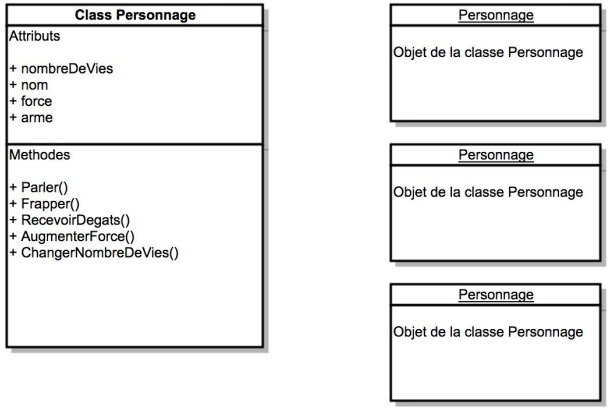
Concepts et structures avancées

1. Les classes en javacript

Notion de classe

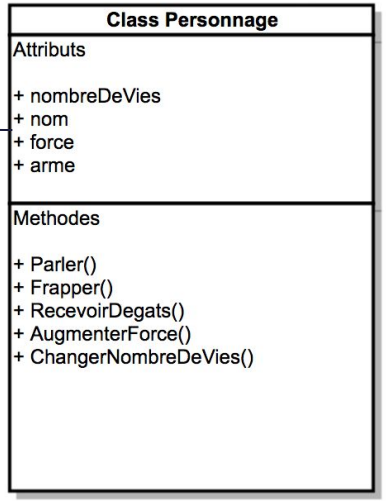
- Une classe est un plan qui nous permet de créer plusieurs objets à partir de ce plan.
- Exemple : Plusieurs personnages d'un jeux vidéo. On parle alors d'instances de la classe Personnage.



Une classe et ses attributs

- Une classe contient aucun ou plusieurs attributs qui vont définir les propriétés de cette même class

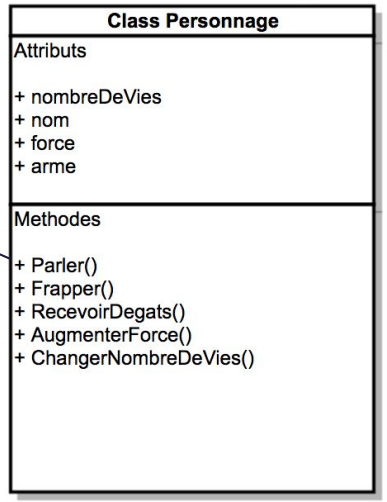
La classe personnage possède 4 attributs



Une classe et ses méthodes

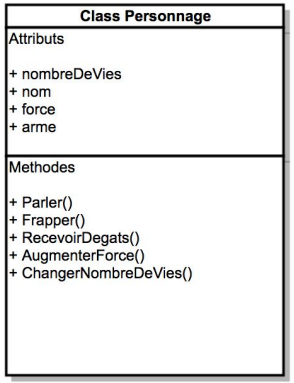
- Une classe contient aucune ou plusieurs méthodes (également appelées fonctions) qui vont définir des comportement pour cette même classe

La classe personnage possède 5 méthodes



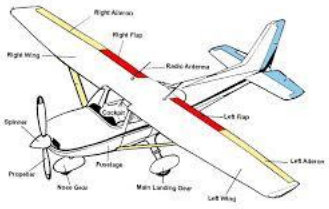
Comment créer une classe

- **Nom** : c'est quoi ? Employé, compte bancaire, joueur, document, album...
- **Attributs** : ce qui la décrit. Largeur, hauteur, couleur, type de fichier, score… On les appel aussi des propriétés.
- **Comportements** : que peut elle faire ? Jouer, ouvrir, chercher, enregistrer, imprimer… On les appel le plus souvent des méthodes

