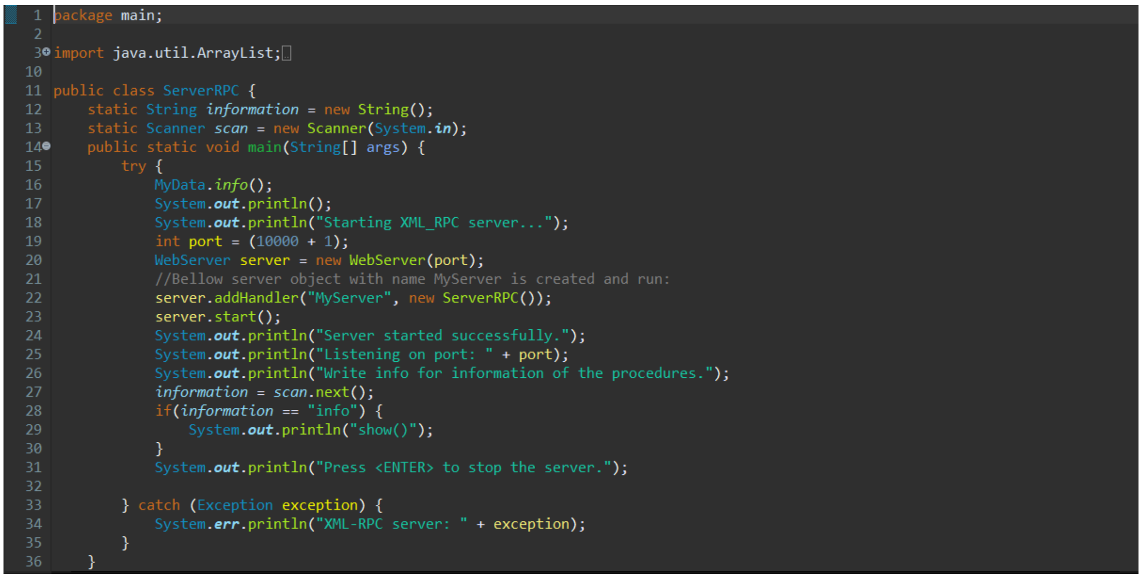
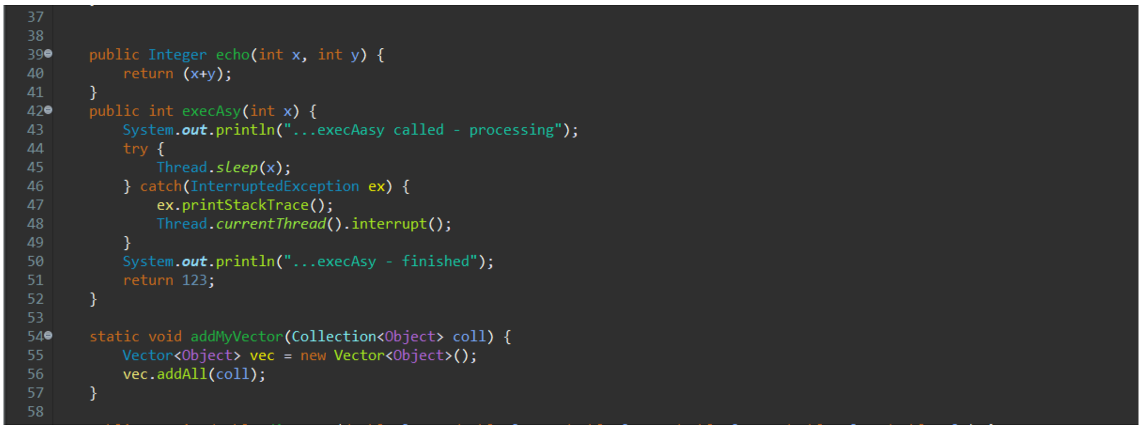
EXERCICE 1

PART B

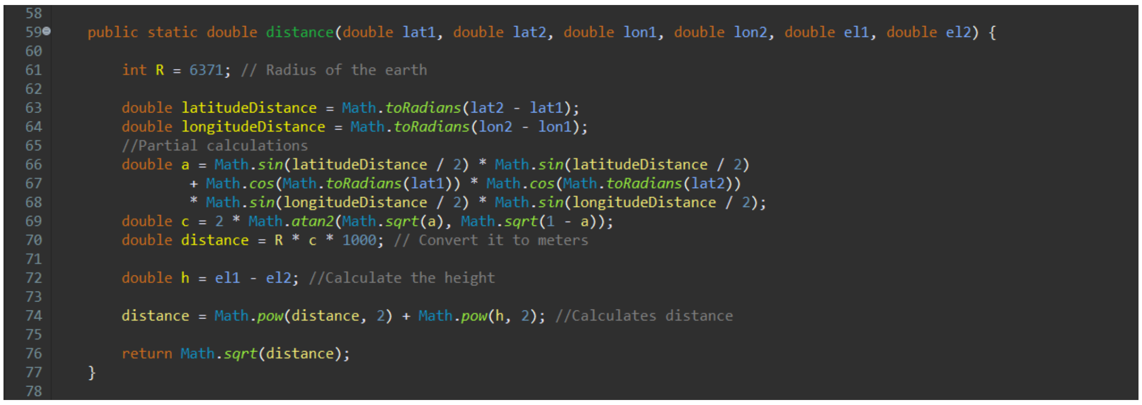
Álvaro López Pereda and Alberto Hernandez Lado



