الجامعة الدولية للرباط
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵜⴰⵅⵔⵕⵕⵉⵜ ⵏ ⵕⵕⴱⴰⵟ
Université Internationale de Rabat
L'UNIVERSITÉ INNOVANTE

Chapitre 5: Classes abstraites et interfaces

M. NAJIB

Objectifs

L'objectif de ce chapitre est de présenter deux éléments essentiels en programmation orientée objets, en JAVA, et qui sont:

- Les classes abstraites
- Les interfaces

À la fin de ce chapitre vous êtes censés maîtriser:

- la syntaxe de la programmation des classes abstraites et des interfaces.
- L'utilisation des classes abstraites et des interfaces pour améliorer la structuration de votre code JAVA.

Classe abstraite: définition

- Une classe abstraite est une classe qui ne permet pas d'instancier des objets
- Elle ne peut servir que d'une classe de base pour une dérivation

Une classe abstraite se déclare de la manière suivante:

```
abstract Class NomClass{

}
```

Classe abstraite

Dans une classe abstraite on déclare:

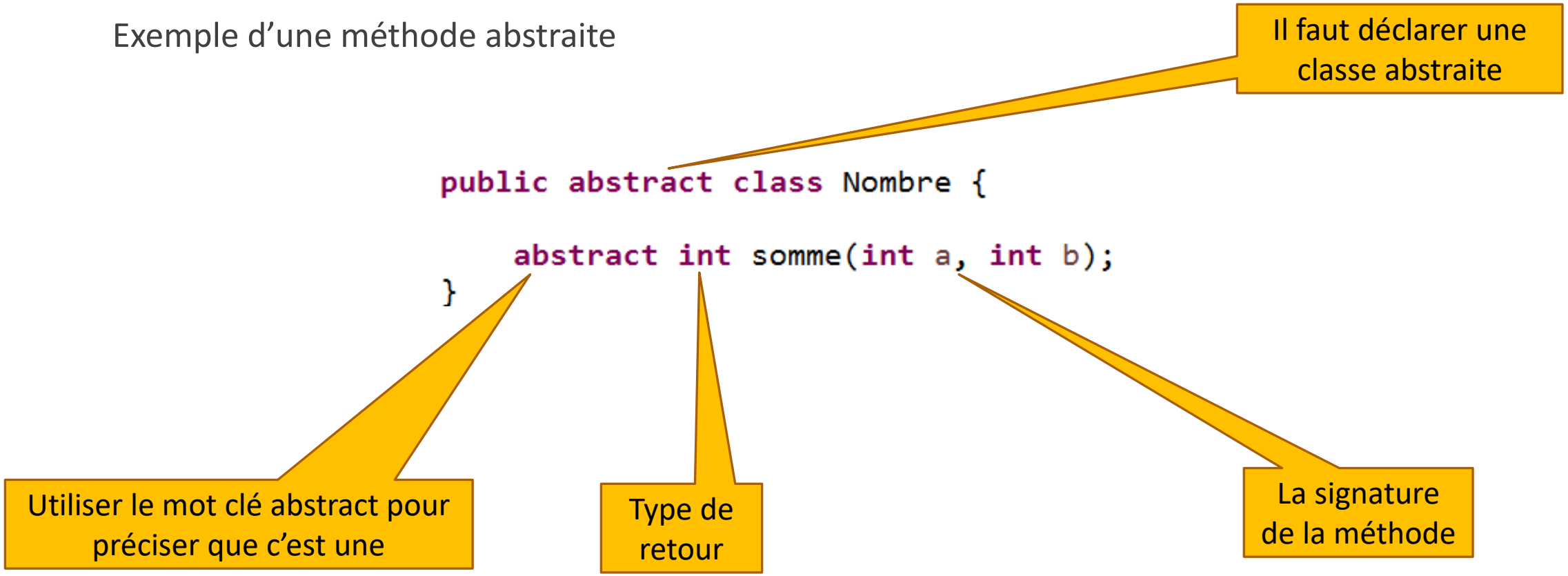
- Les méthodes et les attributs qui seront hérités par toutes les classes dérivées (classes filles).
- Les méthodes abstraites

Une méthode abstraite est une méthode dont on ne spécifie que la signature et le type de la valeur de retour.

