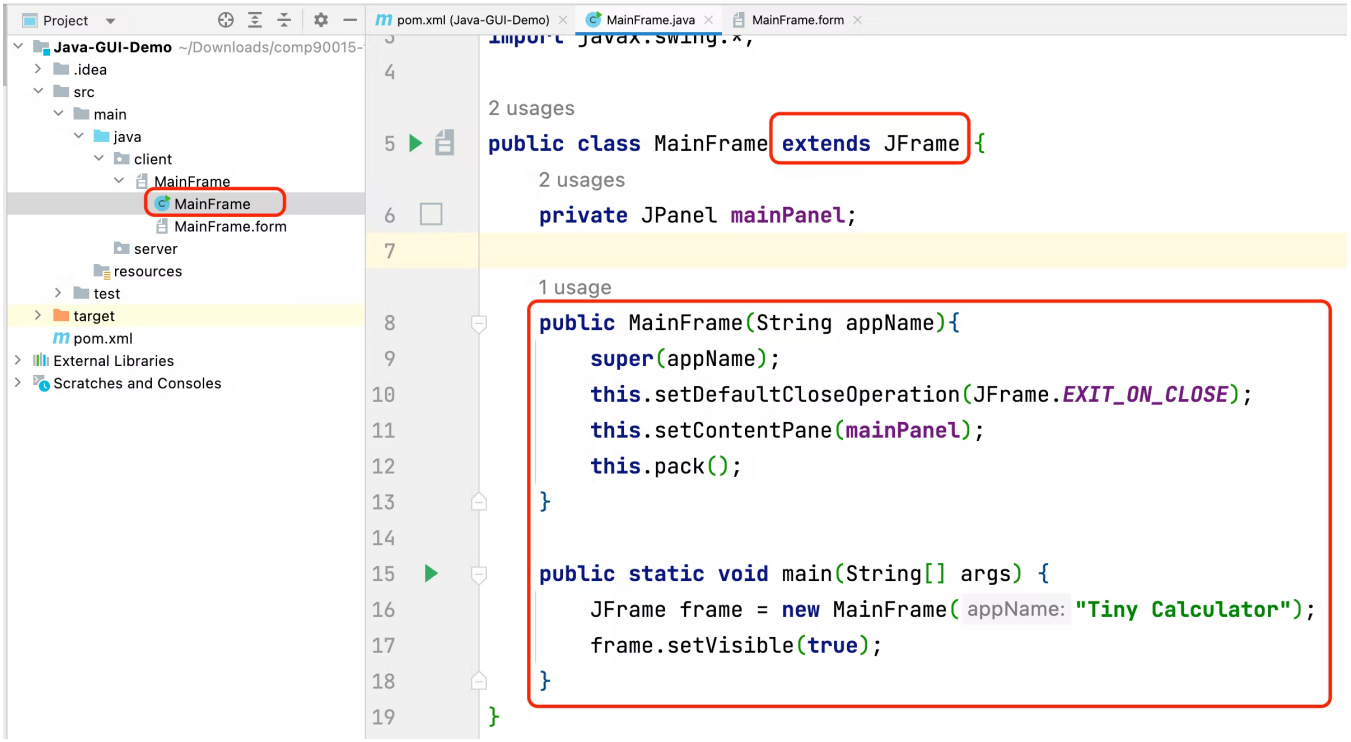




- Rename default JPanel

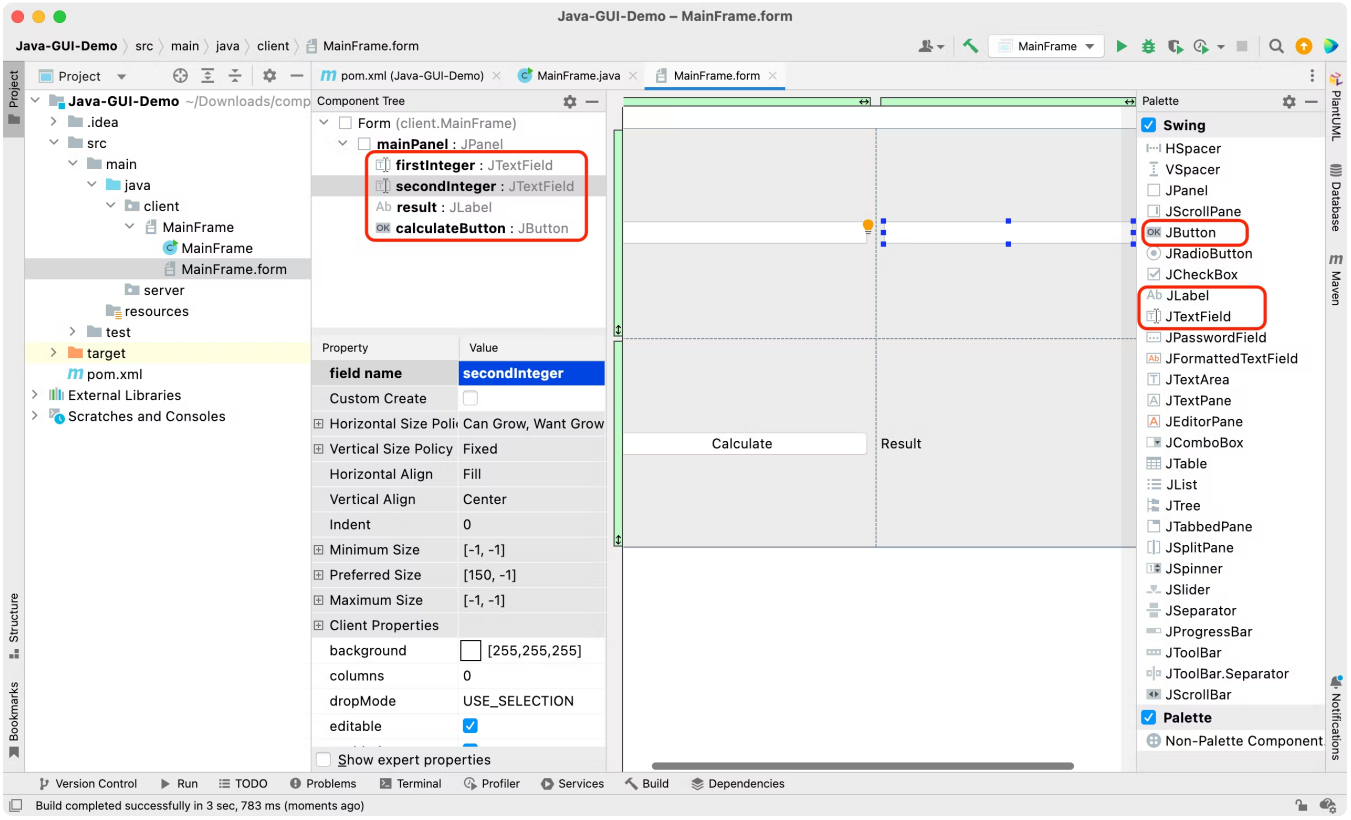


- Update **MainFrame.java** file

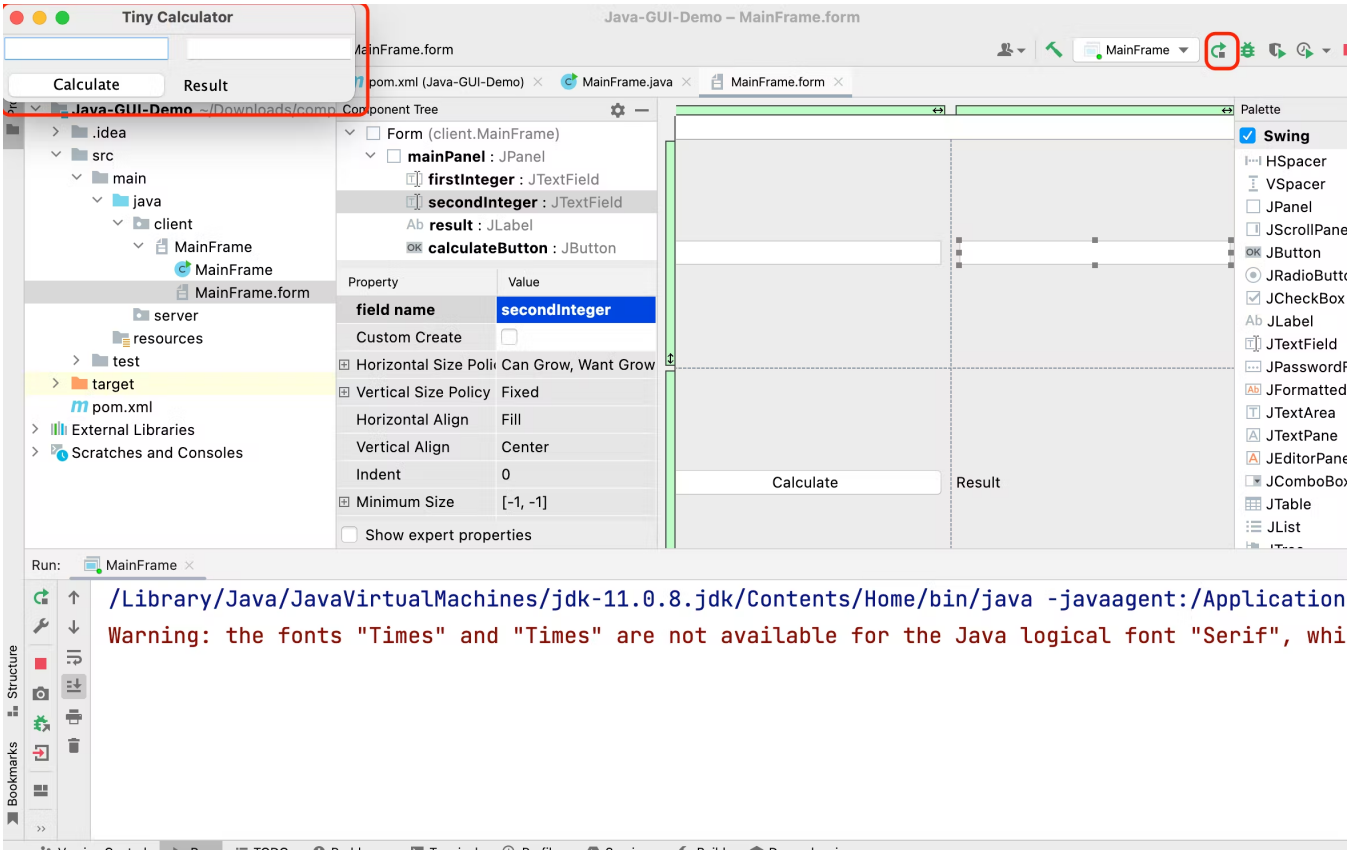



```
public MainFrame(String appName){
        super(appName);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setContentPane(mainPanel);
        this.pack();
    }

public static void main(String[] args) {
        JFrame frame = new MainFrame("Tiny Calculator");
        frame.setVisible(true);
}
```

