Révision POO

Dr F.Guerroudji

2023/2024

Concepts Fondamentaux

- **Classes et Objets :** Les classes sont des modèles pour créer des objets, qui sont des instances de ces classes. Les objets ont des caractéristiques (attributs) et des comportements (méthodes).
- **Encapsulation :** consiste à regrouper les données et les méthodes qui agissent sur ces données dans une seule entité, l'objet. Cela permet de cacher les détails d'implémentation et de protéger les données sensibles.
- **Héritage :** permet à une classe (appelée classe fille ou sous-classe) d'hériter des caractéristiques et des comportements d'une autre classe (appelée classe parent ou superclasse). Cela favorise la réutilisation du code et la hiérarchisation des concepts.
- **Polymorphisme :** permet à un objet de prendre plusieurs formes. Cela peut se manifester à travers le polymorphisme de sous-typage (héritage) ou le polymorphisme de paramétrage (interfaces et surcharge de méthodes).
- **Abstraction :** consiste à définir une interface pour un ensemble de fonctionnalités sans se soucier des détails d'implémentation. Les classes abstraites et les interfaces sont des outils pour créer des abstractions en Java.
- **Modularité :** consiste à diviser un système en modules indépendants et interconnectés. En Java, cela peut être réalisé à l'aide de packages et de la visibilité des membres (public, private, protected).

Classes et Objets

```
public class Animal {
    // Attributs
    String nom;
    int age;

    // Constructeur
    public Animal(String nom, int age) {
        this.nom = nom;
        this.age = age;
    }

    // Méthode
    public void parler() {
        System.out.println("Je suis un animal.");
    }
}

// Création d'objet
Animal monAnimal = new Animal("Léo", 5);
```

Encapsulation

```
public class CompteBancaire {
    private double solde;

    public double getSolde() {
        return solde;
    }

    public void deposer(double montant) {
        solde += montant;
    }

    public void retirer(double montant) {
        if (solde >= montant) {
            solde -= montant;
        } else {
            System.out.println("Solde insuffisant.");
        }
    }
}
```

Héritage

```
public class Chien extends Animal {
    // Constructeur
    public Chien(String nom, int age) {
        super(nom, age);
    }

    // Méthode
    public void aboyer() {
        System.out.println("Wouaf wouaf !");
    }
}
```

Classe abstraite

```
public abstract class Figure {
    abstract double calculerAire();
}

public class Carre extends Figure {
    double cote;

    @Override
    double calculerAire() {
        return cote * cote;
    }
}
```

Interface

```
public interface Forme {
    double calculerAire();
}
public class Rectangle implements Forme {
    double longueur;
    double largeur;
    @Override
    public double calculerAire() {
        return longueur * largeur;
    }
}
public class Cercle implements Forme {
    double rayon;

    @Override
    public double calculerAire() {
        return Math.PI * rayon * rayon;
    }
}
```

Bonne Continuation