PECANWOOD COLLEGE

*Prepared for Life*

# INFORMATION TECHNOLOGY JULY PRACTICAL EXAMINATION GRADE 11

**NAME:** _____ **GRADE:** _____

**DATE: 16 JULY 2020** **MARKS: 90**
**MODERATOR: MR NAIRAR** **TIME: 180 MINUTES**
**EXAMINER: MR SC EILERTSEN**

---

**INSTRUCTIONS:**

1. This test is made up of 9 pages including one addendum.
2. All learners were told beforehand that this practical examination written on the school campus offers a more supportive environment than writing the same examination at home.
   a. Learners writing at home are expected to have a stable internet connection with a working computer.
   b. Java 8 SE installed as well as a suitable Java IDE (jGRASP or NetBeans)
   c. SQLBrowser must also be installed to allow learners to practically code the SQL queries in section three.
3. Note the mark allocation for each question.
4. Addendum One is provided to ensure that all learners, both online and on the school campus have the same advantages.

---

**Section One** **Java One Class with Static Methods**

Create a one class program that uses static methods to convert US Dollars to British Pounds and British Pounds to US Dollars. (26)

**The conversion rates are as follows.**

- US Dollars to pounds multiply by 0.792
- British pounds to US Dollars multiply by 1.262

Use variables for the conversion rates so that they can be updated more easily.

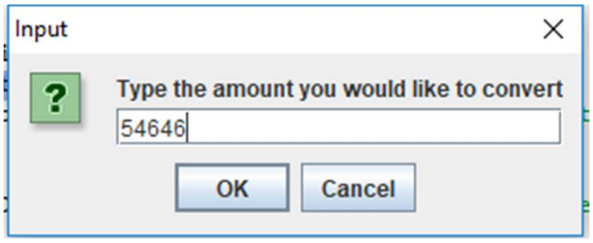
**Input**
Ask the user: US Dollars to British Pounds OR British Pounds to US Dollars.
The amount of currency that needs to be converted.

**Processing**
Call the relevant method to do the processing

**Output**
Report the original amount with the conversion to the user

**Defensive programming**
1. Good use of upper-case and lower-case conversions as needed.
2. Reject a conversion from the keyboard that does not exist.
3. Reject a negative currency amount.

Your program should look a lot like this . . .


```
----jGRASP exec: java CurrencyConverter
Pounds to Dollars
The amount to be converted was 54646.0
The new amount is 68963.25200000001

----jGRASP: operation complete.
```

**NOTES**
To save time in this exam **no marks** have been allocated for . . .
- The correct currency symbols.
- Rounding off to two decimal places.
- Only 2 marks are allocated for the output so text mode is fine (as above).

Marks **are** allocated for good layout, indentation, variable names, class names and whitespace.

**Section Two** **Java Two-Class with Non-Static Methods**

Create a two-class program that uses non-static methods to convert US Dollars to British Pounds and British Pounds to US Dollars. It must read in the daily rates from a text file and store them in a one dimensional array. In this case, the text file will only have two variables. Your program will have a UI class and a manager class to store all the methods.

Much of your code from section one can be re-used here in section two. (38)

**The conversion rates are as follows and are stored in a text file.**

- US Dollars to pounds multiply by 0.792
- British pounds to US Dollars multiply by 1.262

Therefore, you text file will look like this . . .

0.792
1.262

**Input**
Ask the user to input their first and last name eg steve eilertsen
What do they want to convert? US Dollars to British Pounds OR British Pounds to US Dollars.
The amount of currency that needs to be converted.

**Processing**
Call the relevant String handling method in the manager class
Call the relevant method to do the currency conversion, also in the manager class.

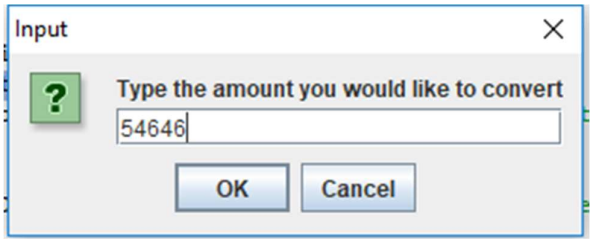
**Output**
Report the name of the user in the following format - EILERTSEN S
Report the original amount with the converted amount to the user

**Defensive programming**

4. Good use of upper-case and lower-case conversions as needed.
5. Reject a conversion from the keyboard that does not exist.
6. Reject a negative currency amount.

Your program should look a lot like this . . .





```
 ----jGRASP exec: java CurrencyConverterUI
EILERTSEN S
The amount to be converted was 54646.0
The new amount is 68963.25200000001

 ----jGRASP: operation complete.
```

**NOTES**
To save time in this exam **no marks** have been allocated for . . .

- The correct currency symbols.
- Rounding off to two decimal places.
- Only 2 marks are allocated for the output, so text mode is fine (as above).