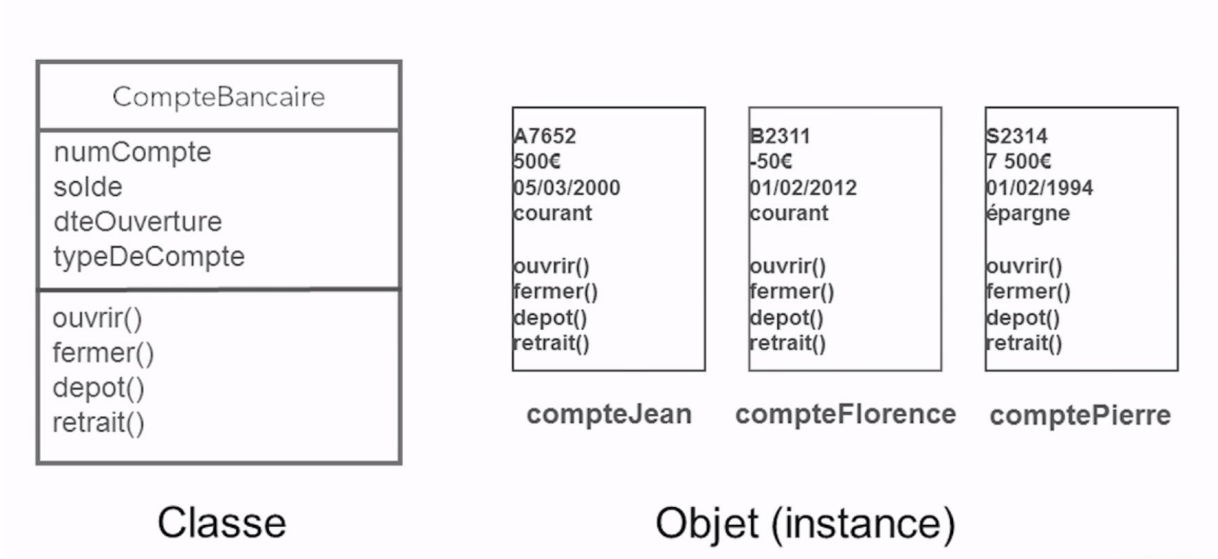


Notion d’instance de classe

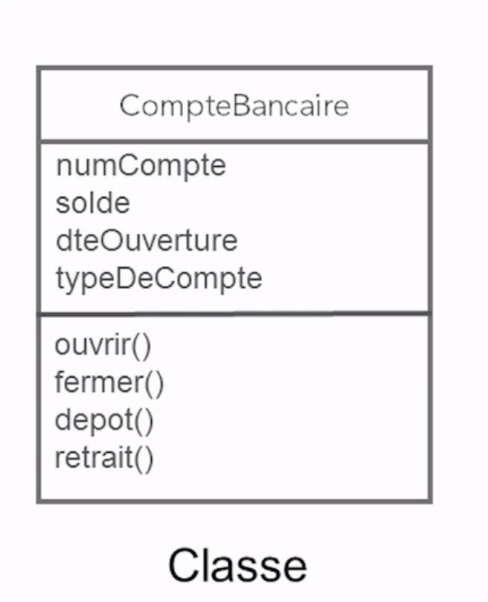
- On peut créer des instances de notre classe
- Chaque instance (aussi appelé objet parfois) à une vie qui lui est propre
- Chaque instance peut avoir des propriétés différentes (une tasse peut être vide / l’autre à moitié pleine, une pomme peut avoir une couleur différente…)
- Chaque instance à ses propres comportements (un avion vole alors qu’un téléphone sonne)



Classe versus instance de ma classe



Exemple



Classe compte bancaire

```
class CompteBancaire {
    constructor(nomCompte, solde, dteOuverture, typeCompte) {
        this.nomCompte = nomCompte;
        this.solde = solde;
        this.dteOuverture = dteOuverture;
        this.typeCompte = typeCompte;
    }

    ouvrir() {
        // Implémenter la méthode ouvrir
    }
    fermer() {}
    depot() {}
    retrait() {}
}
```

Getters et setters

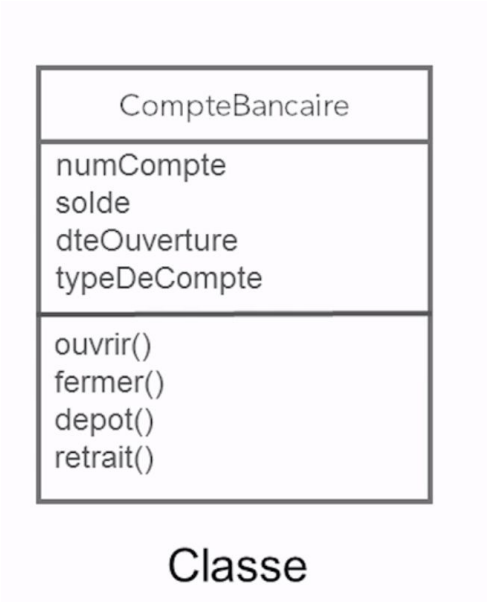
- Getter : méthode spéciale pour lire la valeur d'une propriété.
- Setter : méthode spéciale pour modifier la valeur d'une propriété
- Permettent de contrôler l’accès aux propriétés / modification (**encapsulation)**

```
// Getter pour solde
get solde() {
    return this._solde;
}

// Setter pour solde avec contrôle (pas de solde négatif)
set solde(nouveauSolde) {
    if (nouveauSolde >= 0) {
        this._solde = nouveauSolde;
    } else {
        console.log("Erreur : le solde ne peut pas être négatif.");
    }
}
```

