



GRADE 12

AUGUST / SEPTEMBER 2022

Time: 3 hours

Total: 150 Marks

EXAMINERS: Cluster

MODERATORS: Another Cluster

Memorandum

SECTION A STRUCTURED QUERY LANGUAGE

QUESTION 1

1.1	SELECT * ✓ FROM tblcountries ✓ WHERE stageWins = true ✓ ORDER BY CountryName ✓	4
1.2	SELECT * FROM tblRiders ✓ WHERE position IS NOT NULL ✓ AND ✓ (Age <= 25 ✓ OR Age > 35) ✓ Accept: WHERE Position > 0 WHERE Position <> 0	5

1.3	SELECT RiderName, RiderSurname, Position ✓ FROM tblRiders WHERE Position > 0 ✓ ORDER BY Position ✓ LIMIT 10 ✓	4
1.4	SELECT RiderName, RiderSurname, Age, Position ✓ FROM tblRiders WHERE Age ✓ = ✓ (SELECT MIN ✓ (Age) ✓ FROM tblRiders) ✓	6
1.5	SELECT CountryName , TotalRiders, TotalRankedRiders , ✓ CONCAT ✓ (LEFT(CountryName,3), ✓ FLOOR(RAND() ✓ * 99 + 1)) ✓ AS Code FROM tblCountries ✓ WHERE TotalRiders = TotalRankedRiders ✓	7
1.6	SELECT CountryName , AVG(Position) ✓ AS AvePosition ✓ FROM tblRiders, tblCountries ✓ WHERE tblRiders.CountryID = tblCountries.CountryID ✓ GROUP BY ✓ CountryName ✓ HAVING ✓ AvePosition <= 30 ✓ Accept: INNER JOIN	8

1.7	SELECT RiderName, RiderSurname, CountryName, Colour ✓ FROM tblRiders, tblCountries, tblJerseys ✓✓ WHERE tblRiders.CountryID = tblCountries.CountryID ✓ AND ✓ tblRiders.RiderID = tblJerseys.RiderID ✓ Accept: INNER JOIN	6
1.8	INSERT INTO tblRiders ✓ (RiderName, RiderSurname, Age, Position, YoungRider, CountryID) ✓ SELECT "Sebastian", "Schonberger" ✓, Age, 34, ✓ YoungRider, CountryID ✓ FROM tblRiders WHERE RiderID = 45 ✓	6
1.9	UPDATE tblRiders ✓ SET ✓ Position = 120 ✓ WHERE RiderID = 46 ✓	4

50 marks

SECTION B

OBJECT ORIENTED PROGRAMMING

Downloaded from eduresourceza.com

QUESTION 2

```
package memo;
```

```
import java.time.LocalDateTime;
```

```
//Question 2.1 (1)
```

```
public class YellowRider✓  
{
```

```
//Question 2.2 (2)
```

```
private String name, team; ✓ -1 not private  
LocalDateTime overAllTime; ✓
```

```
//Question 2.3 (2)
```

```
public static final ✓ String JERSEY_COLOUR = "[ YELLOW JERSEY ]"; ✓
```

```
//Question 2.4 (3)
```

```
public YellowRider ✓ (String n, String t, LocalDateTime o) ✓  
{  
    this.name = n; ✓  
    this.team = t;  
    this.overAllTime = o;  
}
```

```
//Question 2.5 (2)
```

```
public String getName() ✓✓  
{  
    return name;  
}
```

```
public String getTeam()  
{  
    return team;  
}
```

```
public LocalDateTime getOverAllTime()
```

Downloaded from eduresourceza.com

```

{
    return overAllTime;
}

public void setOverAllTime(LocalDateTime ovt)
{
    overAllTime = ovt;
}

```

//Question 2.6 (2)

```

@Override
public String toString() ✓
{
    return name + "\t" + team + "\t" + overAllTime; ✓
}

