Licence Informatique 3ème année

Dirigé par Olivier Baudon

7 juin 2018

Table des matières

1 Introduction **2**
- 1.1 Sujet du stage . . . 2
- 1.2 Le contexte du stage . . . 2

2 Déroulement du stage **2**
- 2.1 première semaine . . . 2
  - 2.1.1 Les classes abstraites et les interfaces . . . 2
  - 2.1.2 Les exceptions . . . 3
  - 2.1.3 Les varargs . . . 3
- 2.2 Deuxième semaine . . . 3
  - 2.2.1 Les collection . . . 3
  - 2.2.2 La généricité . . . 3
  - 2.2.3 Git . . . 4
- 2.3 Troisième semaine . . . 4
  - 2.3.1 Diagramme UML . . . 4
  - 2.3.2 Classe interne, locale et anonyme . . . 4
  - 2.3.3 Corretion des warnings . . . 4
- 2.4 Quatrième semaine . . . 4
  - 2.4.1 Optimisation du code . . . 4
  - 2.4.2 Lambda expression . . . 4
  - 2.4.3 Entête et licence . . . 4

3 Bilan du stage **4**

4 Référence **4**

1 Introduction

Dans cette partie, nous allons présenter le thème du stage, les prérequis exigés, le travaille demandé et dans quel contexte ce sujet a-t-il été posé.

1.1 Sujet du stage

Le sujet du stage est *Publier une bibliothèque sur les graphes et la compléter*, une version beta de la bibliothèque était déjà implémenté avant le début du stage, les taches principales était de compléter le code là ou il y manquait, améliorer le code en s'inspirant des nouvelles version du langage *java* et aussi documenter ansi que commenter la bibliothèque.
Des connaissance en "Programmation orienté objet" et en "Théorie des graphes" m'étais nécessaire pour comprendre la conception et l'implémentation de la bibliothèque, heureusement pour j'avais déjà aqcuis les notions de base sur ces deux domaines au cours de mes études universitaires dans mon pays natale, mais cela ne m'été pas suffisant car il me fallait des connaissances plus profondes, alors j'ai lu quelques cours en ligne et codé programmes avant le début du stage pour étre au niveau éxigé.

1.2 Le contexte du stage

Ce stage rentre dans le cadre du cycle licence, c'est stage obligatoire de 4 semaines minimum, il resprésente l'élément pédagogique 4TTVP18U et il fait partie du semèstre 6 du parcours informatique, l'obtention de ce stage été faite avec une candidature spontané adressé au Maître de Conférences Mr BAUDON Olivier qui fait partie de l'équipe des chercheurs du LaBRI.

2 Déroulement du stage

Dans cette partie nous allons parler du travaille fait par l'étudiant, des compétences qui ont été mobilisé pour le réaliser ansi que les connaissances et compétences acquis et métrisés.

2.1 première semaine

2.1.1 Les classes abstraites et les interfaces

Une classe est définie comme abstraite avec le mot clé **abstract**, Les classes abstraites sont à utiliser lorsqu'une classe mère ne doit pas être instanciée, cette dernière peux t'avoir ce qu'on appelle une méthode abstraite qui est caractérisé par le fait qu'elle n'a pas de corps, comme elle peux t'avoir des méthodes normales, par contre si une classe contient une méthode abstraite, cette classe doit alors être déclarée abstraite.
Une interface n'est q'une classe abstraite à 100%, ce qui signifie aucune méthode d'une interface n'a de corps, elle sert à définir un supertype et à utiliser le polymorphisme, pour implémenter dans une classe il faut utiliser le mots clé **implements**, et on peut implémenter autant qu'on veux dans la même classe, pour que cela soit fait correctement il faut redéfinir toutes les méthodes de l'interface (ou des interfaces) dans la classe concerné.

2.1.2 Les exceptions

Lorsqu'un événement que la JVM ne sait pas gérer apparaît, une exception est levée (exemple : division par zéro). une exception correspond donc à une erreur, la superclasse qui gère les exceptions s'appelle **Exception**. C'est possible aussi de créer une classe d'exception personnalisée qui génère ce que le programmeur considère comme exception mais peux être pas par le JVM, pour y procéder il faut lui hériter de la classe Exception.
L'instruction qui permet de capturer des exceptions est le bloc **try...catch**, si une exception est levée dans le bloc try, les instructions figurant dans le bloc catch seront exécutées pour autant que celui-ci capture la bonne exception levée. On peut aussi prévenir la JVM qu'une méthode est risque de générer une exeption grâce au mot clé **throws**. Une instanciation d'une exeception est lancée par le biais de l'instruction **throw**.

2.1.3 Les varargs

Les varargs pemettent de passer un nombre variable d'arguments à une méthode (pourvu qu'ils soient tous du type déclaré dans la signature de la méthode), sans créer explicitement de tableau, en revanche, dans le corps de la méthode, la variable var-arg est considéré comme un banal tableau et peut donc être parcourue par un « foreach », fournir sa taille avec la propriété length. Il y a toutefois deux petites limitations à cette syntaxe :

- Il ne peut y avoir qu'un seul var-arg par signature de méthode.
- Le var-arg doit toujours être le dernier paramètre.

2.2 Deuxième semaine

2.2.1 Les collection

Les collection permet de stocker un nombre variable d'objets, il y a principalement trois types de collection : **List** et **Set** qui hérite de l'interface Collection et aussi le type **Map**. chaque type a ses avantages et ses inconvénients et fournies des fonctionnalités propre à lui.
Voici quelques différences les différents types de collections :

- Les Collection stockent des objets alors que les Map stockent un couple (clé-valeur).
- Si on insére fréquemment des données en milieu de liste, **LinkedList** est plus aprioprié.
- Si on veut rechercher ou accéder à une valeur via une clé de recherche, vaut mieux opter pour une collection de type **Map**.
- Si on veut traiter une grande quantité de données, le type **Set** convient le plus.

2.2.2 La généricité

La généricité est un concept très utile pour développer des objets travaillant avec plusieurs types de données, ça nous fait gagné du temps au lieu de développer des classes qui traite de façon identique des données de type différent.
On peut utiliser la généricité sur les objets servant à gérer des collections, et avec l'outil **wildcard (?)** on indiquer que n'importe quel type peut être traité, mais cela revient à rendre ladite collection en lecture seule ce qui peux être désavantagant.

2.2.3 Git

Pour Facilité la communication et entre le stagière et l'encadrant pendant tout le stage, même les weekend, la platforme github a été utiliser par pour sauvegader les changements fait sur la bibliothèque et aussi pour encadrer et guider le stagiere au fure et à mesure de son aprentissage.

2.3 Troisième semaine

2.3.1 Diagramme UML

2.3.2 Classe interne, locale et anonyme

2.3.3 Corretion des warnings

2.4 Quatrième semaine

2.4.1 Optimisation du code

2.4.2 Lambda expression

2.4.3 Entête et licence

3 Bilan du stage

4 Référence

https ://openclassrooms.com/courses/apprenez-a-programmer-en-java/les-classes-abstraites-et-les-interfaces https ://openclassrooms.com/courses/apprenez-a-programmer-en-java/les-exceptions http ://thecodersbreakfast.net/index.php ?post/2008/07/23/77-java-les-var-args https ://openclassrooms.com a-programmer-en-java/les-collections-d-objets