



Contrôle Finale JAVA

2017-2018

I)

A)

1) Relever et justifier l'anomalie dans la déclaration suivante :

```
Class x{ public void f(){}
          public String f(){return null;};
          public String f(int i){return null};
```

2) Donner le résultat de l'exécution du code suivant :

```
Class AA{ String c;
          AA (String x) {c=x;}
          Public static void main (String [ ] args) {
              AA a1=new AA("12"); AA a2=new AA("12");
              System.out.println (a1.equals (a2) :: a1==a2);
          }
      }
```

3) En considérant les déclarations suivantes :

Interface I { } ;

Class A { } ;

Class L extends A implements I { } ;

Les instructions suivantes sont-elles valides ?

(Expliquez brièvement)

I x1 = new I();

.....

L x2 = new A();

.....

I x3 = new A();

.....

I x4 = new L();

.....

B) On considère les classes Produit et Registre la même paquetage

Public Class Produit{

String code ;

Int quantité;

//1)Donner un constructeur pour <<Produit>>, utilisant ses deux attributs

//2) Redéfinir, relativement au <<code>> la méthode <<equals>> dans <<Produit>>

II)

A)

Public Class Registre extends LinkedList<Produit> {

Registre css(int qt) {

//Cette méthode retourne l'ensemble des produits de quantité > qt

//Il faut :

//Définir, par une Lambda expression, un prédictat

Testant si un produit est de

// quantité > qt

//Utiliser un itérateur explicite pour parcourir un registre

//3) Compléter

}

```
Map <String,Integer> scommandes() {  
    //retourne une <> qui, pour chaque produit  
    //désigné par son code, associe la somme des  
    //quantités de ce produit  
    //4) Compléter
```

```
.....  
.....  
}
```

5) Définir dans <> une méthode renvoyant le code du produit le plus commandé (i.e : ayant la plus grande somme des quantités)

```
}//Fin Classe Registre
```

B) En considérant la déclaration suivante :

```
Interface I { } ;
```

```
Class A { } ;
```

```
Class L extends A implements I { } ;
```

Les instructions <> suivantes sont-elles valides ?

```
List<I> l = new ArrayList<> () ;
```

```
l.add (new A());
```

```
l.add (new L());
```

```
List<? Extends A> ll = new ArrayList<>();  
ll.add (new A());
```

```
List<? super A> lll = new ArrayList<>();  
lll.add (new A());
```

