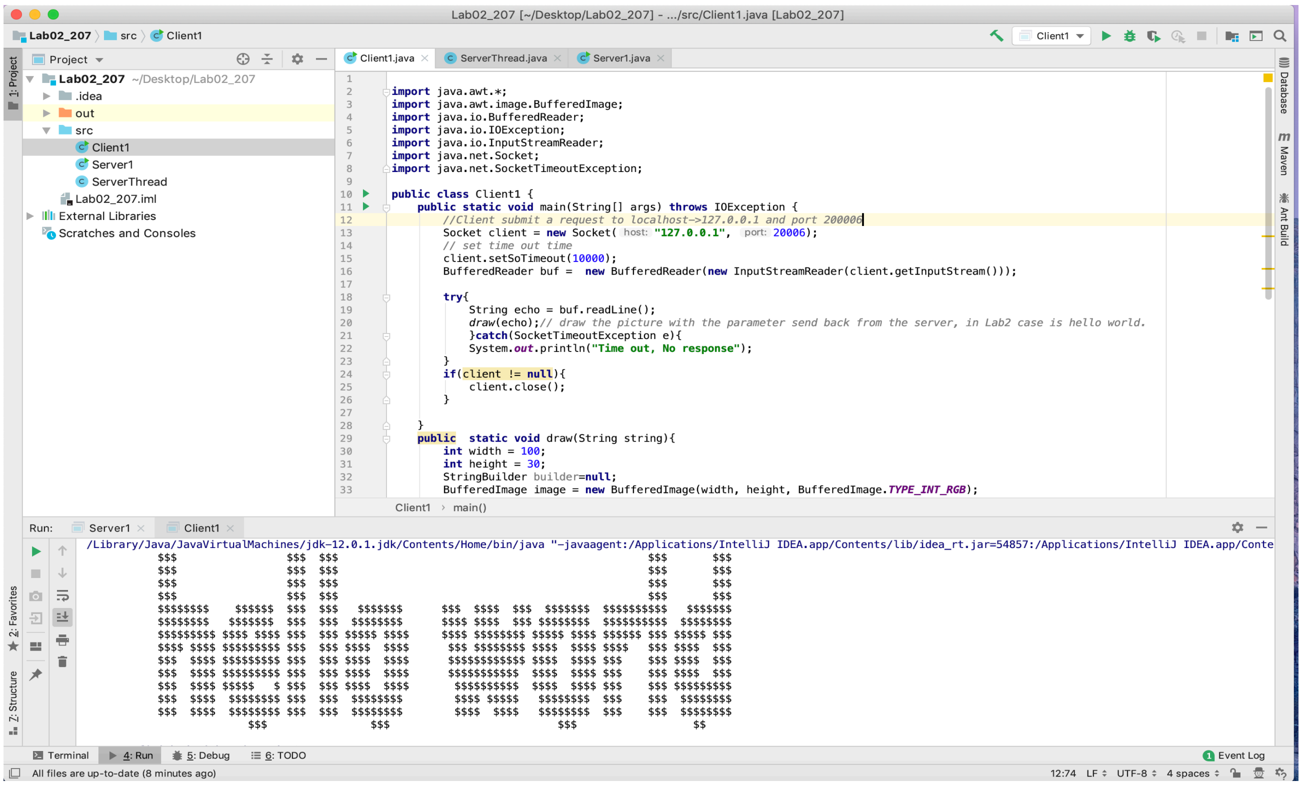
HW # 2

Simple TCP Server

Github Link : https://github.com/Tejas945/cmpe207/tree/master/Assignments/HW_2

Team Members:

