PRELIMINARY EXAMINATION 2016
INFORMATION TECHNOLOGY: Paper II

Grade 12

Examiners: Mr T Bothma, Mr R Viljoen
Moderator: Ms M. Ounaceur, Ms M. Walker

Date: September 2016

Time: 3 hours

120 marks

Memorandum

SECTION A: STRUCTURED QUERY LANGUAGE

QUESTION 1

1.1 (3)

```
SELECT Name, Surname, School ✓
FROM tblStudents ✓
ORDER BY School, Surname ✓
```

1.2 (3)

```
SELECT *✓
FROM tblStudents✓
WHERE Gender="F"✓
```

1.3.1 (6)

```
INSERT INTO tblStudents✓
( studentID, Name, Surname, DoB, Gender, School, Team, Grade, Meal )  ✓
SELECT ✓40, 'Carmel', 'Naidoo' , '2001/11/26', 'F',  ✓  School, Team, Grade, Meal ✓
FROM tblStudents WHERE studentID = 41; ✓
```

1.3.2 (2)

```
DELETE *✓FROM tblStudents
WHERE studentID = 41✓
```

1.4 (2)

```
UPDATE tblStudents✓ SET Surname = "Blackburn"
WHERE Surname="Blackburm";✓
```

1.5 (7)

```
SELECT tblSchools.School, Grades, COUNT(Grades)  ✓ AS [Number of Participants]  ✓
FROM tblGrade, tblStudents, tblSchools ✓
WHERE tblGrade.GrID = tblStudents.Grade AND tblStudents.School = tblSchools.SchoolID✓
GROUP BY tblSchools.School, Grades✓
ORDER BY tblSchools.School, ✓  DESC Grades;  ✓
```

1.6 (5)

```
SELECT INT✓ (COUNT(*)/4) ✓    AS [Number of groups]          // round would be wrong
FROM tblstudents
WHERE Year(DOB)✓    <>    SELECT MIN (YEAR(DOB)✓) FROM tblstudents;✓
```

1.7 (8)

```
SELECT tblSchools.School, Team✓,
5-COUNT(studentID)✓ AS [To Recruit]   ✓
FROM tblSchools, tblStudents ✓
WHERE tblSchools.SchoolID = tblStudents.School ✓
GROUP BY tblSchools.School, Team ✓
HAVING✓ COUNT(studentID) < 5   ✓
```

1.8 (4)

```
SELECT Meal, Sum([Price]*5)   ✓ AS [Price per Meal]   ✓
FROM tblMeals, tblStudents
WHERE tblMeals.MealID = tblStudents.Meal ✓
GROUP BY Meal;   ✓
```

SECTION A: 40

SECTION B: OBJECT-ORIENTED PROGRAMMING

QUESTION 2

```
package userinterface_java;

import javax.smartcardio.CardTerminals;

/**
 *
 * @author Hans Arnold Peuckert
 */
```

Please note: I have used the exact mark allocation & comments of the Delphi memorandum

//2.1

```
public class Race  {   ✓ (class created)
```
(3)

```
   private String raceTime;      ✓ (time field & correct type)
   private int parTime;          ✓ (par time field & correct type))
```

//2.2

```
   public Race(String r, int p) {   ✓ (correct header)
      raceTime = r;                 ✓ (both correctly assigned)
      parTime = p;
   }
```
(2)

```
//2.3
public String toString() {              ✓ (correct header)                    (2)
   return "Time: " + raceTime;          ✓ (correct format & returned)
}

//2.4                                                                         (4)
public double convertTime() {
   double minutes = 0;

   int hours = Integer.parseInt(raceTime.substring(0, 1));
   int posPoint = raceTime.indexOf(".");
   int min = Integer.parseInt(raceTime.substring(2, posPoint));
   int sec = Integer.parseInt(raceTime.substring(posPoint + 1, raceTime.length()));

   minutes = hours * 60 + min + sec / 60.0;      //or: ... + (double) sec/60;

   return minutes;

            (-1 each if hh or mm or ss not correctly converted)
            (-1 if not correctly added)
            (-1 if not returned)
}

//2.5                                                                         (1)
public double getTimeDiff() {
   return convertTime() - parTime;    ✓
}

}                                                                   TOTAL: 12
```

QUESTION 3

```
package userinterface_java;

/**
 *
 * @author Hans Arnold Peuckert
 */

//3.1                                                                         (4)
public class Rider {    ✓ (class created)

   private String name;      ✓ (all field names and types)
   private String dob;
   private String school;
   private Race[] race = new Race[3];          ✓ (array of size 3)  ✓ (of type Race)
```

//3.2

```
public Rider(String n, String d, String s) {                                    (4)
   name = n;
   dob = d;
   school = s;          ✓✓ (all assigned)

   for (int i = 0; i < 3; i++) {
      race[i] = null;                    ✓✓ (all set to null, no marks lost if done one by one)
   }
}
```

//3.3

```
public String getName() {                                                       (1)
   return name;              ✓ (all correct)
}
```