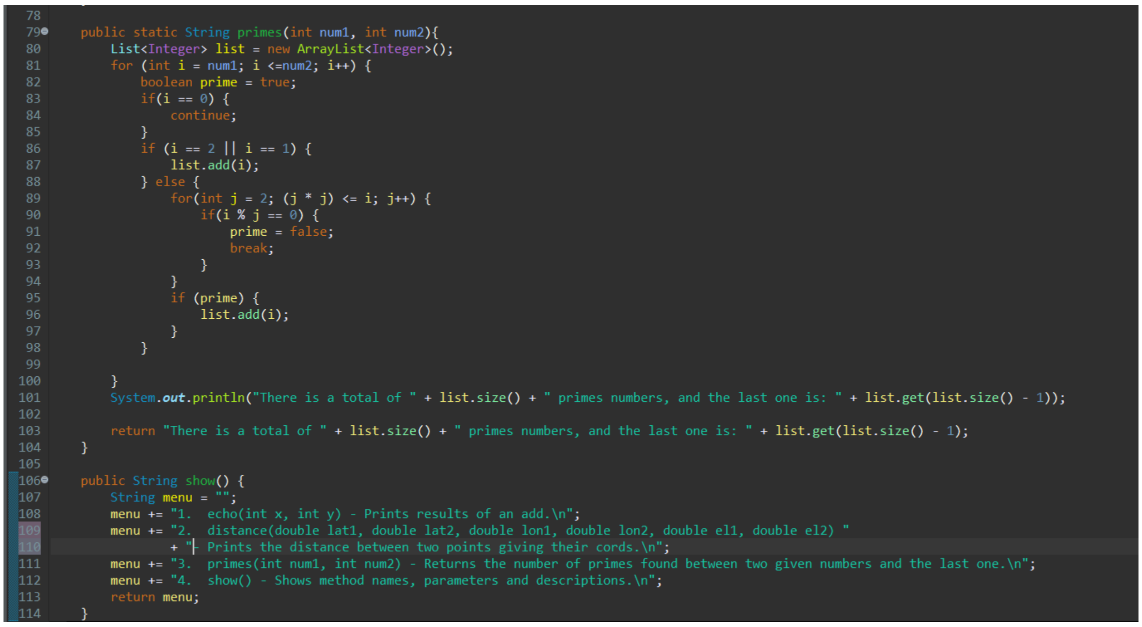
```java
package main;

import java.util.ArrayList;

public class ServerRPC {
    static String information = new String();
    static Scanner scan = new Scanner(System.in);
    public static void main(String[] args) {
        try {
            MyData.info();
            System.out.println();
            System.out.println("Starting XML_RPC server...");
            int port = (10000 + 1);
            WebServer server = new WebServer(port);
            //Bellow server object with name MyServer is created and run:
            server.addHandler("MyServer", new ServerRPC());
            server.start();
            System.out.println("Server started successfully.");
            System.out.println("Listening on port: " + port);
            System.out.println("Write info for information of the procedures.");
            information = scan.next();
            if(information == "info") {
                System.out.println("show()");
            }
            System.out.println("Press <ENTER> to stop the server.");

        } catch (Exception exception) {
            System.err.println("XML-RPC server: " + exception);
        }
    }
```



```java


    public Integer echo(int x, int y) {
        return (x+y);
    }
    public int execAsy(int x) {
        System.out.println("...execAasy called - processing");
        try {
            Thread.sleep(x);
        } catch(InterruptedException ex) {
            ex.printStackTrace();
            Thread.currentThread().interrupt();
        }
        System.out.println("...execAsy - finished");
        return 123;
    }

    static void addMyVector(Collection<Object> coll) {
        Vector<Object> vec = new Vector<Object>();
        vec.addAll(coll);
    }

