Examen Java

Janvier 2019

Durée : 3 heures

Instructions

(Barême ⋆)

- L'examen sera réalisé avec **Eclipse** et **git**. Répondre aux questions libres sur copie.
- Utiliser
  - `JDK >= 1.8`
  - `JUnit 5`
  - `Eclipse Photon`
- **RESPECTER LES INSTRUCTIONS SUIVANTES À LA LETTRE**
- Le repository à utiliser est **https://github.com/CNAMPRO/18_stmn_java**.
- Travailler sur **votre** branche.
- Créer un nouveau projet `Eclipse` **CR** localisé sous `[racine_18_stmn_java]/Exam/CR`, `[racine_18_stmn_java]` étant le répertoire de votre clone git de **https://github.com/CNAMPRO/18_stmn_java**.
- Pour chaque exercice, créer un nouveau package portant le nom de l'exercice en minuscule (e.g. **exercice1**) et y inclure toutes les classes créées pour répondre à l'exercice.
- Lorsque des vérifications ou des tests sont demandés dans un exercice, ne **PAS** utiliser de `main` ..., sauf instructions contraires.
- Ne **PAS** modifier la signature des fonctions/méthodes donnée dans les énoncés.
- À la fin de l'examen, faire un `commit+push` de tout le **code** correspondant à vos réponses.
  hint :

```
git add <files>
git commit
git push
```

- Vérifier que vos réponses sont disponibles dans votre branche sur **https://github.com/CNAMPRO/18_stmn_java**.

Set up initial

- Télécharger le fichier **https://github.com/CNAMPRO/18_stmn_java/blob/teacher/setup/cr/alpha/setup.zip**
- Le décompresser à la racine de votre projet `CT`. Le répertoire `src` doit alors contenir le répertoire `exercice3`.
- Actualiser votre projet **Eclipse** en le sélectionnant dans **Eclipse** et en appuyant sur la touche *F5*, ou par un clic droit -> *Refresh*.

Exercice 1

(Barême ⋆⋆)
On définit un polynôme P de degré au plus n comme suit :

$$P(x) = \sum_{i=0}^{n} a_i x^i, \quad \text{où } a_i \in \mathbb{R} \text{ désigne un } \textit{coefficient}.$$

La classe `Polynom`, définie dans la figure 1, permet de modéliser un tel polynôme.
Cette classe possède les méthodes suivantes :

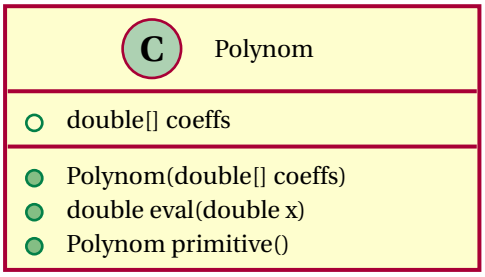


FIGURE 1 – Polynom class diagram

- `eval` permettant d'évaluer le polynôme en un point `x`.
- `primitive` permettant de calculer *une* primitive du polynôme.

1. Implémenter complètement la classe `Polynom`.
2. Tester l'implémentation de la méthode `eval` avec le polynôme $P(x) = 2 - 3x + x^2$ au points $0, 1, 2$.
3. Tester l'implémentation de la méthode `primitive` avec les polynômes $P_i(x) = x^i$ avec $i \in [0, 3]$.
   Hints :
   - Si Q_i =`primitive`(P_i), sous quelle forme s'écrit $Q_i(x)$?
   - Pour vérifier que 2 objets `Polynom` sont égaux, on pourra vérifier que leur attribut `coeffs` sont égaux.

