Cours 4

- Héritage
- Classe abstraite
- Interface

1

On définit une classe dérivée à partir d'une classe existante comme une extension de celle-ci :

super classe, classe de base, classe mère

la classe dérivée hérite de la classe de base

- toutes les méthodes publiques
- toutes variables (attributs) instance et statiques

le code est réutilisé sans copier

- la classe dérivée peut définir de nouvelles méthodes et variables
- la classe dérivée peut redéfinir les variables ou les méthodes de la classe de base

Réutiliser une classe existante en l'adaptant et factoriser le code

2

Définition d'une classe dérivée

```
class Point
{private int x,y;
public Point(int a, int b){x=a;y=b;}
public void modif(int coef){x=x*coef; y=y*coef;}
public void affiche_pt()
{System.out.println("coordonnees du point :"+x+ " "+y);}
}
class Point_Couleur extends Point
{private String couleur;
public Point_Couleur(int a, int b, String c)
{super(a,b); couleur=c;}
public void affiche_pt(){super.affiche_pt();
System.out.println("et sa couleur " +couleur);}
}
public class point_heritage{
public static void main(String args[])
{ Point pt1=new Point (-44,6);
pt1.affiche_pt(); pt1.modif(2); pt1.affiche_pt();
Point_Couleur ptc=new Point_Couleur (3,4,"blue");
ptc.affiche_pt(); ptc.modif(2); ptc.affiche_pt();
}}
```

3

Dans Java 5.0, il est possible de changer le type de la valeur retournée s'il s'agit d'un type classe

```
public class BaseClass
{ . . .
    public Employee getSomeone(int someKey)
    . . .
}
public class DerivedClass extends BaseClass
{ . . .
    public HourlyEmployee getSomeone(int someKey)
```

```
classe de base :private void doSomething()
classe dérivée : public void doSomething()
```

L'inverse n'est pas possible

```
classe de base:public void doSomething()
classe dérivée :private void doSomething()
```

4

- On peut définir plusieurs niveaux d'héritages et l'héritage est transitive
- Il peut y avoir une seule classe de base
- Toute classe est dérivée de la classe racine `Objet` définie dans `java.lang`
- On peut interdire qu'une classe soit étendue

```
final class A{  
}
```

- Tout constructeur autre que la classe `Objet` fait appel
 - soit à un constructeur de sa classe de base (`super ()`)
 - soit à un autre constructeur de sa classe (`this()`)
 - si il n'y a pas ceci en première ligne, le compilateur ajoute `super()`; en première ligne du constructeur

5

On invoque le constructeur de la classe de base par **super**

```
public derivedClass(int p1, int p2, double p3)  
{  
    super(p1, p2);  
    instanceVariable = p3; }
```

super

- la première instruction
- s'il n'existe pas dans le constructeur de la classe dérivée, le constructeur de la classe de base est invoquée (erreur s'il n'existe pas)
- il faut invoquer pour initialiser les variables d'instance

this

pour invoquer un autre constructeur dans la même classe

```
super.methode(); this.methode();
```

6

```

import java.io.*;
class A{
public A()
{System.out.println("constructeur de A");}
}
class B extends A {
public B()
{System.out.println("constructeur de B");}
public B(int a)
{this();System.out.println("autre constructeur de B");}
}
class C extends B{
public C()
{super(3);System.out.println("constructeur de C");}
}
public class Tst_const
{public static void main(String args[])
{ C c= new C();}
}

```

constructeur de A
constructeur de B
autre constructeur de B
constructeur de C

7

```

import java.io.*;
class A{
public void affiche()
{System.out.println("je suis un A");}
}
class B extends A {}
class C extends A
{public void affiche()
{System.out.println("je suis un C");}
}
class D extends C{
public void affiche()
{System.out.println("je suis un D");super.affiche();}
}
class E extends B{}
class F extends C {}
public class deriv_heritage
{public static void main(String args[])
{ A a= new A(); a.affiche(); System.out.println();
  B b= new B(); b.affiche(); System.out.println();
  C c= new C(); c.affiche(); System.out.println();
  D d= new D(); d.affiche(); System.out.println();
  E e= new E(); e.affiche(); System.out.println();
  F f= new F(); f.affiche(); System.out.println();}
}

```

je suis un A
je suis un A
je suis un C
je suis un D
je suis un C
je suis un A
je suis un C

