TP 2 : Généricité

Matière : ATELIER PROGRAMMATION OBJET AVANCEE	**Niveau :** DSI 2
Enseignants : Equipe pédagogique	

Exercice 1 :

1. Créer une classe générique Paire qui contient deux éléments de n'importe quel type (les deux éléments peuvent être de types différents). La classe devrait avoir les méthodes suivantes :
   - Paire(…, …) : un constructeur avec deux paramètres.
   - getPremier : getter de premier.
   - getSecond : getter de second.
   - setPremier : setter de premier.
   - setSecond : setter de second.
   - afficher() : affiche les deux élément de la paire.
2. Soit le tableau "tPaysDrapeaux" suivant :

Pays		**Drapeau**	
nom	**continent**	**symboles**	**couleurs**
Tunisie	Afrique	Cercle + Croissant + Etoile	Rouge + Blanc
Algérie	Afrique	Croissant + Etoile	Rouge + Blanc + Vert
Maroc	Afrique	Etoile	Rouge + Vert
Libye	Afrique	Croissant + Etoile	Rouge + Blanc + Vert + Noir
Mauritanie	Afrique	Croissant + Etoile	Rouge + Vert + Jaune

3. Donner les deux classes Pays et Drapeau.
4. Ecrire une classe TestPaire qui permet de charger les données du tableau "tPaysDrapeaux" et de les afficher de deux façons (for et for each).

Exercice 2 :

Créer une classe générique Pile qui implémente une pile simple en utilisant un tableau de CAPACITE=7. La classe devrait avoir les méthodes suivantes :

- Pile(Class<?> classeDeT) : constructeur avec paramètre.
  NB :
  Pour instancier le tableau tBriques, utiliser l'instruction suivante :
  ```
  tElements = (T[]) Array.newInstance(classeDeT, CAPACITE);
  ```
- empiler(…) : ajoute un élément à la pile (peut lever l'exception PilePleine).
- getSommet : retourne le sommet de la pile (peut lever l'exception PileVide).
- depiler() : retire et retourne le dernier élément ajouté à la pile (peut lever l'exception PileVide).
- estVide() : retourne vrai si la pile est vide, faux sinon.
- getHauteur : retourne le nombre d'élément de la pile.
- afficher() : affiche les éléments de la pile

Donner une classe Couleur et une classe TestPile pour tester une pile d'entiers, une pile de Strings et une pile de Couleurs.

Exercice 3 :

Définir une classe générique pour représenter une famille. Il peut s'agir d'une famille d'être humain, d'une famille de chiens ou d'une famille de toute espèce vivante. Une famille comprend le père, la mère et un groupe d'enfants. Vous trouverez ci-dessous le code la classe de Test.

```
01. public class Test {
02.
03.   public static void main(String[] args) {
04.
05.     Human hf = new Human("Adam", 50);
06.     Human hm = new Human("Eve", 45);
07.     Human[] hc = new Human[3];
08.     hc[0] = new Human("Cain", 20);
09.     hc[1] = new Human("Abel", 15);
10.     hc[2] = new Human("Seth", 10);
11.
12.     Family<Human> fm = new Family<Human>(hf,hm,hc);
13.     fm.getFather().print();
14.     fm.getMother().print();
15.     for (int i=0; i<hc.length; i++)
16.       fm.getChild(i).print();
17.
18.     Dog df = new Dog("Jimmy", 8);
19.     Dog dm = new Dog("Julie", 7);
20.     Dog[] dc = new Dog[2];
21.     dc[0] = new Dog("toto", 1);
22.     dc[1] = new Dog("nimo", 2);
23.
24.     Family<Dog> fd = new Family<Dog>(df,dm,dc);
25.     fd.getFather().print();
26.     fd.getMother().print();
27.     for (int i=0; i<dc.length; i++)
28.       fd.getChild(i).print();
29.   }
30. }
```

Vous devez implémenter la classe générique **Family** ainsi que les classes conteneurs **Human** et **Dog**. Les classes **Human** et **Dog** doivent implémenter toutes les méthodes appelé le code donné précédemment.