```

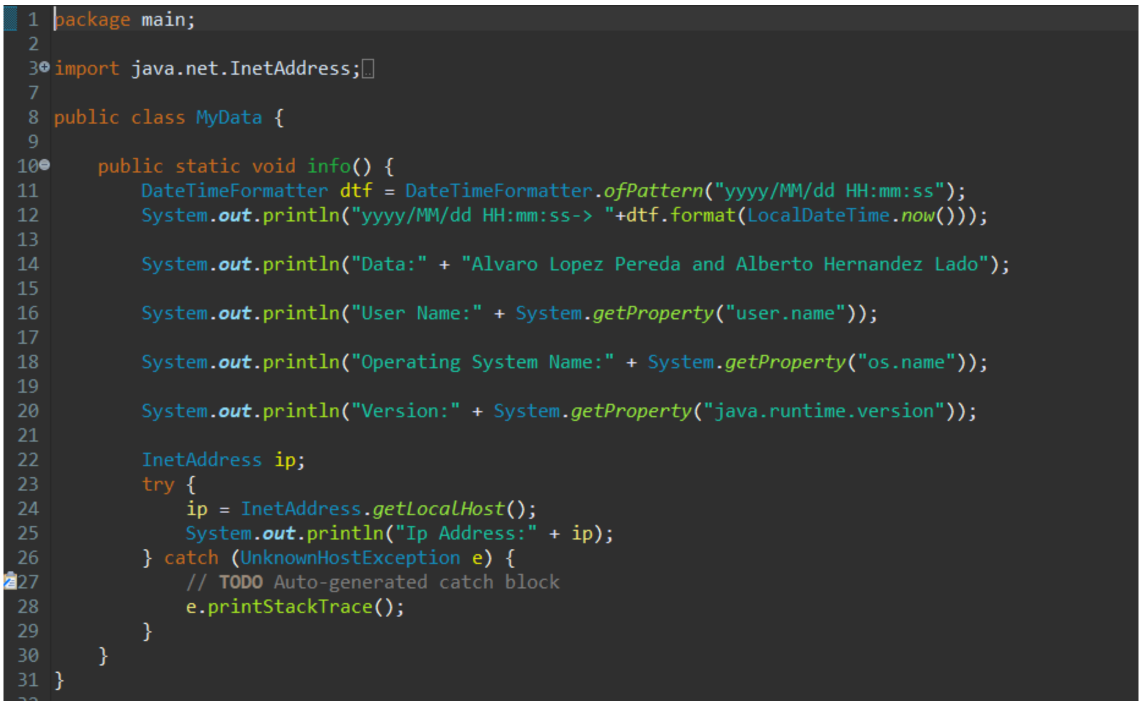


```java

    public static double distance(double lat1, double lat2, double lon1, double lon2, double el1, double el2) {

        int R = 6371; // Radius of the earth

        double latitudeDistance = Math.toRadians(lat2 - lat1);
        double longitudeDistance = Math.toRadians(lon2 - lon1);
        //Partial calculations
        double a = Math.sin(latitudeDistance / 2) * Math.sin(latitudeDistance / 2)
                + Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))
                * Math.sin(longitudeDistance / 2) * Math.sin(longitudeDistance / 2);
        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
        double distance = R * c * 1000; // Convert it to meters

        double h = el1 - el2; //Calculate the height

        distance = Math.pow(distance, 2) + Math.pow(h, 2); //Calculates distance

        return Math.sqrt(distance);
    }

```



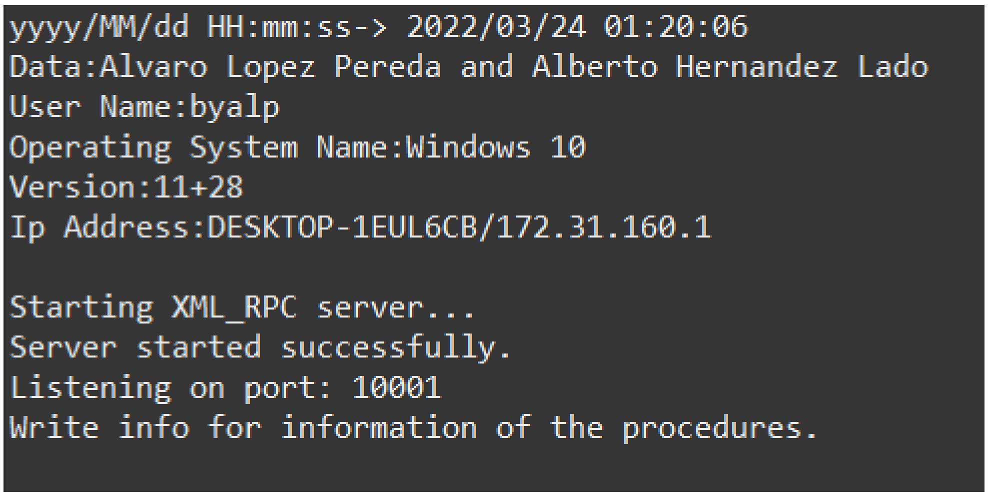
```java

    public static String primes(int num1, int num2){
        List<Integer> list = new ArrayList<Integer>();
        for (int i = num1; i <=num2; i++) {
            boolean prime = true;
            if(i == 0) {
                continue;
            }
            if (i == 2 || i == 1) {
                list.add(i);
            } else {
                for(int j = 2; (j * j) <= i; j++) {
                    if(i % j == 0) {
                        prime = false;
                        break;
                    }
                }
                if (prime) {
                    list.add(i);
                }
            }

        }
        System.out.println("There is a total of " + list.size() + " primes numbers, and the last one is: " + list.get(list.size() - 1));

        return "There is a total of " + list.size() + " primes numbers, and the last one is: " + list.get(list.size() - 1);
    }

    public String show() {
        String menu = "";
        menu += "1.  echo(int x, int y) - Prints results of an add.\n";
        menu += "2.  distance(double lat1, double lat2, double lon1, double lon2, double el1, double el2) "
                + "- Prints the distance between two points giving their cords.\n";
        menu += "3.  primes(int num1, int num2) - Returns the number of primes found between two given numbers and the last one.\n";
        menu += "4.  show() - Shows method names, parameters and descriptions.\n";
        return menu;
    }
```

