Méthodes et Bonne Pratique

Guide de la méthodologie à suivre pour le Projet

# Introduction

Le présent document vise à détailler l’ensemble des méthodes choisies pour le développement du projet afin d’uniformiser les méthodes de travail entre les différent intervenants. Il détaille les technologies et les outils utiliser dans le projet ainsi que les « bonnes pratiques » mise en place.

Table des matières

[Introduction 1](#_Toc27772526)

[I. Environnement de développement 2](#_Toc27772527)

[A) IDE 2](#_Toc27772528)

[B) Gestionnaire de version 2](#_Toc27772529)

[1. Présentation 2](#_Toc27772530)

[2. Utilisation 2](#_Toc27772531)

[II. Bonnes pratiques de programmation 3](#_Toc27772532)

[A) Convention de nommage 3](#_Toc27772533)

[B) Commentaire et documentation 4](#_Toc27772534)

[1. Commenter son code 4](#_Toc27772535)

[2. Les document comments 4](#_Toc27772536)

[C) Numérotation des versions 4](#_Toc27772537)

[III. Bonnes pratiques pour le suivi du projet 4](#_Toc27772538)

[A) Rapport d’avancement 4](#_Toc27772539)

[B) Roadmap 4](#_Toc27772540)

# Environnement de développement

## IDE

Le développement du projet ce fait dans l’environnement de développement intégré Eclipse. Le code source du projet est ainsi structurer sous la forme d’un projet java d’Eclipse.

Le développement ce fait grâce à la perspective Java et le versionnage via la perspective Git. L’import du projet dans Eclipse se fait via l’importation du repository Git du quel on extrait le projet.

## Gestionnaire de version

### Présentation

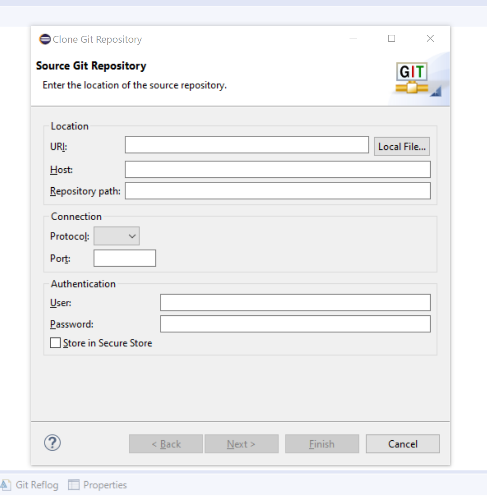
Tout projet informatique doit être versionner, c’est-à-dire que les différentes versions du code lors du développement doivent être accessible. Cela est fait grâce à un gestionnaire de version. Pour ce projet nous utiliserons Git avec la plateforme GitHub.

Git est un gestionnaire de version qui permet d’enregistrer les modifications du code via des commit qui sont des images du code à un instant données. Il permet également de faire des branches qui permettent de modifier indépendamment différentes partie du code en même temps. Le dossier dans le quel est stocker le code est appelé repository.

GitHub est un plateforme ligne qui permet d’héberger des repository Git. L’intérêt est multiple, cela notamment permet d’avoir un sauvegarde du code accessible en permanence et de simplifier le transfert d’information entre les intervenant. GitHub offre également de nombreuses fonctionnalités pour assister le développement du projet.

### Utilisation

#### Initialisation

Pour récupérer le projet sur sa machine et l’importer dans Eclipse on utilise la fonction git clone présente dans la perspective git d’Eclipse.

Après avoir cliquer sur l’icone correspondante, la fenêtre ci-contre s’ouvre. Dans le champs URL collez l’adresse du repository GitHub (on l’obtient en cliquant sur le bouton vert « clone or download » sur le site GitHub). Ensuite dans les champs User et Password, entrez vos identifiant GitHub. Cliquer ensuite sur « next » jusqu’à que le bouton « finish » devienne utilisable. Avant de cliquer dessus, choisissez l’emplacement ou mettre le fichier dans l’encart « Destination » et cocher la case « Import all existing Eclipse projects after clone finishes ». Puis Cliquer sur « finish ».

Maintenant que le projet est dans votre environnement Eclipse, il ne vous reste plus qu’à créer une nouvelle branche en allant dans la l’arborescence Git Repositories. Faites un clique droit sur votre projet puis sélectionner « switch to > new branch ». Dans la fenètre qui s’ouvre entrez le nom de votre nouvelle branch dans le champs « branch name » puis cliquer sur finish.

Félicitation, vous avez configurer votre environnement de travail. Retourner sur la perspective Java pour coder !

