Diploma Programme
Programme du diplôme
Programa del Diploma

Computer science
Standard level
Paper 2

Monday 8 May 2017 (morning)

1 hour

Instructions to candidates

- Do not open this examination paper until instructed to do so.
- Answer all of the questions from one of the options.
- The maximum mark for this examination paper is **[45 marks]**.

Option	Questions
Option A — Databases	1 – 3
Option B — Modelling and simulation	4 – 6
Option C — Web science	7 – 9
Option D — Object-oriented programming	10 – 11

2217 – 7015

14 pages

Option D — Object-oriented programming

A large company with locations in different cities has taken an OOP approach in creating an administration program that manages all aspects of its business. These aspects include:

- the sale of all of the different products that the company manages
- the salaries for managers, office staff and sales personnel.

10. (a) By making use of an example from the above scenario, distinguish between a class and an instantiation of a class. [3]

The different modules in the program each open up a graphical user interface (GUI). Each GUI has a similar design, but contains differences specific to each module.

(b) By giving **two** examples, explain how the principles of inheritance can be incorporated into the design of this administration program. [4]

(c) Describe how the use of libraries can facilitate the development of programs like this company's administration program. [3]

(Option D continues on the following page)

(Option D continued)

11. The company employs several sales personnel to sell its products to different retailers. Each branch of the company keeps track of its own sales with a suite of programs that include the two classes `SalesPerson` and `Sales`.

```
class SalesPerson // each object contains details of one salesperson
{
  private String id;
  private Sales[] salesHistory; // details of the different sales
  private int count = 0;       // number of sales made

  //constructor for a new salesperson
  public SalesPerson(String id)
  {
    // code missing
  }

  // constructor for a salesperson transferred (together with
  // their sales details) from another branch

  public SalesPerson(String id, Sales[] s, int c)
  {
    // code missing
  }

  public int getCount(){return count;}
  public String getId() {return id;}

  public void setSalesHistory(Sales s)
  {
    salesHistory[count] = s;
    count = count + 1;
  }

  public double calcTotalSales() // calculates total sales for the
                                 // salesperson
  {
    // code missing
  }

  public Sales largestSale() // calculates the sale with the largest
                             // value
  {
    // code missing
  }
 }
```

Each instance variable is initialized when a `SalesPerson` object is instantiated.

(a) Complete the constructor **`public`** `SalesPerson(String id)`, from the `SalesPerson` class. [2]

(b) Explain why accessor methods are necessary for the `SalesPerson` class. [3]

(Option D continues on the following page)

(Option D, question 11 continued)

```
class Sales  // each object contains details of one sale
{
  private String itemId;  // id of the item
  private double value;  // the price of one item
  private int quantity;  // the number of the items sold

  // constructor missing

  public double getValue() {return value;}
  public int getQuantity() {return quantity;}
}
```

(c) (i) Construct unified modelling language (UML) diagrams to clearly show the relationship between the `SalesPerson` and `Sales` classes.
Note: There is no need to include mutator or accessor methods or a constructor. [4]

(ii) Outline a negative effect that a future change in the design of the `Sales` object might have on this suite of programs. [2]

The company employs several sales personnel. The different `salesPerson` objects are held in the array `salesPeople`.

The `Main` class contains various methods that operate on the `SalesPerson` and `Sales` classes. The array `salesPeople` is declared globally. The `Main` class contains the following code:

```
SalesPerson[] salesPeople = new SalesPerson[6];
salesPeople[0] = new SalesPerson("100");
salesPeople[1] = new SalesPerson("101");
salesPeople[2] = new SalesPerson("102");
salesPeople[0].setSalesHistory(new Sales("A100",300.00,10));
salesPeople[0].setSalesHistory(new Sales("A200",1000.00,2));
salesPeople[1].setSalesHistory(new Sales("A300",2550.40,10));
System.out.println(salesPeople[2].getId());
System.out.println(salesPeople[0].getCount());
System.out.println(salesPeople[1].getSalesHistory(0).getValue());
System.out.println(salesPeople[0].calcTotalSales());
```

(d) State the output after running this code. [4]

(e) Construct the method `calcTotalSales()`, in the `SalesPerson` class that calculates the **total value** of the sales for a specific `SalesPerson` object. [5]

The `salesPeople` array contains 100 instantiated objects.

The company wishes to reward the salesperson whose sales have the largest total value.

(f) By making use of any previously written methods, construct the method `highest()`, that returns the ID of the salesperson whose sales have the largest total value. [5]

(Option D continues on the following page)

(Option D, question 11 continued)

(g) Construct the method `addSales(Sales s, String id)`, in the `Main` class, that will add a new `Sales` object `s`, to the salesperson with a specified ID.
Note: You can assume that the ID is a valid one. [4]

A further class in this suite of programs is the `Payroll` class. This class is run at the end of each month to calculate each salesperson's salary, which is based on the sales that have been made during that month.

(h) Suggest changes that must be made to the `SalesPerson` class and/or the `Sales` class to allow these calculations to be made. [3]

(i) Discuss the use of polymorphism that occurs in this suite of programs. [3]

End of Option D