Server code.



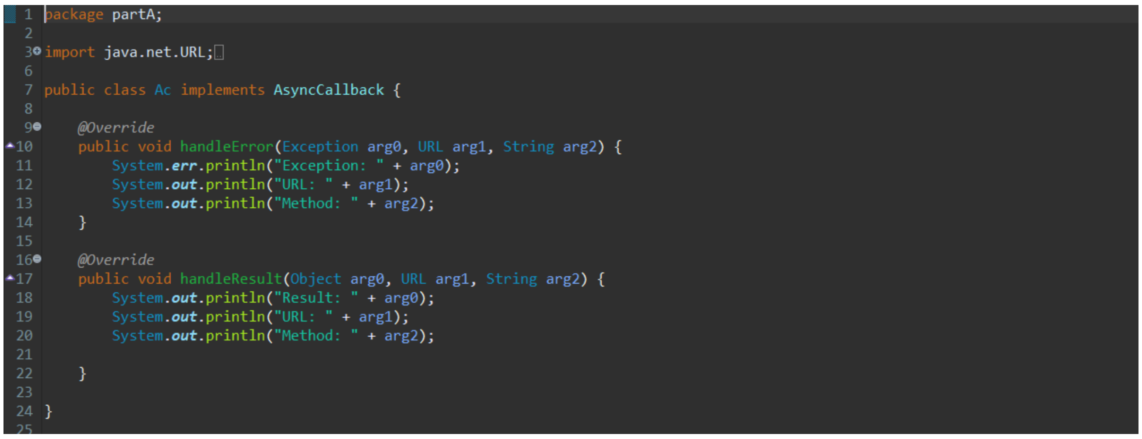
```java
package main;

import java.net.InetAddress;

public class MyData {

    public static void info() {
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
        System.out.println("yyyy/MM/dd HH:mm:ss-> "+dtf.format(LocalDateTime.now()));

        System.out.println("Data:" + "Alvaro Lopez Pereda and Alberto Hernandez Lado");

        System.out.println("User Name:" + System.getProperty("user.name"));

        System.out.println("Operating System Name:" + System.getProperty("os.name"));

        System.out.println("Version:" + System.getProperty("java.runtime.version"));

        InetAddress ip;
        try {
            ip = InetAddress.getLocalHost();
            System.out.println("Ip Address:" + ip);
        } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

MyData code.



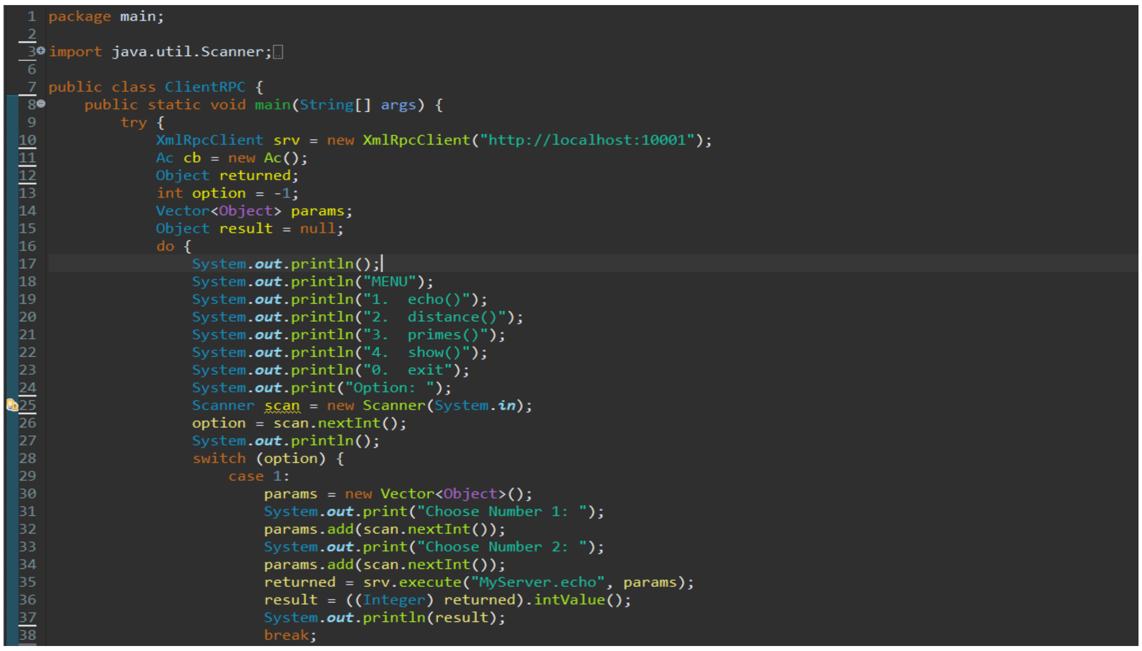
```
yyyy/MM/dd HH:mm:ss-> 2022/03/24 01:20:06
Data:Alvaro Lopez Pereda and Alberto Hernandez Lado
User Name:byalp
Operating System Name:Windows 10
Version:11+28
Ip Address:DESKTOP-1EUL6CB/172.31.160.1