8

Polymorphisme

Le polymorphisme est une notion liée à la redéfinition des méthodes et à la liaison dynamique.

```
public class deriv_heritage1
{public static void main(String args[]){
  A a= new A(); a.affiche();
System.out.println();
  B b= new B(); b.affiche();
  a=b; a.affiche(); System.out.println();
  C c= new C(); c.affiche();
  a=c; a.affiche(); System.out.println();
  D d= new D(); d.affiche();
  a=d; a.affiche();
  c=d; c.affiche(); System.out.println();
  E e= new E(); e.affiche();
  a=e; a.affiche();
  b=e; b.affiche(); System.out.println();
  F f= new F(); f.affiche();
  a=f; a.affiche();
  c=f; c.affiche(); System.out.println();
}}
```

je suis un A
je suis un A je suis un A
je suis un C je suis un C
je suis un D je suis un D je suis un D
je suis un A je suis un A je suis un A
je suis un C je suis un C je suis un C

9

```
class Orateur{
String action(){return "parle";}
}
class Grenouille extends Orateur{
String action(){return "coasse";}
}
class Fourmi extends Orateur{
String action(){return "croonde";}
}
public class Tst_Orateur{
public static void main(String args[]){
Orateur p = new Orateur();
System.out.println("orateur "+ p.action());
Grenouille q= new Grenouille();
System.out.println("grenouille "+ q.action());
p=q; System.out.println("orateur "+ p.action());
Orateur x= new Fourmi();
System.out.println("orateur "+ x.action());}
}
```

10

Classes Abstraites

- Une méthode abstraite est définie uniquement par son intitulé sans code
- Une classe abstraite est une classe dont au moins une méthode est abstraite

```
abstract class figure
{protected double long;
abstract void calcul_long();}
```

- Dans les classes dérivées, la méthode abstraite doit être décrite

11

```
abstract class Figure{
protected double longueur;
abstract void calcul_long();
}
class Segment extends Figure
{private int xA, yA, xB, yB;
public Segment(int xAp,int yAp,int xBp,int yBp)
{xA=xAp; yA=yAp;xB=xBp; yB=yBp;longueur=0;}

public void calcul_long(){
longueur=Math.sqrt((xB-xA)*(xB-xA)+
(yA-yB)*(yA-yB));
System.out.println("long "+longueur);}
}
class Cercle extends Figure{private int rayon;
public Cercle( int r){rayon=r;longueur=0;}
public void calcul_long(){
longueur=2*3.14*rayon;System.out.println("long "+longueur);}
}public class Figabs{
public static void main(String args[]){
Segment s=new Segment(1,3,4,5);s.calcul_long();
Cercle c=new Cercle(3);c.calcul_long();}}
```

12

Interface

Une interface est une classe abstraite ayant les caractéristiques suivants:

- toutes les méthodes sont abstraites et public, alors qu'une classe abstraite peut avoir des méthodes non abstraites
- toutes les variables sont static et constantes déclarées par le modificateur final, alors qu'une classe abstraite peut avoir des variables ordinaires
- toute classe peut hériter d'une seule classe mais elle peut hériter de plusieurs interfaces
- une interface peut hériter d'une ou plusieurs interfaces mais elle ne peut pas hériter d'une classe

- il n'est pas nécessaire d'indiquer abstract pour les méthodes et static final pour les variables
- une interface définie par le mot **interface**
- si une classe hérite d'une interface **implements**

13

```
interface I
    liste de constantes;
    liste de prototypes de méthodes
```

```
class A extends B implements I{ }
```

```
class A implements I1,I2 { }
```

```
interface I extends I1,I2 { }
```

14

```

interface Itf_calcul{
double[] valeur={33.2, 13.2, 5.8};
public void calcul_prime();}
class Pers_admin implements Itf_calcul{
private String nom; private String service;
private int categorie; private int age; private double prime;
public Pers_admin(String n, String s, int c, int a){
nom=n; service=s; age=a; categorie=c;}
public void calcul_prime(){prime=valeur[categorie]*age;}
public void affiche() {
System.out.println("nom: "+nom+" service: "+service+ " prime "+prime);}
}
class Pers_exterieure implements Itf_calcul{
private String nom; private String entreprise;
private int categorie; private double prime;
public Pers_exterieure(String n, String s, int c){
nom=n; entreprise=s; categorie=c;}
public void calcul_prime(){ prime=valeur[categorie]*20;}
public void affiche() {
System.out.println("nom: "+nom+" entreprise:"+entreprise+" prime: "+prime)
}
public class Tstinterface{
public static void main(String args[]){
Pers_admin pa= new Pers_admin("Dupont", "comptabilite", 2, 40);
pa.calcul_prime(); pa.affiche();
Pers_exterieure pe=new Pers_exterieure("Bernard", "EDF", 1);
pe.calcul_prime(); pe.affiche();}}

```