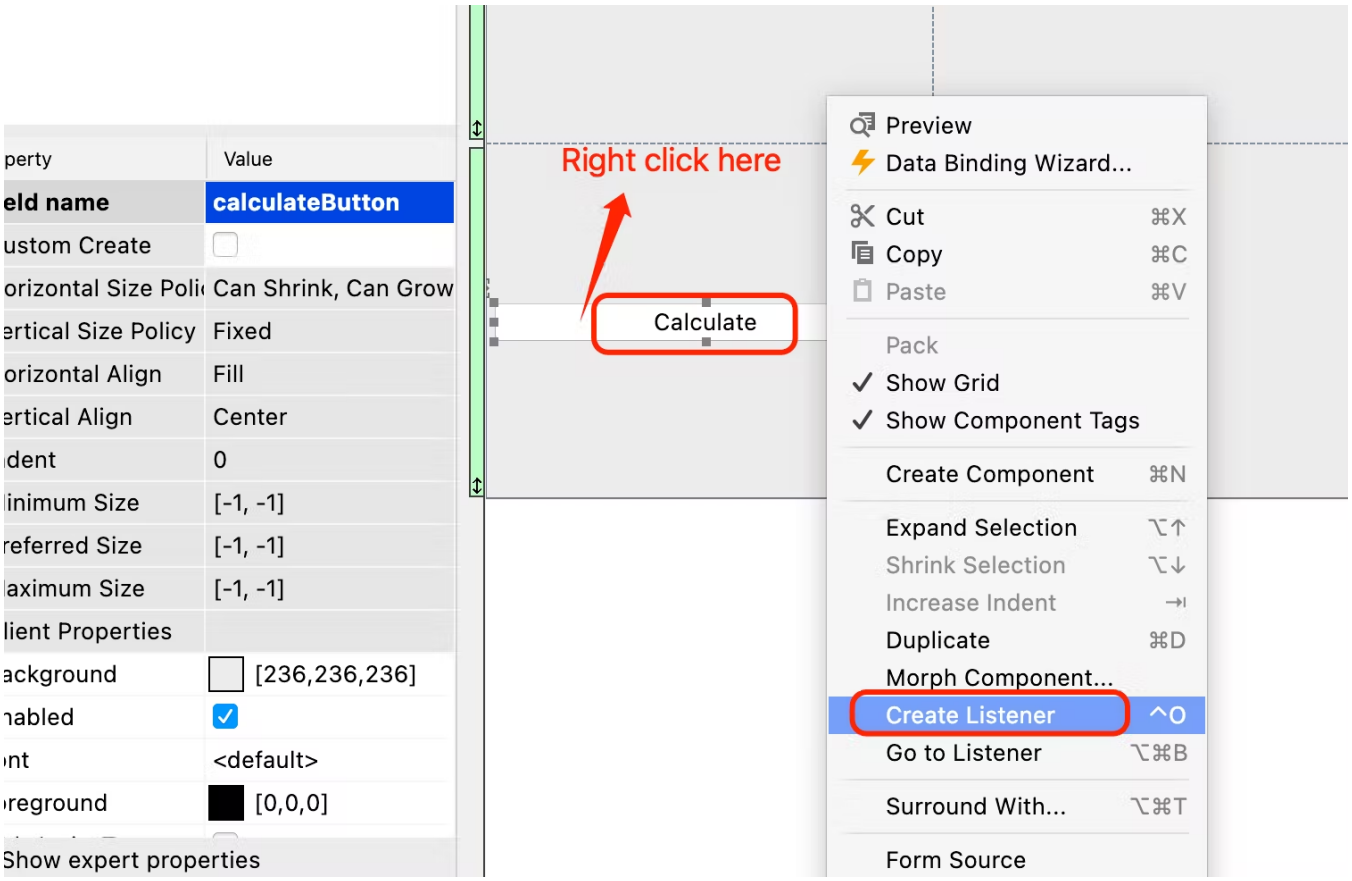
- Go to form file to update GUI by drag & drop components. Name the **fieldName** as the screenshot shows.



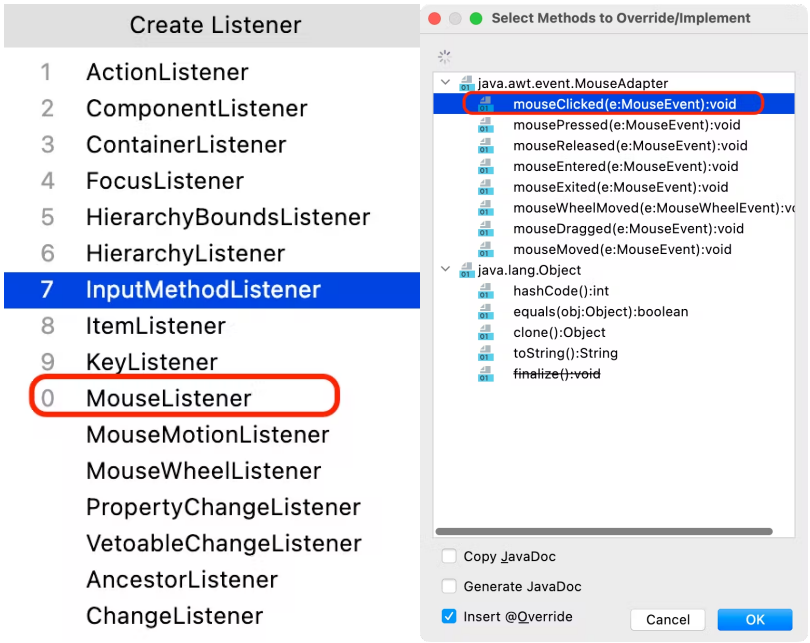
- Run the program, should see the pop up window.