Marks **are** allocated for good layout, indentation, variable names, class names and whitespace.

**Section Three** **Structured Query Language**

You have been supplied with a SQL compatible database i.e. "adventure motorcycles.mdb" – this is a one table database. The table's name is "Specifications" – be careful to **spell it correctly** in your SQL statements. Here is an overview of the fields in the table. Write SQL queries to answer the given questions. (26)

Specifications

| Field Name | Data Type | Description |
|---|---|---|
| ID | AutoNumber | Primary key. |
| bikeName | Short Text | The general everyday name of the motorcycle. |
| category | Short Text | The genre/style/catagory of the motorcycle. |
| firstYear | Number | First year of manufacture. |
| lastYear | Number | Last year of manufacture. |
| dateOfLaunch | Date/Time | Date of motorcycle launch - day, month, year. |
| model | Short Text | The full and correct name of the motorcycle. |
| cooling | Short Text | How the engine is cooled. |
| gearbox | Number | The number of gears the motorcycle has. |
| transmission | Short Text | How the power is transferred from the engine to the rear wheel. |
| power | Number | The power output of the engine in horsepower. |
| torque | Number | The torque output of the engine in Nm (newton meters). |
| engineType | Short Text | Style/design of the engine. |
| capacity | Number | The size of the engine measured in ccm (cubic centimeters). |
| seatHeight | Number | The height of the seat off the ground in millimeters. |
| weight | Number | The weight of the motorcycle without oil or fuel measured in kilgrams. |

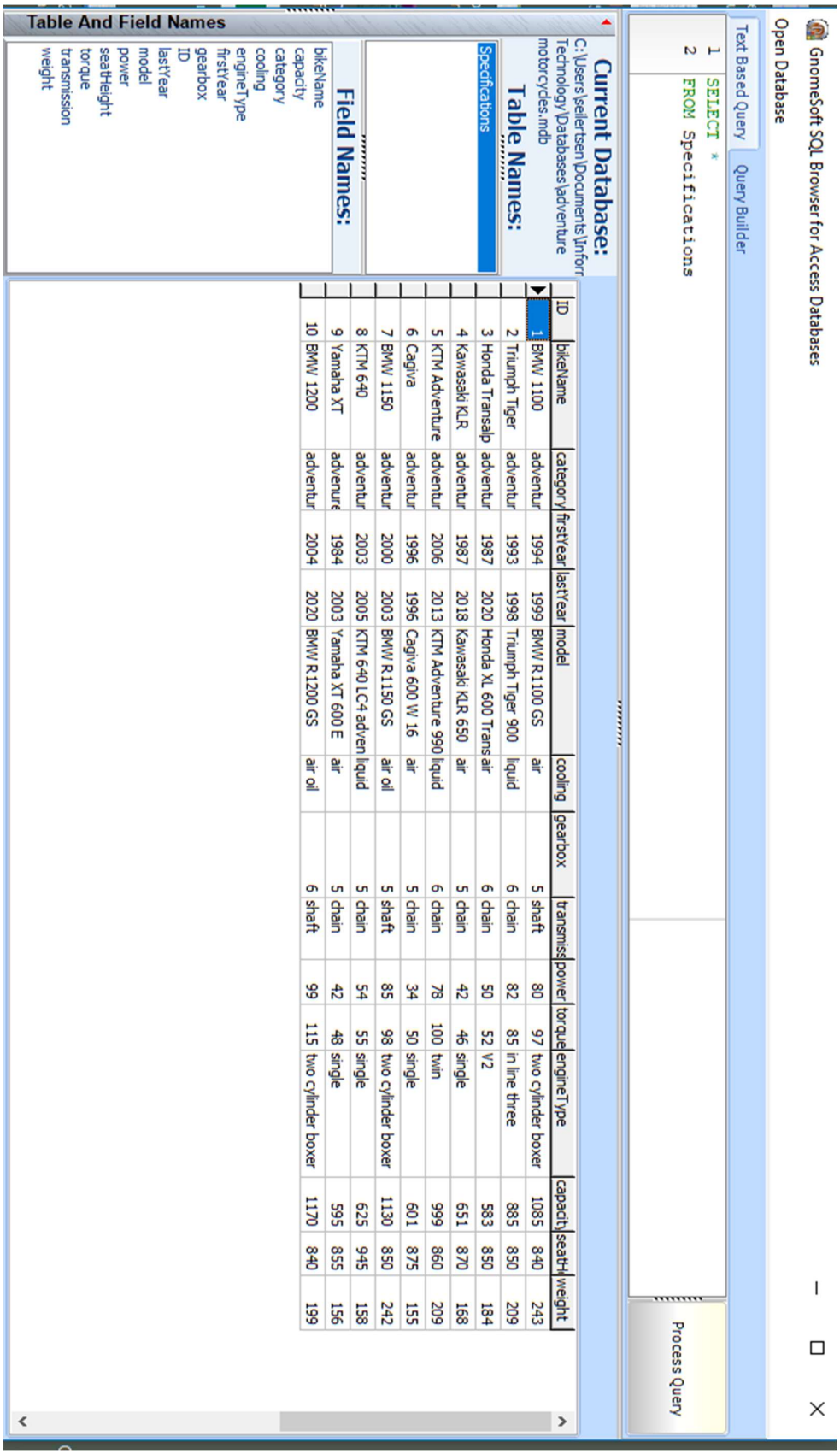
Using the utility program SQLBrowser . . .