Exercice 1

Exemple



Classe compte bancaire

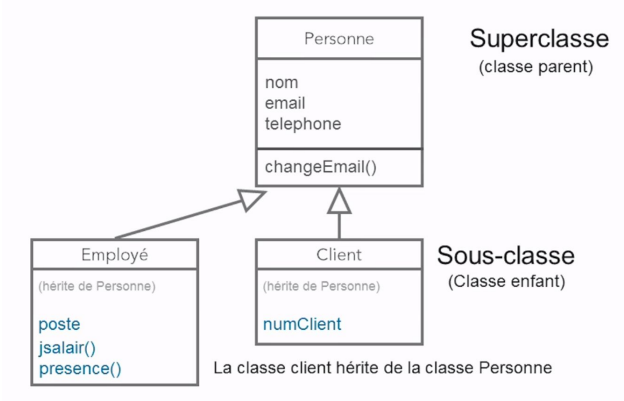
```
class CompteBancaire {
    constructor(nomCompte, solde, dteOuverture, typeCompte) {
        this.nomCompte = nomCompte;
        this.solde = solde;
        this.dteOuverture = dteOuverture;
        this.typeCompte = typeCompte;
    }

    ouvrir() {
        // Implémenter la méthode ouvrir
    }
    fermer() {}
    depot() {}
    retrait() {}
}
```

2. Héritage

Notion d’héritage

- L’héritage est une manière très pratique de réutiliser du code. Les classes enfants héritent des attributs et méthodes de la classe parent.
- Si on modifie la classe parent, toutes les classes enfants bénéficient de ces modifications.



Identifier les situations d'héritage

Une voiture est un véhicule.
Un bus est un véhicule.
~~Une voiture est un bus.~~

Un employé est une personne.
Un client est une personne.
~~Un client est un panier.~~

← Héritage

Un compte courant est un type de compte en banque.
Un compte d'épargne est un type de compte en banque.

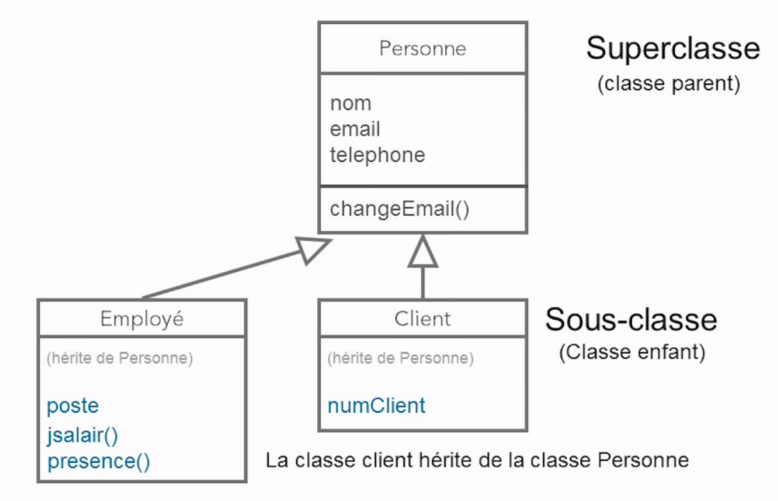
← Double Héritage

Une Peugeot 108 est une voiture et un véhicule.

Un caniche est un chien et un mammifère et un animal.

← Triple Héritage

Exemple d’héritage




```
class Personne {
    constructor(nom, email, telephone) {
        this.nom = nom;
        this.email = email;
        this.telephone = telephone;
    }

    changeEmail(newEmail) {
        this.email = newEmail;
    }
}

class Employe extends Personne {
    constructor(nom, email, telephone, poste, salaire) {
        super(nom, email, telephone);
        this.poste = poste;
        this.salaire = salaire;
    }

    presence() {
        console.log("Présence de l'employé enregistrée.");
    }
}

class Client extends Personne {
    constructor(nom, email, telephone, numClient) {
        super(nom, email, telephone);
        this.numClient = numClient;
    }
}
```

diginamic
FORMATION

Exercice 2

TP 3 : Validation des acquis