Starting XML_RPC server...
Server started successfully.
Listening on port: 10001
Write info for information of the procedures.
```

Server initialized.

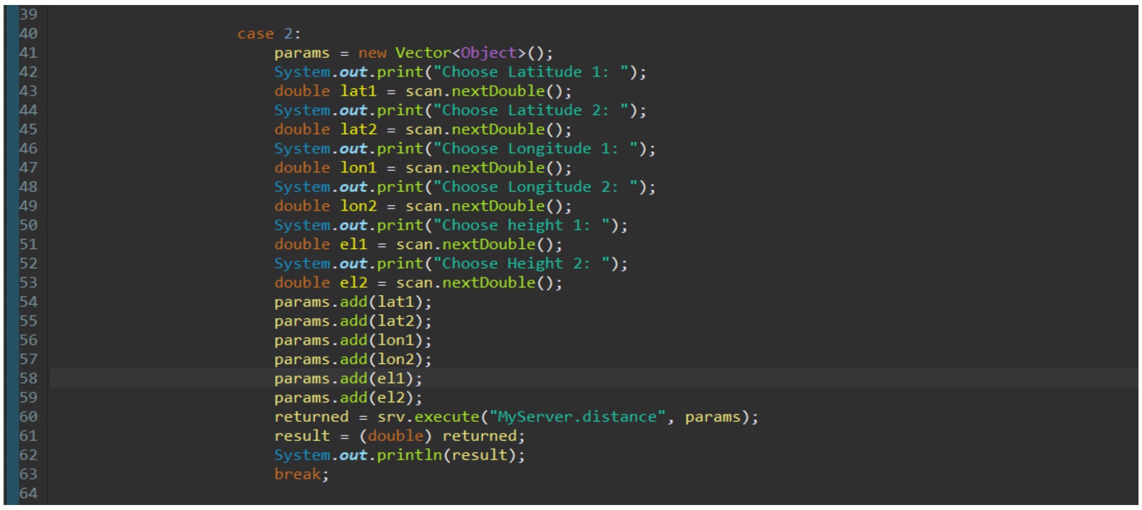


```java
package partA;

import java.net.URL;

public class Ac implements AsyncCallback {

    @Override
    public void handleError(Exception arg0, URL arg1, String arg2) {
        System.err.println("Exception: " + arg0);
        System.out.println("URL: " + arg1);
        System.out.println("Method: " + arg2);
    }

    @Override
    public void handleResult(Object arg0, URL arg1, String arg2) {
        System.out.println("Result: " + arg0);
        System.out.println("URL: " + arg1);
        System.out.println("Method: " + arg2);

