INFORMATION TECHNOLOGY – PAPER I

GRADE 12

JUNE/JULY 2022

Time: 3 hours **Total: 150 Marks**

Examiners: L Coetzee (Section A), H Peuckert (Section B)
Moderators: C Kader (Section A), R Viljoen (Section B), L Bothma (Section A & B)

Memorandum

SECTION A STRUCTURED QUERY LANGUAGE

QUESTION 1

1.1 (4)

```
SELECT *
FROM tblMembers✓
WHERE DateJoined = #1945/10/24#✓
ORDER BY ✓CountryName DESC; ✓
```

1.2 (3)

```
SELECT CountryName✓
FROM tblMembers
WHERE SeatOnCouncil ✓ = Yes✓
```

1.3 (4)

```
UPDATE ✓tblMembers
SET ✓SeatOnCouncil = Yes✓
WHERE CountryID = 'IND45'✓
```

1.4 (4)

```
SELECT ✓DISTINCT CountryName
FROM tblMembers, tblDonations✓
WHERE tblMembers.CountryID = tblDonations.CountryID✓ AND SupportID = "HUM09"✓
```

1.5 (8)

SELECT ~~CountryName,~~ ✓ SUM(Quantity) ✓ AS [No of Howitzers✓]

FROM tblMembers, tblDonations✓

WHERE tblMembers.CountryID = tblDonations.CountryID✓ AND✓ Description LIKE✓

"*Howitzer*"✓**must have both ** on both sides for this mark**

~~GROUP BY tblDonations.CountryID, CountryName✓~~

1.6 SELECT CountryName, GeneralDescription, ✓ correct fields included (8)

FROM tblMembers, tblDonations, tblSupportType ✓✓ all tables

WHERE tblMembers.CountryID = tblDonations.CountryID✓ AND✓ tblDonations.SupportID = tblSupportType.SupportID✓

GROUP BY ✓CountryName, ✓tblSupportType.GeneralDescription

1.7 SELECT GeneralDescription✓, COUNT(*)✓AS ✓ [How Much] correct field name (7)

FROM tblSupportType, tblDonations✓

WHERE tblSupportType.SupportID = tblDonations.SupportID✓ AND tblDonations.SupportID LIKE "HUM*"✓

GROUP BY GeneralDescription✓

1.8 SELECT DISTINCT ✓CountryID (5)

FROM tblDonations

WHERE CountryID ✓NOT IN✓

(SELECT ✓CountryID

from tblMembers✓)

<u>OR</u>

SELECT DISTINCT✓ tblDonations.CountryID✓

FROM tblDonations ✓

LEFT ✓JOIN tblMembers ON tblDonations.CountryID = tblMembers.CountryID

WHERE tblMembers.CountryID IS NULL✓

1.9 INSERT INTO tblDonations✓ (DonationID, CountryID, SupportID, Description, Quantity, QuantityUnit, DateOfDonations) (4)

SELECT DonationID, CountryID, SupportID, Description, Quantity, QuantityUnit, Now()✓

All other fields✓ also accept Date()

FROM tblDonations

WHERE DonationID = 220✓

1.10 DELETE (3)

FROM tblDonations✓

WHERE CountryID = 'LIT91'✓ AND SupportID = 'FINA02'✓

---

50 marks

SECTION B OBJECT ORIENTED PROGRAMMING

QUESTION 2

```
// Question 2.1 - 1

public class Equipment {                    ✓ class header

    // Question 2.2 - 2

    private int deliveryID;                 ✓ typed and named correctly
                                            ✓ properties made private
    private String name;

    private int quantity;


    // Question 2.3 - 2
       ✓ correct header, parameter names and types

    public Equipment(int id, String n, int q) {
        deliveryID = id;
        name = n;                           ✓ fields set to parameters
        quantity = q;
    }


    // Question 2.4 - 2
       ✓ correct header and return types
       ✓ returning correct fields

    public int getID() {
        return deliveryID;
    }

    public int getQuantity() {
        return quantity;
    }


    // Question 2.5 - 2
       ✓ correct header and returning quantity and name fields
       ✓ fields correctly combined and returned as a string

    public String toString() {
        return " * " + quantity + " x " + name;
    }


}
```

QUESTION 3 & 7.2

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;


// Question 3.1 - 1

public class Delivery {                    ✓ class header

    // Question 3.2 - 4

    private int deliveryID;                ✓ all fields private

    private double latitude;

    private double longitude;

    private LocalDate deliveryDate;        ✓ Date object declared

    private Equipment [] equipmentArray; ✓ Equipment array declared
                                                    and array size set to 30

    private String dangerLevel;     ✓ all other fields correctly typed with correct names

    private boolean isDelivered;

