TP4 : LES CONCEPTS AVANCES DE LA POO

Exercice 1 :

Définir une classe **Personne** contenant deux attributs privés : **nom** et **prenom.** Munir cette classe d'un constructeur permettant l'initialisation de ses attributs et de la méthode **toString**

Définir une classe **Etudiant** héritant de **Personne** et disposant d'un attribut privé **numInscription,** de type entier. Ajouter un constructeur qui permet de créer un étudiant à partir de son nom, son prénom et son numéro d'inscription et la méthode **toString**

Compléter la classe **Etudiants** qui gère un ensemble d'étudiants et changer les classes **Etudiant** et **Personne** en cas de besoin

```
class Etudiants {

   // attributs

   private Etudiant [] listeEtudiants;

   private int nbEtudiants ;

   // constructeur, reçoit en paramètre la capacité max du tableau
   listeEtudiants

   Etudiants (int n){//…}

   //ajoute un étudiant dans le tableau s'il n'existe pas déjà et si le tableau
   //n'est pas plein

   void ajouterEtudiant (Etudiant e){//…}

   // retourne l'étudiant ayant le numéro d'inscription passé en paramètre et
   // null sinon

   Etudiant rechercherEtudiant (int num) {//…}

   // affiche les informations sur tous le étudiants dans le tableau
   void listerEtudiants (){//…}

}
```

Ecrire une classe **Departement** contenant la méthode **main** et dans laquelle on testera les méthodes de la classe **Etudiants**

Exercice 2

Écrire les classes nécessaires au fonctionnement du programme suivant, en ne fournissant que les méthodes nécessaires à ce fonctionnement :

```
class TestMetiers {
  public static void main(String [] args) {
  Personne[] personnes = new Personne[4];
  personnes[0] = new Personne("Salah");
```

```
  personnes[1] = new Forgeron("Ali");
  personnes[2] = new Menuisier("Mohamed");
  personnes[3] = new Forgeron("Amor");
  for (int i=0 ; i<personnes.length ; i++)System.out.println(personnes[i]);

 } }
```

Sortie de ce programme :

```
Je suis Salah
Je suis Ali le forgeron
Je suis Mohamed le menuisier
Je suis Amor le forgeron
```

Exercice 3

Il s'agit de compléter le programme suivant :

```
interface Son {
	void parle();
}
absract class Mammifere implements Son {

	// Attributs
	String nom, son;
	int âge;
	// Constructeur
	Mammifere(String nom, String son, int age)
	{
		this.nom = nom;
		this.son = son;
		this.age = age;
	}
	// Méthode parle affiche le son d'un mammifère
	public void parle()
	{
		System.out.println(son+this);
	}
	// Méthode abstraite vitesse
	abstract void vitesse();
	//... à compléter
}
class Homme extends Mammifere {
	boolean marie;
	// ... à compléter
}
class Chien extends Mammifere {
	String race;
	// .... à compléter
}
class Test{
	public static void main(String [] args)	{
		Son []listeSons = new Son[2];
		Mammifere []listeMammiferes = new Mammifere[2];
		Homme h  = new Homme("Ali", "Bonjour", 20, true);
		Chien ch = new Chien("Snoopy", "Wouah!", 2,"caniches");
		listeSons[0] = h;
		listeSons[1] = ch;
		listeMammiferes[0] = h;
		listeMammiferes[1] = ch;
		for (int i=0; i<listeSons.length; i++)
		{
			// la méthode parle est polymorphe
			listeSons[i].parle();
			// la méthode vitesse est polymorphe
```

```
                    listeMammiferes[i].vitesse();
                }
        } // fin main
} // fin Test
```

A l'exécution de Test on obtient à l'écran :

```
Bonjour mon nom est Ali j'ai 20 ans je suis marié

Je cours à une vitesse de 20 km/h

Wouah! mon nom est Snoopy j'ai 2 ans j'appartiens à la race des caniches

Je cours à une vitesse de 30 km/h
```

Exercice 4

On vous demande d'écrire :

1- Une interface nommée **Homme** possédant une seule méthode :

`void identite( )` dont le rôle est d'afficher les informations concernant un homme.

2- Une classe **Personne** implémentant cette interface et possédant deux attributs privés "nom" et "prenom" de type String et un constructeur paramétré avec le nom et le prénom d'une personne.

3- Une classe **Client** héritant de la classe Personne et implémentant l'interface Homme avec un attribut privé supplémentaire nommé "numero" de type entier et un constructeur paramétré en conséquence.

4- Une classe **Peuple** contenant deux attributs privés : un tableau d'Hommes nommé pays de capacité 100 et un entier nbHommes représentant le nombre d'hommes dans pays.

Cette class contient également deux méthodes :

`void naissance (Homme h)` : permettant d'ajouter un homme dans pays

`void explorer ( )` : affiche l'identité de chaque homme dans pays

5- Tester les méthodes naissance ( ) et explorer ( ) après avoir placé dans pays des personnes et des clients.