TP 1 : RÉÉCHANTILLONNAGE

Introduction

Pour le premier TP nous allons faire du rééchantillonnage. Votre programme va recevoir en entrées une suite de valeurs qui représente une courbe. Ensuite, le logiciel va rééchantillonner les points sur la courbe en utilisant une méthode d'approximation. Finalement, la nouvelle séquence de valeurs est affichée sur la console. Les points en entrées sont équidistants sur l'axe des x, cette distance sera dénotée h. Les points en sorties seront aussi équidistants, la distance entre deux points en sortie sera dénotée h'.

Description

Entrées

Dans un premier temps, votre application devra lire le nom d'un fichier au clavier. Ensuite, il devra lire les entrées dans ce fichier. Le fichier va contenir une valeur par ligne. La première valeur (κ) sera un entier entre 1 et 3 inclusivement. Il représente le degré de l'équation utilisé pour trouver les valeurs du rééchantillonnage. Ces équations sont décrites plus loin dans le texte.

Les valeurs suivantes sont de type `double`. Elle représente la valeur : x_0. Elle est suivi de la distance (h) sur l'axe des x entre deux valeurs avant rééchantillonnage. Ensuite, il y a une valeur représentant la distance (h') sur l'axe des x entre deux valeurs après rééchantillonnage.

Finalement, il y aura une suite de n valeurs, toutes de type double, qui représente les échantillons ($y_i \mid 0 \leq i < n$). Maintenant que nous connaissons ces valeurs, nous pouvons déduire les valeurs des $x_i = x_0 + i \times h$.

Exemple de valeurs dans un fichier :

```
2
1.0
5.0
2.0
3.0
4.0
8.5
9.1
1.2
```

κ=2
x_0=1.0
h =5.0
h'=2.0
y_0=3.0
y_1=4.0
y_2=8.5
y_3=9.1
y_4=1.2

Toute information incorrecte implique l'affichage d'un message d'erreur et la fin du programme.

Exemple de lecture au clavier (vous **ne pouvez pas** utiliser la classe Clavier).

```
Scanner clavier = new Scanner( System.in );
String nom = clavier.nextLine();
clavier.close();
```

Exemple de message d'erreur et d'arrêt d'un programme.

```
System.err.println( mssg );
System.exit( -1 );
```

Voici un code de départ pour la lecture dans un fichier.

```
String nom = "nom.txt";
int x = 0;
double d = 0.0;

try {
    Path path = FileSystems.getDefault().getPath(nom);
    Scanner sc = new Scanner(Files.newBufferedReader(path));

    sc.useLocale( Locale.CANADA );

    if( sc.hasNextInt() ) {
        x = sc.nextInt();
    }

    if( sc.hasNextDouble() ) {
        d = sc.nextDouble();
    }

    sc.close();
} catch ( InvalidPathException e ) {
    // Traitement d'erreur ici.
} catch ( IOException e ) {
    // Traitement d'erreur ici.
}
```

Explication

Le fichier contient donc certains points (x_i, y_i) de la courbe pour $0 \leq i < n$. L'objectif est d'approximer de nouveaux points (x'_j, y'_j) sur la courbe, nous voulons seulement les valeurs de y'_j. Pour cela, votre logiciel doit :

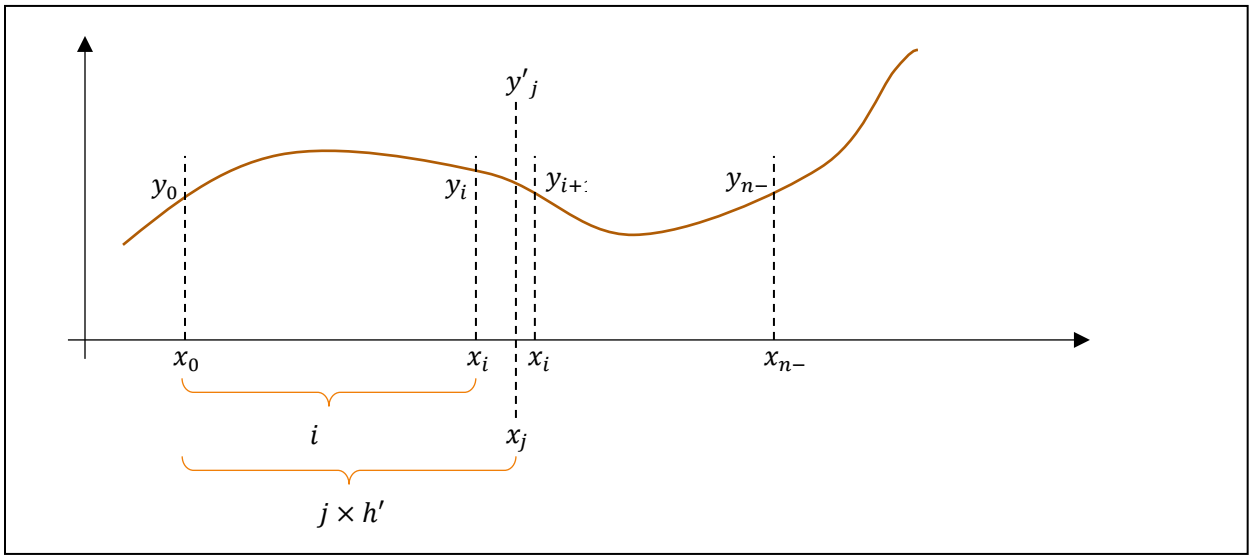
1. Calculer le nombre de nouvelles valeurs (m).
2. Calculer les nouveaux x'_j où nous voulons réévaluer la courbe.
3. Pour chaque nouveau x'_j :
    a. Trouver l'intervalle des anciens x_i qui comprend le nouveau x'_j.
    b. Construire une équation locale de degré κ pour cet intervalle.
    c. Évaluer y'_j à partir de cette équation locale et x'_j.