- Right click **Calculate** button, create a listener.



- Select MouseListener → mouseClicked



- Update listener

```
25        this.setContentPane(mainPanel);
```

```
26          this.pack();
27          calculateButton.addMouseListener(new MouseAdapter() {
28              @Override
29              public void mouseClicked(MouseEvent e) {
30                  System.out.println("pressed calculate button");
31                  try {
32                      Socket socket = new Socket( host: "localhost",  port: 4321)
33                      DataInputStream input = new DataInputStream(socket.getI
34                      DataOutputStream output = new DataOutputStream(socket.g
35
36                      output.writeUTF( str: "I want to connect!");
37                      output.flush();
38
39                      JSONObject newCommand = new JSONObject();
40
41                      newCommand.put("command_name", "Math");
42                      newCommand.put("method_name","add");
43                      newCommand.put("first_integer",Integer.parseInt(firstIn
44                      newCommand.put("second_integer",Integer.parseInt(second
45
46                      System.out.println(newCommand.toJSONString());
47
48                      // Read hello from server.
49                      String message = input.readUTF();
50                      System.out.println(message);
51
52                      // Send message to Server
53                      output.writeUTF(newCommand.toJSONString());
54                      output.flush();
55
56                      // Parse result
57                      JSONParser parser = new JSONParser();
58                      JSONObject res =(JSONObject) parser.parse(input.readUTF
59                      System.out.println("Received from server: "+res.get("re
60                      result.setText("Result: "+res.get("result"));
61                  } catch (IOException ex) {
62                      throw new RuntimeException(ex);
63                  } catch (ParseException ex) {
64                      throw new RuntimeException(ex);
65                  }
66              }
```

```
67          });
68      }
69
70      public static void main(String[] args) {
71          JFrame frame = new MainFrame( appName: "Tiny Calculator");
```

```
System.out.println("pressed calculate button");
                try {
                    Socket socket = new Socket("localhost", 4321);
                    DataInputStream input = new DataInputStream(socket.getInputStream());
                    DataOutputStream output = new DataOutputStream(socket.getOutputStream());

                    output.writeUTF("I want to connect!");
                    output.flush();

                    JSONObject newCommand = new JSONObject();

                    newCommand.put("command_name", "Math");
                    newCommand.put("method_name","add");
                    newCommand.put("first_integer",Integer.parseInt(firstInteger.getText()));
                    newCommand.put("second_integer",Integer.parseInt(secondInteger.getText()));

                    System.out.println(newCommand.toJSONString());

                    // Read hello from server.
                    String message = input.readUTF();
                    System.out.println(message);

                    // Send message to Server
                    output.writeUTF(newCommand.toJSONString());
                    output.flush();

                    // Parse result
                    JSONParser parser = new JSONParser();
                    JSONObject res =(JSONObject) parser.parse(input.readUTF());
                    System.out.println("Received from server: "+res.get("result"));
                    result.setText("Result: "+res.get("result"));
                } catch (IOException ex) {
                    throw new RuntimeException(ex);
                } catch (ParseException ex) {
                    throw new RuntimeException(ex);
                }
```

- Create **Math** and **Server** classes under **server** package

Math class code

```
package server;

public class Math {
    public int add(int x, int y){
        return (x + y);
    }

    public int subtract(int x, int y){
        return (x - y);
    }

    public int multiply(int x, int y){
        return (x * y);
    }
}
```

Server class code

```
package server;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
```

