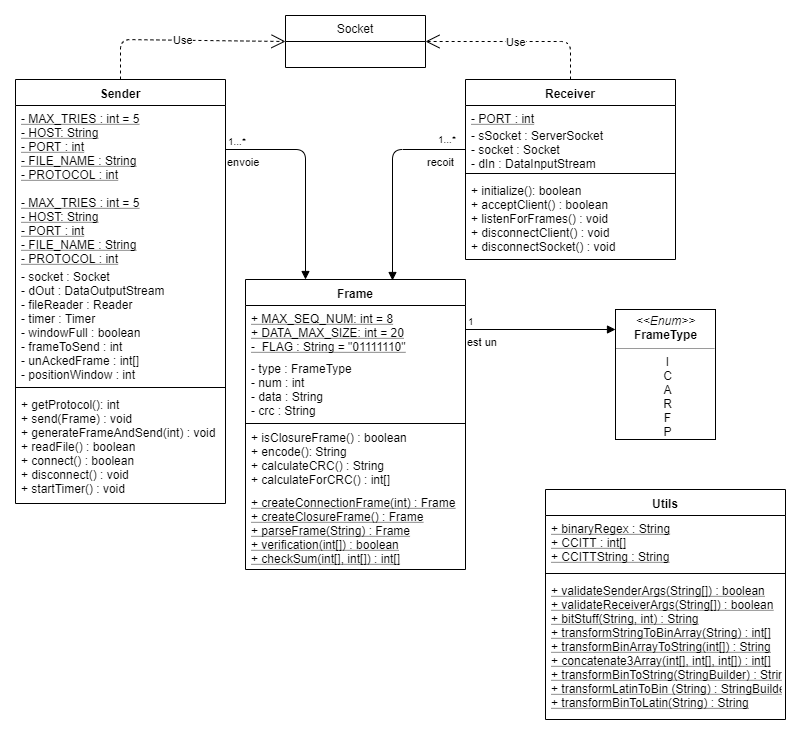
**Alexandre Dufour (p1054564) - Jonathan Caspar (p1129743 - 20059041)**

# **Téléinformatique - Devoir #2 - Rapport**

Nous avons organisé notre programme en 3 classes principales : **Sender**, **Receiver** et **Frame** :



L’idée générale est que deux entités (**Sender** et **Receiver**) se transmettent des suites de bit (dans notre programme, il s’agit de String composées uniquement de 0 et de 1) via un socket.

Cette suite de bits est une représentation bas-niveau d’un **Frame**.   
Une fois un frame construit, on l’encode (**encode()**) dans sa représentation bas-niveau (String de 0/1), on l’envoie au destinataire qui reconstruit un objet Frame à partir de ce String.

Ainsi, quand l’objet Frame est reconstruit : on peut vérifier grâce à un **cheksum**() si les données n’ont pas été corrompues durant la transmission.

# **Classes principales**

# 1) Frame

Un objet Frame est une représentation plus haut niveau d’un paquet de bits. Il respecte un certain pattern et certains de ses champs ont des tailles fixes (ce qui nous permet de « parser » facilement une suite de bits).

Il peut être instancié et donc **encodé** (Frame -> String) ou « **parsé** » (String -> Frame).  
Ses méthodes sont les suivantes :

* **isClosureFrame()** : retourne s’il s’agit d’une **trame de fermeture de connexion** (Type F) ou non
* **encode()**: retourne une représentation binaire (en String) d’un objet Frame

Il contient également des méthodes statiques pour « parser » des trames et créer facilement des trames qui ne sont pas des trames d’information et qui introduit le **bit-stuffing** (grâce à la méthode **bitStuff**() de la classe utilitaire **Utils**) des champs qui n’occupent pas tout l’espace binaire alloué :

* **createConnectionFrame(int protocol)** : retourne un objet Frame paramétré pour une demande de connexion avec un protocole spécifique
* **createClosureFrame()** : retourne un objet Frame paramétré pour une indication de fermeture de connexion
* **parseFrame(String frame)**: retourne un objet Frame reconstruit à partir d’un String binaire passé en paramètre

Ou pour faire un calcul de division polynomial :

* **verification**() : …
* **checksum**() : …

# 2) Sender

<description>

Les méthodes sont les suivantes :

# **getProtocol()** :

# **send(Frame frame)** :

# **generateFrameAndSend(int frameToSend)** :

# **readFile()** :

# **connect()** :

# **disconnect()** :

# **startTimer()** :

# 3) Receiver

<description>

Les méthodes sont les suivantes :

* **initialize**(): …
* **acceptClient**() :
* **listenForFrames**() :
* **disconnectClient**() :
* **disconnectSocket**() :

# **Classes secondaires**

# 4) FrameType (enum)

<description>

# 5) Utils (Utilitaires)

<description>

Les méthodes sont les suivantes :

* **validateSenderArgs**(String[] args) :
* **validateReceiverArgs**(String[] args) :
* **bitStuff**(String bits, int maxSize) :
* **transformStringToBinArray**(String) :
* **transformBinArrayToString**(int[]) :
* **concatenate3Array**(int[], int[], int[]) :
* **transformBinToString**(StringBuilder) :
* **transformLatinToBin** (String) :
* **transformBinToLatin**(String) :

# 6) Test

<description>

Les méthodes sont les suivantes :