Quel médecin 1

TP du module 03 - La création de classes

Proposition de solution

2 - Création de la classe `MedecinGeneraliste` :

```
package fr.eni.ecole.quelMedecin.bo;

/**
 * Classe modélisant un médecin généraliste
 * @date 17 oct. 2018
 * @version POO - V1.0
 * @author hboisgontier
 */
public class MedecinGeneraliste {
	private String nom;
	private String prenom;
	private String numeroDeTelephone;
	private static int tarif = 25;

	/**
	 * Constructeur : crée une instance de Medecin
	 * @param nom nom du médecin
	 * @param prenom prénom du médecin
	 * @param numeroDeTelephone numéro de téléphone
	 */
	public MedecinGeneraliste(String nom, String prenom, String numeroDeTelephone) {
		this.nom = nom.toUpperCase();
		this.prenom = prenom;
		this.numeroDeTelephone = numeroDeTelephone;
	}

	/**
	 * Getter pour numeroDeTelephone.
	 * @return le numéro de téléphone
	 */
	public String getNumeroDeTelephone() {
		return this.numeroDeTelephone;
	}

	/**
	 * Setter pour numeroDeTelephone.
	 * @param numeroDeTelephone
	 *            le numéro de téléphone
	 */
	public void setNumeroDeTelephone(String numeroDeTelephone) {
		this.numeroDeTelephone = numeroDeTelephone;
	}

	/**
	 * Getter pour tarif.
	 * @return le tarif de la consultation
```

```
	 */
	public static int getTarif() {
		return MedecinGeneraliste.tarif;
	}

	/**
	 * Setter pour tarif.
	 * @param tarif
	 *            le tarif de la consultation
	 */
	public static void setTarif(int tarif) {
		MedecinGeneraliste.tarif = tarif;
	}

	/**
	 * Affiche sur la console sous la forme :
	 * NOM Prénom
	 * Téléphone : XXXXXXXXXX
	 * Tarif : XX€
	 */
	public void afficher() {
		System.out.printf("%s %s%nTéléphone : %s%nTarif : %d€%n",
				this.nom, this.prenom,
				this.numeroDeTelephone, MedecinGeneraliste.tarif);
	}
}
```

L'attribut `tarif` et les méthodes `getTarif()` et `setTarif()` sont déclarés avec le mot-clé `static` pour indiquer que ce sont des éléments de classe, c'est-à-dire communs à toutes les instances et non spécifiques à chacune.

Pour coder la fonction `afficher()`, il y a différentes possibilités. Celle proposée ici s'appuie sur la méthode `printf()`, mais il est également possible de créer la chaîne de caractères à l'aide de la classe `StringBuilder`. Cela est néanmoins déconseillé, pour éviter la création de plusieurs instances en mémoire de `String`.

3 - Création de la classe `Patient` :

```
package fr.eni.ecole.quelMedecin.bo;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;

/**
 * Classe modélisant un patient
 * @date 17 oct. 2018
 * @version POO - V1.0
 * @author hboisgontier
 */
public class Patient {
	private String nom;
	private String prenom;
	private String numeroDeTelephone;
	private char sexe;
	private long numSecu;
	private LocalDate dateNaissance;
```

```
        private String commentaires;

        /**
         * Constructeur
         * @param nom
         *            nom du patient
         * @param prenom
         *            prénom du patient
         * @param numeroDeTelephone
         *            numéro de téléphone du patient
         * @param sexe
         *            sexe du patient : 'F' pour Féminin ou 'M' pour Masculin
         * @param numSecu
         *            numéro de Sécurité sociale du patient
         * @param dateNaissance
         *            date de naissance du patient
         * @param commentaires
         *            commentaires associés à ce patient
         *            (allergies, antécédents médicaux…)
         */
        public Patient(String nom, String prenom, String numeroDeTelephone, char sexe,
                            long numSecu, LocalDate dateNaissance, String commentaires) {
                this.nom = nom.toUpperCase();
                this.prenom = prenom;
                this.numeroDeTelephone = numeroDeTelephone;
                this.sexe = sexe;
                this.numSecu = numSecu;
                this.dateNaissance = dateNaissance;
                this.commentaires = commentaires;
        }

        /**
         * Affiche sur la console sous la forme :
         * NOM Prénom
         * Téléphone : XXXXXXXXXX
         * Sexe : Féminin ou Masculin
         * Numéro de Sécurité sociale XXXXXXXXXXXXXXX
         * Date de naissance : XX mois XXXX
         * Commentaires : XXXXXXXXXXXXXXX ou [aucun commentaire]
         */
        public void afficher() {
                System.out.printf("%s %s%nTéléphone : %s%nSexe : %s%n" +
          "Numéro de Sécurité sociale : %d%nDate de naissance : %s%nCommentaires : %s%n",
                                        this.nom, this.prenom, this.numeroDeTelephone,
                                this.sexe == 'F' ? "Féminin" : "Masculin", this.numSecu,
          this.dateNaissance.format(DateTimeFormatter.ofLocalizedDate(FormatStyle.LONG)),
                        this.commentaires != null ? this.commentaires : "[aucun commentaire]");
        }
}
```

