Université de Carthage
École Supérieure de la Statistique et de l'Analyse de l'Information

Mme Aïcha El Golli Jabbes

Année Universitaire **2022-2023**

Durée : 45 minutes **Java 1** $2^{ème}$ année

Nom et prénom : .....

Groupe : **AB**

---

1. (6 points) Soient les deux classes Mere et Fille suivantes. Qu'affiche le code suivant :

```
public class Mere {
        protected int j=3;
        public Mere(){
                j++;
                System.out.println("const Mere "+j);}
        public static int f(Mere u, int p) {
                 u.j= 2*p;
                 p=u.j;
                 p++;
                 return p;
                 }
        public String toString(){return ("j= "+j);}
        public void test() {System.out.println("Mere.test : " + this);}
}
public class Fille extends Mere {
        public int n ;
        public Fille(int n) {
                this.n = n ;
                System.out.println("const Fille "+ n);}
        public String toString() { return super.toString()+ " n= "+n ; }
        public void test1() {
                this.n++;
        System.out.println("Fille.test1 : " + this );
        }
        public void test() {
        super.test();
        System.out.println( "Fille.test : " + n +" " + j);
        }

        public static void main(String[] args) {
                Mere m = new Fille(2) ;
                System.out.println(m);
                m.j = 4 + Mere.f(m, 3);
                System.out.println(m);
                m.test() ;
                Fille c= new Fille(4);
                c.test1() ;
                System.out.println(c);
                c.test();}}
```

Solution: const Mere 4 (0.5pt)
const Fille 2 (0.5pt)
j= 4 n= 2 (0.5pt)
j= 11 n= 2 (0.5pt)
Mere.test : j= 11 n= 2 (0.5pt)
Fille.test : 2 11 (0.5pt)
const Mere 4 (0.5pt)
const Fille 4 (0.5pt)
Fille.test1 : j= 4 n= 5 (0.5pt)
j= 4 n= 5 (0.5pt)

Mere.test : j= 4 n= 5 (0.5pt)
Fille.test : 5 4 (0.5pt)

2. (7 points) Indiquez si les instructions suivantes sont correctes ou pas (ce qui se passe à la compilation et à l'exécution) et, pour les instructions qui vous semblent correctes, indiquez ce qui serait affiché, si il y a affichage.

	Compilation	**Exécution**
1) Mere b = new Fille(8) ; b.test1() ;	upcast (ok) erreur ne compile pas car test1 non membre de Mere **2pt**	
2) Mere b = new Fille(8) ; Fille h= (Fille)b ;	upcast (ok) downcast (ok) **1pt**	(ok) **1pt** const Mere 4 (**0.5pt**) const Fille 8 **0.5pt**
3) Object O = new Mere() ; Fille f= (Fille)O ;	upcast (ok) downcast (ok) **1pt**	Erreur ClassCastException **1pt** : l'objet O ne pointe pas Fille ou fille de la classe Fille

3. (7 points) Soient la classe A et la classe B suivantes :

```
public class A {
        protected int [] a;
        public ClassA(int[] n) {
                a= n;} }
class B extends A {
        protected int membreB;
        public B(int t, int x, int y, int z) {
                //a finir...
        }}
```

1. Finir d'écrire le constructeur de la classe B qui prend 4 entiers t, x, y et z et permet :
   — d'initialiser son tableau a avec ces 4 entiers et
   — d'initilaiser le membre b à la valeur 10 ;

Solution:

```
public B(int t, int x, int y, int z) {
                super(new int[] {t,x, y, z}); //1pt
                membreB=10;//1pt
                }
```

2. redéfinir la méthode equals au niveau de la classe A qui permet de vérifier si deux objets de classe A sont les mêmes ou pas. En effet, pour considérer deux objets de classe A égaux si ils ont deux tableaux avec la même taille et avec exactement les mêmes valeurs à la même position.
exemple : si ojb1 possède un tableau (a) constitué des valeurs $1, 2, 3$ et obj2 possède un tableau (a) constitué des valeurs $2, 1, 3$ votre méthode equals devra retourner false. En revanche : si ojb1 possède un tableau (a) constitué des valeurs $1, 2, 3$ et obj2 possède un tableau (a) constitué des valeurs $1, 2, 3$ votre méthode equals devra retourner true.

Solution:

```
@Override
        public boolean equals(Object o) {//0,5pt
                        if (o==this)return true; //0,5pt
                        if (!(o instanceof A)) return false; //1pt
        A r= (A)o; //1pt downcast pour pouvoir acceder aux attributs d'un objet de la classe A
        if(r.a.length!=a.length) return false; //0,5pt
        else{
                                for(int i=0; i<a.length; i++)//0,5pt
                                {
                                        if(r.a[i]!=a[i]) return false;//0,5pt
                                }
                                return true;}//0,5pt
                        }
```