TP02 Correction: Généricité

Matière : ATELIER PROGRAMMATION OBJET AVANCEE | **Classes : DSI2**

Enseignants : Equipe pédagogique

Exercice 1 :

Repérer les erreurs commises dans les instructions suivantes :

```
class C <T>{
    T x;
    T[] t1;
    T[] t2;
    public static T inf;
    public static int compte;

    void f() {
        x = new T();
        t2 = t1;
        t2 = new T[5];
    }
}
```

```
class C {
    T x; // OK
    T[] t1; // OK
    T[] t2; // OK
    public static T inf; // champ statique d'un type générique interdit
    public static int compte;

    void f() {
        x = new T(); // instanciation d'un type générique impossible
        t2 = t1; // OK
        t2 = new T[5]; // réservation mémoire d'un tableau d'un type générique
                       // impossible
    }
}
```

Exercice 2 :

Quels seront les résultats fournis par ce programme ?

```
public class TestStatic {
    public static void main(
            String args[]) {
        C ci = new C();
        ci.affiche();
```

```
        C cd = new C();
        ci.affiche();
        cd.affiche();
        Class cci = ci.getClass();
        Class ccd = cd.getClass();
        if (cci == ccd)
            System.out.println("ci et cd sont de la meme classe");
        else
            System.out.println("ci et cd ne sont pas de la meme classe");
        System.out.println(cci.getName() + " " + ccd.getName());
    }
}

class C {
    private static int compte = 0;
    public C() {
        compte++;
    }
    public void affiche() {
        System.out.println("compte = " + compte);
    }
}
```

compte = 1

compte = 2

compte = 2

ci et cd sont de la meme classe

C C

Exercice 3 :

```
public class Paire <T,U>{
    private T premier;
    private U second;

    public Paire(T premier, U second) {
        this.premier = premier;
        this.second = second;
    }

    public T getPremier() {
        return premier;
    }

    public void setPremier(T premier) {
        this.premier = premier;
    }

    public U getSecond() {
        return second;
    }

    public void setSecond(U second) {
        this.second = second;
    }
    public void afficher() {
        System.out.println("("+premier.toString()+", "+second.toString()+")");
```

```
    }

}
public class Pays {
    private String nom;
    private String continent;

    public Pays(String nom, String continent) {
        this.nom = nom;
        this.continent = continent;
    }

    @Override
    public String toString() {
        return "Pays:"+nom + ", " + continent;
    }
}
public class Drapeau {
    private String symboles;
    private String couleurs;

    public Drapeau(String symboles, String couleurs) {
        this.symboles = symboles;
        this.couleurs = couleurs;
    }
    @Override
    public String toString() {
        return "Drapeau:"+symboles + ", " + couleurs;
    }
}
public class TestPaire {
    public static void main(String[] args) {
        //façon1
        System.out.println("---------------TestPaire--------------");
        Paire<Pays, Drapeau>[] tPaysDrapeaux = new Paire[5];
        tPaysDrapeaux[0] = new Paire<Pays, Drapeau>(new Pays("Tunisie", "Afrique"), new
Drapeau("Cercle + Croissant + Etoile", "Rouge + Blanc"));
        tPaysDrapeaux[1] = new Paire<Pays, Drapeau>(new Pays("Algérie", "Afrique"), new
Drapeau("Croissant + Etoile", "Rouge + Blanc + Vert"));
        tPaysDrapeaux[2] = new Paire<Pays, Drapeau>(new Pays("Maroc", "Afrique"), new
Drapeau("Etoile", "Rouge + Vert"));
        tPaysDrapeaux[3] = new Paire<Pays, Drapeau>(new Pays("Libye", "Afrique"), new
Drapeau("Croissant + Etoile", "Rouge + Blanc + Vert + Noir"));
        tPaysDrapeaux[4] = new Paire<Pays, Drapeau>(new Pays("Mauritanie", "Afrique"), new
Drapeau("Croissant + Etoile", "Rouge + Vert + Jaune"));
        System.out.println("Affichage: Façon 1");
        for (int i = 0; i < tPaysDrapeaux.length; i++)
            tPaysDrapeaux[i].afficher();
        System.out.println("Affichage: Façon 2");
        for (Paire<Pays, Drapeau> p : tPaysDrapeaux)
            p.afficher();
        System.out.println("-------------Fin TestPaire------------");
    }
}
```