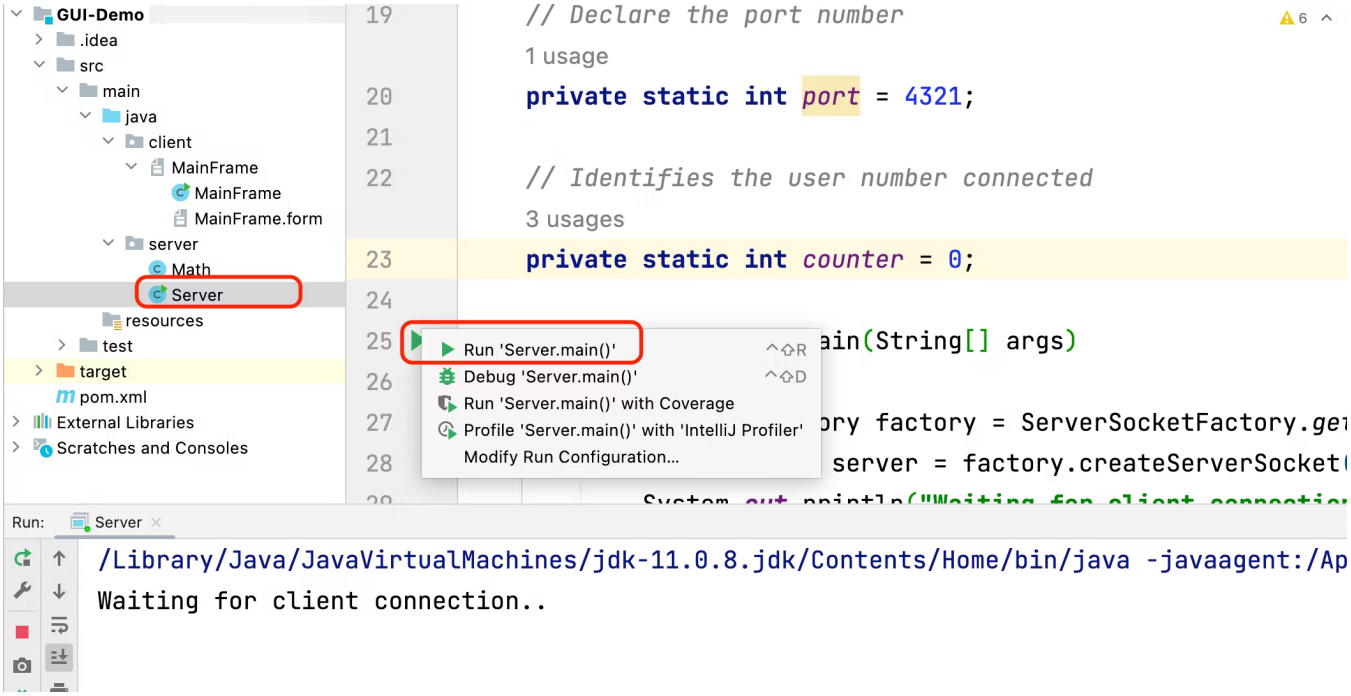
```
import javax.net.ServerSocketFactory;

import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class Server {

    // Declare the port number
    private static int port = 4321;

    // Identifies the user number connected
    private static int counter = 0;

    public static void main(String[] args)
    {
        ServerSocketFactory factory = ServerSocketFactory.getDefault();
        try(ServerSocket server = factory.createServerSocket(port)){
            System.out.println("Waiting for client connection..");

            // Wait for connections.
            while(true){
                Socket client = server.accept();
                counter++;
                System.out.println("Client "+counter+": Applying for connection!");

                // Start a new thread for a connection
                Thread t = new Thread(() -> serveClient(client));
                t.start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void serveClient(Socket client)
    {
        try(Socket clientSocket = client)
        {

            // The JSON Parser
            JSONParser parser = new JSONParser();
            // Input stream
            DataInputStream input = new DataInputStream(clientSocket.getInputStream());
            // Output Stream
            DataOutputStream output = new DataOutputStream(clientSocket.getOutputStream());

            System.out.println("CLIENT: "+input.readUTF());
```

```
            output.writeUTF("Server: Hi Client "+counter+" !!!");

            // Receive more data..
            while(true){
                if(input.available() > 0){
                    // Attempt to convert read data to JSON
                    JSONObject command = (JSONObject) parser.parse(input.readUTF());
                    System.out.println("COMMAND RECEIVED: "+command.toJSONString());
                    Integer result = parseCommand(command);
                    JSONObject resObj = new JSONObject();
                    resObj.put("result", result);

                    output.writeUTF(resObj.toJSONString());
                }
            }
        } catch (IOException | ParseException e) {
            e.printStackTrace();
        }
    }
    private static Integer parseCommand(JSONObject command) {

        int result = 0;

        if(command.containsKey("command_name")){
            System.out.println("IT HAS A COMMAND NAME");
        }

        if (command.get("command_name").equals("Math"))
        {
            Math math = new Math();
            Integer firstInt = Integer.parseInt(command.get("first_integer").toString());
            Integer secondInt = Integer.parseInt(command.get("second_integer").toString());

            switch((String) command.get("method_name"))
            {
                case "add":
                    result = math.add(firstInt,secondInt);
                    break;
                case "multiply":
                    result = math.multiply(firstInt,secondInt);
                    break;
                case "subtract":
                    result = math.subtract(firstInt,secondInt);
                    break;
                default:
                    try
                    {
                        throw new Exception();
                    } catch (Exception e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
            // TODO Auto-generated method stub
            return result;
        }

    }
```

