Travaux pratiques d'informatique N° 22

Le but principal de cette séance est de vous permettre de vous familiariser avec la programmation générique, en général, et avec les notions suivantes, en particulier : la définition et l'instanciation des classes génériques, la définition et l'appel des méthodes génériques, ainsi que le mécanisme d'effacement de type et ses conséquences.

1. On présente ci-dessous le contenu du fichier ***CP_TP22Exo1.java***.

```
package cms_tp22;

abstract class FigurePlane{ }

class Ellipse extends FigurePlane{ }

class Polygone extends FigurePlane{ }

class Solitaire<T> {
	private T element;

	public T getElement() {
		return element;
	}

	public void setElement(T element) {
		this.element = element;
	}
}

public class CP_TP22Exo1 {
	public static void main(String[] args) {
		//ici seront introduite les lignes mentionnées ci-dessous
	}//fin de la méthode main
}//fin de la classe principale
```

1.1 Les trois lignes de code suivantes sont introduites dans le corps de la méthode `main`.

```
1		Solitaire<FigurePlane> sol = new Solitaire<>();
2		sol.setElement(new Ellipse());
3		Ellipse e = sol.getElement();
```

Choisir et encercler **la** réponse correcte.

Suite à l'exécution de la méthode `main` ci-dessus :

a) il n'y a pas d'erreur (ni à la compilation ni à l'exécution) ;

b) il y a une erreur à la compilation à la ligne 2 ;

c) il y a une erreur à l'exécution à cause de la ligne 2 ;

d) il y a une erreur à la compilation à la ligne 3 ;

e) il y a une erreur à l'exécution à cause de la ligne 3.

1.2 Les trois lignes de code suivantes sont introduites dans le corps de la méthode `main`.

```
1          Solitaire<FigurePlane> solBis = new Solitaire<>();
2          solBis.setElement(new Ellipse());
3          Polygone pol = (Polygone)solBis.getElement();
```

Choisir et encercler **la** réponse correcte.

Suite à l'exécution de la méthode `main` ci-dessus :

a) il n'y a pas d'erreur (ni à la compilation ni à l'exécution) ;

b) il y a une erreur à la compilation à la ligne 2 ;

c) il y a une erreur à l'exécution à cause de la ligne 2 ;

d) il y a une erreur à la compilation à la ligne 3 ;

e) il y a une erreur à l'exécution à cause de la ligne 3.

2. On présente ci-dessous le contenu du fichier ***CP_TP22Exo2.java***.

```
package cms_tp22;

import java.util.List;      //interface
import java.util.ArrayList; //classe qui implémente l'interface List

abstract class FigurePlane{ public abstract void afficher();}

class Ellipse extends FigurePlane{
      @Override public void afficher(){
         System.out.println("Je suis une ellipse !");
     }
}

class Cercle extends Ellipse{
      @Override public void afficher(){
         System.out.println("Je suis un cercle !");
     }
       public void afficherPlus(){
            System.out.println("Comme je suis rond !");
       }
}
```

```
class Polygone extends FigurePlane{
      @Override public void afficher(){
         System.out.println("Je suis un polygone !");
     }
}
class Presentation {
    static void afficherFigures(ArrayList<FigurePlane> figures){
        for(FigurePlane fig : figures){
            fig.afficher();
        }
    }
}
public class CP_TP22Exo2 {
    public static void main(String [] args){

        List<FigurePlane> figures = new ArrayList<FigurePlane>();

        figures.add(new Ellipse());

        figures.add(new Cercle());

        figures.add(new Polygone());

        figures.add(Integer.valueOf(11));

        System.out.println(figures.get(1));

        FigurePlane fp = figures.get(1);

        fp.afficherPlus();

        Integer i = figures.get(1);

        Cercle c1 = figures.get(1);

        Cercle c2 = (Cercle)figures.get(1);

        c2.afficherPlus();

        Presentation.afficherFigures(figures);

    }//fin de la main
}//fin de la classe principale
```

Le code ci-dessus contient plusieurs erreurs. Corriger d'abord ces erreurs et préciser ensuite quels seront les résultats affichés dans la fenêtre console suite à l'exécution du code corrigé.

3. On présente ci-dessous le contenu du fichier ***CP_TP22Exo3.java***.

```
package cms_tp22;

public class TabGen<T>{
     T[] tab;
     static int taille;
     static T pivot;
```

```
	TabGen(int taille){
		this.taille = taille;
		tab = new T[taille];
	}

	void addHead(T head){
		tab[0] = head;
	}

	void addHead(){
		tab[0] = new T();
	}

	T[] copy() {
		T[] tabRes = new T[taille];
		for(int i=0; i<taille; i++) {
			tabRes[i] = tab[i];
		}
		return tabRes;
	}
}//fin de la classe TabGen
```

Le code ci-dessus contient plusieurs erreurs. Trouver ces erreurs et expliquer leurs origines.

4. Ecrire du code source placé dans un fichier appelé ***CP_TP22Exo4.java***, en respectant les consignes suivantes :

- le fichier contient la définition de deux classes appartenant au package ***cms_tp22*** ;
- la première classe :
  - est appelée ***Solitaire*** et est définie sans modificateur d'accès ;
  - est générique avec une seule variable de type ;
  - déclare un champ de type variable de type, sans modificateur d'accès, nommé ***el*** et sans valeur initiale ;
  - déclare un champ statique de type entier, sans modificateur d'accès, nommé ***nb***, avec la valeur initial 0 et qui sert à compter les nombres d'objets créés ;
  - définit un constructeur avec un argument de type variable de type ;
  - redéfinit la méthode ***toString*** afin de retourner la concaténation entre le résultat retourné par l'appel de la méthode d'origine, le texte ***Je suis*** et le résultat retourné par l'appel de la méthode ***toString*** pour le champ ***el*** ;
- la deuxième classe :
  - est appelée ***CP_TP22Exo4***, est définie publique et contient la méthode ***main*** ;
  - sert à tester la classe ***Solitaire*** (en particulier les aspects concernant l'instanciation d'une classe générique, le typage sécurisé ("*safe type*" en anglais), l'effacement de type et le type brut, les champs statiques d'une classe générique).