Java Avancé
Master 1 Informatique UPEC 2015/2016
TD 1 : Rappels de Java et Généricité

Exercice 1: *Choix des méthodes polymorphes*

Soient données les trois classes suivantes :

```
class A {
    void f (A a) {System.out.println("AA");}
    void f (C c) {System.out.println("AC");}
}
class B extends A {
    void f (B b) {System.out.println("BB");}
}
class C extends B {
    void f (A a) {System.out.println("CA");}
    void f (C c) {System.out.println("CC");}
}
```

Dites ce qu'affiche la méthode suivante et expliquez pourquoi

```
public static void main (String[] args) {
    A aa = new A();
    B bb = new B();
    A ac = new C();
    A ab = new B();
    aa.f(ac);
    bb.f(ac);
    ab.f(bb);
}
```

Exercice 2: *Sous-typage*

Soient données les classes suivantes :

```
class A<T> {}
class B<T> extends A<T> {}
class C<T> extends B<T> {}
class X{}
class Y1 extends X{}
class Y2 extends X{}
```

Dessinez les relations de sous-typage entre les types suivants, en expliquant la raison :

```
A<? extends X> {}
B<Y1>
C<X>
C<? super Y2>
C<? extends Y2>
```

Exercice 3: *Méthode générique 1*

Montrez avec un exemple pourquoi la signature

```
static void f (List<Object> ls) ;
```

et la signature

```
static <T> void f (List<T> ls) ;
```

ne sont pas équivalentes

Exercice 4: *Méthode générique 2*

Pour chacune de ces méthodes générique, dire si elles compilent ou pas, en expliquant pourquoi.

```
static <T> void f (T x) {
    System.out.print(x);
}
static <T> T g () {
    return null;
}

static <T> T h (List<? super T> ls) {
    return ls.get(2);
}

static <T,S> void k (List<T> lt, List<S> ls) {
    lt.add(ls.get(i));
}
```

Exercice 5: *Wild Card*

Considérez des méthodes avec les signatures suivantes.

```
static <T> T f (List<? super T> ls) ;

static <T> void g (List<? super T> lt, List<? extends T> ls);

static <T,S> void h (List<T> lt, List<S> ls);
```

Soient `A,B,C` trois classes, déclarées comme suit :

```
class A{}
class B extends A{}
class C extends B{}
```

Indiquez les erreurs dans le code suivant :

```
public static void main (String[] args) {
    List<A> laa = new ArrayList<>();
    List<? extends A> lac = new ArrayList<C>();
    List<? extends B> lbc = new ArrayList<C>();
    ArrayList<? super B> lba = new ArrayList<A>();
    A a = f(lba);
    B b = f(lbc);
    h(lbc,lba);
    g(lbc,lba);
    g(laa,lac);
```

Exercice 6: *Wild card ou généricité ?*

Écrivez le corps d'une méthode de signature

```
static <T> void f (List<T> ls) ;
```

sans mentionner le type `T` dans le corps, et tel que il ne puisse pas être le corps d'une méthode de signature

```
static void f (List<?> ls) ;
```

Exercice 7: *Paramètre borné*

Écrivez le corps d'une méthode de signature

```
static <T extends List<?>> void f (List<T> ls) ;
```

tel que il ne puisse pas être le corps d'une méthode de signature

```
static <T> void f (List<T> ls) ;
```

Exercice 8: *** *Wild card sauvage*

Considérez les classes

```
abstract class Wild<T> {
    abstract void sup(Wild<? super T> w);
    abstract void ext(Wild<? extends T> w);
}
class A{}
class B extends A{}
```

Considérez les déclarations

```
Wild<A> wa;
Wild<B> wb;
Wild<? super A> wsa;
Wild<? super B> wsb;
Wild<? extends A> wea;
Wild<? extends B> web;
```

Dire, parmi les lignes suivantes, quelles donnent une erreur à la compilation

```
wa.sup(wb);
wsa.ext(web);
web.ext(web);
wb.ext(wea);
```

Essayez pour toutes les combinaisons possibles avec le 6 variables et les 2 méthodes (72 possibilités...)