Exercice 4 :

```
public class PilePleine extends RuntimeException{
    public PilePleine() {
        super("La pile est pleine!");
    }
}
public class PileVide extends RuntimeException{
    public PileVide() {
        super("La pile est Vide!");
    }
}
import java.lang.reflect.Array;

public class Pile<T> {
    private static final int CAPACITE = 7;
    private T[] tElements;
    private int nbElements;
    public Pile(Class<?> classeDeT) {
        tElements = (T[]) Array.newInstance(classeDeT, CAPACITE);
        nbElements = 0;
    }
    public void empiler(T element){
        if(nbElements<CAPACITE){
            tElements[nbElements]=element;
            nbElements++;
        }else
            throw new PilePleine();
    }
    public T getSommet() {
        T sommet=null;
        if(nbElements>0)
            sommet=tElements[nbElements-1];
        else
            throw new PileVide();
        return sommet;
    }
    public T depiler(){
        T elementDepile=null;
        if(nbElements>0){
            elementDepile=tElements[nbElements-1];
            tElements[nbElements-1]=null;
            nbElements--;
        }else
            throw new PileVide();
        return elementDepile;
    }
    public boolean estVide(){
        return nbElements==0;
    }
    public int getHauteur(){
        return nbElements;
    }
    public void afficher(){
        if(nbElements>0) {
            System.out.println("------------");
            for (int i = nbElements - 1; i >= 0; i--) {
                System.out.println("\t" + tElements[i].toString());
                System.out.println("------------");
            }
        }else
            System.out.println("La pile est vide");
    }
}
public class Couleur {
    private String nom;
    public Couleur(String nom) {
        this.nom = nom;
```

```
    }
    @Override
    public String toString() {
        return nom;
    }
}
public class TestPile {
    public static void main(String[] args) {
        //Pile d'entiers
        Pile<Integer> pileInt=new Pile<Integer>(Integer.class);
        pileInt.empiler(10);
        pileInt.empiler(11);
        pileInt.empiler(12);
        pileInt.empiler(14);
        pileInt.empiler(16);
        pileInt.empiler(18);
        pileInt.empiler(20);
        System.out.println("Pile des entiers");
        System.out.println("----------------------------------------");
        System.out.println("Sommet de la pile: "+pileInt.getSommet());
        System.out.println("Hauteur de la pile: "+pileInt.getHauteur());
        pileInt.afficher();
        while(!pileInt.estVide())
            pileInt.depiler();
        pileInt.afficher();
        System.out.println("Hauteur de la pile: "+pileInt.getHauteur());

        //Pile de Strings
        Pile<String> pileString=new Pile<String>(String.class);
        pileString.empiler("C");
        pileString.empiler("B-");
        pileString.empiler("B");
        pileString.empiler("B+");
        pileString.empiler("A-");
        pileString.empiler("A");
        pileString.empiler("A+");
        System.out.println("Pile des Strings");
        System.out.println("----------------------------------------");
        System.out.println("Sommet de la pile: "+pileString.getSommet());
        System.out.println("Hauteur de la pile: "+pileString.getHauteur());
        pileString.afficher();
        while(!pileString.estVide())
            pileString.depiler();
        pileString.afficher();
        System.out.println("Hauteur de la pile: "+pileString.getHauteur());

        //Pile de Strings
        Pile<Couleur> pileCouleur=new Pile<Couleur>(Couleur.class);
        pileCouleur.empiler(new Couleur("VIOLET"));
        pileCouleur.empiler(new Couleur("INDIGO"));
        pileCouleur.empiler(new Couleur("BLEU"));
        pileCouleur.empiler(new Couleur("VERT"));
        pileCouleur.empiler(new Couleur("JAUNE"));
        pileCouleur.empiler(new Couleur("ORANGE"));
        pileCouleur.empiler(new Couleur("ROUGE"));
        System.out.println("Pile des Couleurs");
        System.out.println("----------------------------------------");
        System.out.println("Sommet de la pile: "+pileCouleur.getSommet());
        System.out.println("Hauteur de la pile: "+pileCouleur.getHauteur());
        pileCouleur.afficher();
        while(!pileCouleur.estVide())
            pileCouleur.depiler();
        pileCouleur.afficher();
        System.out.println("Hauteur de la pile: "+pileCouleur.getHauteur());
    }
}
```

Exercice 5 :

```
public class MinMax <T extends Comparable<T>> {
    private T first;
    private T second;

    public MinMax(T first, T second) {
        this.first = first;
        this.second = second;
    }

    public T getMin() {
        return first.compareTo(second) < 0 ? first : second;
    }

    public T getMax() {
        return first.compareTo(second) > 0 ? first : second;
    }
}
public class Couleur implements Comparable<Couleur> {
    private String nom;

    public Couleur(String nom) {
        this.nom = nom;
    }

    @Override
    public String toString() {
        return nom;
    }

    @Override
    public int compareTo(Couleur o) {
        return nom.compareTo(o.nom);
    }
}
public class TestMinMax {
    public static void main(String[] args) {
        //MinMax d'entiers
        MinMax<Integer> mmInt= new MinMax<Integer>(15,19);
        System.out.println("----------------------------------------");
        System.out.println("          MinMax des entiers");
        System.out.println("----------------------------------------");
        System.out.println("Min des entiers: "+mmInt.getMin());
        System.out.println("Max des entiers: "+mmInt.getMax());

        //MinMax de Strings
        MinMax<String> mmString= new MinMax<String>("Mohamed","Ali");
        System.out.println("----------------------------------------");
        System.out.println("        MinMax des Strings");
        System.out.println("----------------------------------------");
        System.out.println("Min des Strings: "+mmString.getMin());
        System.out.println("Max des Strings: "+mmString.getMax());

        //MinMax de Strings
        MinMax<Couleur> mmCouleur= new MinMax<Couleur>(new Couleur("ROUGE"),new
Couleur("BLEU"));
        System.out.println("----------------------------------------");
        System.out.println("       MinMax des Couleurs");
        System.out.println("----------------------------------------");
        System.out.println("Min des Couleurs: "+mmCouleur.getMin());
        System.out.println("Max des Couleurs: "+mmCouleur.getMax());


    }
}
```