Examen de rattrapage

(Durée 1h30)

Exercice 1 :

On appelle monôme une application $x \rightarrow ax^n$, $x \in R$, $a \in R$, $n \in N$. ***a*** est appelé le coefficient du monôme et ***n*** son degré. On veut modéliser un monôme sous la forme d'un objet défini par son degré et son coefficient.

1. Coder une classe *Monome* qui encapsulera le degré et le coefficient. Coder également le(s) constructeur(s) nécessaire(s).
Une exception *ArithmeticException* devra être levée si on essaye de construire un monôme avec un degré négatif.
Coder les méthodes *getDegre* et *getCoefficient* qui permettront une encapsulation efficace.

2. Coder une méthode appelée *calcul* permettant de calculer la valeur du monôme pour une valeur donnée en paramètre de x.

3. Surcharger les méthodes

- String toString()permettant de représenter le monôme sous la forme de la chaîne ax^n (en remplaçant a et n par leurs valeurs)
- boolean equals(Monome m) permettant de comparer l'instance courante à un autre monôme (2 monômes sont égaux si leurs coefficients et leurs degrés sont identiques)

4. Coder une méthode statique appelée derivee qui crée et retourne un nouveau monôme qui est la dérivée du monôme passé en paramètre.
Rappel : la dérivée de ax^n est anx^{n-1} si n>0 ; la dérivée de ax^0 est $0x^0$.

5. Coder la méthode main permettant de saisir au clavier le coefficient et le degré d'un monôme qui sera ensuite construit. Répéter la saisie d'un nombre x et l'affichage de la valeur du monôme pour ce paramètre x jusqu'à ce qu'on saisisse la valeur 0.

6. On cherche au cours de cet exercice à modéliser un polynôme comme un tableau de Monomes.
Ecrire une classe *Polynome* qui sera composée d'un tableau T de Monome et de sa taille t.

- Ecrire la méthode *getMonome* qui retourne le Monome à l'indice i du tableau.
- Ecrire la méthode public int Ajoute (Monome m) qui ajoute un monome à T et qui retourne le nombre de monome dans T. Si un monome de même degré est déjà présent dans T, on le remplacera par un monome de ce degré et dont le coefficient est la somme des coefficients du monome ajouté et du monome déjà présent, sauf si cette somme est nulle.
- Ecrire une méthode public *String toString()* qui présente T sous la forme d'une chaîne. On veillera à obtenir une présentation élégante du style P(x)=3x^2-2x+1 en évitant d'avoir plusieurs signes dans le cas d'un monome négatif.
- Ecrire une méthode *public double calcul(double x)* qui calcule la valeur du polynome pour une valeur de x, en faisant la somme de toutes les valeurs des monomes de T pour cette valeur de x

7. La méthode *calcul* n'est pas optimale, car on effectue de nombreuses fois le calcul de x^n. On cherche à mettre en œuvre le schéma de Horner qui écrit le polynome

$P(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_o$ sous la forme :

$P(x) = ((a_4 x + a_3) x + a_2) x + a_1) x + a_0$

Ecrire une méthode récursive *calculHorner* qui effectue le calcul avec cette méthode. NB : il sera judicieux de trier préalablement la liste des monomes.

8. Ecrire la méthode main où étant donné un polynome permet d'afficher pour un x donné la valeur de ce polynome.

Exercice 2 :

(a) Écrire une classe ***A*** contenant deux données membres ***int*** privées nommées ***i et j***. Cette classe aura un constructeur public, ***A(int a, int b)*** qui initialise ces deux données membres et une méthode publique ***getj*** qui retourne la valeur de la donnée membre ***j***. Redéfinir la méthode ***toString*** de façon à obtenir une chaîne de caractères donnant les valeurs des données membres.

(b) Écrire une classe ***B*** dérivée de la classe ***A*** et qui contient une nouvelle donnée membre privée de type ***int*** nommée ***j***. Combien de données membres contient un objet ***B*** ? Définir un constructeur ***B(int a, int b, int c)*** pour ***B*** qui initialise toutes les données membres de ***B***.

Si un objet de ***B*** est créé par ***new B(1,2,3)***, que retournera la méthode ***getj*** appliquée à cet objet. Redéfinir ***toString*** de façon à obtenir une chaîne de caractères donnant les valeurs de toutes les données membres.

(c)

i. Que produira le code suivant ? (erreur à la compilation, à l'exécution, et s'il n'y a pas d'erreur quelle en est la sortie) :

```
A[] ta= new A[10];
for (int i = 0; i < ta.length; i++) ta[i]=new B(i,i+1,i+2);}
System.out.println(ta[1]);
```

ii. Même question pour :

```
B[] tb= new B[10];
for (int i = 0; i < tb.length; i++) tb[i]=new B(i,i+1,i+2);
A[] tab=tb;
System.out.println(tab[1]);
```

iii. Même question pour :

```
A[] tac= new B[10];
tac[1]= new A(1,2);
```

```
System.out.println(tac[1]);
```

Exercice 3 :

(a) On considère les classes:

class X1 { … }
class Y1 extends X1 { … }

Soit le code suivant (choisir la ou les bonnes réponses) :
X1[ ] tx1=new X1[2] ; tx1[0]=new Y1( ) ;

(a) il provoque une erreur à la compilation
(b) il provoque une erreur à l'exécution (exception)
(c) il ne provoque pas d'erreur

(b) On considère les classes:

class X1 { … }
class Y1 extends X1 { … }

Soit le code suivant (choisir la ou les bonnes réponses) :
X1[ ] tx1=new Y1[2] ; tx1[0]=new X1( ) ;

(a) il provoque une erreur à la compilation
(b) il provoque une erreur à l'exécution (exception)
(c) il ne provoque pas d'erreur

(c) On considère les classes:

class X1 { … }
class Y1 extends X1 { … }

Soit le code suivant (choisir la ou les bonnes réponses) :
Y1[ ] tx1=new X1[2] ; tx1[0]=new X1( ) ;

(a) il provoque une erreur à la compilation
(b) il provoque une erreur à l'exécution (exception)
(c) il ne provoque pas d'erreur

(d) On considère les classes :

```
class A {
        int i=0 ;
        void f() { System.out.println(i) ; }
        void f(int i) {  System.out.println(this.i) ;}
}//fin de A

class B extends A {
        int  i=100 ;
        void f() { System.out.println(i) ;  f(i) ;  }
}
```

Que fera le code : *A a=new B( ) ; a.f( ) ;*
(a) il ne passe pas à la compilation
(b) il affiche 0

(c) il affiche 100 puis 0

(d) il affiche 100 puis 100 ***Bon courage***