- Transtypage, conversion
- Polymorphisme

Conversion et Transtypage

- La conversion d'un type ordinaire vers un autre type ordinaire
  - implicitement vers un type « plus grand »
  - explicitement vers un type « plus petit »
  - double pi=3.14; int p;
  - p=(int) pi;
- transtypage d'une variable de type « classe »
  en une autre d'un autre type « classe »
  - les seules conversions possibles concernent les variables qui référencent des objets d'une hiérarchie des classes construites par héritage et dans le sens ascendant. (*conversion implicite*)

conversion vers la classe Objet explicite

```
class Point{
protected int x,y;
public Point (int a, int b)    {x=a; y=b;}
public void affiche()
{System.out.println("coord:"+x +" " +y);}
}
class Point_Colore extends Point{
private String couleur;
public Point_Colore(int a, int b, String c)
{super(a,b); couleur=c;}
public void affiche()
{System.out.println("coord:"+x +" " +y+  "couleur :"+couleur);}
}
public class Tstranstypage{
public static  void main(String args[]){
Point pp= new Point(4,7);
Point_Colore ppcol=new Point_Colore(7,9, "vert");
 pp=ppcol; pp.affiche();
 // ppcol=pp interdit  Object o1;   Point p1;
 o1=new Point_Colore(1,8,"rouge");  p1= (Point_Colore)o1;
p1.affiche();
//OK compilation, erreur execution
Point p2=new Point(5,5);  Point_Colore pc2= (Point_Colore)p2;
 pc2.affiche();  }}
```

- Un même appel de méthode peut donner lieu à des exécutions différentes

si pp , variable de type Point, a pour contenu la référence à un objet de type Point; pp.affiche() déclenche l'exécution de la méthode Point.

pp=ppcol; pp a pour contenu la référence d'un objet de type Point_colore; pp.affiche() déclenche l'exécution de la méthode de classe Point_Colore.

- Un tableau peut contenir des objets appartenant à des classes d'un même arbre d'héritage

- Le type Objet peut être utilisé pour référencer n'importe quel type d'objet

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
```

Classe racine Object

•ancêtre de toutes les classes définie dans java.lang
•définit les méthodes héritées par toutes les classes
•constructeur Object();

Méthodes

•public String toString()
+ obj +obj.toString()
•public boolean equals (Object o)
•public int hashcode()
•protected Object clone()
•protected void finalize

Transtypage

```
class Cl_Obj
{private Object t;

public Cl_Obj(Object u)
{t=u;}

public Object get()
{return t;}
}

public class Testobjet
{ public static void main(String[] args)
         {Cl_Obj nb1= new Cl_Obj (new Integer(37));
         Cl_Obj st1= new Cl_Obj("bonjour");
         int  v=(Integer)nb1.get();//pas int;
         String s=(String)st1.get();
         System.out.println(v);
         System.out.println(s);
         nb1=st1;
         v=(Integer)nb1.get(); // erreur l'exécution
}}
```