BRUEL Jonathan

BONGOLO-BETO Berdrigue

LIANI Marwan

IACONA Alexandre

Documentation orientée design du projet Spider

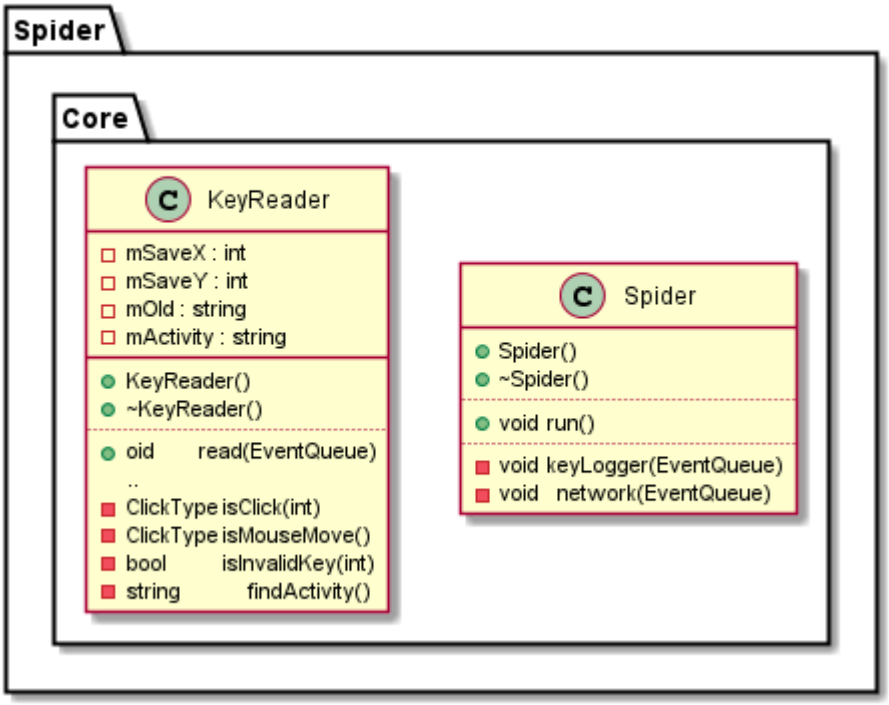
Spider : Design orienté objets

1. Client
    a. Core
    b. Système d'exception
    c. Système d'events
    d. Encapsulation de la OpenSSL
2. Server
    a. Client
    b. Client Manager
    c. Request
    d. Request Handler

I] Client

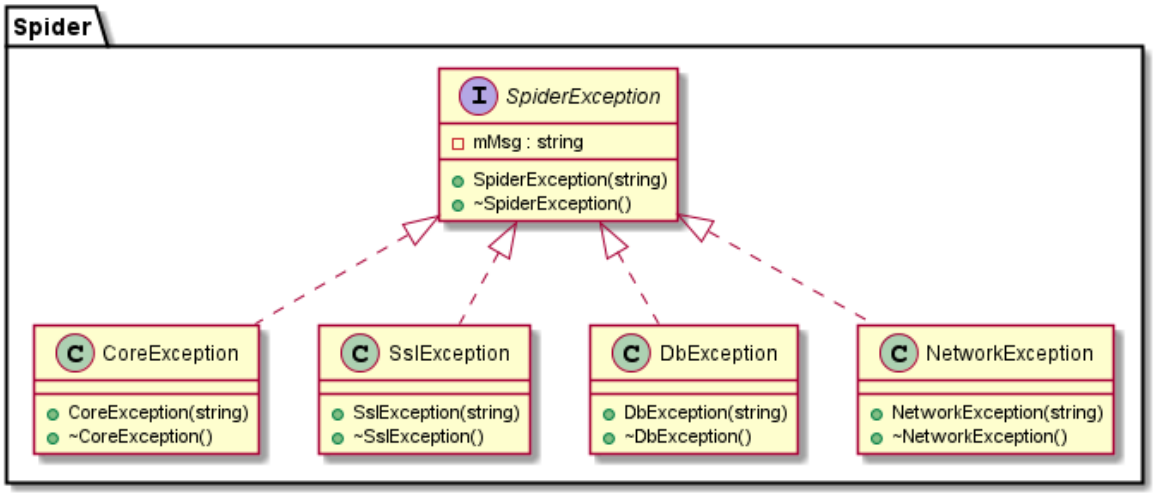
a) Core

Le Core du client est composé d'une classe Spider et d'une classe KeyReader. Le Spider crée deux threads, une première qui gère la récupération des touches en continue et une deuxième qui gère les requêtes (construction, envoi, réception, stockage dans une base de donnée si nécessaire).



