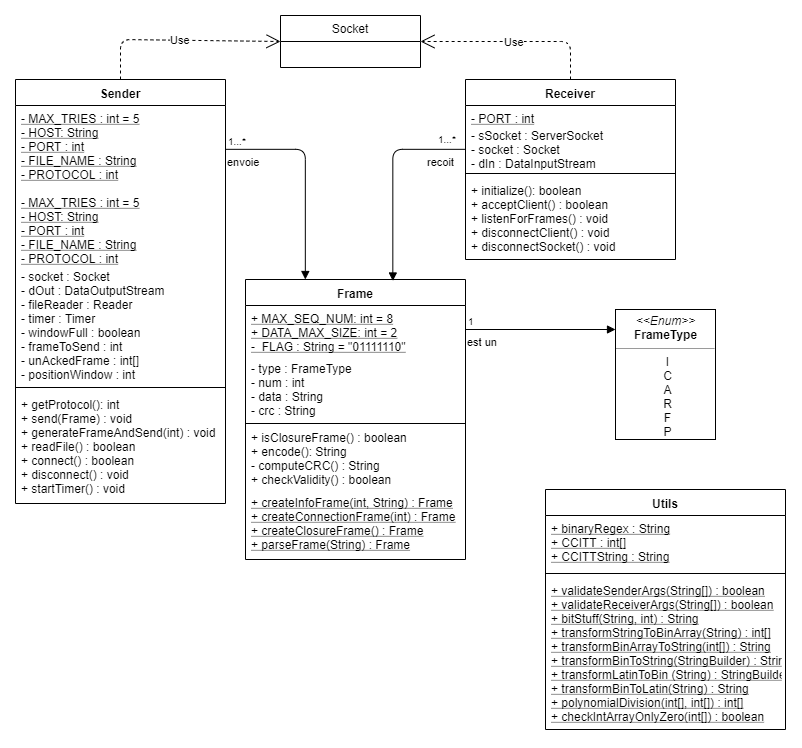
**Alexandre Dufour (p1054564) - Jonathan Caspar (p1129743 - 20059041)**

# **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 **checksum**() 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 uniquement en appelant les méthodes (createInfo/Closure/ConnectionFrame, etc..) puisque le constructeur de Frame est privé. Cela est voulu car on souhaite qu’une création de Frame mène au calcul automatique du code CRC (qui se fait dans le premier constructeur avec **this.crc = this.computeCRC()**)

Il peut aussi être **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 en introduisant des bits supplémentaires si les données n’occupent pas tout l’espace binaire alloué (avec **Utils.bitStuff(String bits, int maxSize)**)
* **createInfoFrame(int num, String data)** : retourne un objet Frame représentant une trame d’information
* **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
* **computeCRC**() : méthode privée (car utilisé uniquement par les méthodes « createInfo/Closure/ConnectionFrame, etc.. ») retournant le code CRC (en String) calculé à partir des attributs « type », « num » et « data » de l’objet Frame ciblé.
* **checkValidity**() : retourne si le frame est corrompu en effectuant un « checksum » grâce au CRC déjà calculé

# 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[]) :
* **transformBinToString**(StringBuilder) :
* **transformLatinToBin** (String) :
* **transformBinToLatin**(String) :
* **polynomialDivision(int[] array, int[] checksum)**: permet d'obtenir le résultat d'une division polynomiale entre deux nombres binaires .
* **checkIntArrayOnlyZero(int[] array)**: vérifie si un tableau de nombres contient uniquement des 0. On l’utilise pour vérifier si une division polynomiale donne un reste égal à 0.

# 6) Test

<description>

Les méthodes sont les suivantes :