IFT3325

Devoir 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Coéquipiers**

**Mihai-Teodor Ionescu** | 1012378

**Tristan Forget** | 1020461

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **Diagramme de classes**

Voir le dossier Diagramme de classe pour les diagrammes de classe.

### **Démarrage du programme**

Le démarrage du programme jar se fait avec la commande : java -jar Main.jar <arguments>

Si aucun argument n’est spécifié, un classe de test démarrera qui permet de démontrer l’efficacité du checksum et de la décomposition de message en trame et de la recomposition.

### **Conception et séparation des classes**

Avant de commencer à coder, nous avons discuté et établit une séparation pertinente de nos classes afin de minimiser le couplage et la quantité de code à écrire.

Cela nous a permis de sauver beaucoup de temps et nous a grandement simplifié la manipulation des données circulant entre l’émetteur et le récepteur.

Voici notre architecture :

* Factories
* Managers
* Models
* Networking
* Utils

#### Factories :

Ce dossier contient des classes qui génèrent des *Trames* et des *Types de trame*. L’utilité ici est de nous fournir un accès simplifié aux constructeurs de ceux-ci.

#### Managers :

Ici nous avons que des classes statiques. Celles-ci s’occupent chacune de tâches bien précises. Nous avons par exemple :

* **CheckSumManager** : S’occupe de tous les manipulations du *CheckSum*.
* **ConversionManager** : S’occupe de toutes les conversions de type.
* Etc.

#### Models

Comme le nom l’indique, ce dossier contient tous les *models* de notre programme.

#### Networking

Nous gardons ici toutes les classes ayant un rapport avec le la transmission et la réception des trames. Donc vous allez y trouver des classes *Sender*, *Receiver*, etc.

#### Utils

Utilitaires génériques au projet.

### **Description des classes**

Pour plus d’information, consultez notre JavaDoc dans le dossier du même nom.