    }

}
```

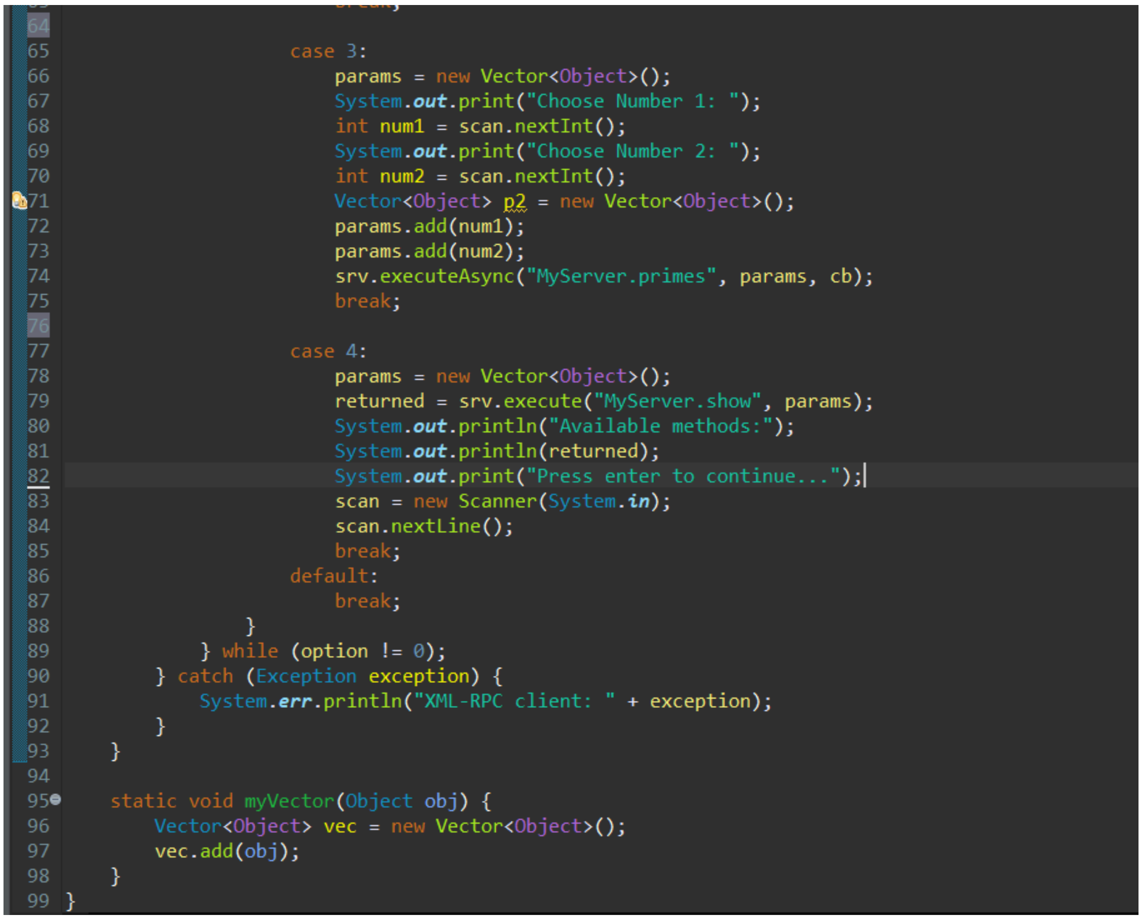
AC's code.



```java
package main;

import java.util.Scanner;

public class ClientRPC {
    public static void main(String[] args) {
        try {
            XmlRpcClient srv = new XmlRpcClient("http://localhost:10001");
            Ac cb = new Ac();
            Object returned;
            int option = -1;
            Vector<Object> params;
            Object result = null;
            do {
                System.out.println();
                System.out.println("MENU");
                System.out.println("1.  echo()");
                System.out.println("2.  distance()");
                System.out.println("3.  primes()");
                System.out.println("4.  show()");
                System.out.println("0.  exit");
                System.out.print("Option: ");
                Scanner scan = new Scanner(System.in);
                option = scan.nextInt();
                System.out.println();
                switch (option) {
                    case 1:
                        params = new Vector<Object>();
                        System.out.print("Choose Number 1: ");
                        params.add(scan.nextInt());
                        System.out.print("Choose Number 2: ");
                        params.add(scan.nextInt());
                        returned = srv.execute("MyServer.echo", params);
                        result = ((Integer) returned).intValue();
                        System.out.println(result);
                        break;
```

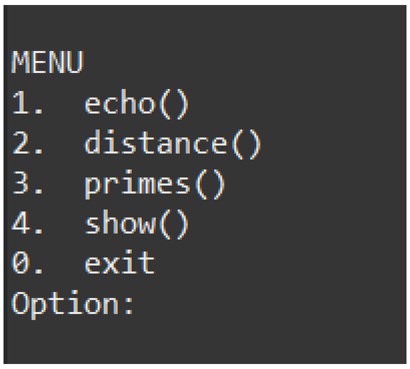


```
                case 2:
                    params = new Vector<Object>();
                    System.out.print("Choose Latitude 1: ");
                    double lat1 = scan.nextDouble();
                    System.out.print("Choose Latitude 2: ");
                    double lat2 = scan.nextDouble();
                    System.out.print("Choose Longitude 1: ");
                    double lon1 = scan.nextDouble();
                    System.out.print("Choose Longitude 2: ");
                    double lon2 = scan.nextDouble();
                    System.out.print("Choose height 1: ");
                    double el1 = scan.nextDouble();
                    System.out.print("Choose Height 2: ");
                    double el2 = scan.nextDouble();
                    params.add(lat1);
                    params.add(lat2);
                    params.add(lon1);
                    params.add(lon2);
                    params.add(el1);
                    params.add(el2);
                    returned = srv.execute("MyServer.distance", params);
                    result = (double) returned;
                    System.out.println(result);
                    break;
```



