UE Programmation Orientée Objet

TD 2

Exercice 1 : Complexes

Q 1 . Quelles fonctionnalités (⇒ méthodes) ?

- Par exemple : `conjugué`, `estRéel`, `partieRéelle`, `partieImaginaire`, `module`, `add`, `mult`.
- pensez aussi aux méthodes "classiques" : `equals` et `toString`.

Q 2 . Constructeur(s) ?

Q 3 . Quels sont les attributs pour réaliser ces méthodes ?

Q 4 . Faire les déclarations (classe, attributs, signatures de méthodes et de constructeurs).

Q 5 . Codez les méthodes.

Exercice 2 : Points et Droites

Q 1 . Définir une classe pour les points du plan.

Q 2 . Définir une classe pour les droites, définies par deux points ($\neq$).

Q 3 . Proposer des méthodes pour ces deux classes (seules les signatures et la documentation sont demandées).

Exercice 3 : Date

Q 1 . Faites une proposition pour représenter (modéliser) des `Date`s avec jour (quantième), mois, annee

Q 2 . méthodes : `equals`, `toString`, comparer 2 dates, calcul de la date du lendemain, écart en jour entre 2 dates, date atteinte n jours après une date donnée.

Q 3 . Comment représenter les mois ? Comment connaître le nombre de jours dans un mois ?

Q 4 . Donnez les tests pour la méthode qui calcule la date du lendemain.

Q 5 . La méthode qui calcule la date atteinte n jours après doit déclenchr une exception `IllegalArgumentException` si son paramètre est négatif.

Q 6 . Ecrire les tests pour cette méthode, y compris le test de l'exception.

Q 7 . Ecrire les codes java des types nécessaires.

Exercice 4 : Ampoules et Interrupteurs (voir le sujet dans le TP2).

Q 1 . Définir une classe `Lightbulb` : constructeurs ? méthodes ? attributs ?

Q 2 . Définir une classe `Switch` : un interrupteur commande une ampoule. Idem.

Q 3 . Définir une classe `MultiSwitch` dont les instances sont des interrupteurs qui permettent de contrôler (« simultanément ») plusieurs ampoules stockées dans un tableau.

Exercice 5 : Temps (durée)
Il s'agit de modéliser des Temps (dans le sens durée) typiquement dans le cadre d'un championnat ou d'une course (100m, Marathon, Formule 1, Tour de france, ...). On veut pouvoir ajouter des temps, en soustraire, les comparer, les avoir sous forme de chaîne de caractère, etc.
En complément on peut envisager l'implémentation d'un constructeur faisant appel à des méthodes de la classe pour initialiser l'instance :

```
public Time(int days, int hours, int min, int sec, int ms){
   this.ms=0;                  // ou remplacer ces lignes
   this.sec=0;                 // par un appel à un constructeur Time()
   this.min=0;                 // qui initialiserait tout à 0
   this.hours=0;               // la syntaxe est alors simplement :
   this.days=0;                // this();

   this.addMs(ms);
   this.addSec(sec);
   this.addMin(min);
   this.addHours(hours);
   this.addDays(days);
}
```

et dans la même idée, par exemple, un autre constructeur se basant sur un début et une fin sous la forme de timestamp (en ms par rapport à une date de référence) `Time(int begin, int end)`