Université Paris Diderot **Programmation Orientée Objet**
L2 **Année 2017-2018**

TD - Séance n°6

Contrôle continu

À partir du début de l'examen, vous avez 1 heure 30 pour le traiter. Si vous finissez en avance, profitez-en pour vous relire ! Une étourderie est toujours possible.

Exercice 1 *Syntaxe, analyse de code et concepts de base*
Pour chacun des codes suivants, indiquer s'il compile ou non. S'il compile, décrire ce qui se passe à l'exécution. S'il ne compile pas, expliquer pourquoi.

Les fonctions `public void main(String[] args)` sont systématiquement dans une classe `public class Test`, dans un fichier séparé `Test.java`, et les classes `A`, `B` et ainsi de suite sont chacune dans un fichier `A.java`, `B.java` ...

— Que donne la classe `A` avec le `main` qui suit ?

```
1 public class A{
    private int i;
3   public A sousA;
    public A(int j){ i = j; }
5   public int f(){ return i + sousA.g(); }
    private int g(){ return i; }
7 }
```

```
1 public static void main(String[] args){
    A a = new A(3);
3   System.out.println(a.f());
  }
```

— Que donne la classe `B` avec le `main` qui suit ?

```
  public class B{
2   private static int i = 5;
    public static void f(int j){ i += j; }
4 }
```

```
  public static void main(String[] args){
2   B b = new B();
    b.f(2);
4   System.out.println(b.i);
  }
```

— Que donne la classe `C` avec le `main` qui suit ?

```
1  public class C{
     public static int i=0;
3    { i = i+1; }
   }
```

```
   public static void main(String[] args){
2    C c1 = new C();
     C c2 = new C();
4    System.out.println(C.i);
   }
```

— Que donne la classe `D` avec le `main` qui suit ?

```
1  public class D{
     private final int[] tab;
3    public D(int[] t){ tab = t; }
     public int lireEntier(int i){ return tab[i]; }
5  }
```

```
1  public static void main(String[] args){
     int[] debutPi = {3, 1, 4};
3    D d = new D(debutPi);
     for(int i = 0; i < 3; i++)
5        debutPi[i] += i;
     for(int i = 0; i < 3; i++)
7        System.out.println(d.lireEntier(i));
   }
```

— Que donne la classe `E` avec le `main` qui suit ?

```
   public class E{
2    private int i;
     public E sousE;
4    public E(int j){ i = j; }
     public int f(){ return i + sousE.g(); }
6    private int g(){ return i; }
   }
```

```
1  public static void main(String[] args){
     E e = new E(7);
3    e.sousE = e;
     System.out.println(e.f());
5  }
```

Exercice 2 *Héritage, surcharge, interprétation de code*

Qu'affiche le code suivant ?

```
class A{
  public void f(){ System.out.println("f de A"); }
  public void g(){ System.out.println("g de A"); }
  public void h(A a){ System.out.println("h de A sur A") ;}

  public static void i(){ System.out.println("i de A"); }
}
class B extends A{
  public void g(){ System.out.println("g de B"); }
  public void h(A a){ System.out.println("h de B sur A");}
  public void h(B b){ System.out.println("h de B sur B") ;}

  public static void i(){ System.out.println("i de B"); }
}

public class Test{
  public static void main(String[] args){
    A a = new A();
    B b = new B();
    A ab = new B();

    a.f();
    b.f();
    ab.f();

    a.g();
    b.g();
    ab.g();

    a.h(ab);
    b.h(ab);
    ab.h(ab);

    a.i();
    b.i();
    ab.i();
  }
}
```

Exercice 3 *Modélisation et héritage*
On définit une classe `Ticket` par le code suivant.

```
public class Ticket{
  private static int prochainNumero = 0;
  public final int numero;
  private boolean valide = true;

  protected void invalider(){ valide = false; }
  public Ticket(){ numero = prochainNumero;
                   prochainNumero++; }
}
```

1. Écrire une fonction `toString()` de façon à ce que le code suivant :

```
System.out.println(new Ticket());
System.out.println(new Ticket());
```

   affiche
   `"Je suis le ticket numéro 0. Je suis valide."`
   `"Je suis le ticket numéro 1. Je suis valide."`
   Prévoir dès maintenant le cas où le ticket serait invalide.

   On va maintenant définir une classe `TicketPerissable` héritant de `Ticket` :

2. Écrire la classe `TicketPerissable`. Elle aura deux champs entiers `jour` et `annee`, qui ne doivent plus changer après l'appel au constructeur, qui a deux arguments entiers.

   Les champs `jour` et `annee` indiquent le dernier jour de validité du ticket (le jour numéro `'jour'` de l'année `'annee'`, le 1er janvier étant le 0ème jour, le 10 janvier le 9ème).

3. Écrire une fonction `boolean testerValidite(int jour, int annee)` qui renvoie `true` si le `TicketPerissable` est encore valide à la date donnée.

4. Écrire la fonction `toString()` de `TicketPerissable` de façon à ce qu'elle renvoie une chaîne de caractère comme :
   `"Je suis le ticket numéro 0. Je suis valide. J'expire le 34eme jour de 2018"`

5. Écrire la fonction `public static void miseAjourValidite(Ticket[] tickets, int jour, int annee)` de la classe `Ticket`, qui parcourt le tableau de tickets et invalide ceux qui doivent être invalidés (considérant qu'on est le jour `'jour'` de l'année `'annee'`).