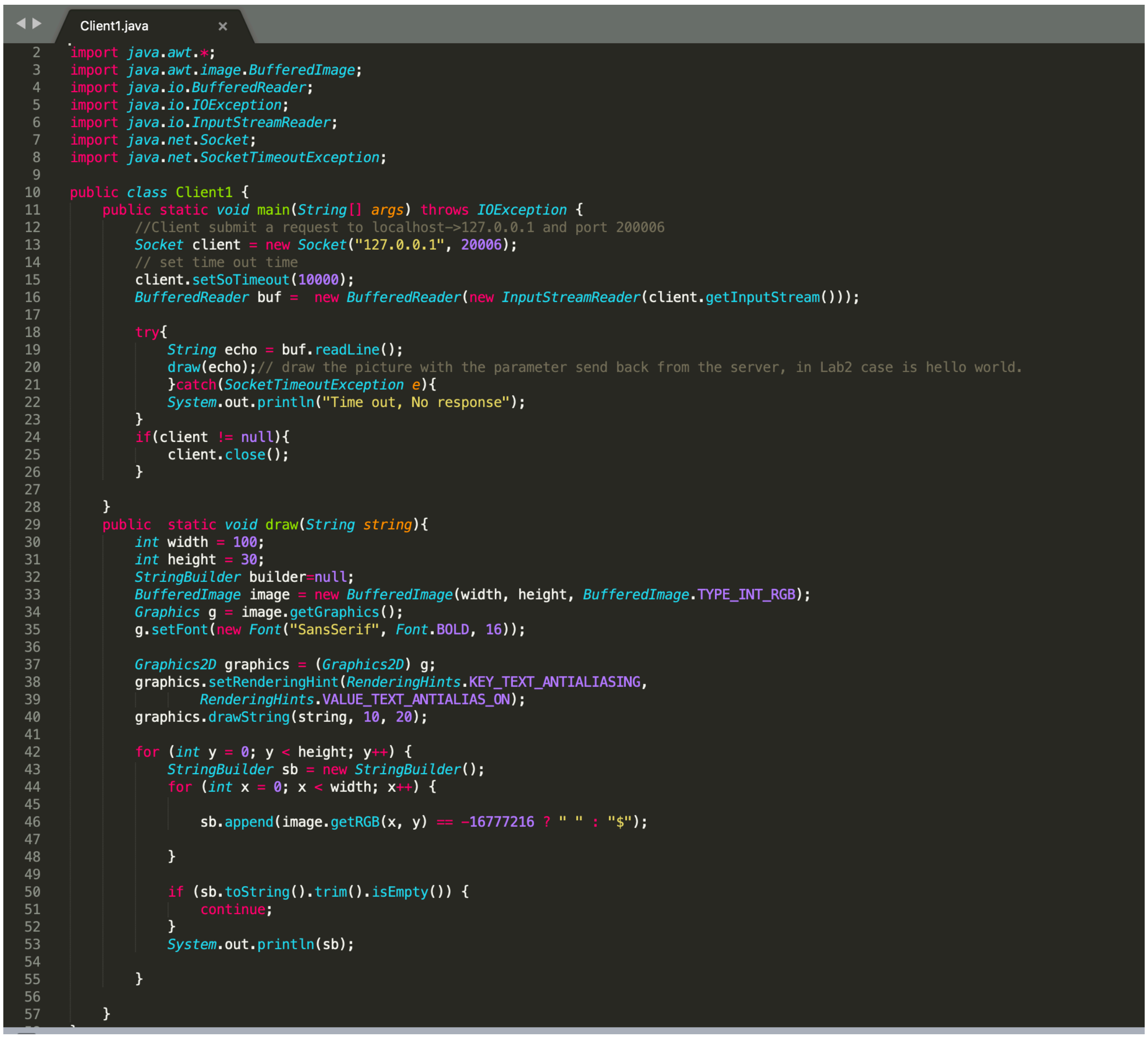
1) Tejas Madappa
2) Chandra Mohan
3) Vedant Bhoj
4) Ketan Rudrurkar
5) Hao Ran Chen



```java

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.SocketTimeoutException;

public class Client1 {
    public static void main(String[] args) throws IOException {
        //Client submit a request to localhost->127.0.0.1 and port 200006
        Socket client = new Socket( host: "127.0.0.1",  port: 20006);
        // set time out time
        client.setSoTimeout(10000);
        BufferedReader buf =  new BufferedReader(new InputStreamReader(client.getInputStream()));

        try{
            String echo = buf.readLine();
            draw(echo);// draw the picture with the parameter send back from the server, in Lab2 case is hello world.
            }catch(SocketTimeoutException e){
            System.out.println("Time out, No response");
        }
        if(client != null){
            client.close();
        }

    }
    public  static void draw(String string){
        int width = 100;
        int height = 30;
        StringBuilder builder=null;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
```