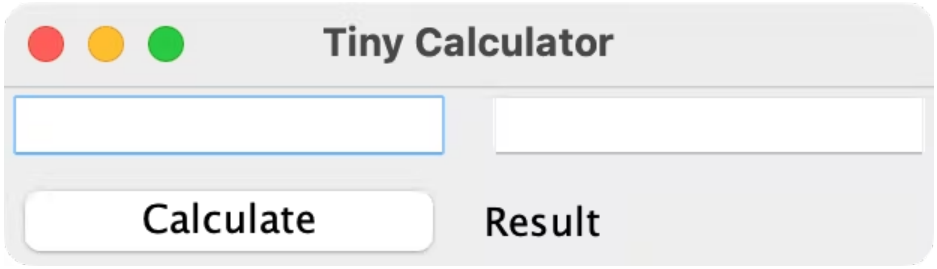
Run program via IDEA

1. Open **Server** file and click **Run** button.

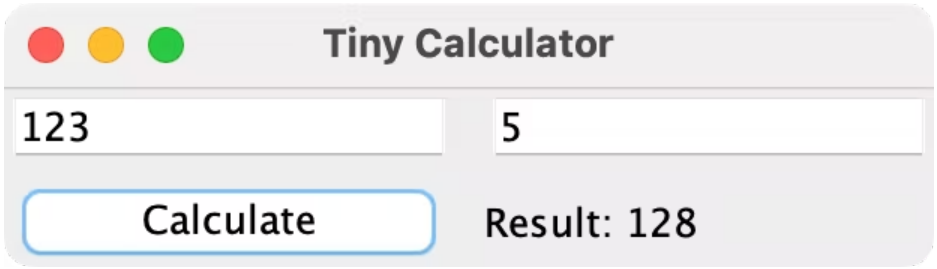


2. Open **MainFrame** file and click **Run** button.

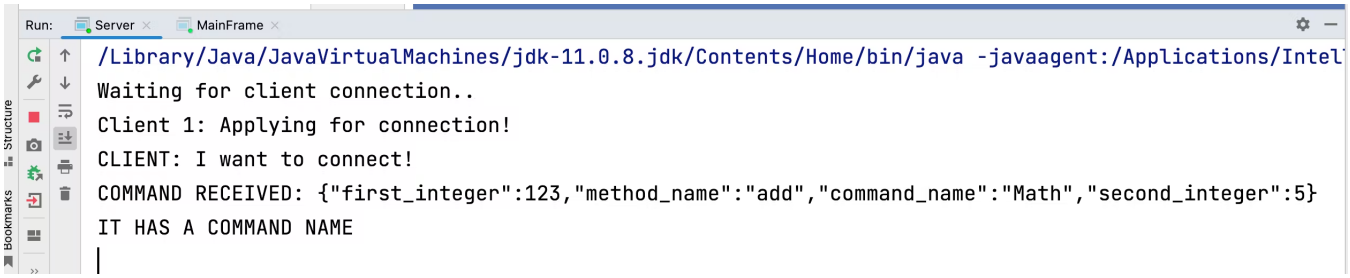
3. Wait a few second, a window will display.