    // Question 3.3 - 3
       ✓ constant declared with final / constant
       ✓ named and typed correctly
       ✓ assigned correct values

    public static final String LEVEL_RED = "High Danger";
    public static final String LEVEL_ORANGE = "Considerable Danger";
    public static final String LEVEL_YELLOW = "Moderate Danger";
    public static final String LEVEL_GREEN = "Minor Danger";
    public static final String LEVEL_UNKNOWN = "Unknown Danger Level";

    // Question 3.4 - 7
       ✓ correct header, parameter names and types

    public Delivery(int id, double lat, double lng, LocalDate dd, int dl, char d) {

        deliveryID = id;
        latitude = lat;                        ✓ four fields set to parameters
        longitude = lng;
        deliveryDate = dd;

        if (dl == 4) {                         ✓ if statements to check levels
            dangerLevel = LEVEL_RED;           ✓ constants assigned to dangerLevel
        } else if (dl == 3) {
            dangerLevel = LEVEL_ORANGE;        ✓ nested correctly assigning
        } else if (dl == 2) {                    default value of unknown
            dangerLevel = LEVEL_YELLOW;
        } else if (dl == 1) {
            dangerLevel = LEVEL_GREEN;
        } else {
            dangerLevel = LEVEL_UNKNOWN;
        }
```

Alternative solution:

```
switch (dl) {
    case 4:
        dangerLevel = LEVEL_RED;
        break;
    case 3:
        dangerLevel = LEVEL_ORANGE;
        break;
    case 2:
        dangerLevel = LEVEL_YELLOW;
        break;
    case 1:
        dangerLevel = LEVEL_GREEN;
        break;
    default:
        dangerLevel = LEVEL_UNKNOWN;
}
```

```
    d = Character.toUpperCase(d);
    isDelivered = false;
    if (d == 'Y') {
        isDelivered = true;
    }
}
```

✓ if statements to check levels and assigning correct Boolean value

✓ check for case sensitivity and assigning false if not Y,y,n or N

Alternative:

```
if (d == 'y' || d == 'Y') {
    delivered = true;
} else {
    delivered = false;
}
```

// Question 3.5 - 3

✓ correct header and return types for getID, getLatitude, getLongitude, getDangerLevel and getIsDelivered methods

✓ returning correct fields

```
public int getID() {
    return deliveryID;
}

public double getLatitude() {
    return latitude;
}

public double getLongitude() {
    return longitude;
}

public String getDangerLevel() {
    return dangerLevel;
}

public boolean getIsDelivered() {
    return isDelivered;
}
```

✓ correct header & return type for getDeliveryDate method, returns correct field

```
public LocalDate getDeliveryDate() {
    return deliveryDate;
}
```

```
// Question 3.6 - 2

   ✓ method header correct accepting Equipment array as parameter
   ✓ assigns parameter to field

public void setEquipmentArray(Equipment [ ] equipArr) {
    equipmentArray = equipArr;
}

// Question 3.7 - 1
   ✓ method header and return correct

public void setDangerLevel(String dl) {
    dangerLevel = dl;
}

// Question 3.8 - 5
   ✓ correct header

public String toString() {        ✓ changing format and returning of date
                                  ✓ using correct dd MMM yyyy format
    DateTimeFormatter formatYMMMD = DateTimeFormatter.ofPattern("dd MMM yyyy");

    String formattedDate = formatYMMMD.format(deliveryDate);


      // Question 7.2 - 4

      String outputEquipment = "";    ✓ check if not null - array and element
                                        (both if statements)
      if (equipmentArray != null) {
          for (int i = 0; i < equipmentArray.length; i++) {  ✓ loop through array
              if (equipmentArray[i] != null) {
                  outputEquipment = outputEquipment + equipmentArray[i].toString()
                                    + "\n";
              }                                         ✓ toString of object added

          }
      }


                ✓ return deliveryID, latitude, longitude, dangerLevel and delivered
                ✓ fields correctly combined and returned as a string

    return "(" + deliveryID + ") " + formattedDate
           + " at " + latitude + "," + longitude
           + " in a " + dangerLevel + " Zone"
           + " - " + isDelivered

           + "\n" + outputEquipment;       ✓ returned at end of string
}                                            on a new line
}
```

QUESTION 4, 6.1, 7.1 & 8.1

```
package prelimjuly2022;

import java.io.File;
import java.io.FileNotFoundException;
import java.time.LocalDate;
import java.time.Month;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