//3.4

```
public int getAge() {                                                           (2)
   return 2016 - Integer.parseInt(dob.substring(0, 4));   ✓✓ (all correct – give 1 mark if almost
                                                              correct solution)

        (Minette wrote on the Delphi memo - Do you think this is acceptable? Rickus said yes for now
         but teach for end of year)
}
```

//3.5

```
public String toString() {                                                      (4)
   String output = "Name: " + name + " Age: " + getAge() + "\n";  ✓ (all correct)

   for (int i = 0; i < 3; i++) {     ✓ (loop)
      if (race[i] != null) {         ✓ (check if exists)
         output = output + "Race " + (i + 1) + " " + race[i].toString() + "\n";  ✓ (return in correct format)
      }
   }

   return output;
}
```

//3.6

```
public void setRace(Race competition) {                                         (4)
   for (int i = 0; i < 3; i++) {                  ✓✓ (loop and ensure to stop when null found)
      if (race[i] == null) {                      ✓ (check if null)
         race[i] = competition;                   ✓ (assign)
         break;
      }                          // Instead of a while loop I used a for loop and a break.
   }                             // It is so less complicated for weaker students and it works perfectly.
}
```

```
//3.7
public boolean equals(String n, String d) {          ✓ (correct header)                    (3)
    if (name.equals(n) && dob.equals(d)) {           ✓ (compare)
        return true;
    } else {                                         ✓(assign and return boolean value)
        return false;
    }
}
```

```
//3.8
public double bestResult() {                                                               (3)
    double max = Integer.MAX_VALUE;

    for (int i = 0; i < 3; i++) {
        if (race[i] != null) {                  ✓✓✓ (any valid method for getting best race result)
            if (race[i].getTimeDiff() < max) {      (1 mark – good attempt, 2 marks – almost correct)
                max = race[i].getTimeDiff();
            }
        }
    }

    return max;
}
}                                                                                          TOTAL: 25
```

QUESTION 4 & 6

```
package userinterface_java;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author Hans Arnold Peuckert
 */
```

```
//4.1
public class Controller {  ✓                                                               (1)
```

```
//4.2
private Rider[] riderArr = new Rider[50];  ✓                                               (2)
private int count = 0;                     ✓
```

```
//4.3
public Rider findRider(String n, String b) {  ✓                                    (5)

   Rider rider = null;

   for (int i = 0; i < count; i++) {          ✓ (conditional linear search with flag / for loop with break)
      if (riderArr[i].equals(n, b)) {         ✓ (compare & method call)
         rider = riderArr[i];          //or: return riderArr[i];   ✓ (return object)
         break;
      }
   }

   return rider;                  //and then here: return null;  ✓ (return null)
}
```

```
//4.4
public Controller() {

   try {
      //4.4.1
      Scanner scFile = new Scanner(new File("Results.txt"));   ✓ (open file for reading)   (6)

      while (scFile.hasNext()) {                    ✓ (loop through file)
         String line = scFile.nextLine();           ✓ (read line from file)

         Scanner scLine = new Scanner(line).useDelimiter("#");

         String riderName = scLine.next();
         String dateOfBirth = scLine.next();
         String school = scLine.next();             ✓✓ (take apart, give 1 mark if mostly correct)
         String time = scLine.next();
         int par = scLine.nextInt();

         Race race = new Race(time, par);           ✓ (create race object)

         //4.4.2
         if (findRider(riderName, dateOfBirth) == null) {          ✓ (if rider does not exist)   (6)

            riderArr[count] ✓ = new Rider(riderName, dateOfBirth, school); ✓ (add rider)

            riderArr[count].setRace(race);   ✓ (add race)
            count++;                                               ✓ (increase counter)
         } else {
            findRider(riderName, dateOfBirth).setRace(race);  ✓ (if exist just add race)

         }
      }
```

```
    } catch (FileNotFoundException ex) {
        JOptionPane.showMessageDialog(null, "Error. The text file could not be found.");
    }

}
```

//4.5

```
public String toString() {                                                    (3)
    String output = "";    ✓

    for (int i = 0; i < count; i++) {  ✓
        output = output + riderArr[i].toString() + "\n";   ✓ (must add "\n" here or at toString from Rider)
    }

    return output;
}
```

//4.6

```
public String ridersQualify() {                                               (4)
    String output = "";
    for (int i = 0; i < count; i++) {                          ✓(loop through all players)
        if (riderArr[i].bestResult() + (riderArr[i].getAge() * 5) < 80) {  ✓ (bestResult+getAge*5)  ✓ (if < 80)
            output = output + riderArr[i].getName() + "\n";    ✓ (put together list of names and return)
        }
    }
    return output;
}
```

