BCS THE CHARTERED INSTITUTE FOR IT

BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 5 Diploma in IT

OBJECT-ORIENTED PROGRAMMING

Friday 17th April 2026 – Afternoon

Answer **any** FOUR questions out of SIX. All questions carry equal marks.

Time: TWO hours

Answer any Section A questions you attempt in Answer Book A
Answer any Section B questions you attempt in Answer Book B

The marks given in brackets are **indicative** of the weight given to each part of the question.

Calculators are **NOT** allowed in this examination.

This page is intentionally blank.

Section A
Answer Section A questions in Answer Book A

A1.

Consider the following classes that will be used in the implementation of a bank computer system:

class bankAccount
balance: Real isActive: Boolean
withdraw(amount: Real) deposit(amount: Real)

class Transaction
amount: Real
execute()

A **Transaction** is associated with a **bankAccount**. It modifies the `balance` attribute of the associated **bankAccount** using the `execute()` operation, provided that the account is set to **active**.

a) Write an Object Constraint Language (OCL) statement to ensure that the `balance` of an object of **bankAccount** is always non-negative.

(10 marks)

b) Write an OCL precondition for `withdraw()` in the **bankAccount** class to ensure that a withdrawal can happen only when an active account contains a balance greater than or equal to the withdrawal amount.

(7 marks)

c) Write an OCL postcondition for the deposit operation on **bankAccount**, ensuring that after a deposit, the balance is increased by the amount of the deposit.

(8 marks)

[Turn Over]

A2.

Ad-hoc polymorphism and parametric polymorphism are two important concepts in object-oriented programming.

a) Define ad-hoc polymorphism and parametric polymorphism, providing a code example of **each**.

(10 marks)

b) Discuss the key differences between ad-hoc polymorphism and parametric polymorphism in terms of:

i. Code reuse.

(5 marks)

ii. Code flexibility.

(5 marks)

iii. The compilation process.

(5 marks)

A3.

a) The SOLID principles are key guidelines for designing flexible, maintainable and scalable object-oriented systems. Explain **each** of the **five** SOLID principles.

(10 marks)

b)

i. Define black box and white box testing.

(5 marks)

ii. Discuss their respective advantages and disadvantages.

(5 marks)

iii. Provide examples of when it is most appropriate to use **each** type of testing in the software development lifecycle.

(5 marks)

Section B
Answer Section B questions in Answer Book B

B4.

The Very Large Security Conference (VLSC) is an annual international conference that is held every spring in a different country. Each year, a call for papers invites academics and industrialists to write papers for the conference.

Authors write their papers and submit them to the conference for review. The conference has a Programme Chair who appoints the reviewers, called the Programme Committee.

A subset of the Programme Committee are chosen to be Regional Chairs. After the closing date for submissions, the Regional Chairs assign each paper from their region a unique number and then send it out to two reviewers. Each reviewer will review their allocated papers, by providing a score for each paper and comments for the author(s).

The Regional Chair will collate the reviewers' comments and will produce a list of papers, ordered by their score, which will be sent to the Programme Chair. The Programme Chair makes the final decision on the papers. Those with high scores will be accepted for the conference, whereas papers with a low score will be rejected outright.

Depending on the reviewers' comments, a further decision will be made as to whether a paper needs to be revised by the authors or can be accepted for the conference without changes. If a paper needs revisions, the authors have to resubmit a new version within a month's deadline.

When the required number of papers have been accepted for the conference, the Programme Chair will produce a conference programme.

a) Draw a use case diagram for the VLSC.

(15 marks)

b) Discuss how use case diagrams and scenarios contribute to the development of a system. Within your answer, include an example scenario from the conference example.

(10 marks)

[Turn Over]

B5.

a) Genealogy looks at the background, or origin, of things.

Discuss what features from structured and procedural languages have contributed to the development of object-oriented languages, indicating how the features have been improved in object-oriented languages. Include examples of these features using code or a diagram.

(10 marks)

b)

i. Explain what coupling and cohesion mean and discuss how they contribute to the quality of a program. Include examples to illustrate your answer.

ii. Discuss in detail how encapsulation helps a programmer produce good quality code when measured against the concepts of coupling and cohesion.

(15 marks)

B6.

a) You have been asked to maintain an existing system and have been presented with the object-oriented code below, which represents a subset of an airline system, showing the relationship between planes, flights and the main pilot (captain).

To aid your understanding of the system, draw a class diagram to represent this information:

```
class Plane{
   static int noOfPlanes = 0;
   String tailNo;
   String planeType;
   int noOfSeats;

   public Plane(String aTailNo, String aPlaneType, int
aNoOfSeats){
         noOfPlanes++;
         tailNo = aTailNo;
         planeType = aPlaneType;
         noOfSeats = aNoOfSeats;
   }
} // Plane

class Pilot{
   int pilotNo;
   String name;
   int available_flight_hours;

   public Pilot(int aPilotNo, String aName, int
availableHours){
         pilotNo = aPilotNo;
         name = aName;
         available_flight_hours = availableHours;
   }
```

```
   public int getAvailableHours() {
        return available_flight_hours;
   }
} // Pilot

import java.util.Date;

class Flight {
   String flightNo;
   String departureAirport;
   String arrivalAirport;
   Date departureTime;
   int noOfHours;
   Plane planeAllocated;
   Pilot captain;

   public Flight(String aFlightNo, String departure, String
arrival, Date aDate, int aHours) {
        flightNo = aFlightNo;
        departureAirport = departure;
        arrivalAirport = arrival;
        departureTime = aDate;
        noOfHours = aHours;
        }

   public void allocatePlane(Plane aPlane) {
        this.planeAllocated = aPlane;
        }

   public void allocateCaptain(Pilot aPilot) {
        int hours;
        hours = aPilot.getAvailableHours();
        if (hours > noOfHours)
             System.out.println("ERROR: Pilot does not have
available hours to fly");
        else
             this.captain = aPilot;
        }
} //Flight
```

You can assume that other appropriate *setter* and *getter* methods have been defined but are not fully documented here and do not need to be included in the class diagram.

(15 marks)

b) Draw **two** object interaction diagrams: one to represent a valid instantiation of at least **two** of the above classes and one to show an invalid one. Explain why the diagram is valid or invalid.

(4 marks)

c) Explain the difference between typed and untyped languages. In your explanation, include an example of how a variable can be defined in **each** language.

(6 marks)

END OF EXAMINATION