Objectifs du TD

Ce TD a pour but de présenter la dynamique des programmes Java (notamment lors de l'emploi de l'héritage), ainsi que le concept de cœrcion de type (appelé aussi *cast*).

Exercice 1 : Rappels sur l'Héritage

Soient les classes suivantes :

```
class Animal{
    boolean vivant ;
    int age;

    Animal(int a) {
        age=a;
        vivant=true;
        System.out.println("Un animal de " + a + " an(s) vient d'être créé.");
    }
    void vieillir(){
        ++age;
        System.out.println("C'est l'anniversaire de cet animal.");
        System.out.println("Il a maintenant "+age+ " ans.");
    }
    void mourir(){
        vivant=false;
        System.out.println("cet animal est mort ");
    }
    void crier(){}
}

class Canari extends Animal{
    Canari(int a){
         super(a);
    }
    void crier(){
         System.out.println(" Cui-cui ");
    }
}
```

Quel est le résultat de la fonction `main()` contenant les lignes suivantes ?

```
Canari titi = new Canari(3);
titi.vieillir();
titi.crier();
```

Exercice 2 : Liaison dynamique

Rappel La liaison dynamique est le coeur de la programmation orientée objet. Avec la liaison dynamique, la méthode appelée est celle correspondant au type de l'objet et non au type de la variable qui le référence. Il s'agit d'un procédé dynamique dans le sens où la méthode appelée est déterminée à l'exécution (par le type de l'objet référencé) et non à la compilation :

- il y a remplacement de la méthode `f` de la classe `B` dans la classe `A` (dérivée de `B`) si les deux fonctions ont les mêmes *signatures*,
- attention, si la méthode `f` de la classe `A` est appelée, elle est appelée dans le *contexte* de la classe `A` : les associations des noms aux variables et méthodes sont ceux de la classe `A`.

Soient les classes suivantes :

```
// fichier Essai.java
class B{
    public int i=0;
    public void f(){System.out.println(i);}
    public void g(){System.out.println(i);}
}
class A extends B {
    public int i=1;
    public void f(){System.out.println(i);}
    public void h(){System.out.println(i);}
}
class Essai{
  public static void main(String[] args){
    A a = new A();
    B b = new B();
    b.f();
    b.g();
    a.f();
    a.g();
    a.h();
    b=a;
    b.f();
    b.g();
    b.h();
  }
}
```

Qu'affiche ce programme ?

Exercice 3 : Liaison dynamique (suite)

Soit le programme Java suivant :

```
// fichier Liaison.java
class A {
        int i = 1;
```

```
        public int getI(){return i;}
}
class B extends A {
        int i = 89;
        public int  getI(){return i;}
}
class C extends A {
        int i = 123;
        public int getI(){return i;}
}
public class Liaison{
        public static void main(String[] args) {
                B classeB = new B();
                C classeC = new C();
                A resultat;
                if (args.length == 1) {
                        resultat = classeB;
                } else {
                        resultat = classeC;
                }
                System.out.println("resultat.i -> "+resultat.i);
                // la variable est recherchee de facon statique

                System.out.println("resultat.getI() -> "+resultat.getI()) ;
                // la methode est recherchee de facon dynamique
        }
}
```

Que donnent les commandes suivantes :

```
javac Liaison.java
java Liaison
java Liaison Titi
```

Exercice 4 : Cœrcion de type

Rappel Le *cast* (ou conversion de type) consiste à forcer un changement de type. Pour cela, il suffit de placer le type entre parenthèses à gauche de la variable à convertir. **Attention :** ce changement peut induire des réductions de résolution qui sont parfois considérables. Par exemple, au cours de la conversion d'une valeur décimale en valeur entière, on perd la partie décimale du chiffre.

Cast sur des types de base

```
float a;            // permet de stocker des valeurs
                    // entre -3.4E38 et +3.4E38.
double b;           // permet de stocker des valeurs
                    // entre -1.7E308 et +1.7E308
b = 1 / 1.2346789;  // stocke 0.810..... dans b avec
```

```
                    // une résolution double
a = (float)b;       // copie la valeur et réduit la
                    // résolution du chiffre à float
```

Soient les instructions suivantes :

```
double a = 1.989898;
int b = (int)a;
```

Que contient alors la variable b ?

Cast sur des objets

```
class Chien extends Mammifere { ... }
...
    Chien milou = new Chien();
    Mammifere telAnimal = milou;   // cast implicite
    milou = (Chien)telAnimal;      // cast explicite
    milou = telAnimal;
```

Ce code est-il correct ? Expliquez pourquoi.

Exercice 5 : Classes abstraites

Rappel Une classe abstraite est un classe contenant au moins une méthode abstraite, *i.e.* une méthode qui ne contient pas de corps. Cette méthode doit être implémentée dans les sous-classes non abstraites.

A faire :

- Ecrire une classe abstraite `Animal` contenant une méthode abstraite `crier`.
- Ecrire deux classes héritées d'`Animal`, respectivement `Canari` et `Chien`, possédant une redéfinition de la méthode `crier()`. La méthode `crier()` affiche à l'écran le cri de l'animal.
- Dans la méthode `main()` de la classe `Test`, créez un objet milou et un objet titi et faites les crier.