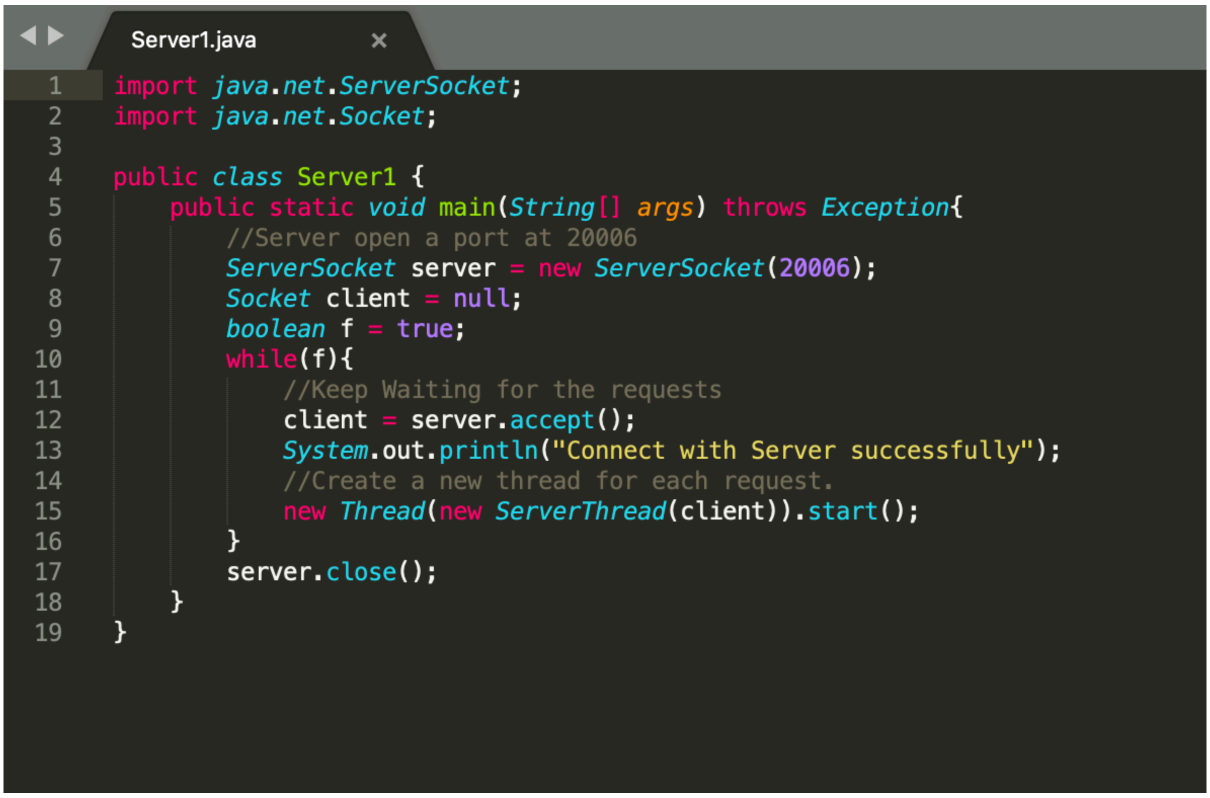
```
/Library/Java/JavaVirtualMachines/jdk-12.0.1.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=54857:/Applications/IntelliJ IDEA.app/Conte
           $$$                 $$$  $$$                                                $$$       $$$
           $$$                 $$$  $$$                                                $$$       $$$
           $$$                 $$$  $$$                                                $$$       $$$
           $$$                 $$$  $$$                                                $$$       $$$
           $$$$$$$$    $$$$$$  $$$  $$$   $$$$$$$      $$$  $$$$  $$$  $$$$$$$  $$$$$$$$$$   $$$$$$$
           $$$$$$$$   $$$$$$$  $$$  $$$  $$$$$$$$      $$$$ $$$$  $$$ $$$$$$$$  $$$$$$$$$$  $$$$$$$$
           $$$$$$$$$ $$$$ $$$$ $$$  $$$ $$$$$ $$$$     $$$$ $$$$$$$$ $$$$$ $$$$ $$$$$$ $$$ $$$$$ $$$
           $$$$ $$$$ $$$$$$$$$ $$$  $$$ $$$$  $$$$      $$$ $$$$$$$$ $$$$  $$$$ $$$$   $$$ $$$$  $$$
           $$$  $$$$ $$$$$$$$$ $$$  $$$ $$$$  $$$$      $$$$$$$$$$$$ $$$$  $$$$ $$$    $$$ $$$$  $$$
           $$$  $$$$ $$$$$$$$$ $$$  $$$ $$$$  $$$$      $$$$$$$$$$$  $$$$  $$$$ $$$    $$$ $$$$  $$$
           $$$  $$$$ $$$$$   $ $$$  $$$ $$$$  $$$$       $$$$$$$$$$  $$$$  $$$$ $$$    $$$ $$$$$$$$$
           $$$  $$$$  $$$$$$$$ $$$  $$$  $$$$$$$$        $$$$ $$$$$   $$$$$$$$  $$$    $$$  $$$$$$$$
           $$$  $$$$  $$$$$$$$ $$$  $$$  $$$$$$$$        $$$$  $$$$   $$$$$$$$  $$$    $$$  $$$$$$$$
                         $$$                $$$                          $$$                  $$
```

