Diploma Programme
Programme du diplôme
Programa del Diploma

Computer science
Standard level
Paper 2

Monday 2 November 2020 (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 – 12

8820 – 7015

Option D — Object-oriented programming

10. A company provides car parking services in large cities and needs a computer system to keep track of the number of vehicles using its parking areas.

When a vehicle arrives, its registration plate is recorded on the system and it is allocated a number that identifies where it should park. When the vehicle leaves, that space is made available for other vehicles to use.

Vehicles are identified by their unique registration plate, which is an alphanumeric code of eight characters (*eg* `X1234567`). This is clearly displayed on the vehicle.

A programmer created the classes `ParkingArea` and `Vehicle` to model the above situation.

```
public class ParkingArea {
    private Vehicle vehicles[];
    private String name;

    ParkingArea(String name, int capacity) {
        this.name = name;
        if (capacity > 300) capacity = 300;
        this.vehicles = new Vehicle[capacity];
    }

    String getName() {
        return name;
    }

    public int getCapacity() {
        return vehicles.length;
    }

    public int findVehicle(String reg) {
        //find where the vehicle is located in the array and
        //return the index not yet written
    }
}
```

(Option D continues on the following page)

(Option D, question 10 continued)

```
public  class Vehicle {
    private String registration;
    private byte colour;
    private boolean broken;

    public final static byte BLACK=1;
    public final static byte WHITE=2;
    public final static byte BLUE=3;
    public final static byte RED=4;
    public final static byte GREEN=5;
    private final static double ADMIN_FEE = 3;

    public Vehicle() {}

    public Vehicle(String registration) {
        this.registration = registration;
    }
    public Vehicle(String registration, byte colour) {
        this.registration = registration;
        this.colour=colour;
    }
    public void setBroken(boolean broken) {
        this.broken=broken;
    }
    public void setColour(byte colour) {
        this.colour=colour;
    }
    public boolean getBroken() {
        return broken;
    }
    public String getRegistration() {
        return registration;
    }
    public double pay(int hours) {
        // code to return admin fee - only if applicable
    }
}
```

(a) Outline **one** effect of using the modifier `static` when declaring a variable.

[2]

(b) Describe the relationship between the classes `Vehicle` and `ParkingArea`.

[3]

(Option D, question 10 continued)

(c) Outline why it is necessary to use the keyword `this` in the `setBroken` method of the `Vehicle` class.

[2]

(b) (i) Construct code to create an instance of the `Vehicle` class that has a registration of `X1234567`.

[2]

(ii) Construct code that sets the colour of the object created in part (i) as black.

[2]

(Option D continues on the following page)

(Option D continued)

11. (a) Construct the method `addVehicle(Vehicle v)` that will add a vehicle to the first empty position of the array `vehicles[]` and return the position (*ie* the index of the array) at which it has added the car. If it is not possible to fit the vehicle into the array then it should return `-1`.

[6]

(b) Outline **two** differences between inheritance and aggregation.

Two further classes, `Car` and `Motorbike`, are created.

```
public class Car extends Vehicle{
    public static double hourlyFee=3.5;
    public double pay(int hours) {
        //code to calculate and return the complete price
    }
}

public class Motorbike extends Vehicle{
    public static double hourlyFee=2.5;
    public double pay(int hours) {
        //code to calculate and return the complete price
    }
}
```

[4]

(Option D continues on the following page)

(c) Construct a UML diagram that shows the relationships between the `ParkingArea`, `Vehicle`, `Motorbike` and `Car` classes. There is no need to include the attributes or methods of each class.

[4]

The method `pay` in the `Vehicle` class returns the administration fee (which is only part of the total price), while the method `pay` of the `Car` class calculates the total price for a car staying in the parking area.

(d) (i) Construct the method `pay` in the `Vehicle` class that returns the admin fee stored in the variable `AdminFee` if the vehicle has stayed for five hours or less; otherwise, it returns 0.

[2]

(ii) Construct the method `pay` in the `Car` class, where it uses the `vehicle` method `pay` but adds the charge for the amount of time spent in the parking area.

[2]

The array `vehicles[]` in the `ParkingArea` class is used to store instances of the `Car` or `Motorbike` class.

(e) Outline why `Vehicle` is a valid type for this array.

[2]

(Option D continues on the following page)

(Option D continued)

12. The management of the company will launch a new scheme to give every 50th car driver and every 60th motorcyclist a free coffee voucher. The code for printing this voucher has already been created and is activated by calling the static method `Vouchers.printCoffeeVoucher()`.

A `getKind()` method has already been added to the `Vehicle` class, which returns a `char` value indicating whether it is a car (`c`) or a motorbike (`m`).

(a) Describe, **without writing code**, any changes required to the `addVehicle` method and the `ParkingArea` class to make the new voucher scheme work.

One test performed on the finished code was defined as follows:

Test data	Vouchers printed
29 cars	0
130 motorbikes	2

[5]

(b) Identify **three** other tests you might perform on the completed code to prove that it functions correctly.

[3]

(Option D continued)

The `removeVehicle` method of the `ParkingArea` class searches in the array for a `Vehicle` object with a specified registration plate, then removes it by setting that array index to `null`.

The method returns a reference to the `Vehicle` object that has been removed from the array, or `null` if no matching registration plate was found.

(c) Construct the `removeVehicle` method.

[6]

End of Option D