TP 1 (partie 2) : environnement Java, utilisation de quelques classes

Exercice 1 : gestion des dates

Le but de cet exercice est d'essayer diverses manières d'obtenir et d'afficher la date courante (encore un prétexte pour vous faire chercher dans la documentation...).

Question 1 : première manière
La méthode `java.lang.System.currentTimeMillis()` donne la date courante, exprimée comme le nombre de millisecondes qui se sont écoulées depuis le 1er janvier 1970 à 0 heures GMT. C'est précis, mais pas très pratique pour organiser sa semaine ! (Retenez quand même l'existence de cette méthode, car elle est bien utile pour mesurer et comparer la performance des programmes).

Écrivez un programme qui affiche le nombre de secondes écoulées depuis le 1er janvier 1970. Exécutez deux fois ce programme, à une minute d'intervalle, et voyez si les valeurs obtenues correspondent à peu près à l'explication.

Question 2 : deuxième manière
Créez un objet de type `java.util.Calendar` par une expression comme

```
Calendar c = Calendar.getInstance();
```

et obtenez séparément les éléments de la date (le jour de la semaine, le jour du mois, le mois, l'année) pour les afficher comme bon vous semble.

En étudiant la documentation de la classe `Calendar` vous découvrirez qu'on obtient les divers composants par des expressions de la forme

```
c.get(Calendar.MONTH);
c.get(Calendar.YEAR) ;
```
etc.

Écrivez un programme pour afficher la date d'aujourd'hui sous la forme : le jour de la semaine (son numéro), le jour dans le mois, le numéro du mois et l'année.

Cette présentation n'est pas satisfaisante. On veut afficher la date du jour sous la forme : jour de la semaine en toutes lettres, puis jour du mois en chiffres, puis mois en lettres, puis année en chiffres (ex : mardi 17 janvier 2023). Des tableaux de chaînes déclarés comme ceci peuvent vous aider à obtenir une présentation adéquate:

```
String[] mois = { "janvier", "février", ... "décembre" };
String[] jSem = { "lundi", "mardi", ... "dimanche" };
```

Au lieu de ces tableaux, vous pouvez utiliser des `ArrayList<String>`. Voir la classe `java.util.ArrayList` dans la documentation.

```
ArrayList<String> listeJoursSemaine = new ArrayList<String>();
listeJoursSemaine.add("lundi");
listeJoursSemaine.add("mardi");
etc.
```

Question 3 : troisième manière (la meilleure)

Une autre manière consiste à construire un objet `d` de type `java.util.Date` et un objet `f` de type `java.text.SimpleDateFormat` et à afficher le résultat du formatage du premier par le second par une expression comme `f.format(d)`.

Ne vous laissez pas effrayer par la documentation de `SimpleDateFormat`. Si vous ne voyez pas comment cela marche, essayez ceci et vous comprendrez :

```
Date d = new Date();
SimpleDateFormat f=new SimpleDateFormat("dd MMMMM yyyy HH:mm");
System.out.println("maintenant: " + f.format(d));
```

Écrivez un programme pour afficher la date d'aujourd'hui (éventuellement l'heure) sous la forme :

- *jj / mois en lettres / année hh:mm*
- *jj / mm / aa* à *hh:mm:ss*
- *jour de la semaine* en toutes lettres, puis *jour du mois* en chiffres, puis *mois* en lettres, puis *année* en chiffres, puis *hh:mm:ss* (ex : `mardi 23 janvier 2018 15:47:29`).

Exercice 2 : Première classe

Le but de cet exercice est de définir et de manipuler une classe élémentaire ne proposant que des méthodes de classe pour les saisies.

Rappel : une classe est définie par le mot clé `class` et peut être structurée comme ceci :

```
class <nom de la classe> {
        <définitions de variables d'instance>      // utilisez le mot clé private pour
                                                   // forcer l'encapsulation
        <définitions de variables de classe>       // utilisez les mot clés static et
                                                   //private
        <définitions des méthodes d'instance>      // vous pouvez utilisez public
        <définitions de méthodes de classe>        // utilisez le mot clé static
}
```

Une **méthode de classe** est définie de la manière suivante :

```
static public <type retourné> <nom méthode> ( <liste d'arguments>) {

            <déclarations et instructions>
 }
```

Si le mot clé `static` est absent, c'est une **méthode d'instance**.

Le mot clé `return` délivre la valeur de la méthode. Si la méthode ne délivre rien (pas d'utilisation du mot clé `return`) le type retourné par la méthode est `void` (ce qui signifie vide). L'annexe donne un exemple de définition de classe.

Question 1/ Définissez la classe `Saisir`. Cette classe devra proposer les méthodes de classe `entier()`, `reeld()`, `reelf()`, `chaine()` et `car()` pour saisir respectivement un type primitif `int`, `double`, `float`, une instance de `String` et un type primitif `char`. Cette classe sera à rajouter dans tous les projets que vous allez créer par la suite (ce qui vous permettra de saisir des types primitifs).

Pour saisir au clavier un type primitif il est nécessaire de mettre la saisie dans une instance de la classe `String`. Les déclarations sont :

```
String ligne;
Scanner entree=new Scanner (System.in));
```

Le code pour une saisie est :

```
// saisie d'une ligne
System.out.println("Tapez une ligne");
ligne = entree.next();
```

Il faut alors convertir la saisie en type primitif. Voici quelques méthodes nécessaires aux conversions :

- la méthode de classe `parseInt(String str)` de la classe `Integer` a comme argument une instance de la classe `String` (elle délivre la conversion de cette instance en type primitif `int`),
- la méthode de classe `valueOf(String str)` de la classe `Double` a comme argument une instance de classe `String` (elle délivre la conversion de la chaîne en instance de `Double`),
- la méthode d'instance `doubleValue()` délivre la conversion d'une instance de `Double` en type primitif `double`,
- la méthode de classe `valueOf(String str)` de la classe `Float` a comme argument une instance de classe `String` (elle délivre la conversion de la chaîne en instance de `Float`),
- la méthode d'instance `floatValue()` délivre la conversion d'une instance de `Float` en type primitif `float`.

Question 2/ Les méthodes que vous avez écrites dans la question précédente sont déjà implémentées en Java. Étudiez et testez la classe `Scanner` de la documentation JAVA en ligne.

Annexe : exemple de classe

```
class Personne
{
	// variable d'instance donc privées
	private String nom;
	private int age;

	// variable de classe donc static
	static private int nb=0;

	// constructeur : de même nom que la classe
	// éventuellement plusieurs constructeurs
	public Personne(String lenom, int lage){
		nom=lenom;
		age=lage;
		nb++;
	}
	public Personne(){
		nom="Inconnu";
		age=-1;
		nb++;
	}
	// Méthodes d'instance
	public int getAge(){
		return this.age;
	}
	public String getNom(){
		return this.nom;
	}
```

```
	public void setNom(String nouveauNom){
		this.nom=nouveauNom;
	}

	public void setAge(int nouvelAge){
		this.age = nouvelAge;
	}

	// Méthode de classe
	static public int obtenirNb(){
		return nb;
	}

	// éventuellement une méthode main pour tester la classe
	static public void main(String args[]){
		Personne cLui, unAutre ;
		cLui = new Personne("Jean", 23) ;
		unAutre = new Personne() ;
		System.out.println("La variable cLui  est une instance de
Personne de nom:"+cLui.getNom()) ;
		System.out.println("La variable unAutre  est une instance de
Personne de nom:"+unAutre.getNom()) ;
	}

}
```