// Question 4.1 - 1

public class DeliveriesManager {                        ✓ class header

    // Question 4.2 - 3
      ✓ Delivery array declared with correct name
      ✓ Array size set to 100

    private Delivery[] delivery = new Delivery[100];

    private int size = 0;     ✓ size initialized correctly and properties private

    // Question 4.3 - 11
      ✓ correct header

    public DeliveriesManager() {

        try {                                       ✓ open file correctly
            Scanner scFile = new Scanner(new File("Deliveries.txt"));

            while (scFile.hasNext()) {              ✓ loop through all the lines

                String line = scFile.nextLine();    ✓ read the next line from the file

                String[] data = line.split(";");    ✓ split the line into requested parts
                                                    ✓ create Date object in correct format
                LocalDate delDate = LocalDate.parse(data[3],
                                         DateTimeFormatter.ofPattern("yyyy/MM/dd"));

                  ✓ create delivery object and add delivery to array
                  ✓ using correct conversions for arguments (double, int and char)
                     in correct order
                  ✓ using Date object as an argument (in correct place)

                delivery[size] = new Delivery(Integer.parseInt(data[0]),
                                              Double.parseDouble(data[1]),
                                              Double.parseDouble(data[2]),
                                              delDate,
                                              Integer.parseInt(data[4]),
                                              data[5].charAt(0));

                size++;         ✓ increment size
            }
            scFile.close();
    } catch (FileNotFoundException ex) {          ✓ handle exception
        System.out.println("Cannot find file");
    }

}
```

// Question 4.4 - 4

✓ method header correct and returns String

```
public String allDeliveries() {
    String output = "";

    for (int i = 0; i < size; i++) {         ✓ loop through delivery array

          ✓ add to string (which has been initialized) ✓ toString of object added

        output = output + delivery[i].toString() + "\n";
    }

    return output;
}
```

// Question 6.1 - 5

✓ method header accepts integer as parameter and Delivery data type

```
public Delivery getDeliveryObject(int id) {
    Delivery del = null;

    int i = 0;
    boolean found = false;                    ✓ loop through delivery array

    while (i < size && !found) {              ✓ exit loop when found (using while loop
                                                and Boolean variable; not break)
        if (delivery[i].getID() == id) {      ✓ check id
            del = delivery[i];                ✓ assign and return correct object
            found = true;
        }

        i++;

    }

    return del;
}
```

Alternative solution (for getDeliveryObject method):

```
  public Delivery getDelivery(int deliveryID){
    for (int i = 0; i < size; i++) {              ✓ loop through delivery array
        if (delivery[i].getDeliveryID() == deliveryID) { ✓ check id
            return delivery[i];               ✓ exit loop when found (must use
        }                                       return here and outside loop; not
    }                                           break)
    return null;                                  ✓ assign and return
                                                    correct object
  }
```

// Question 7.1 - 10

```
   ✓ method header

public void populateEquipment() {

    for (int i = 0; i < size; i++) {                ✓ loop through delivery array

        Equipment[] equipment = new Equipment[100];  ✓ create Equipment array

        int pos = 0;

        try {                                          ✓ open file correctly
            Scanner scFile = new Scanner(new File("Equipment.txt"));

            while (scFile.hasNext()) {           ✓ loop through all the lines
                                                   (correct place - inside for loop)

                String line = scFile.nextLine();   ✓ read next line from file

                String[] data = line.split(",");       and split data items

                int id = Integer.parseInt(data[0]);

                if (delivery[i].getID() == id) {  ✓ check deliveryID
                    String name = data[1];
                    int qty = Integer.parseInt(data[2]);  ✓ instantiate object
                                                            with correct arguments
                                                          ✓ add to correct element
                                                            and increase index
                    equipment[pos] = new Equipment(id, name, qty);
                    pos++;
                } //endIf

            } //endWhile

            ✓ set delivery[i] object, using correct argument (Equipment array)
            delivery[i].setEquipmentArray(equipment);

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

    } //endFor

}
```

Alternative solution (for populateEquipment method):

```
public void populateEquipment() {
    try {
        Scanner scFile = new Scanner(new File("equipment.txt"));
        Equipment[] equip = new Equipment[100];
        int count = 0;

        while (scFile.hasNext()) {
            String[] line = scFile.nextLine().split(",");
            equip[count] = new Equipment(Integer.parseInt(line[0]), line[1],
                                         Integer.parseInt(line[2]));
            count++;