4. Type any number to calculate the sum of two numbers.

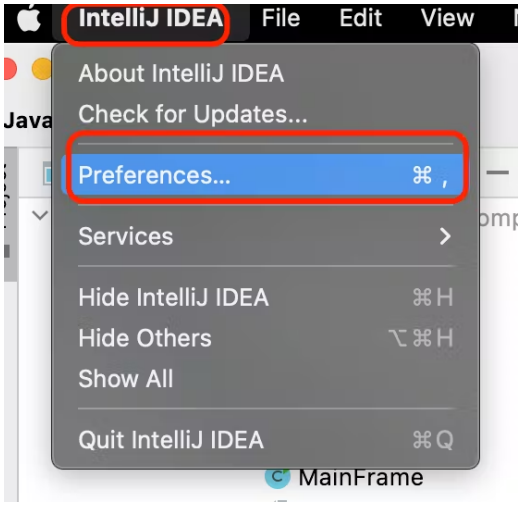


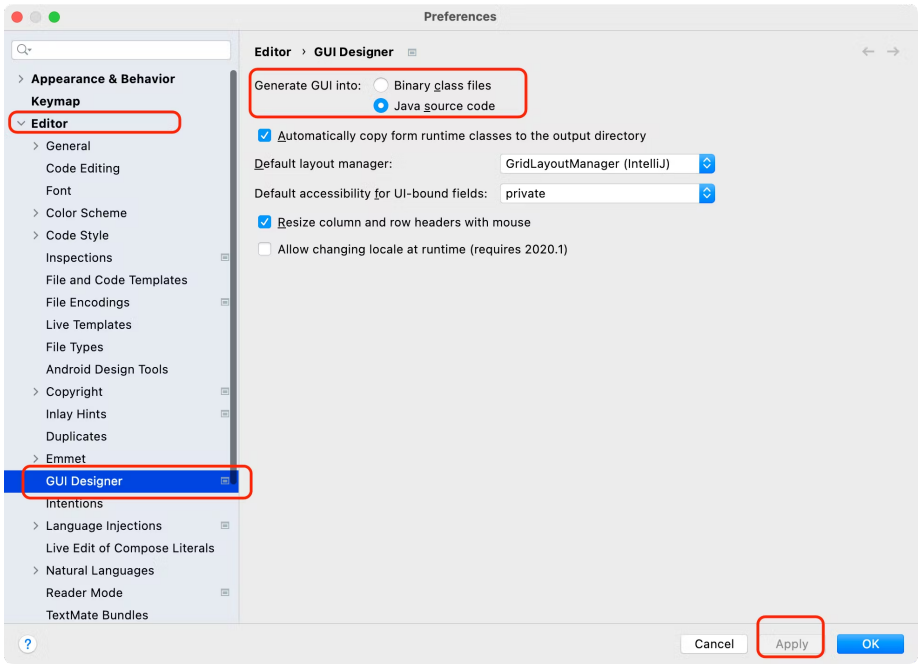
5. Server side console



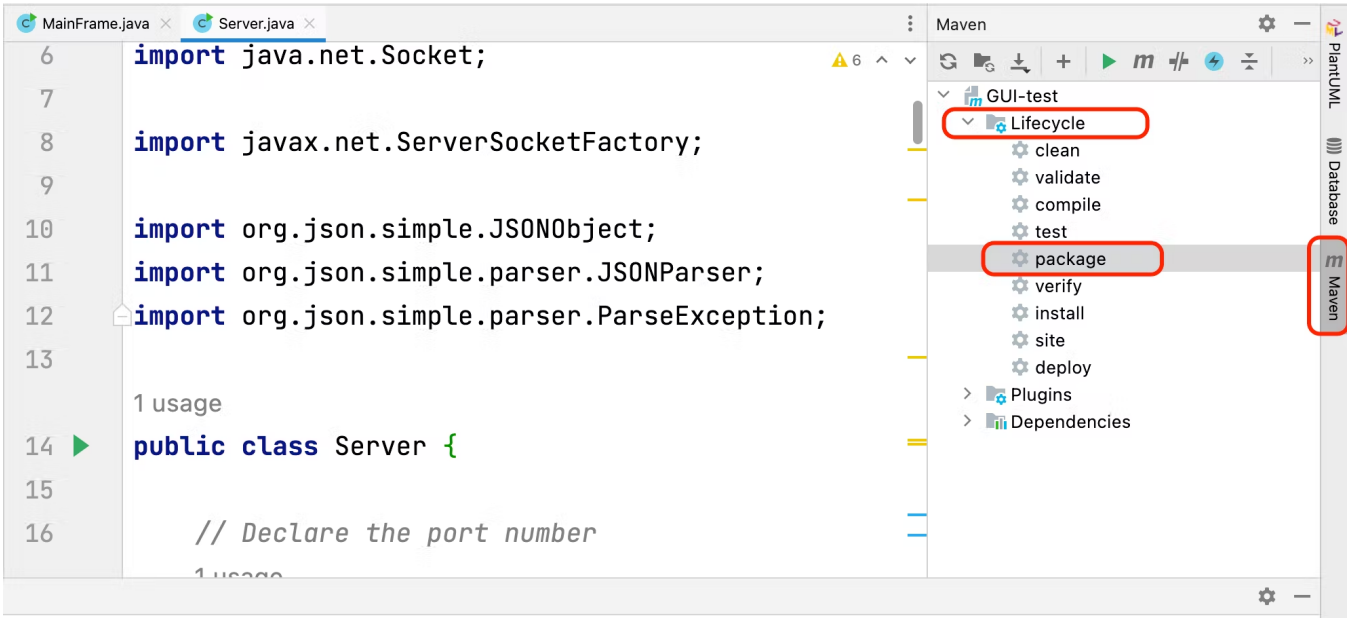
Run program via jar files

0. Configure GUI designer.

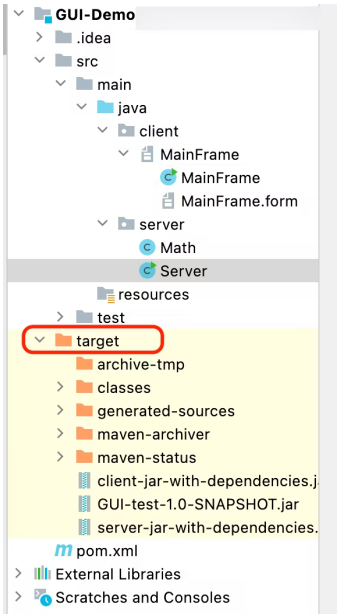




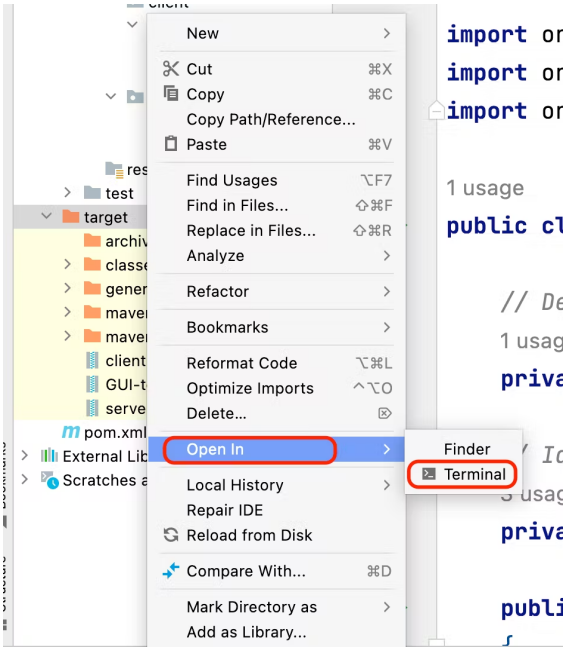
1. Open **Maven** panel on the right. Click **Lifecycle.** Click **package**



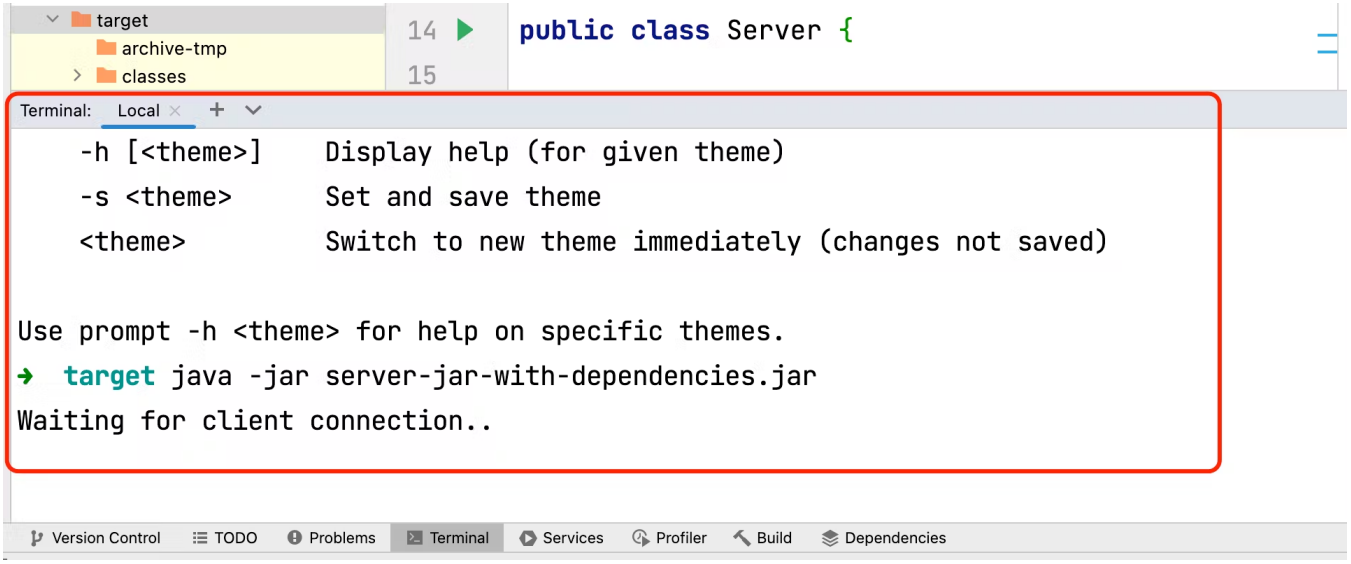
2. A new folder **target** will be generated