TOTAL: 27

// Q6

```
public String eliminator() {                                                  (12)

    String output = "";

    int[] rounds = new int[6];              ✓ (double array or other suitable data structure)
    int[] schools = new int[6];

    for (int i = 0; i < 6; i++) {
        rounds[i] = 0;                      ✓ (initialise)
        schools[i] = i + 1;
    }
```

```
    Scanner scFile;
    try {
        scFile = new Scanner(new File("eliminator.txt"));      ✓ (file access and read)

        boolean stop = false;
        while (scFile.hasNext() && !stop) {
            String line = scFile.nextLine();
            int team = Integer.parseInt(line);
            rounds[team - 1]++;
            for (int i = 0; i < 5; i++) {
                if (rounds[i] == 18) {                          ✓ (count rounds)
                    stop = true;
                }
            }
        }

        for (int i = 0; i < 5; i++) {                           ✓✓
            for (int j = i + 1; j < 6; j++) {

                if (rounds[i] < rounds[j]) {                    ✓

                    int temp = rounds[i];
                    rounds[i] = rounds[j];
                    rounds[j] = temp;                           ✓✓ (both arrays)

                    temp = schools[i];
                    schools[i] = schools[j];
                    schools[j] = temp;
                }
            }
        }

        output = "";
        for (int i = 0; i < 3; i++) {                                       ✓ (top 3)
            output = output + getSchool(schools[i]) + ": " + rounds[i] + "\n";
        }

    } catch (FileNotFoundException ex) {
        JOptionPane.showMessageDialog(null, "Error. The text file could not be found.");
    }
    return output;
}
```

```
   public String getSchool(int s) {
      String school = "";
      switch (s) {
         case 1:
            school = "Menlopark";
            break;                          ✓✓ (one mark if very close to a correct solution)
         case 2:
            school = "Waterkloof";
            break;
         case 3:
            school = "Midstream";
            break;
         case 4:
            school = "Cornwall Hill";
            break;
         case 5:
            school = "Montana";
            break;
         case 6:
            school = "Wonderboom";
            break;
      }
      return school;
   }

}                                                                    TOTAL: 12
```

QUESTION 5

```
package userinterface_java;

/**
 *
 * @author Hans Arnold Peuckert
 */
```

//5.1

```
public class UserInterface_Java {          ✓                                  (1)

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

//5.2

```
      Controller cont = new Controller();  ✓ (declare and instantiate object)  (1)
```

```
    //5.3
    System.out.println(cont.toString());                                    ✓ (call toString)          (2)

    System.out.println("Qualified for nationals: \n" + cont.ridersQualify()); ✓ (call ridersQualify
                                                                              correct output)
```

TOTAL: 4

```
    // Q6
    System.out.println(cont.eliminator());
  }

}
```

SECTION B: 80

GRAND TOTAL: 120 MARKS

OUTPUT OF THE JAVA PROGRAM:

Name: Tielman Roos Age: 17
Race 1 Time: 1h22.34
Race 2 Time: 1h28.44

Name: Peter Hammer Age: 17
Race 1 Time: 1h10.10

Name: James Simpson Age: 17
Race 1 Time: 1h30.44
Race 2 Time: 1h24.59
Race 3 Time: 1h32.30

Name: Barry Waldron Age: 16
Race 1 Time: 1h13.30

Name: Gary Hamilton Age: 16
Race 1 Time: 1h26.14
Race 2 Time: 1h12.30
Race 3 Time: 1h27.41

Name: Mackenzie Crook Age: 17
Race 1 Time: 1h16.26

Name: Thando Mchunu Age: 16
Race 1 Time: 1h24.51

Name: Simphiwe Mkize Age: 15
Race 1 Time: 1h33.39

Name: Ron Naiker Age: 18
Race 1 Time: 1h23.09

Name: Johan van der Walt Age: 15
Race 1 Time: 1h37.23

Name: Gary Bottrill Age: 18
Race 1 Time: 1h34.49

Name: Andre Richardson Age: 15
Race 1 Time: 1h24.47

Name: Stephan van Tonder Age: 15
Race 1 Time: 1h40.18

Name: Chad Fikkert Age: 17
Race 1 Time: 1h18.52
Race 2 Time: 1h14.33

Name: Marc Simpson Age: 20
Race 1 Time: 1h29.19
Race 2 Time: 1h11.33

Name: Arthur White Age: 19
Race 1 Time: 1h15.12

Name: James Clark Age: 17
Race 1 Time: 1h18.29

Name: Musa Mogale Age: 20
Race 1 Time: 1h40.06

Name: Ryan Coetzer Age: 17
Race 1 Time: 1h28.12

Name: John Lark Age: 17
Race 1 Time: 1h20.01

Name: Shivam Moodly Age: 20
Race 1 Time: 1h30.42

Name: John Blackburm Age: 16
Race 1 Time: 1h21.30

Name: Jeremy Johnson Age: 18
Race 1 Time: 1h28.24

Name: Fred Langeveld Age: 20
Race 1 Time: 1h24.34

Name: Jethro Chamberland Age: 15
Race 1 Time: 1h20.08

Name: Alex Wellworth Age: 15
Race 1 Time: 1h39.07

Name: Muza Botlhele Age: 16
Race 1 Time: 1h33.03

Name: Khumo Xaba Age: 17
Race 1 Time: 1h18.23

Name: Josh Dewrance Age: 16
Race 1 Time: 1h23.09

Name: Martin Sebethoma Age: 16
Race 1 Time: 1h21.48

<u>Qualified for nationals:</u>
Simphiwe Mkize
Andre Richardson
Chad Fikkert
Jethro Chamberland
Josh Dewrance

Midstream: 18
Waterkloof: 17
Menlopark: 17