CMT205 Examination Answer Sheet

c1912705

Question 1

(a) The values of r1, r2, r3 and r4 are:

r1 = **4.2**

r2 = **9.100000000000001**

r3 = **2.7+6.4**

r4 = **2.$7+#4**

(b) i. The answer is coded below:

```
package exam2020.q1;

import java.util.Scanner;

/**
 * The type Average positive.
 *
 * @author Tian Z
 */
public class AveragePositive {
    /**
     * The entry point of application.
     *
     * @param args the input arguments
     */
    public static void main(String[] args) {
//        create a double array with the volume of 10
        double[] nums = new double[10];
        Scanner scanner = new Scanner(System.in);
//        create a double variable sum to record the sum of the numbers
        double sum = 0;
//        create a int count to count the positive number
        int count = 0;
//        use a for loop to record all the 10 double input
        for (int i = 0; i < nums.length; i++) {
            System.out.println("Please enter the next double:");
            nums[i] = scanner.nextDouble();
        }
//        use another for loop to add numbers to sum
        for (double num : nums) {
//            check if the number is positive, if so add it to the sum and add 1
to count
            if (num > 0) {
                sum += num;
```

```
                count += 1;
            }
        }
        double average = sum / count;
        System.out.println("The average of all the positive values is " +
average);
    }
}
```

(b) ii. The answer is coded below:

```
package exam2020.q1;

import java.util.Scanner;

/**
 * The type Positive int.
 *
 * @author Tian Z
 */
public class PositiveInt {
    /**
     * The entry point of application.
     *
     * @param args the input arguments
     */
    public static void main(String[] args) {
//        Initialize val
        int val;
        Scanner scanner = new Scanner(System.in);
        do {
            System.out.println("Please enter a String:");
            String input = scanner.nextLine();
            if (isNumeric(input)) {
                val = Integer.parseInt(input);
                System.out.println("The value of val is: " + val);
                break;
            }
        } while (true);

    }

    /**
     * Create a method to check if the String entered could be transformed to a
positive integer
     *
     * @param string the string
     * @return return true if the string could be transformed to a positive
integer
     */
    public static boolean isNumeric(String string) {
        for (int i = string.length(); --i >= 0; ) {
            if (!Character.isDigit(string.charAt(i))) {
                return false;
```

```
            }
        }
        return true;
    }
}
```

(c) i. The answer is coded below:

```
package exam2020.q1;

/**
 * The type Student account.
 *
 * @author Tian Z
 */
public class StudentAccount {
    /**
     * The entry point of application.
     *
     * @param args the input arguments
     */
    public static void main(String[] args) {
        String[] accounts = {"c1111111", "c2222222", "c3333333", "c4444444",
"c5555555"};
        boolean found = false;
        String query = "c1111111";
        for (String account : accounts) {
            if (account.equals(query)) {
                found = true;
                break;
            }
        }
        System.out.println("Variable found is " + found);
    }
}
```

(c) ii. The answer is coded below:

```
package exam2020.q1;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

/**
 * The type Student account collection.
 *
 * @author Tian Z
 */
public class StudentAccountCollection {
    /**
     * The entry point of application.
     *
```

```
     * @param args the input arguments
     */
    public static void main(String[] args) {
        String[] accounts = {"c1111111", "c2222222", "c3333333", "c4444444",
"c5555555"};
        boolean found = false;
        String query = "c1111111";
//        Use a HashSet to store all the Strings
        Set<String> accountSet = new HashSet<>(Arrays.asList(accounts));
        if (accountSet.contains(query)) {
            found = true;
        }
        System.out.println("Variable found is " + found);
    }
}
```

Question 3

SprinterThread.java

```
package exam2020.q3;

/**
 * The type Sprinter thread.
 *
 * @author Tian Z
 */
public class SprinterThread extends Thread {
    private int ID;
    private int time = 0;

    /**
     * Instantiates a new Sprinter thread.
     *
     * @param id the id
     */
    public SprinterThread(int id) {
        ID = id;
    }

    /**
     * Gets time.
     *
     * @return the time
     */
    public int getTime() {
        return time;
    }

    @Override
    public void run() {
//        total distance of 10000cm
        final int totalDistance = 10000;
        int distanceFinished = 0;
//        forward from 180cm to 250cm distance
```

```
        int maxDistanceStep = 250;
        int minDistanceStep = 180;
//        time step from 14ms to 18ms
        int maxTimeStep = 18;
        int minTimeStep = 14;

        while (distanceFinished < totalDistance) {
            int distanceStep = minDistanceStep + (int) (Math.random() *
(maxDistanceStep - minDistanceStep + 1));
            distanceFinished += distanceStep;
            int timeStep = minTimeStep + (int) (Math.random() * (maxTimeStep -
minTimeStep + 1));
            time += timeStep;
            try {
                Thread.sleep(timeStep);
            } catch (InterruptedException ie) {
                ie.printStackTrace();
            }
            System.out.println("Sprinter " + ID + " finished " + distanceStep +
" distance using " + timeStep + " time;");
        }

        System.out.println("Sprinter " + ID + " finished at time " + time +
".");
    }
}
```