- Open the database (done only once)
- Code your SQL queries into the "Text Based Query" area.
- Hit the "Process Query" button.
- Copy and paste your queries into your SQL answer sheet.

**NOTES:**

1. **In order to see ALL the columns in SQLBrowser you need to grab the ones you CAN see and drag them smaller – this brings the hidden ones on the right into view.**
2. **Use SQLBrowser OR Ms Access. You cannot use them both at the same time.**

On the following page you can see a screen shot of what your SQLBrowser should look like once fully operational.




| ID | bikeName | category | firstYear | lastYear | model | cooling | gearbox | transmiss | power | torque | engineType | capacity | seatH | weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | BMW 1100 | adventur | 1994 | 1999 | BMW R1100 GS | air | 5 | shaft | 80 | 97 | two cylinder boxer | 1085 | 840 | 243 |
| 2 | Triumph Tiger | adventur | 1993 | 1998 | Triumph Tiger 900 | liquid | 6 | chain | 82 | 85 | in line three | 885 | 850 | 209 |
| 3 | Honda Transalp | adventur | 1987 | 2020 | Honda XL 600 Trans | air | 6 | chain | 50 | 52 | V2 | 583 | 850 | 184 |
| 4 | Kawasaki KLR | adventur | 1987 | 2018 | Kawasaki KLR 650 | air | 5 | chain | 42 | 46 | single | 651 | 870 | 168 |
| 5 | KTM Adventure | adventur | 2006 | 2013 | KTM Adventure 990 | liquid | 6 | chain | 78 | 100 | twin | 999 | 860 | 209 |
| 6 | Cagiva | adventur | 1996 | 1996 | Cagiva 600 W 16 | air | 5 | chain | 34 | 50 | single | 601 | 875 | 155 |
| 7 | BMW 1150 | adventur | 2000 | 2003 | BMW R1150 GS | air oil | 5 | shaft | 85 | 98 | two cylinder boxer | 1130 | 850 | 242 |
| 8 | KTM 640 | adventur | 2003 | 2005 | KTM 640 LC4 adven | liquid | 5 | chain | 54 | 55 | single | 625 | 945 | 158 |
| 9 | Yamaha XT | adventure | 1984 | 2003 | Yamaha XT 600 E | air | 5 | chain | 42 | 48 | single | 595 | 855 | 156 |
| 10 | BMW 1200 | adventur | 2004 | 2020 | BMW R1200 GS | air oil | 6 | shaft | 99 | 115 | two cylinder boxer | 1170 | 840 | 199 |

**Write a SQL statements for all the following.**

Copy and paste your solutions into a Ms Word file that has **your name** on the first line.

3.1) To view all the records in the database. (2)

3.2) To count the number of records in the database. Call this new aggregate column "Number of Motorcycles". (2)

3.3) To list the motorcycles that were first manufactured in 1994. (2)

3.4) To list all the motorcycle that were first manufactured between 1990 and 2000. List them from oldest to newest. (2)

3.5) To list the models by capacity from smallest to biggest. Also display the motorcycle model as well. (2)

3.6) To determine the average capacity of the motorcycles listed. Call the new aggregate column "Average capacity". (1)

3.7) Select all the fields from the motorcycle that has the largest capacity (only one record must be shown) (2)

3.8) Select all the motorcycle models that were launched in the month of January or February of any year. (3)

3.9) Select all the motorcycle models that were launched in the month of January or February of 1994. (4)

3.10) To show the motorcycle that was on the market for the longest period. (only one record must be shown). (2)

3.11) To calculate the number of years ago when each motorcycle model was launched. (2)

3.12) Find every motorcycle that has the letters "GS" somewhere in the model name. (2)
NOTE: My solution ran in SQLBrowser but did not give any results. When I tried it in Ms Access I got the results I was expecting. This is fine because you need to know that subtle differences do exist in SQL.
TIP: If you are confident of your solution and it runs in SQLBrower without an error message (but does not display any results), you will be fine.