```
                case 3:
                    params = new Vector<Object>();
                    System.out.print("Choose Number 1: ");
                    int num1 = scan.nextInt();
                    System.out.print("Choose Number 2: ");
                    int num2 = scan.nextInt();
                    Vector<Object> p2 = new Vector<Object>();
                    params.add(num1);
                    params.add(num2);
                    srv.executeAsync("MyServer.primes", params, cb);
                    break;

                case 4:
                    params = new Vector<Object>();
                    returned = srv.execute("MyServer.show", params);
                    System.out.println("Available methods:");
                    System.out.println(returned);
                    System.out.print("Press enter to continue...");
                    scan = new Scanner(System.in);
                    scan.nextLine();
                    break;
                default:
                    break;
            }
        } while (option != 0);
    } catch (Exception exception) {
        System.err.println("XML-RPC client: " + exception);
    }
}

static void myVector(Object obj) {
    Vector<Object> vec = new Vector<Object>();
    vec.add(obj);
}
}
```

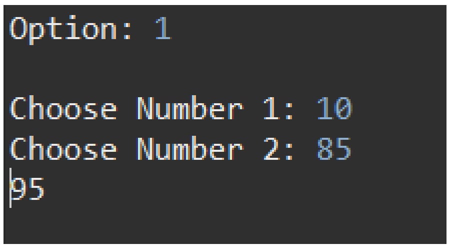
Client’s code.



```
MENU
1.   echo()
2.   distance()
3.   primes()
4.   show()
0.   exit
Option:
```

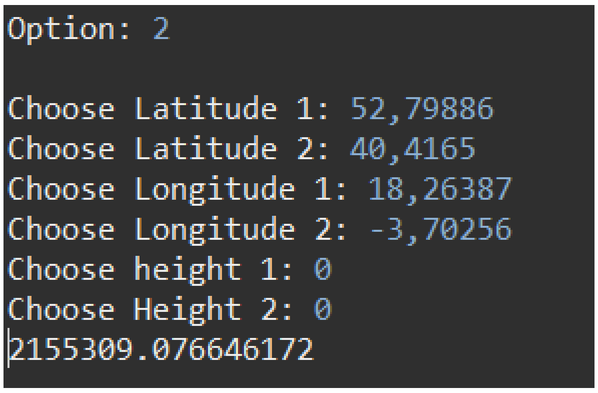
Client initialized.

OPTIONS FROM THE CLIENT



```
Option: 1

Choose Number 1: 10
Choose Number 2: 85
95
```

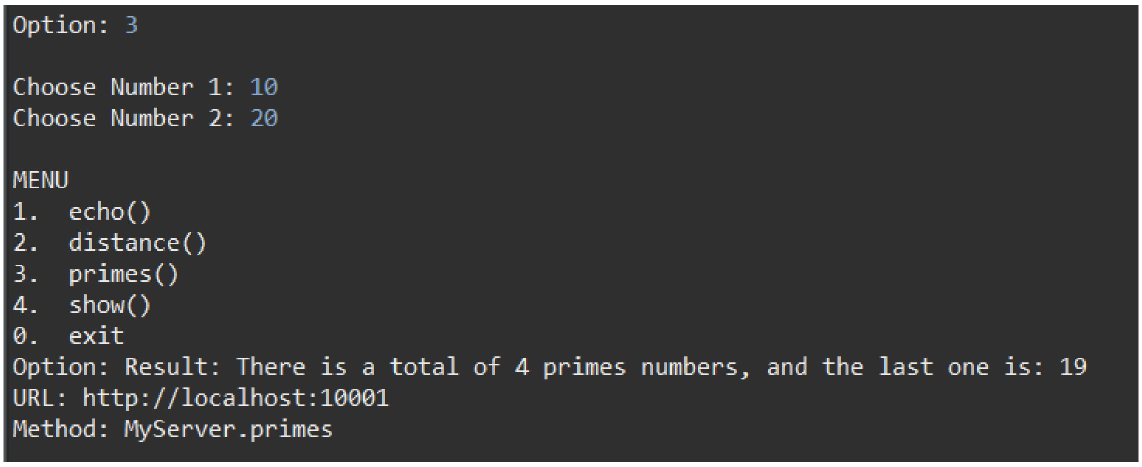
The first option returns the addition from the 2 numbers.



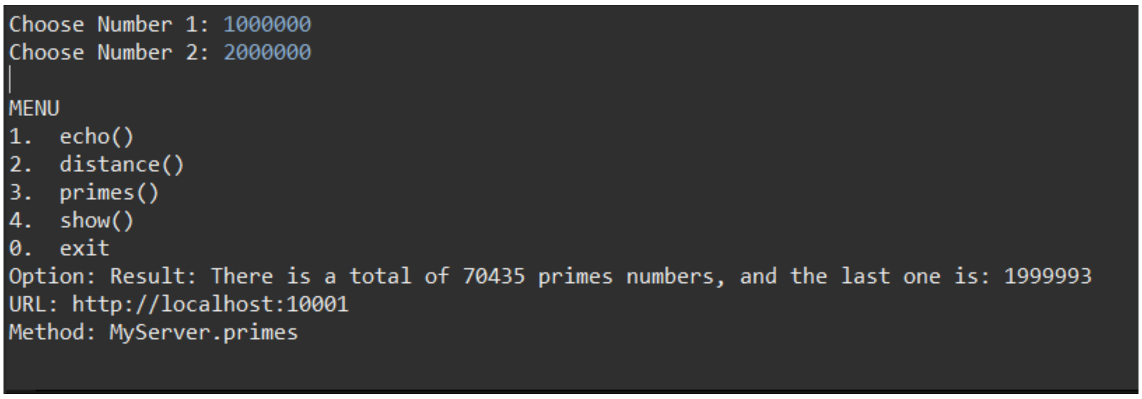
```
Option: 2