Méthode abstraite

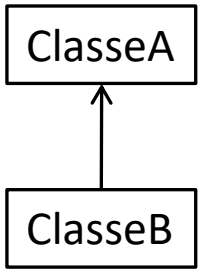
Une méthode abstraite ne peut être déclarée que dans une classe abstraite

Exemple d'une méthode abstraite



Exemple(1)

Soit les deux classes A et B qui ont une relation d'héritage.




```java
public abstract class ClasseA {
    public int nbr1;
    public int nbr2;
    // méthode abstraite
    abstract int somme(int a, int b);

    // méthode
    public void afficher(){
        System.out.println("nbr1 = " + nbr1 +
                " nbr2 = " + nbr2);
    }
}
```

Classe A

Exemple(2)

L'implémentation de la classe B doit spécifier le traitement réalisé par toutes les méthodes abstraites déclarées dans la classe A :

Accès aux attributs de la classe mère

Spécification du traitement de la méthode somme

```
public class ClasseB extends ClasseA {

    public ClasseB(int a, int b){
        super.nbr1 = a;
        super.nbr2 = b;
    }
    int somme(int a, int b){
        int c = a + b;
        return c;
    }
}
```

Classe B

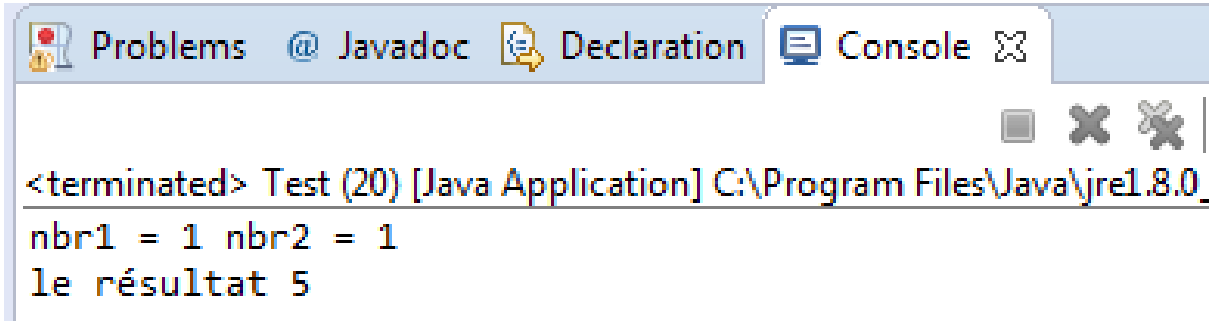
Exemple(3)

- Une classe abstraite ne peut pas être instanciée
- Dans le programme main on n'instancie que ces classes dérivées
- Les instances des classes dérivées ont accès à toutes les méthodes et attributs de la classe mère

ClasseTest

```
public static void main(String[] args) {
    // appel de la méthode afficher
    ClasseB cb = new ClasseB(1,1);
    cb.afficher();
    // appel de la méthode somme
    int cc = cb.somme(2, 3);
    System.out.println("le résultat " + cc);
}
```

Résultat



Règles à retenir

- Si une classe comporte des méthodes abstraites, elle est obligatoirement une classe abstraite.
- Il faut spécifier le type et le nom des paramètres dans une méthode abstraite
- Si une classe est dérivée d'une classe abstraite, elle doit redéfinir toutes les méthodes abstraites sinon elle doit être considéré comme une classe abstraite.
- Une classe dérivée d'une classe non abstraite peut être déclarée abstraite et/ou contenir des méthodes abstraites

Exercice d’application

Soit les classes suivantes qui illustrent la relation entre les classes salarié, professeur administrateur.