```

//Question 7.1 (6)

```

public void updateOverAllTime(String newTime) ✓
{
    int hour = Integer.parseInt(newTime.substring(0, 2)); ✓✓
    int mm = Integer.parseInt(newTime.substring(3, 5));
    int ss = Integer.parseInt(newTime.substring(6));

    LocalDateTime updateTime1 = overAllTime.plusHours(hour); ✓✓
    LocalDateTime updateTime2 = updateTime1.plusMinutes(mm);
    LocalDateTime updateTime3 = updateTime2.plusMinutes(ss);

    overAllTime = updateTime3; ✓
}
}

```

QUESTION 3

```

package memo;

import java.time.LocalDateTime;
import java.time.LocalTime;

```

//Question 3.1 (2)

```
public class GreenRider✓ extends YellowRider✓ {
```

```
//Question 3.2 (2)
```

```
    private int✓ points;✓
```

```
//Question 3.3 (1)
```

```
    public static final String JERSEY_COLOUR = "[ GREEN JERSEY ]";✓
```

```
//Question 3.4 (3)
```

```
    public GreenRider(String n, String t, LocalDateTime ovt, int p)✓ {
        super(n, t, ovt);✓
        points = p; ✓
    }
```

```
//Question 3.5 (2)
```

```
    public int getPoints()✓ {
        return points;
    }
```

```
    public void setPoints(int points)✓ {
        this.points += points;
    }
```

```
//Question 3.6 (2)
```

```
    @Override
    public String toString() {
        return super.toString()✓ + "\t" + points; ✓
    }
}
```

QUESTION 4, 6 and 7

```
package memo;
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;
import javax.swing.text.html.parser.DTDConstants;
```

//Question 4.1 (1)

```
public class RiderManager✓
{
```

//Question 4.2 (4)

```
private✓ YellowRider[] arrRiders✓ = new YellowRider[200];✓
private int size = 0;✓
```

//Question 4.3 (16)

```
public RiderManager(String fileName✓)
{
    try
    {
        Scanner scFile = new Scanner(new File(fileName));✓
        String n, tm, time;
        int pts;
        while (scFile.hasNext())✓
        {

            String line = scFile.nextLine();✓

            Scanner scLine = new Scanner(line).useDelimiter(",");✓
            n = scLine.next();✓
            tm = scLine.next();
            time = scLine.next();

            int hour = Integer.parseInt(time.substring(0, 2));✓
            hour = hour - 24;✓

            int mm = Integer.parseInt(time.substring(3, 5));✓
            int ss = Integer.parseInt(time.substring(6));✓

            LocalDateTime ldt = LocalDateTime.of(2022, 7, 1, hour, mm, ss);✓
```

```

        if (scLine.hasNext()) ✓
        {
            pts = scLine.nextInt(); ✓

            arrRiders[size] = new GreenRider(n, tm, ldt, pts); ✓
        } else
        {

            arrRiders[size] = new YellowRider(n, tm, ldt); ✓
        }

        size++; ✓
    }

} catch (FileNotFoundException ex)
{
    System.out.println("File not found");
}
}

```

//Question 4.4 (5)

```

public String toString() ✓
{
    String temp = "";
    for (int i = 0; i < size; i++) ✓
    {
        temp += arrRiders[i].toString() + "\n"; ✓
    }
    return temp; ✓
}

```

//Question 4.5 (6)

```

public int searchRider(String name) ✓
{
    for (int i = 0; i < size; i++) ✓
    {
        if (arrRiders[i].getName().equalsIgnoreCase (name)) ✓
        {
            return i; ✓
        }
    }
}

```

```

    }
}
return -1; ✓
}

```

//Question 6.1 (8)

```

public GreenRider ✓ findFastestGreenRider()
{
    int lead = 0;
    GreenRider temp = null; ✓

    for (int i = 1; i < size; i++) ✓
    {
        if (arrRiders[i] instanceof GreenRider) ✓
        {
            if (lead < ((GreenRider) ✓ arrRiders[i]).getPoints()) ✓
            {
                lead = ((GreenRider) arrRiders[i]).getPoints(); ✓
                temp = (GreenRider) arrRiders[i]; ✓
            }
        }
    }
    return temp;
}

