https://github.com/alina-puscasu/pao_lab_2022

# Programare avansata pe obiecte – laborator 5

***Alina Puscasu*** *alina.puscasu@endava.com*

## Clase abstracte si interfete

Folosim o **clasa abstracta** atunci cand vrem sa:

- Implementam doar unele metode din clasa
- Reutilizam o serie de metode si membri din aceasta clasa in clasele derivate
- Nu vrem sa instantiem clasa

Particularitati:

- Putem avea metode/date membre cu **orice** modificator/non modificator de acces

Folosim **interfete** atunci cand vrem sa:

- Avem o descriere a structurii fara implementari
- Metodele sunt implicit public
- Definim un contract intre clase

Particularitati interfete:

- Putem crea folosind cuvantul cheie: **interface**
- Pentru a defini o clasa conforma cu o interfata folosim cuvantul cheie **implements**
- Pentru a defini o interfata care mosteneste alta interfata folosim cuvantul cheie **extends**
- Putem avea **campuri**, iar acestea sunt in mod implicit **static** si **final**
- Combinarea unor interfete care contin o metoda cu acelasi nume e posibila doar daca metodele nu au tipuri intoarse diferite si aceeasi lista de argumente. Este preferabil ca in interfete care trebuie combinate sa nu existe metode cu acelasi nume, pentru a evita confuziile.
- Inainte sa folosim o interfata ne trebuie o clasa care sa o implementeze, ele **nu pot fi instantiate**
- **Putem avea metode** fara implementare (public si abstract), default (vizibilitate public, apar din java 8), statice (vizibilitate public, apar din java 8), cu vizibilitate private (din java 9, statice sau nestatice)

> **Abstract classes should be used primarily for objects that are closely related, whereas interfaces are best suited for providing a common functionality to unrelated classes**

## Comparator si Comparable

- Interfete folosite pentru sortare
- Pentru a folosi **Comparable** clasa trebuie sa implementeze aceasta interfata, fiecare clasa care face asta putand defini **un criteriu** de sortare implementand metoda **compareTo**

https://github.com/alina-puscasu/pao_lab_2022

- Pentru a folosi **Comparator**, cream o clasa separata care implementeaza interfata si prin urmare metoda **compare** in care definim criteriul de sortare dorit.

Clasele pot defini **mai multe criterii** de sortare.

## Exercitii

1. Declarati o interfata Task care contine o metoda execute(), care returneaza void. Pe baza acestei interfete implementati 3 clase: RandomTask, OutTask si CounterOutTask.

   - Pentru OutTask afisati un mesaj in consola, mesaj specificat in constructor
   - Pentru RandomTask generati un numar aleator si afisati un mesaj cu el. Generarea se face in constructor
   - Pentru CounterOutTask, incrementati un contor global si afisati-i valoarea dupa fiecare incrementare

   Creati o noua clasa Container in care puteti adauga si elimina elemente.

2. Declarati o clasa Album care are campurile: nume, artist, rating si anul publicarii.

   - Sortati un array de albume pe baza numelui, rating-ului si anului publicarii. Folositi ambele interfete de comparare.
   - Creati o clasa Main unde declarati array-ul si afisati-l inainte si dupa sortare.

3. Creati 4 interfete Minus, Plus, Mult si Div care contin cate o metoda aferenta numelui si are ca argument un numar de tipul float. Declarati o clasa Operation care sa le implementeze si care are un camp de tip float, modificat de metodele implementate de voi.