SprinterSim.java

```
package exam2020.q3;

/**
 * The type Sprinter simulator.
 *
 * @author Tian Z
 */
public class SprinterSim {
    /**
     * The entry point of application.
     *
     * @param args the input arguments
     */
    public static void main(String[] args) {
        final int SPRINTER_NUM = 8;
//        Create a new Thread array of 8 length
        SprinterThread[] sprinter = new SprinterThread[SPRINTER_NUM];
//        use for loop to create Thread and start them
        for (int i = 0; i < SPRINTER_NUM; i++) {
            sprinter[i] = new SprinterThread(i);
            sprinter[i].start();
        }
        try {
            for (int i = 0; i < SPRINTER_NUM; i++) {
                sprinter[i].join();
            }
```

```
        } catch (InterruptedException ie) {
            ie.printStackTrace();
        }

        int bestSprinter = -1;
        int bestTime = Integer.MAX_VALUE;

        for (int i = 0; i < SPRINTER_NUM; i++) {
            int t = sprinter[i].getTime();
            if (t < bestTime) {
                bestTime = t;
                bestSprinter = i;
            }
        }
        System.out.println("Winner is: Sprinter " + bestSprinter + "!");
    }
}
```

Question 4

(a) The Singleton Pattern would be used in this scenario to restricts the instantiation of the NetworkMonitor Class to one single instance. One approach to implement this design pattern is coded below (Lazy initialization):

```
package exam2020.q4;

/**
 * @author Tian Z
 */
public class NetworkMonitor {
    private static NetworkMonitor networkMonitor;

    private NetworkMonitor() {}

    public static NetworkMonitor getInstance(){
        if (networkMonitor == null){
            networkMonitor = new NetworkMonitor();
        }
        return networkMonitor;
    }
}
```

(b) The answer is coded below:

TransServer.java

```
package exam2020.q4;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
```

```java
import java.net.Socket;
import java.net.SocketException;

/**
 * The type Trans server.
 *
 * @author Tian Z
 */
public class TransServer {
    /**
     * The entry point of application.
     *
     * @param args the input arguments
     */
    public static void main(String[] args) {
        double sum = 0;
        ServerSocket sSock;
        try {
            sSock = new ServerSocket(8000);
            Socket socket = sSock.accept();
            while (true) {
                InputStream in = socket.getInputStream();
                byte[] data = new byte[1024];
                int len = in.read(data);
                String purchaseStr = new String(data, 0, len);
//                add a exit to the while loop, when receiving "stop" shut down
the server
                if ("stop".equals(purchaseStr)) {
                    break;
                } else {
                    System.out.println("Received:");
                    System.out.println(purchaseStr);
                    String[] purchaseArr = purchaseStr.split(" ");
                    double price = Double.parseDouble(purchaseArr[0]);
                    double unitNum = Double.parseDouble(purchaseArr[1]);
                    sum += price * unitNum;
                    OutputStream out = socket.getOutputStream();
                    System.out.println("Send back:");
                    System.out.println(Double.toString(sum));
                    out.write(Double.toString(sum).getBytes());
                }
            }
            System.out.println("Server terminated.");
            socket.close();
            sSock.close();
        } catch (SocketException e) {
            e.printStackTrace();
        } catch (IOException e) {
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

TransClient.java

```
package exam2020.q4;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

/**
 * The type Trans client.
 *
 * @author Tian Z
 */
public class TransClient {
    /**
     * The entry point of application.
     *
     * @param args the input arguments
     * @throws IOException the io exception
     */
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("127.0.0.1", 8000);
        while (true) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Please enter the price and unit number:" +
                    "\n (Enter \"stop\" to exit.)");
            String line = scanner.nextLine();
//            If user entered "stop" then send it to the server then shut down
the client
            if ("stop".equals(line)) {
                OutputStream out = socket.getOutputStream();
                out.write(line.getBytes());
                break;
            } else {
                OutputStream out = socket.getOutputStream();
                out.write(line.getBytes());

                InputStream in = socket.getInputStream();
                byte[] data = new byte[1024];
                int len = in.read(data);
                System.out.println(new String(data, 0, len));
            }
        }
        System.out.println("Client closed.");
        socket.close();
    }
}
```