Fig1. Code executed on localhost(127.0.0.1) on port(2006).



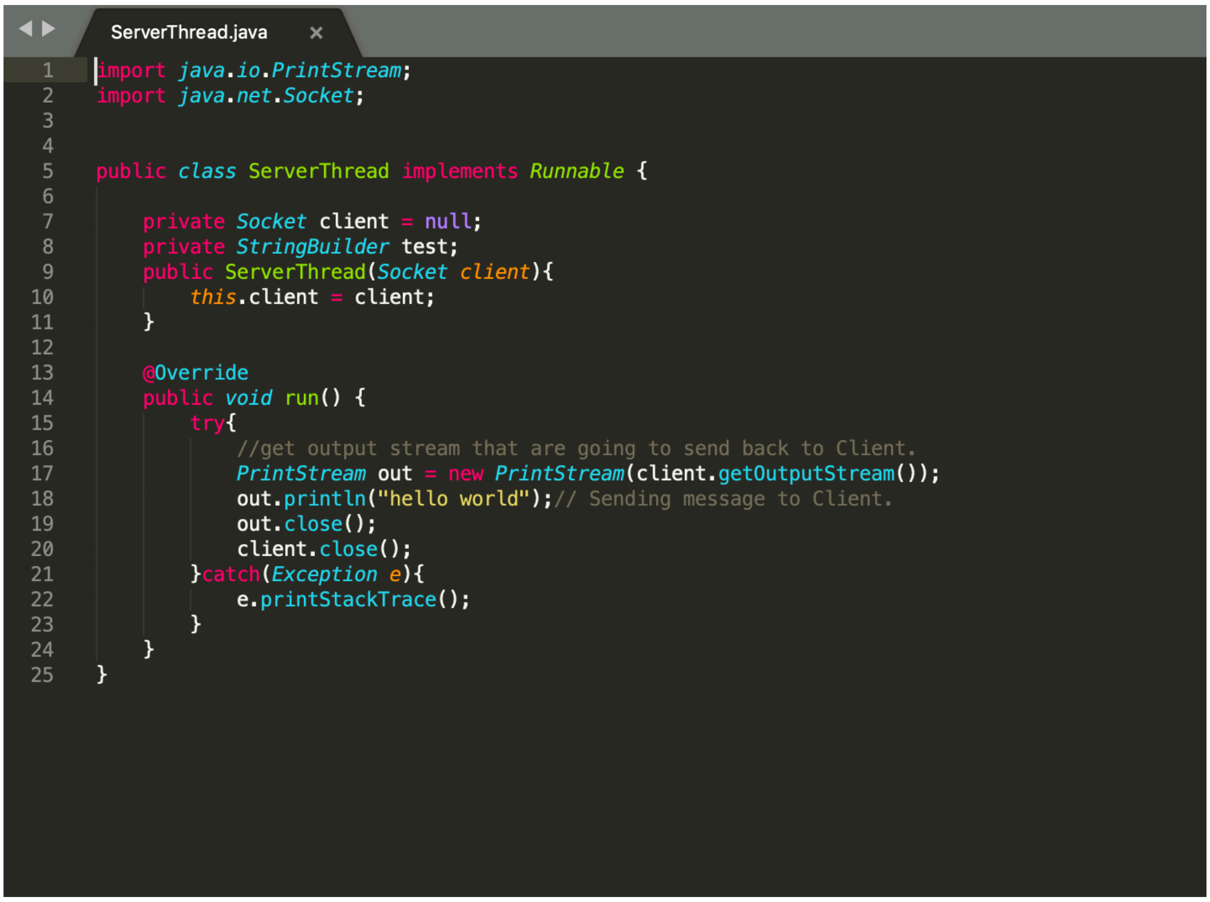
```java
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.SocketTimeoutException;

public class Client1 {
    public static void main(String[] args) throws IOException {
        //Client submit a request to localhost->127.0.0.1 and port 200006
        Socket client = new Socket("127.0.0.1", 20006);
        // set time out time
        client.setSoTimeout(10000);
        BufferedReader buf =  new BufferedReader(new InputStreamReader(client.getInputStream()));

        try{
            String echo = buf.readLine();
            draw(echo);// draw the picture with the parameter send back from the server, in Lab2 case is hello world.
            }catch(SocketTimeoutException e){
            System.out.println("Time out, No response");
        }
        if(client != null){
            client.close();
        }

    }
    public  static void draw(String string){
        int width = 100;
        int height = 30;
        StringBuilder builder=null;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();
        g.setFont(new Font("SansSerif", Font.BOLD, 16));

        Graphics2D graphics = (Graphics2D) g;
        graphics.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
                RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
        graphics.drawString(string, 10, 20);

        for (int y = 0; y < height; y++) {
            StringBuilder sb = new StringBuilder();
            for (int x = 0; x < width; x++) {

                sb.append(image.getRGB(x, y) == -16777216 ? " " : "$");

            }

            if (sb.toString().trim().isEmpty()) {
                continue;
            }
            System.out.println(sb);

        }

    }
```

Fig 2. Client which submits a request to connect to server to parse data sent by the server.



```java
import java.net.ServerSocket;
import java.net.Socket;

public class Server1 {
    public static void main(String[] args) throws Exception{
        //Server open a port at 20006
        ServerSocket server = new ServerSocket(20006);
        Socket client = null;
        boolean f = true;
        while(f){
            //Keep Waiting for the requests
            client = server.accept();
            System.out.println("Connect with Server successfully");
            //Create a new thread for each request.
            new Thread(new ServerThread(client)).start();
        }
        server.close();
    }
}
```

Fig3. Server1.java accepts the connection.



```java
import java.io.PrintStream;
import java.net.Socket;


public class ServerThread implements Runnable {

    private Socket client = null;
    private StringBuilder test;
    public ServerThread(Socket client){
        this.client = client;
    }

    @Override
    public void run() {
        try{
            //get output stream that are going to send back to Client.
            PrintStream out = new PrintStream(client.getOutputStream());
            out.println("hello world");// Sending message to Client.
            out.close();
            client.close();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Fig4. ServerThread sends “hello world” message to connecting client.