Choose Latitude 1: 52,79886
Choose Latitude 2: 40,4165
Choose Longitude 1: 18,26387
Choose Longitude 2: -3,70256
Choose height 1: 0
Choose Height 2: 0
2155309.076646172
```

Option 2, distance from Wroclaw to Madrid.

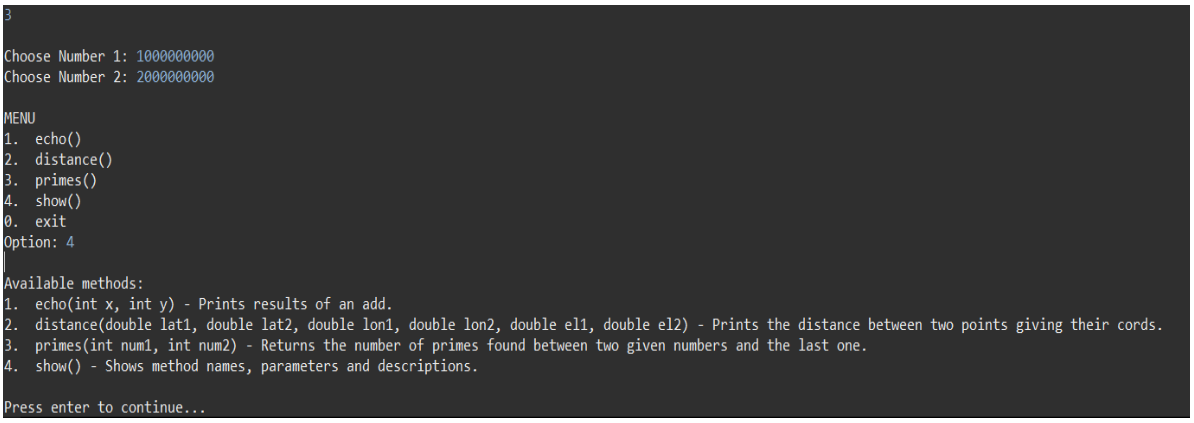
Option 3, to search for prime numbers:



```
Option: 3

Choose Number 1: 10
Choose Number 2: 20

MENU
1.  echo()
2.  distance()
3.  primes()
4.  show()
0.  exit
Option: Result: There is a total of 4 primes numbers, and the last one is: 19
URL: http://localhost:10001
Method: MyServer.primes
```



```
Choose Number 1: 1000000
Choose Number 2: 2000000

MENU
1.  echo()
2.  distance()
3.  primes()
4.  show()
0.  exit
Option: Result: There is a total of 70435 primes numbers, and the last one is: 1999993
URL: http://localhost:10001
Method: MyServer.primes
```



```
3

Choose Number 1: 1000000000
Choose Number 2: 2000000000

MENU
1.  echo()
2.  distance()
3.  primes()
4.  show()
0.  exit
Option: 4

Available methods:
1.  echo(int x, int y) - Prints results of an add.
2.  distance(double lat1, double lat2, double lon1, double lon2, double el1, double el2) - Prints the distance between two points giving their cords.
3.  primes(int num1, int num2) - Returns the number of primes found between two given numbers and the last one.
4.  show() - Shows method names, parameters and descriptions.

Press enter to continue...
```

The bigger the numbers, the longer it will take to process, but because it is asynchronized, we can do other tasks meanwhile (such as the number 4).

```
0
```

We end with the 0.