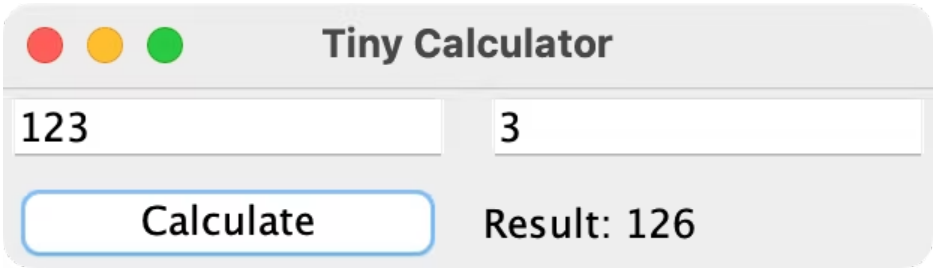
3. **Right click** target folder



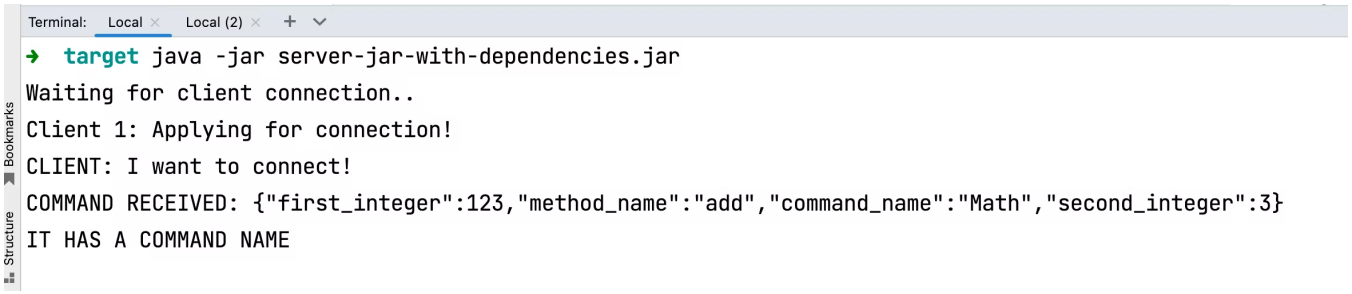
4. A terminal will display, run **java -jar server-jar-with-dependencies.jar**

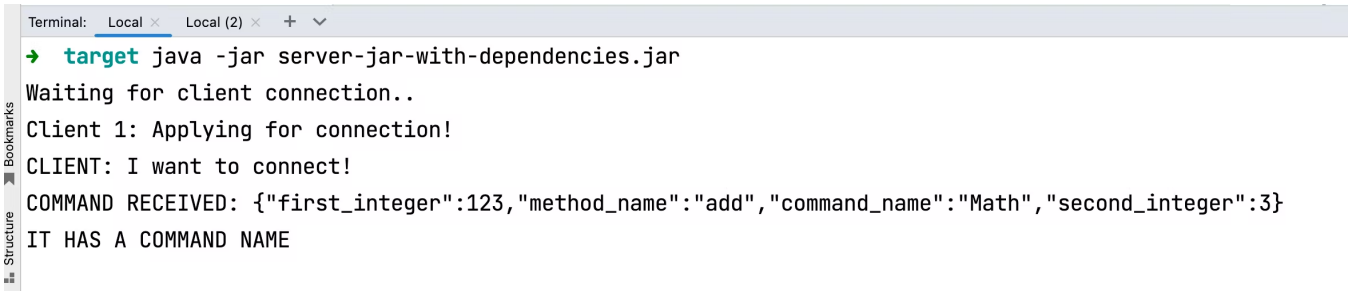


5. **Right click** target folder and open a new terminal as above step. Run **java -jar client-jar-with-dependencies.jar.** A window will display.



8. Server side console


Terminal: Local Local (2)
target java -jar server-jar-with-dependencies.jar
Waiting for client connection..
Client 1: Applying for connection!
CLIENT: I want to connect!
COMMAND RECEIVED: {"first_integer":123,"method_name":"add","command_name":"Math","second_integer":3}
IT HAS A COMMAND NAME
Bookmarks
Structure


Terminal: Local Local (2)
target java -jar server-jar-with-dependencies.jar
Waiting for client connection..
Client 1: Applying for connection!
CLIENT: I want to connect!
COMMAND RECEIVED: {"first_integer":123,"method_name":"add","command_name":"Math","second_integer":3}
IT HAS A COMMAND NAME
Bookmarks
Structure