Calcule

Votre logiciel doit calculer les nouvelles valeurs (y'_j) pour chaque nouvelle valeur de rééchantillonnage (x'_j). Pour cela vous avez besoin de trouver le nombre de valeurs de x dont nous aurons besoin pour le rééchantillonage (dénoté m).

$$m = \frac{(n-1)h}{h'} + 1$$

Il suffit ensuite de rééchantillonner pour chaque valeur $x'_j = x_0 + j \times h'$ ou $0 \leq j < m$. Pour cela, nous devons trouver l'intervalle $[x_i .. x_{i+1}]$ qui contient x'_j. Il faut donc le plus grand x_i tel que $x_i \leq x'_j$ (à vous de



trouver comment faire ce calcul).

Ensuite, il faut trouver l'équation d'approximation pour l'intervalle $[x_i .. x_{i+1}]$. Il y a trois équations d'approximation possible, dépendant du degré (κ) demandé par l'utilisateur.

Degré (κ)	Calcule	Polynôme résultant
$\kappa = 1$	$\dot{\delta}_i = y_{i+1} - y_i$	$y'_j = a(x'_j) + b$

	$a = \frac{\dot{\delta}_i}{h}$ $b = y_i - ax_i$	
$\kappa = 2$	$\dot{\delta}_i = y_{i+1} - y_i$ $\dot{\delta}_{i+1} = y_{i+2} - y_{i+1}$ $\ddot{\delta}_i = \dot{\delta}_{i+1} - \dot{\delta}_i$ $a = \frac{\ddot{\delta}_i}{2h^2}$ $b = \frac{\dot{\delta}_i}{h} - a(x_i + x_{i+1})$ $c = y_i - bx_i - ax_i^2$	$y'_j = a(x'_j)^2 + b(x'_j) + c$
$\kappa = 3$	$\dot{\delta}_i = y_{i+1} - y_i$ $\dot{\delta}_{i+1} = y_{i+2} - y_{i+1}$ $\dot{\delta}_{i+2} = y_{i+3} - y_{i+2}$ $\ddot{\delta}_i = \dot{\delta}_{i+1} - \dot{\delta}_i$ $\ddot{\delta}_{i+1} = \dot{\delta}_{i+2} - \dot{\delta}_{i+1}$ $\dddot{\delta}_i = \ddot{\delta}_{i+1} - \ddot{\delta}_i$ $a = \frac{\dddot{\delta}_i}{6h^3}$ $b = \frac{\ddot{\delta}_i}{2h^2} - a(x_i + x_{i+1} + x_{i+2})$ $c = \frac{\dot{\delta}_i}{h} - b(x_i + x_{i+1}) - a(x_i^2 + x_i x_{i+1} + x_{i+1}^2)$ $d = y_i - cx_i - bx_i^2 - ax_i^3$	$y'_j = a(x'_j)^3 + b(x'_j)^2 + c(x'_j) + d$

Pour trouver les valeurs de y'_j vous n'avez qu'à appliquer x'_j au polynôme trouvé.

Résultat et affichage

Votre programme affiche simplement les valeurs de tous les y'_j à l'écran, à raison d'une valeur par ligne.

Directives

1. Le TP est à faire seul ou en équipe de deux.
2. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaitre que dans les `switch`.
    c. Un seul `return` par méthode.
    d. Additionnez le nombre de `if`, `for`, `while`, `switch` et `try` pour chaque méthode. Ce nombre ne doit pas dépasser 7.
3. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.
4. Commentaires
    - Commentez l'entête de chaque classe et méthode.
    - Une ligne contient soit un commentaire, soit du code, pas les deux.
    - Utilisez des noms d'identificateur significatif.

- Une ligne de commentaire ne devrait pas dépasser 120 caractères. Continuez sur la ligne suivante au besoin.
- Nous utilisons Javadoc :
    - La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la méthode ou la classe.
        - Courte.
        - Complète.
        - Commencez la description avec un verbe.
        - Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
    - Ensuite, au besoin, une description détaillée de la méthode ou classe va suivre.
        - Indépendant du code. Les commentaires d'entêtes décrivent ce que la méthode fait, ils ne décrivent pas comment c'est fait.
        - Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
    - Ensuite, avant de placer les **tags,** placez une ligne vide.
    - Placez les **tag** @param, @return et @throws au besoin.
        - @param : décris les valeurs acceptées pour la méthode.
    - Dans les commentaires, placer les noms de variable et autre ligne de code entre les tags <code> … </code>.
    - Écrivez les commentaires à la troisième personne.

Remise

Remettre le TP par l'entremise de Moodle. Placez vos fichiers `'*.java'` dans un dossier compressé (**zip**) de **Windows**, vous devez remettre l'archive. Le TP est à remettre avant le 14 octobre 23:55.

Évaluation

- Fonctionnalité (7 pts) : des tests partiels vous seront remis. Un test plus complet sera appliqué à votre TP.
- Structure (2 pts) : veillez à utiliser correctement le mécanisme d'héritage et de méthode. Vous devez avoir une classe de base pour vos fonctions d'échantillonnage. Chaque polynôme ($d = 1,2\ ou\ 3$) doit avoir une classe qui hérite de la classe de base. Utilisez correctement les mécaniques de généralisation et spécialisation pour éliminer la duplication de code.
- Lisibilité (2 pts) : commentaire, indentation et noms d'identificateur significatif.