Exercice 2

(Barême ⋆ ⋆ ⋆)
On souhaite évaluer différentes méthodes de calcul d'intégrale d'une fonction $f : [a, b] \implies \mathbb{R}$ et évaluer la quantité $\int_a^b f(x)dx$.
Dans un premier temps, on s'intéresse à la somme de Riemann de la fonction $sinus$.
Pour rappel, la somme de Riemann associée à une fonction f est :

$$S_n = \frac{b-a}{n} \sum_{k=1}^{n} f(a + k\frac{b-a}{n}), n \in \mathbb{N}^*$$

1. Implémenter une classe permettant de calculer la somme de Riemann de la fonction $sinus$ sur $[a, b]$ pour $n \in \mathbb{N}^*$
   Hint :

```
public static double integrate(double a, double b, int n)
Math.sin
```

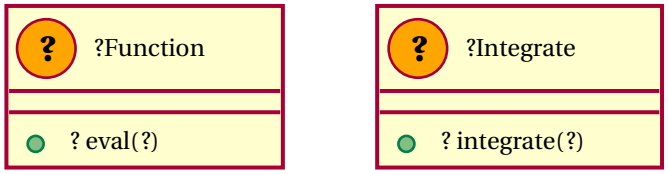
2. Tester l'implémentation pour $a = 1, b = 2, n = 10$.

On souhaite généraliser le code précédent pour considérer **toute** fonction $f : \mathbb{R} \implies \mathbb{R}$ et **différentes** méthodes de calcul d'intégrale, et plus particulièrement la méthode des trapèzes.
Pour rappel, la méthode des trapèzes associée à une fonction f est :

$$T_n = \frac{b-a}{n}\left[\frac{f(a)}{2} + \sum_{i=1}^{n-1} f(x_i) + \frac{f(b)}{2}\right], \quad \text{avec } x_i = a + i\frac{b-a}{n}$$

3. En vous inspirant du code précédent, proposer une nouvelle modélisation, et son implémentation correspondate, répondant au besoin expimé ci-dessus,. La modélisation devra *à moindre coût et effort* permettre de :
   - de prendre en compte toute fonction f
   - de prendre en compte la somme de Riemann, la méthode des trapèzes et éventuellement d'autres méthodes de calcul d'intégrales.

   Hint :



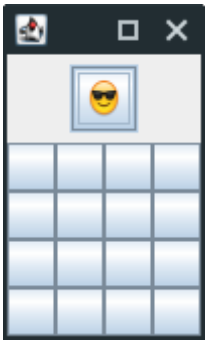
4. Tester l'implémentation pour $a = 1, b = 2, n = 4$ en considérant :
   a) la fonction sin, la somme de Riemann, la méthode des trapèzes
   b) la fonction cos, la somme de Riemann, la méthode des trapèzes

Partie optionnelle/facultative/bonus

6. Tester l'implémentation pour $a = 1, b = 2, n = 10$ en considérant le polynôme de l'Exercice 1.2, la somme de Riemann, la méthode des trapèzes.
7. En déduire les inégalités entre S_n, T_n et la valeur exacte de $\int_a^b P(x)\mathrm{d}x$.

Exercice 3

(Barême ⋆ ⋆ ⋆⋆)



Etudier attentivement le code fourni dans le package `exercice3`.
Ce code correspond à un projet d'application graphique permettant de jouer à un démineur dans une version à l'état de prototype.
On souhaite réaliser les évolutions suivantes au projet :

(a) On souhaite pouvoir revenir à l'état initial en appuyant sur le button avec l'icone d'un *smiley*.

(b) On souhaite mettre fin à la partie lorsqu'on déclenche une bombe. Pour cela, lorsqu'on clique sur une bombe, cela révèle l'ensemble de la grille (i.e. comme si on cliquait sur toutes les cases)

(c) On souhaite que le placement des bombes initiales soit aléatoire. Le principe est le suivant :
- On considère une grille de n lignes et m colonnes.
- On choisit un nombre de bombes b.
- On effectue b tirages aléatoires entre $[0, n-1]$.
- On effectue b tirages aléatoires entre $[0, m-1]$.
- En regroupant par couple les 2 séries de tirages, on obtient des coordonnées de bombes dans la grille.
  Remarque : Il est ainsi possible qu'on obtienne des doublons. On a donc *au plus b* bombes distinctes.

1. Réaliser l'évolution (a) *reset*
   - Faire toutes les adaptations du code nécessaires.
   - Ecrire une fiche de test permettant de valider l'évolution.
   - *Hint*

```
class Grid {
  public void reset();
}
```

2. Réaliser l'évolution (b) *fin de partie*
   - Faire toutes les adaptations du code nécessaires.
   - Ecrire une fiche de test permettant de valider l'évolution.

Partie optionnelle/facultative/bonus

3. Réaliser l'évolution (c) Placement aléatoire