```

//Question 6.2 (6)

```

public String listJerseyRiders() ✓
{
    String temp = YellowRider.JERSEY_COLOUR ✓ + "\n";
    temp += arrRiders[0].toString() + "\n"; //1
    temp = temp + GreenRider.JERSEY_COLOUR ✓ + "\n";
    temp = temp + findFastestGreenRider() ✓.toString() ✓ + "\n\n";
    return temp;
}

```

//Question 6.2 (16)

```

public String stageResults(String fn) ✓
{

```

```

String tempSt = "";
try
{
    Scanner scFile = new Scanner(new File(fn)); ✓
    while (scFile.hasNext())
    {
        String[] line = scFile.nextLine().split(","); ✓
        String name = line[0]; ✓
        int pos = searchRider(name); ✓
        if (pos != -1) ✓
        {
            String tm = line[1]; ✓
            arrRiders[pos].updateOverAllTime(tm); ✓
            if (line.length == 3) ✓
            {
                int pts = Integer.parseInt(line[2]); ✓
                if (arrRiders[pos] instanceof GreenRider) ✓
                {
                    GreenRider temp = (GreenRider) arrRiders[pos]; ✓
                    temp.setPoints(pts); ✓
                } else
                {
                    String team = arrRiders[pos].getTeam();
                    LocalDateTime ovt = arrRiders[pos].getOverallTime();
                    arrRiders[pos] = new GreenRider(✓name, team, ovt, pts); ✓
                }
            }
            tempSt = tempSt + arrRiders[pos] + "\n"; ✓
        }
    }
} catch (FileNotFoundException ex)
{
    System.out.println("File not found");
}
return tempSt;
}
}

```

```
package memo;
```

```
//Question 5.1 (1)
```

```
public class LeTourUI ✓
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        //Question 5.2 (2)
```

```
        RiderManager rm✓ = new RiderManager✓ ("overallstage10.txt");
```

```
        //Question 5.3 (1)
```

```
        System.out.println(rm.toString());✓
```

```
        //Question 6.3 (2)
```

```
        System.out.println✓ (rm.listJerseyRiders()✓);
```

```
        //Question 7.3 (2)
```

```
        System.out.println(rm.stageResults✓ ("stage11results.txt"));✓
```

```
    }
```

```
}
```

QUESTION 3

Question 3 [13]

Question 3.1 (2)

```
public class GreenRider extends YellowRider { //1 class name //1 extends
```

Question 3.2 (2)

```
    private int points; //1 private //1 field
```

Question 3.3 (1)

```
    public static final String JERSEY_COLOUR = "[ GREEN JERSEY ]"; //1
```

Question 3.4 (3)

```
    public GreenRider(String n, String t, String o, int p) { //1
        super(n, t, o); //1
        points = p; //1
    }
```

Question 3.5 (2)

```
    public int getPoints() { //1
        return points;
    }
```

```
    public void setPoints(int points) { //1
        this.points += points;
    }
```

Question 3.6 (2)

```
    @Override
    public String fileLine() { //1 override //1 return
        return super.fileLine() + "#" + points;
    }
```

Question 3.7 (1)

```
    @Override
    public String toString() { //1
        return super.toString() + "\t" + points;
    }
```

```
}
```