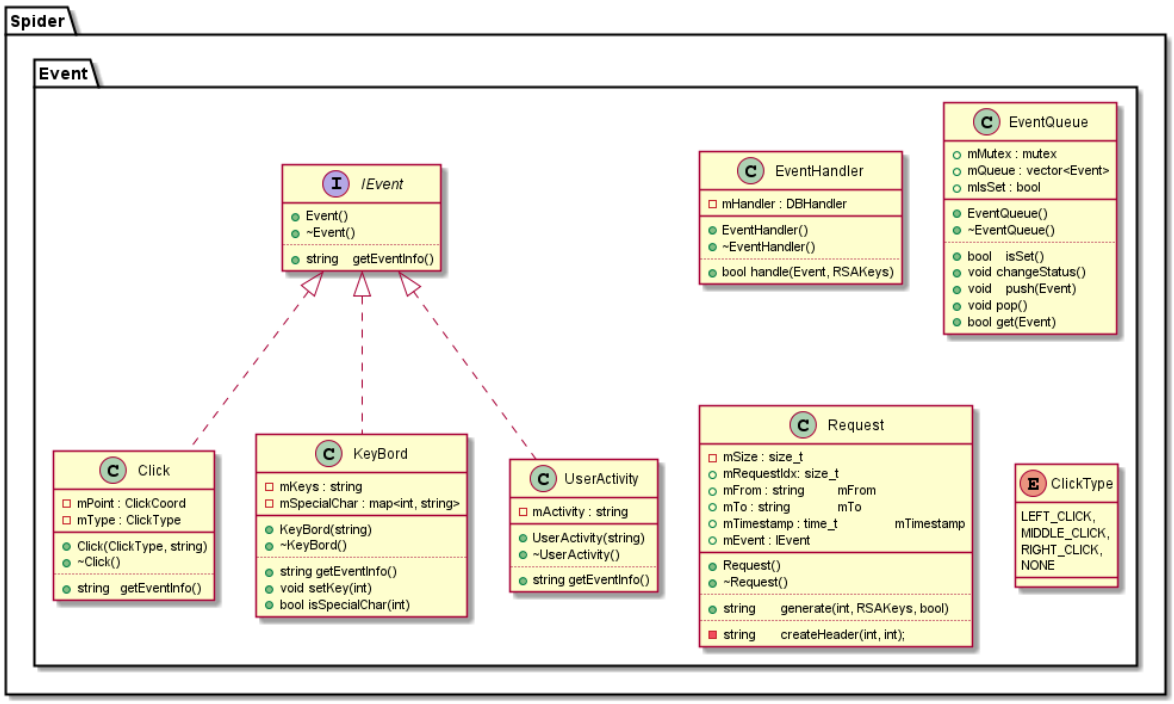
b) Système d'exception

Le client possède ces propres exceptions avec une interface qui se nomme « SpiderException ». Chaque entitée du Spider possède sa propre exception qui hérite du SpiderException. Par exemple, la classe « CoreException » qui appartient au Core.



c) Système d'events.

Un gestionnaire d'events enregistre chaque événement dans une queue. Le système d'events est aussi composé d'une queue d'évènements. Les deux threads discutent via cette queue. Chaque évènements sont différents et ils sont tous abstraits via une interface.



II] Serveur

Le serveur est pour rôle d'authentifier les clients (Spider) et d'enregistrer les informations qui lui y sont transférés, comme le mouvement de la sourie, les textes du clavier. La communication entre le serveur et les spiders se font à travers le protocole TCP. Le serveur est composé de plusieurs classes permettant de gérer de manières efficaces la communication avec l'extérieur

a) Client

Chaque client enregistrer et authentifié sur le serveur est représenté par une classe. Cette classe possède les méthodes pour traiter la requête du client : Authentification, Réponse aux requêtes, enregistrement dans une base donnée les informations sur le client et envoi de commande au client.

b) Client Manager

Le client manager a pour but de gérer les clients enregistrés sur le serveur, il peut donc enregistre un nouveau client, supprimer un client, supprimer tous les clients et appliquer une fonction sur tous les clients.

c) Request

Le serveur gère qu'un certain nombre de requetés, ses requêtes sont de type Request. Chaque requête possède une méthode pour transformer la requête en JSON ou construire une requête à partir d'un JSON.

d) Request Handler

La gestion et la réponse à une requête étant différent selon le type de requête, le Request Handler a pour but de gérer les requêtes, le Client n'a donc pas à gérer lui-même les requêtes.