- Salarié{CIN, nom, salaireB}
- Prof{CIN, nom, salaireB, grade (I, II, III)}
- Admin{CIN, nom, salaireB, nbrExp}

Formule1 Le calcul du salaire des prof est le salaire de base + 300 * grade actuelle

Formule2 Le calcul du salaire d’un administrateur est le salaire de base + 5% si (1-5ans d’expérience), 10% (6-10ans d’expérience)

Donnez une implémentation de ces classes en prenant en considération les formules différentes de calcul de salaire.

Les interfaces

Définition: une interface est une collection nommée qui définit les en-têtes d'un certain nombre de méthodes, ainsi que des constantes.

Une interface en JAVA est déclarée de la manière suivante:

```
public interface NomInterface{

    // liste des méthodes

}
```

La déclaration d'une interface ressemble à la déclaration des classes. On y utilise simplement le mot clé interface à la place de class

Les interfaces

L'implémentation d'une interface passe par l'utilisation du mot clé **implements**

```
public interface Affichage {
    // constante
    public int nombreMax = 100;
    // méthode
    public void afficher(int a);

}
```

```
public class ClasseEtudiant implements Affichage{
    public void afficher(int a) {
        System.out.println("la valeur du nombre à afficher est = " + a);

    }
    public void afficheCST(){
        System.out.println("la valeur de la constante est " + nombreMax);
    }

}
```

Les interfaces

Remarque: une classe qui implémente une interface doit redéfinir toutes les méthodes déclarées dans cette interface

Une classe peut implémenter plusieurs interfaces. Cette opération est réalisée de la manière suivante:

```
public class NomClass implements interface1, interface2{

    // redéfinition des méthodes de l'interface 1 et
    l'interface2

}
```

Les interfaces

Les conflits rencontrés lors de l'implémentation de deux interfaces qui définissent la même méthode:

1. Si les deux méthodes ont la même signature, alors il suffit d'implémenter une seule méthode qui satisfait les deux interfaces.
2. Si les deux méthodes ont le même type de retour avec deux signatures différentes alors il faut implémenter les deux méthodes pour satisfaire les deux interfaces.
3. Si les deux méthodes n'ont pas le même type de retour alors il faut modifier la conception des deux interfaces

Les interfaces et le polymorphisme

On peut déclarer des objets de type interface

```
public static void main(String[] args) {

    Affichage[] t ;
    t = new Affichage[2];

    t[0] = new ClasseEtudiant();
    t[1] = new ClasseProf();


    for(int i = 0; i < t.length ; i++){
        t[i].afficher(10);
    }
```

Les interfaces et le polymorphisme

Le code de la classe professeur

```
public class ClasseProf implements Affichage{

    public void afficher(int a) {
        System.out.println("affichage prof le nombre "
                + "à afficher est = " + a);

    }
}
```

Le résultat de l'exécution

```
<terminated> Test (20) [Java Application] C:\Program Files\Java\jre1.8.0_
la valeur du nombre à afficher est = 10
affichage prof le nombre à afficher est = 10
```

Les interfaces et l’héritage

L’héritage entre les interfaces est réalisé comme suite:

```
public interface Affichage2 extends Affichage {
    public void methode2();
}
```

Les interfaces et l'héritage

L'héritage entre les interfaces est équivalent à l'implémentation des deux interfaces

```
public class ImplementationHeritage implements Affichage2{

    @Override
    public void afficher(int a) {
        // TODO Auto-generated method stub

    }

    @Override
    public void methode2() {
        // TODO Auto-generated method stub

    }

}
```

Exercice d’application

Nous souhaitons implémenter une classe entité qui représente les entités Etudiant et Professeur.

Pour imposer au développeur de procéder à l’implémentation des méthodes de bases pour la gestion de ces entités (**C**reate **R**ead **U**pdate **D**elete)

Contrainte:

La fonction create donne comme résultat un objet de la classe de déclaration.

Question

Proposer une solution pour imposer l’implémentation et le respect de la signature de ces méthodes

Merci de votre attention