Université de Tlemcen
Faculté des Sciences
Département d'Informatique

Année universitaire 2018-2019
2ème année licence informatique
Programmation Orientée Objet

TD N° 2

Exercice 1

Soient les classes suivantes :

class A {
int a;
A(int a) { a=a; }
}
	class B extends A {
	int b;
	B(int b) { b=b; }
	}
		class C extends B {
		int c ;
		}

Corrigez les erreurs **sur les constructeurs** des classes A, B et C.

Exercice 2

```
class A {
   void m(){System.out.println("A");  }
}


class B extends A {
  void n(){   System.out.print("a  b");  }
}


class C extends B {
  void m(){ System.out.println("C");}
}
```

```
public class Polym {
    public static void main(String args[]){
       A a=new A();
        B b1=new B();
        b1.m();
        b1.n();
        System.out.println();
        C c=new C();
        B b2= c;
        c.m();
        b2.m();
        b2.n();    }
}
```

1. Qu'affiche ce programme ? Expliquez ?
2. Redéfinissez la méthode **n()** dans la classe **C** de telle façon qu'elle **complète** celle de la classe **B** en affichant : a b c
3. Que se passe t-il si on ajoute à la fin du main les instructions : **a=c ; a.m() ; a.n() ;**

Exercice 3

```
class Entier{
    int n ;
    Entier (int nn) { n = nn ; }
    void incr (int dn) { n += dn ; }
    void affiche () { System.out.println (n) ; }
    }
```

1. Ecrire une classe principale **TestEntier** qui permettra de :
   a) créer deux objets Entier (**e1** et **e2**) avec des valeurs différentes de n.
   b) afficher les valeurs n des deux objets e1 et e2
   c) **incrémenter** la valeur n de l'un des 2 objets de telle sorte que **e1.n soit égale à e2.n**
   d) comparer et afficher le résultat de la comparaison entre e1 et e2 en utilisant l'opérateur == puis la méthode **equals()** de la classe Object.

2. Dans la classe Entier, **Spécifier** la méthode **equals** () afin qu'elle compare 2 objets selon les **valeurs de leurs attributs**.

3. Réécrire TestEntier en déplaçant la création des objets e1 et e2 **en dehors du main**. Que faut-il modifier ?

Règles de transtypage ou de Casting (UpCasting + DownCasting)

1. Dans une instruction d'affectation (refVar1 = refVar2;), le compilateur vérifie que le type de la variable refVar1 est le même ou **un type parent** du type de la variable refVar2.

"**UpCasting**" implicite/explicite : permet de convertir le type d'une référence vers un type parent.

Exemple

```
Object obj1 = new String(); // UpCasting implicite
Object obj2 = (Object) new String(); // UpCasting explicite
```

2. La conversion (implicite ou explicite) vers un type parent est toujours acceptée par le compilateur et elle ne posera aucun problème à l'exécution du programme : une référence d'un type parent peut, sans risque, lire/modifier les attributs ou invoquer les méthodes dont elle a accès, indépendamment si l'instance référée et du même type ou un sous-type du type de la référence.

"DownCasting" permet de convertir le type d'une référence vers un sous type.

Exemple

```
String str1 = new Object(); // DownCasting implicite (erreur de compilation : Type
mismatch: cannot convert from Object to String)

String str2 = (String) new Object(); // DownCasting explicite (erreur à l'exécution :
java.lang.ClassCastException: java.lang.Object cannot be cast to java.lang.String)
```

3. La conversion implicite (String str1 = new Object();) vers un sous type est toujours refusée par le compilateur.

4. Par contre, la conversion explicite (String str2 = (String) new Object();) vers un sous type est toujours acceptée par le compilateur. Cependant à l'exécution du programme, la JVM va vérifier si le type de l'instance (Object) est le même ou un sous type du type qu'on a spécifiée pour la conversion (String) : si ce n'est pas le cas, la JVM déclenchera une exception.

Exercice 4

a) Que représente phraseObject et phraseString dans les deux codes suivants :

1er code

```
Object phraseObject = "Ceci est une chaine de caractères";
String phraseString = (String) phraseObject;
```

2ème code

```
String phraseString = "Ceci est une chaine de caractères";
Object phraseObject = (Object) phraseString;
```

b) Donnez des exemples de « downCasting » implicite, de « downCasting » explicite qui ne fonctionne pas et un autre qui fonctionne.