**TOTAL: 90 marks**

**Addendum One**

```
/*
 * Steve Eilertsen. Divides players into 2 teams based on their birth month.
 * Odds versus evens. Each team must have 3 boys and 3 girls.
 * Program allocates a name to each team using a random number.
 * The text names are in a separate text file.
 */
package sixasidebeachtext;

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

public class SixASideBeachText
{
   public static void main(String[] args)
   {
      String name,genderString;
      char gender = 'z';  // char often demands to be initialised to something
      int month = 0;
      boolean monthOdd = false, monthEven=false;
      int teamEvenBoys = 0, teamEvenGirls = 0;
      int teamOddBoys = 0, teamOddGirls = 0;
      String [] evenTeamArray = new String[6];
      String [] oddTeamArray = new String[6];
      int evenTeamCounter = 0;
      int oddTeamCounter = 0;
      String [] teamName = new String[20];

      // Populate the teamName array.
      // Reads in a text file, one name per line, no delimiter
      try {
         Scanner scFile = new Scanner(new File("SixASideBeachText.txt"));
         int size = 0;
         while (scFile.hasNext()) {
            String line = scFile.nextLine();
            teamName[size] = line;
            size++;
         }
      } catch (FileNotFoundException ex) {
         JOptionPane.showMessageDialog(null, "Error! File not found");
         System.exit(0);
      }

      // for debugging purposes
      for (int i = 0; i < 20; i++)
         System.out.println(teamName[i]);

      //INPUT via while loop
      // Loop executes until the teams are full - equal numbers of boys and girls
      while (teamEvenBoys < 3||teamEvenGirls < 3||teamOddBoys < 3||teamOddGirls < 3){
```

```
name = JOptionPane.showInputDialog("Enter your name");

genderString = JOptionPane.showInputDialog("Enter your gender - M/F");
gender = Character.toUpperCase(genderString.charAt(0));

// PROCESSING. Determine odd or even month of birth. Integer only.
month = Integer.parseInt(JOptionPane.showInputDialog(
   "Enter birth month as a number" + "\n" +
   "1 - January" + "\n" +
   "2 - February" + "\n" +
   "3 - March" + "\n" +
   "4 - April" + "\n" +
   "5 - May" + "\n" +
   "6 - June" + "\n" +
   "7 - July" + "\n" +
   "8 - August" + "\n" +
   "9 - September" + "\n" +
   "10 - October" + "\n" +
   "11 - November" + "\n" +
   "12 - December" + "\n"
   ));

// Determines if the month is even or odd using MOD
// There should be erro checking here, ie smaller than 1 and
// bigger than 12.
if (month % 2 == 0){
   monthEven = true;
   monthOdd = false;
} else {
   monthEven = false;
   monthOdd = true;
}

// Allocates into teams based on gender and month of birth
// Teams of 6 - 3 boys and 3 girls
if (monthOdd) {
if (gender == 'M') {
   if (teamOddBoys < 3 ){
      oddTeamArray[oddTeamCounter] = name;
      teamOddBoys++;
      oddTeamCounter++;
   }
}
else if (gender == 'F') {
   if (teamOddGirls < 3 ){
      oddTeamArray[oddTeamCounter] = name;
      teamOddGirls++;
      oddTeamCounter++;
   }
}
} // End monthOdd

if (monthEven) {
if (gender == 'M') {
```

```
            if (teamEvenBoys < 3 ){
               evenTeamArray[evenTeamCounter] = name;
               teamEvenBoys++;
               evenTeamCounter++;
            }
         }
         else if (gender == 'F'){
            if (teamEvenGirls < 3 ){
               evenTeamArray[evenTeamCounter] = name;
               teamEvenGirls++;
               evenTeamCounter++;

            }
         }

         } // End monthEven

         monthEven = false; // NB. Reset the monthEven flag for re-use in loop
         monthOdd = false;  // NB. Reset the monthOdd flag for re-use in loop

      } // end while

   // Generate 2 random numbers from 1 to 20 inclusive for teamName
   // Loop does not allow a duplicate name to be chosen
   int myRandom1 = 0, myRandom2 = 0;
   while(myRandom1 == myRandom2){
      myRandom1 = (int)(Math.random() * (20 + 1) + 1);
      myRandom2 = (int)(Math.random() * (20 + 1) + 1);
   }

   // OUTPUT The even array team with a name from array
   System.out.println(teamName[myRandom1]);
   System.out.println("---------");
   for (int i = 0; i < 6; i++){
         System.out.print(evenTeamArray[i] + " ");

      }
   System.out.println("");
   System.out.println("==================================");

   // OUTPUT The odd array team with a name from the array
   System.out.println(teamName[myRandom2]);
   System.out.println("--------");
      for (int i = 0; i < 6; i++){
         System.out.print(oddTeamArray[i] + " ");

      }

   System.out.println("");

}
}
```