Programmation Orientée Objet
TD N°2 : classes, instances et références

Exercice 1

Soit la classe suivante décrivant une montre:

```
public class Montre
{
      private int heure  ;
      private int minute ;

      public Montre()
      {
        this.heure  = 0 ;
        this.minute = 0 ;
      }

      public Montre(int h, int m)
      {
        this.heure = h ;
        this.minute = m ;
      }

      public int getHeure()
      {
            return this.heure ;
      }

      public int getMinute()
      {
            return this.minute ;
      }

      public void setHeure(int h)
      {
            this.heure = h ;
      }

      public void setMinute(int m)
      {
            this.minute  = m ;
      }

      public static void  main(String[] args)
      {
          (1) Montre m1 ;
          (2)  m1 = new Montre( ) ;

          (3) Montre m2 ;
          (4) m2 = new Montre(13, 42) ;

          (5) Montre m3 = m2 ;

          (6) System.out.println(m2.getHeure()) ;
          (7) System.out.println(m2.heure) ;
          (8) System.out.println(m3.getHeure()) ;

          (9) m1.setMinute(21) ;
          (10) m1.minute = 22   ;
```

```
        (11) System.out.println(m1.getMinute()) ;

        (12) m1 = m2 ;
    }
```

} *// fin classe Montre*

a) Préciser quelles sont les variables d'instances, les constructeurs, les méthodes d'accès en lecture, et les méthodes d'accès en écriture.
b) Préciser les services que rend cette classe (donner la vue publique).
c) Dessiner les références, les instances et leurs liens créés par les instructions (1) (2) (3) (4) et (5).
d) Qu'affiche les instructions (6), (7) et (8) ? Expliquer pourquoi…
e) Qu'affiche l'instruction (11) ?
f) Redessiner les liens entre instances et références après l'instruction (12). Que devient l'instance qui était référencée par *m1* avant cette instruction ?

Soit la classe suivante :

```
public class TestMontre
{
    public static void main(String[] args)
    {
        (i) Montre m1 = new Montre( ) ;
        (ii) Montre m2 = new Montre(13, 42) ;

        (iii) System.out.println(m2.getHeure()) ;
        (iv) System.out.println(m2.heure) ;

        (v) m1.setMinute(21) ;
        (vi) m1.minute = 22    ;
        (vii)) System.out.println(m1.getMinute()) ;

    } // fin main
```

} *// fin classe TestMontre*

g) Quelles instructions sont refusées par le compilateur et pourquoi ?
h) Qu'en déduisez-vous concernant le test d'une classe ?

Exercice 2

Soit la classe *Telephone*, qui décrit la gestion d'un poste téléphonique, rendant les services suivants (vue publique) :

Constructor Summary	
Telephone(int num)	initialise une instance de *Telephone* avec *num* comme numéro d'appel. à sa création le téléphone est libre et n'a ni émis ni reçu d'appel.

Method Summary	
	affiche() : affiche l'état (valeur des variables) du téléphone
	appelle() : le téléphone effectue un appel
	decroche() : décroche le téléphone (qui devient occupé)
	estLibre() : retourne *true* si le téléphone est libre, *false* sinon
	getNbAppelsEmis() : retourne le nombre d'appels émis par le téléphone
	getNbAppelsRecus() : retourne le nombre d'appels reçus par le téléphone
	getNumero() : retourne le numéro d'appel du téléphone
	raccroche() : raccroche le téléphone (qui devient libre)
	repond() : le téléphone répond à un appel
	setNumero(int nouveauNumero) : *nouveauNumero* devient le nouveau numéro d'appel du téléphone

Nous donnons ci-dessous une partie (les variables d'instances) de la classe *Telephone* qu'il faudra compléter de façon à assurer les services proposés :

```
public class Telephone
{
      private int     numero        ; // numéro du poste
      private boolean libre         ; // true = libre, false = occupé
      private int     nbAppelsRecus ; // nombre d'appels recus
      private int     nbAppelsEmis  ; // nombre d'appels emis
      ...
} // fin classe Telephone
```

De plus, pour tester la classe *Telephone* on utilisera une classe *TestTelephone* ne comprenant qu'une méthode *main* qui sera aussi complétée au fur et à mesure des questions.

a) Écrire le constructeur de *Telephone*. Compléter la méthode *main* de *TestTelephone* en créant 2 téléphones *t1* et *t2* ayant *10* et *20* comme numéros d'appels respectifs. Dessiner les instances, les références et leur lien.
b) Écrire les méthodes *getNumero()*, *getNbAppelsEmis()*, *getNbAppelsRecus()*, *estLibre()*. Compléter le *main* de *TestTelephone* en affichant les valeurs des variables d'instances de *t1*.
c) Écrire la méthode *affiche*
d) Écrire la méthode *setNumeroAppel(...)*. Compléter le *main* de *TestTelephone* en changeant le numéro d'appel de *t1* avec la valeur *30*, puis afficher l'état de *t1* (les valeurs de ses variables d'instances).
e) Écrire les méthodes *decroche* et *raccroche*. Compléter le *main* de *TestTelephone* : vérifier que *t1* est libre, si oui décrocher *t1*, afficher son état, puis raccrocher *t1* et afficher de nouveau son état.
f) Écrire les méthodes *appelle* et *repond* chacune d'elle se contentant de décrocher et de mettre à jour les variables d'instances concernées par la situation (on suppose que tout appel réussit). Compléter le *main* de *TestTelephone* : vérifier que *t2* est libre, si oui faire appeler *t2*, afficher son état, puis raccrocher *t2* et afficher de nouveau son état. Faire la même chose pour *t1* avec *repondre*.