        }
        for (int i = 0; i < size; i++) {
            Equipment[] temp = new Equipment[100];
            int tempEquip = 0;
            for (int j = 0; j < count; j++) {
                if (delivery[i].getID() == equip[j].getID()) {
                    temp[tempEquip] = equip[j];
                    tempEquip++;
                }
            }
            delivery[i].setEquipmentArray(temp);
        }
    } catch (FileNotFoundException ex) {
        System.out.println("File not found");
    }
}
```

// Question 8.1 – 18

```
   ✓ method header and returns String, accept two strings as parameters

public String updateDangerLevels(String coordinates, String fromDate) {
    String output = "";

    LocalDate dateFrom = LocalDate.parse(fromDate,
            DateTimeFormatter.ofPattern("yyyy/MM/dd"));  ✓ instantiate Date object

    ✓ extract first coordinate
        ✓ extract top left latitude
        ✓ extract top left longitude

    ✓ extract second coordinate
        ✓ extract bottom right latitude and bottom right longitude

        ✓ converting to real numbers

    int posSemicolon = coordinates.indexOf(";");

    String topLeft = coordinates.substring(0, posSemicolon);

    String bottomRight = coordinates.substring(posSemicolon + 1);

    int posComma = topLeft.indexOf(",");

    double topLeftLat = Double.parseDouble(topLeft.substring(0, posComma));

    double topLeftLong = Double.parseDouble(topLeft.substring(posComma + 1));

    posComma = topLeft.indexOf(",");

    double bottomRightLat = Double.parseDouble(bottomRight.substring(0, posComma));

    double bottomRightLong = Double.parseDouble(bottomRight.substring(posComma + 1));
```

```
        Alternative solution:

        String[] data = coordinates.split(";");

        String[] topLeft = data[0].split(",");

        double topLeftLat = Double.parseDouble(topLeft[0]);

        double topLeftLong = Double.parseDouble(topLeft[1]);

        String[] bottomRight = data[1].split(",");

        double bottomRightLat = Double.parseDouble(bottomRight[0]);

        double bottomRightLong = Double.parseDouble(bottomRight[1]);

    for (int i = 0; i < size; i++) {              ✓ loop through delivery array

        double lat = delivery[i].getLatitude();      ✓ get latitude & longitude

        double longit = delivery[i].getLongitude();

                                               ✓ check range - latitudes
        if ((lat >= bottomRightLat && lat <= topLeftLat)

                                               ✓ check range - longitudes
                && (longit >= topLeftLong && longit <= bottomRightLong)

                                               ✓ check danger level with constant
                && !delivery[i].getDangerLevel().equals(Delivery.LEVEL_RED)

                                               ✓ check delivery date is after parameter
                && delivery[i].getDeliveryDate().isAfter(dateFrom)

                                               ✓ check not delivered
                && !delivery[i].getIsDelivered()) {

            output = output + delivery[i].getID()

                    + "\t" + delivery[i].getLatitude()        ✓ all fields added

                    + "," + delivery[i].getLongitude()        ✓ correct format

                    + "\t" + delivery[i].getDangerLevel()

                    + "\n";

            ✓ set danger Level using constant
            delivery[i].setDangerLevel(Delivery.LEVEL_RED);
        }

    }

    return output;
  }

}
```

QUESTION 5, 6.2, 7.3, 7.4 & 8.2

```
package prelimjuly2022;

    // Question 5.1 - 1
      ✓ application class created with main method

public class LogisticsUI {

    public static void main(String[] args) {

        // Question 5.2 - 1
           ✓ DeliveriesManager object created in appropriate place in the code

        DeliveriesManager dm = new DeliveriesManager();

        // Question 5.3 - 1
           ✓ allDeliveries method called in output statement

        System.out.println("ALL DELIVERIES: \n\n" + dm.allDeliveries());

        // Question 6.2 - 2
            ✓ getDeliveryObject method called in output statement
            ✓ with argument of 463217

        System.out.println("DETAILS OF DELIVERY 463217:\n"
                + dm.getDeliveryObject(463217));

        // Question 7.3 - 1
            ✓ populateEquipment method called correctly (not in output statement)

        dm.populateEquipment();


       // Question 8.2 - 2
            ✓ updateDangerLevels method called in output statement
            ✓ with correct arguments

        System.out.println("ESCALATING TO HIGH RISK: \n\n"
                + dm.upDateDangerLevels("47.12,31.74;46.61,32.98", "30/04/2022"));

        // Question 7.4 - 1

           ✓ allDeliveries method called again in output statement

        System.out.println("ALL EQUIPMENT:\n\n" + dm.allDeliveries());

    }

}
```

100 marks

Total: 150 marks