

Object-Oriented Programming & Design  
Practice Work 4  
**2 points + 1 bonus**

**Instructor:** Izbassar Assylzhan

**Deadline:** 8<sup>th</sup> week, in your practice time. The last defense on week 9. Please, read the class & interface requirements carefully. Recall about naming conventions. Keep your classes & interfaces separately. Use the proper structured packages.

## Problem 1

**Total points:** 0,2.

Write short, concise answers to the following questions in your documentation:

1. What is the core difference between a class and an interface?
2. Can interfaces have fields? If yes, what are the implicit modifiers for them?
3. Can a class implement multiple interfaces? Explain why this is useful.

## Problem 2

**Total points:** 0,6.

Define an interface `CanHavePizza` with a method `void eatPizza()`. Implement the following:

- Create a class `Cat` that implements `CanHavePizza`, printing a message when eating.
- Create a class `Person` that does **not** implement this interface.
- Create a class `Student` that extends `Person` and implements `CanHavePizza`, `CanHaveRetake`, and `Movable`. Use dummy print statements for the methods (e.g., ‘eatPizza’, ‘retakeExam’, ‘dance’, ‘move’).

Create a class `Restaurant` with a method `servePizza(CanHavePizza eater)`.

---

```
1 boolean servePizza(CanHavePizza eater) {  
2     eater.eatPizza();  
3     if (eater instanceof Person) {  
4         // process payment  
5     }  
6     return true;  
7 }
```

---

Inside this method, call `eater.eatPizza()` and use the `instanceof` operator to check if the eater is a `Person`; if so, print a “Processing payment” message. In your `Main` class, demonstrate this by passing both a `Cat` and a `Student` to the restaurant.

## Problem 3

**Total points:** 0,6.

Demonstrate interface hierarchies and multiple inheritance:

- Define interface Game (methods a(), b(), c()) and an interface IGame that extends Game (adding method d()). Implement these in LogicGame and MemoryGame. Create a method getStatistics(Game g) in an App class to show polymorphism.
- Create interfaces Sellable and Pluggable. Create a third interface SellableAndPluggable that extends both, and implement it in a class iPhone.

## Problem 4

**Total points:** 0,6.

Create a class Student with fields name and gpa.

1. Implement Comparable<Student> to provide a default sort by gpa.
2. Create a separate class NameComparator that implements Comparator<Student> to sort by name.
3. In Main, create a List of students and demonstrate sorting using Collections.sort(list) and Collections.sort(list, new NameComparator()).

## Bonus Task

**Total points:** 1.

Implement the **Counting Sort** algorithm for an array of integers (range 0 to 10). The output should display the sorted array based on the frequency of the elements.

**Package Structure:**

- pr4.interfaces.model (Interfaces and data classes)
- pr4.interfaces.services (Logic classes like Restaurant/App)
- pr4.interfaces.main (Main entry point)

**Hint:** Remember that an interface can extend multiple interfaces, but a class can only extend one class. Use this to your advantage when designing the SellableAndPluggable hierarchy.