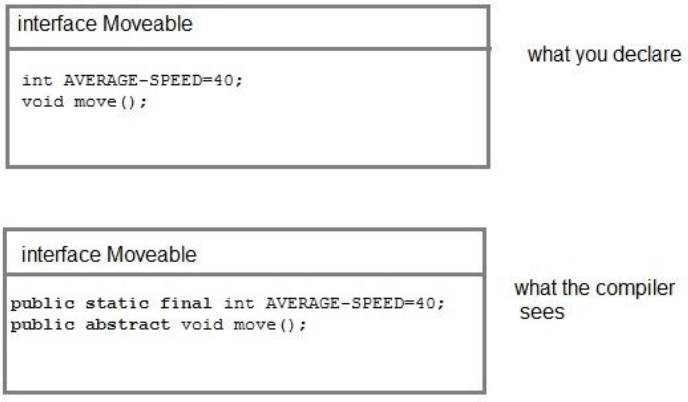
Laborator 5

Interfeţe:

- Sunt folosite pentru a definii comportamentul unei clase.
- O interfata poate fi considerată o “clasa abstracta pură”, deoarece ne lasă sa stabilim o “forma” pentru o clasa (numele metodelor, lista de argumente, valori întoarse), dar fără **nicio implementare.**
- Interfața este folosită pentru a descrie un **contract** între clase: o clasă care implementează o interfață va implementa metodele definite în interfață.
- Metodele declarate în interfaţa sunt în mod implicit public
- O clasa implementează o interfata folosind keyword-ul **implements**. Şi spre deosebire de mostenire, o clasa poate implementa mai multe interfeţe
- O interfata poate să extindă mai multe interfeţe.

```
public interface A {

    void afisareA();
}
```



- Campurile conţinute de o interfata sunt implicit **static** si **final**.
- Iar metodele din interfaţa sunt implicit **public.**

```java
public interface Alarm {

    void setAlartm();

    default String turnAlarmOn() {
        return "Turning the alarm on.";
    }

    default String turnAlarmOff() {
        return "Turning the alarm off.";
    }

    static int getHorsePower(int rpm, int torque) {
        return (rpm * torque) / 5252;
    }

}
```

Interfete folosite pentru sortare:

- **Comparable**

```java
public class Persoana implements Comparable<Persoana> {

    String nume;

    int varsta;

    public Persoana(String nume, int varsta) {
        this.nume = nume;
        this.varsta = varsta;
    }

    @Override
    public int compareTo(Persoana p) {
        if (this.varsta == p.varsta) {
            return 0;
        } else if (this.varsta > p.varsta) {
            return -1;
        } else {
            return 1;
        }
    }
```

- **Comparator**

```java
public class SortByName implements Comparator<Student> {

    @Override
    public int compare(Student a, Student b) {
        return a.name.compareTo(b.name);
    }
}
```

Probleme:

1. Declaraţi o interfata *Task* care contine o metoda **execute**(), tipul returnat al metodei este void. Pe baza interfeţei respective, o sa implementati 3 clase: **RandamTask**, **RandomOutTask** si **CounterOutTask**.
    - pentru **OutTask**, implementaţi un task care sa afişeze un mesaj în fluxul de ieşire. Iniţializaţi mesajul în constructor.
    - pentru **RandomOutTask**, implementati un task care genereaza un număr aleator şi afişează un mesaj cu numărul generat la output. Generarea se va face în constructor. (folositi-va de metodele clasei Math.class pentru generare)
    - pentru **CounterOutTask**, implementaţi un task care incrementeaza un contor global şi afişează valoarea contorului după fiecare incrementare.
2. Declaraţi o clasă Album care are campurile nume, anul publicării şi rating.
    - implementaţi metodele clasei : constructori, getteri, setteri
    - folosiţi una din cele două interfeţe de comparare (Comparable/Comparator) pentru a sorta o colecţie de albume pe baza numelui şi al ratingului.
    - declarati-va o clasa Main în care sa creaţi o lista de albume şi sa afisati colecţia înainte şi după sortare.