La principale difficulté de cette classe est l'affichage de la date de naissance. Pour cela, la méthode `format()` est utilisée. Elle prend en paramètre une instance de `DateTimeFormatter`. La documentation de cette classe indique qu'il existe une méthode de classe `ofLocalizedDate()` permettant de créer une telle instance. Cette méthode prend en paramètre une valeur de l'énumération `FormatStyle` (`FULL`, `LONG`, `MEDIUM` ou `SHORT`) permettant de choisir le niveau de détail souhaité.

*4 - Création de la classe **Adresse** :*

```
package fr.eni.ecole.quelMedecin.bo;

/**
 * Classe modélisant une adresse française en respectant
 * les recommandations de la poste.
 * @date 17 oct. 2018
 * @version POO - V1.0
 * @author hboisgontier
 */
public class Adresse {
	private String mentionsCompl;
	private int numero;
	private String complNumero;
	private String rue;
	private int cp;
	private String ville;

	/**
	 * Constructeur
	 * @param mentionsCompl
	 *            mentions complémentaires éventuelles (comme l'appartement,
	 *            l'étage, l'escalier, « chez… », le bâtiment ou la résidence)
	 * @param numero
	 *            numéro dans la voie
	 * @param complNumero
	 *            complément facultatif de numéro tel bis, ter, quater...
	 * @param rue
	 *            type de voie (rue, avenue, etc.) et nom de celle-ci
	 * @param cp
	 *            code postal
	 * @param ville
	 *            nom de la commune
	 */
	public Adresse(String mentionsCompl, int numero, String complNumero, String rue,
			int cp, String ville) {
		this.mentionsCompl = mentionsCompl;
		this.numero = numero;
		this.complNumero = complNumero;
		this.rue = rue;
		this.cp = cp;
		this.ville = ville.toUpperCase();
	}

	/**
	 * Constructeur
	 * @param numero
	 *            numéro dans la voie
	 * @param complNumero
	 *            complément facultatif de numéro tel bis, ter, quater...
	 * @param rue
	 *            type de voie (rue, avenue, etc.) et nom de celle-ci
	 * @param cp
	 *            code postal
	 * @param ville
	 *            nom de la commune
	 */
```

```
	public Adresse(int numero, String complNumero, String rue, int cp, String ville){
		this(null, numero, complNumero, rue, cp, ville);
	}

	/**
	 * Affiche sur la console sous la forme :
	 * Complément
	 * XXbis rue XXXXXXXXX
	 * 00000 XXXXXXXXXXXX
	 */
	public void afficher() {
		if (this.mentionsCompl != null)
			System.out.println(mentionsCompl);
		System.out.printf("%d%s %s%n%05d %s%n", this.numero,
	this.complNumero != null ? this.complNumero : "", this.rue, this.cp, this.ville);
	}
}
```

Une petite subtilité est d'arriver à afficher un zéro pour les codes postaux dont le numéro de département est inférieur à 10. Une première manière de faire est de vérifier si le code postal est inférieur à 10000 et, auquel cas, écrire un zéro. Dans le code ci-dessus, une autre approche a été employée : l'utilisation de `%05d` comme indicateur dans le format de la fonction `printf()`. Le `%d` indique l'affichage d'un nombre entier en base 10 et le `05` indique que la valeur doit s'afficher sur cinq caractères en complétant le cas échéant avec des zéro non significatifs devant.