#### Faire un commit

Une fois que vos modifications sont effectuées, il faut les enregistrer dans le gestionnaire de version. Cet action est ce que l’on nomme un commit.

Pour faire un commit retournez sur la perspective Git. Dans la parti inférieur, sélectionnez l’onglet « Git Staging ». Il faut tout d’abort abort ajouter les modifications effectuées dans le commit, pour cela cliquer sur l’icone avec deux symboles « + ». Il faut en suite rédiger le message de commit dans le cadre « Commit Message ».

Le message doit être structurer de ma façon suivante :

Titre (Objectif principale de la modification)

(ligne vide)

- Element modifier vX.X

- Element modifier vX.X

- Element modifier vX.X

(…)

Les « vX.X » au numéro de versions des éléments. La façon du numéroter les versions est détaillée plus loin dans ce documents.

Une fois le message de commit rédigé, il ne reste plus qu’a le valider en cliquant sur « commit and push ». La première fois qu’un commit est fait sur une branche, Eclipse demande de configurer l’envoie sur GitHub en ouvrant une fenètre. Laisser les paramètres par défault.

Il est important de faire régulièrement des commits afin de séparer les modifications les unes des autres et ainsi pouvoir facilement les retrouver.

# Bonnes pratiques de programmation

## Convention de nommage

Afin d’uniformiser l’écriture du code et de simplifier sa compréhension par les autres développeurs, on utilise des conventions pour le nommage des éléments. Dans ce projet, les conventions s’inspire de celle officiel du langage Jave :

* L’ensemble des noms doit être en anglais
* Les nom de packages sont en minuscule (lowercase) et court (dans l’idéal un seul mot) les sous packages ont des noms composer des packages auquel il appartiennent séparer par des points. Ainsi le sous-package « ui » du package « dicewars » a pour nom « dicewars.ui »
* Les nom de classe, d’interface et d’énumération suivent la notation UpperCamelCase : Ils commencent par une majuscule et chacun des mots qui les composent commence par une majuscule pour marquer la séparation.
* Les nom de variable et de méthode suivent la notation lowerCamelCase : comme la notation UpperCamelCase à la seule différence que la première lettre est minuscule.
* Les nom de constante sont écrit en majuscule (UPPERCASE)

Observer ces quelques règles permet d’améliorer la lisibilité du code en permettant d’identifier facilement la nature des éléments présents.

## Commentaire et documentation

### Commenter son code

Un autre point important dans la lisibilité d’un code est la présence de commentaire. Ils expliquent le rôle des différents blocs de code. Ils sont donc indispensable pour faire un code facile à comprendre et donc à modifier et maintenir.

Il n’est pas nécessaire de commenter chaque ligne, ce comportement est même plutôt à proscrire. On préfère découper le code en petit bloc ayant une tache simple, il suffit alors de le précéder d’un commentaire qui décrit la tâche.

Si commenter chaque ligne est à éviter, certaines on des rôles importants qui ne sont pas forcément simple à comprendre avec la seule instruction. Dans ce cas-là il faut alors mettre un commentaire en fin de ligne pour expliquer.

### Les document comments

Un *document comment* est un type de commentaire particulier utiliser pour générer automatique la documentation du code. Ils décrivent le rôle des classes et des méthodes dans un format particulier. Ils sont indispensables et contiennent des champs pour les différentes informations. Pour plus d’information, renseigner vous sur la javadoc.

## Numérotation des versions

Afin de pouvoir suivre l’évolution des classes est des méthodes, on leurs attribue un numéro de version. Celui-ci doit être mis à jour à chaque modification et est écrit dans le champs correspondant du *document comment* associé. Pour ce projet il suit la forme « x.y ». « x » est le numéro de version principal qui correspond à un ensemble de fonctionnalité pour une classe ou à une faction de fonctionner pour une méthode. « y » est le numéro de modification, il change dès qu’il y a modification du code et est remis à zéro à chaque changement de version principale.

La mise à jour du numéro de version est associée au fait de faire un commit et doit y être mentionné.

# Bonnes pratiques pour le suivi du projet

## Rapport d’avancement

Le rapport d’avancement est un document qui contient l’état du projet au moment de son édition. Il est produit à chaque séance de TP projet afin de rendre compte de l’évolution du projet.

Chacun des membres de l’équipe l’alimente avec les taches effectuées et les problèmes rencontrés.

## Roadmap

La roadmap est un document qui contient la liste des tâches à effectuer et leur progression. Elle permet ainsi d’évaluer la progression du projet.