

# Examination (with solutions)

## Object-oriented programming

*Course code: DIT044*

<i>Date:</i>	2025-08-19
<i>Questions:</i>	4
<i>Results:</i>	Posted within 15 working days.
<i>Allowed aids:</i>	No aids (books or calculators) are allowed.

Please observe carefully the following:

- Write in legible English (illegible translates to “no points”).
- Motivate your answers, and clearly state any assumptions made.
- Start each task on a new sheet and write on only one side of the paper.
- Before handing in your exam, number and sort the sheets in task order.
- Write your anonymous code and page number on every page.
- The exam is anonymous, do not leave any information that would reveal your name on your exam sheets.
- For all class diagrams in your exam, write all methods and attributes relevant to your code, including constructors, getters and setters.

Not following these instructions will result in the deduction of points.

## Question 1 (20pt)

1.1. Describe the relationship between these two OOP concepts. (10pt)

1. Class methods
2. Encapsulation


Solution: 5pt for correct answer (relationship explanation makes sense, 0 points if these are independent definitions), 5pt for using right terms (e.g., method, parameters, initialize, etc.).

1.2. Explain two types of errors in Java and why they occur. (10pt)

Solution: 5pt for mentioning real errors (of any type, any level), 5pt for saying whether they are compilation or runtime errors and what does that mean

## Question 2 (30pt)

2.1. Write the output of this program (i.e., running Tab) and answer whether we could add sparkling water to the tab. (10pt)



```
1 import java.util.LinkedList;
2
3 public class Tab {
4     static LinkedList<FoodItem> currentOrder;
5
6     public static void main(String[] args) {
7         currentOrder = new LinkedList<FoodItem>();
8
9         currentOrder.add(new BurrataSalad());
10        currentOrder.add(new PatatasBravas());
11        currentOrder.add(new PimientosPadron());
12        currentOrder.add(new ColdBrewLemonade());
13
14        for (FoodItem foodItem : currentOrder) {
15            System.out.println(foodItem.getName());
16        }
17    }
18 }
```

```
Tab.java FoodItem.java +
1 public interface FoodItem {
2     String getName();
3 }
4
5 class Salad implements FoodItem {
6     public String getName() { return "Salad"; }
7 }
8
9 class BurrataSalad extends Salad implements FoodItem {
10     public String getName() { return super.getName() + ": Burrata Salad"; }
11 }
12
13 class Tapa implements FoodItem {
14     int ID = 0;
15     void increaseID() { ID += 1; }
16     public String getName() { return "Tapa"; }
17 }
18
19 class PatatasBravas extends Tapa implements FoodItem {
20     public String getName() {
21         super.increaseID();
22         return super.getName() + " " + String.valueOf(ID) + ": Patatas Bravas";
23     }
24 }
25
26 class PimientosPadron extends Tapa {
27     public String getName() {
28         super.increaseID();
29         return super.getName() + " " + String.valueOf(ID)
30             + ": Pimientos del Padrón";
31     }
32 }
33
34 abstract class Drink {
35     public String getName() { return "Drinks"; }
36 }
37
38 class Lemonade extends Drink {}
39
40 class ColdBrewLemonade extends Drink implements FoodItem {
41     public String getName() { return super.getName()
42         + ": Cold Brew Lemonade"; }
43 }
44
45 class SparklingWater extends Drink {
46     public String getName() { return super.getName() + ": Sparkling Water"; }
47 }
```

Solution: 2pt per correct line + 2pt for something like “SparklingWater does not extend FoodItem and therefore can’t be added to the List.”

Salad: Burrata Salad

Tapa 1: Patatas Bravas

Tapa 1: Pimientos del Padrón

Drinks: Cold Brew Lemonade

**2.2. What OOP concepts are used to be able to store instances of different classes (e.g., salads, bravas, etc.) in the list called *currentOrder* in the code in exercise 2.1? (10pt)**

**Solution: 5pt for polymorphism and explanation, 5pt for interfaces and explanation**

**2.3. Can you spot any errors in the code in exercise 2.1? What would you change? (10pt)**

**Solution: 5pt for “some classes do not use the interface” plus explanation, 5pt for the wrong ID increase plus explanation (the class is always initialised at ID = 0, it is not global)**

## Question 3 (30pt)

**3.1. We created a social network app for artists as a fun side project. Our app has room for different types of content, and all users should be able to access all types of content. Whenever a new art piece is published, it reaches all the subscribed users, that receive a notification. Users are (of course) able to subscribe to and unsubscribe from the feed of a particular artist at any time. What design pattern did we use? Why? (10pt)**

**Solution: observer (5pt), explanation (5pt)**

**3.2. When looking for strategies to simplify complex software, we often use the facade pattern (one of the 23 “GoF”). The advantages of the facade design pattern are obvious; however, the use of the facade design pattern has some disadvantages. Write about the advantages and disadvantages of using the façade design pattern. (20pt)**

**Solution: 5pt per each explicitly stated advantage (“simplify complex software”, minimize complexity of sub-systems, support loose coupling, flexible and easily expandable software) with explanation, 5pt per explicitly stated disadvantage (complex implementation, difficult to integrate with pre-existing code, additional level of abstraction, high degree of dependence at facade interface, etc.) with explanation. Max 20pt.**

## Question 4 (20pt)

**4.1. Draw a UML diagram representing the relationship between the classes in exercise 2.1 (the one with food items). (10pt)**

**Solution: 10pt for all correct (not a single mistake), 5pt for some elements missing or incorrect, 0pt for more than half of the elements missing or incorrect, or completely unrecognizable UML diagrams**

**4.2. Cohesion and coupling are used to measure our code quality and ensure it is maintainable and scalable. If classes and modules are heavily dependent on each other, what can we say about cohesion and coupling? What do these indicators tell us? (10pt)**

**Solution: 5pt mentioning that in tight coupling, classes and modules are heavily dependent on each other, 5pt for explaining that changes in one module can significantly impact the functionality of others. It makes software complex and difficult to maintain, extend, and test.**