Sujet du TD 2 1920v1

Objectifs

Ce TD est appelé TD2 non pas parce-que c'est le deuxième, mais parce qu'il appartient à la séquence 2. Il a pour but de vous entraîner à écrire sur papier *(comme en contrôle)* une classe complète, avec ses attributs et ses méthodes.
Pour chaque exercice, **lire les 2 parties Q**(uestion) **et R**(éponse) **avant** de commencer à le résoudre.

Quelques rappels

- Une classe (ex: `Cercle`) possède des attributs (ex: `aDiametre`) d'un certain type (ex: `int`).

- Un constructeur permet d'initialiser les attributs, soit à des valeurs fixées dans ses instructions, soit à des valeurs qu'on lui passe en paramètres (entre les parenthèses). Il ne comporte aucun mot entre `public` et son nom (qui doit être identique à celui de la classe). Il est appelé automatiquement à la création de chaque objet de sa classe.

- Une procédure (ex: `vaDroite`) ne retourne rien, ce qui est indiqué dans sa définition par le mot `void` entre `public` et le nom de la procédure.

- Mais on peut aussi définir une fonction qui retourne toujours une et une seule valeur, dont le type doit être précisé entre `public` et le nom de la fonction. Ses instructions se terminent toujours par un `return` suivi de la valeur qu'elle retourne.

- Lorsqu'il n'y a pas de paramètres, tant la définition que l'appel d'une fonction, d'une procédure, ou d'un constructeur se termine quand-même obligatoirement par `()`.

- On notera que la classe `Cercle` possède ainsi deux méthodes de même nom : `Cercle`. Ceci ne crée pas d'ambiguïté car ces deux méthodes ont des signatures différentes. Cette possibilité qu'offre Java est appelée *surcharge*.

- `a = b % c;` met dans `a` le reste de la division entière de `b` par `c`. `%` est l'opérateur souvent appelé « modulo ».

- le `+` appliqué à une `String` permet de lui concaténer la représentation textuelle de n'importe quel objet ou nombre.

- ne pas confondre l'affectation `a = b;` qui change la valeur de `a`, avec la comparaison `a == b` qui vaut vrai ou faux, mais qui ne modifie rien.

- l'instruction SI condition_est_vraie ALORS instructions_1 SINON instructions_2 FINSI se traduit par :
`if (condition) { instructions_1 } else { instructions_2 }`

- Un appel récursif survient quand une méthode s'appelle elle-même (avec une valeur de paramètre qui a changé).

- Il est possible d'afficher les valeurs de vos variables avec `System.out.println(…);`

- Ci-après, le code java (corrigé) de `MaPremiereClasse` pour pouvoir vous en inspirer.

Ma première classe

```
public class MaPremiereClasse
{
  // ### Attributs ###
  private int     aPremierAttribut;
  private boolean aDeuxiemeAttribut;
  private String  aTroisiemeAttribut;

  // ### Constructeurs ###
  /**
   * Construit toujours le meme objet : 2, true, "exemple"
   */
  public MaPremiereClasse()
  {
      this.aPremierAttribut   = 2;
      this.aDeuxiemeAttribut  = true;
      this.aTroisiemeAttribut = "exemple";
  } // MaPremiereClasse()

  /**
   * Constructeur naturel
   * @param pI entier  pour initialiser aPremierAttribut
   * @param pB booleen pour initialiser aDeuxiemeAttribut
   * @param pS chaine  pour initialiser aTroisiemeAttribut
   */
  public MaPremiereClasse( final int pI, final boolean pB, final String pS )
  {
      this.aPremierAttribut   = pI;
      this.aDeuxiemeAttribut  = pB;
      this.aTroisiemeAttribut = pS;
  } // MaPremiereClasse()

  // ### Accesseurs ###
  public int     getPremierAttribut()   { return this.aPremierAttribut;   }
  public boolean getDeuxiemeAttribut()  { return this.aDeuxiemeAttribut;  }
  public String  getTroisiemeAttribut() { return this.aTroisiemeAttribut; }

  // ### Modificateurs ###
  public void setPremierAttribut(   final int     pPremierAttribut )
      { this.aPremierAttribut   = pPremierAttribut; }
  public void setDeuxiemeAttribut(  final boolean pDeuxiemeAttribut )
      { this.aDeuxiemeAttribut  = pDeuxiemeAttribut; }
  public void setTroisiemeAttribut( final String  pTroisiemeAttribut )
      { this.aTroisiemeAttribut = pTroisiemeAttribut; }

  // ### Autres methodes ###
  /**
   * Un exemple de procedure (qui ne retourne rien)
   */
  public void procedure()
  {
    System.out.println( "La procedure s'est bien executee." );
    return; // facultatif
  } // procedure()

  /**
   * Un exemple de fonction booleenne
   * @return toujours true
   */
  public boolean fonction()
  {
    return true;
  } // fonction()
} // MaPremiereClasse
```