QUESTION 4

Downloaded from eduresourceza.com

```
public class Controller { //Q 4.1 //new class 1

    //Question 4.2 (4)
    private YellowRider[] arr = new YellowRider[200]; //1 private //1 YellowRider arr 200
    private int count = 0; //1 counter //1 counter =0

    //Question 4.3 (12)
    public Controller(String fileName) //1 Method header with filename
    {
        try //1 if file exists
        {
            Scanner scFile = new Scanner(new File(fileName)); //1 open file

            while (scFile.hasNext()) //1 loop
            {
                String[] line = scFile.nextLine().split("#"); //1 read line
                String temp = ""; //1 tokenise

                //1 test line
                if (line.length == 3) { //1 add to array
                    arr[count] = new YellowRider(line[0], line[1], line[2]); //1 create object
                } else { //1 correct constructor
                    arr[count] = new GreenRider(line[0], line[1], line[2], Integer.parseInt(line[3]));
                }

                count++; //1 increment counter
            }

        } catch (FileNotFoundException ex) {
            System.out.println("File not found"); //1 display message
        }
    }

    //Question 4.4 (5)
    public String listAll() {

        String temp = listJerseyRiders(); //1 initiate string
        //listJerseyRiders can be called here or UI

        for (int i = 0; i < count; i++) { //1 loop
            temp += i + 1 + "\t" + arr[i].toString() + "\n"; //1 rider pos
        } //1 return + object.toString
        return temp; //1 new line
    }
}
```

Downloaded from eduresourceza.com

//Question 4.6 (6)

```

public GreenRider findGreenRider () { //1 method header
    int lead = 0;
    GreenRider temp = null;

    for (int i = 1; i < count; i++) {
        if (arr[i] instanceof GreenRider) { //1
            //1 type cast
            if (lead < ((GreenRider) arr[i]).getPoints()) //1 compare
            {
                lead = ((GreenRider) arr[i]).getPoints(); //1 new points leader
                temp = (GreenRider) arr[i];
            }
        }
    }
    return temp; //1 return object
}

```

//Question 4.7 (5)

```

public String listJerseyRiders() {
    String temp = "\t" + YellowRider.JERSEY_COLOUR + " ]\n"; //1;
    temp += arr[0].toString() + "\n"; //1
    temp = temp + "\t" + GreenRider.JERSEY_COLOUR + " ]\n"; //1
    temp = temp + findGreenRider().toString() + "\n\n"; //1 add heading and rider
    //1 method call
    return temp;
}

```

//Question 4.5 (6)

```

public void sort() {
    for (int i = 0; i < count - 1; i++) //1 both counters
    {
        for (int j = i + 1; j < count; j++) //1 both counters
        {
            if (arr[i].getOverall().compareTo(arr[j].getOverall()) > 0) { //1 compare //1 object method call
                YellowRider temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp; //2 swop
            }
        }
    }
}

```

//Question 6.2 (17)

```

public void stageResults(String fn) {
    LocalTime start = LocalTime.of(12, 35, 0); //1
    try {

```

```

Scanner scFile = new Scanner(new File(f1));

while (scFile.hasNext()) //marks for file reading already assigned
{
    String[] line = scFile.nextLine().split("#"); //1 read and tokenise
    int i = 0;
    boolean found = false;
    while (!found || i < count) //2 flagged search to match object
    {
        if (line[0].equals(arr[i].getName())) //1 match object
        {
            LocalDateTime stageTime = LocalDateTime.parse(line[2],
DateTimeFormatter.ofPattern("HH:mm:ss")); //2 read and parse
            arr[i].setOverall(start, stageTime); //1 method call

            if (line.length == 4) { //1
                if (!(arr[i] instanceof GreenRider)) { //1
                    arr[i] = new GreenRider(arr[i].getName(), arr[i].getTeam(), arr[i].getOverall(),
Integer.parseInt(line[3]));
//1
                }
                ((GreenRider) arr[i]).setPoints(Integer.parseInt(line[3])); //2 set points
            }
            found = true;
        }
        i++;
    }

    }

    sort(); //1 sort
    PrintWriter p = new PrintWriter(new File("src/resources/overallstage11.txt")); //1 printwriter

    for (int i = 0; i < count; i++) {
        p.println(arr[i].fileLine()); //1 write to file
    }
    p.close(); //1 save

} catch (FileNotFoundException ex) {
    System.out.println("File not found");
}

}
//*****
}

```

100 marks

Total: 150 marks

eduresourceza.com