TP Projet LO43

ROMET Pierre

Automne 2019

1 Présentation du projet

1.1 Introduction

Le TP de LO43 va prendre la forme d'un mini-projet, que vous allez devoir **réaliser en solitaire**, qui va vous permettre de mettre en pratique les connaissances acquises en dehors et lors du cours; il sera également une préparation au projet Java, que vous devrez rendre en fin de semestre.

Votre projet sera réalisé en dehors des séances de TP.
Ces dernières seront consacrés:

- à la présentation et explications des différents points techniques essentiels au TP
- aux points d'avancements
- à la résolution des points de blocages
- vos questions
- etc...

À compter de votre première séance de TP, vous disposez de 5 semaines pour réaliser votre projet, comme illustré ci-dessous:

<table>
<tr><th>Semaine 1</th><th>Semaine 2</th><th>Semaine 3</th><th>Semaine 4</th><th>Semaine 5</th></tr>
<tr><td rowspan="2">A Faire :
Tuto Git
Tuto Eclipse</td><td></td><td rowspan="2">Rendu
Intermédiaire</td><td></td><td rowspan="2">Rendu
Projet</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td rowspan="4">Pusher sur Git
Code première
version projet</td><td></td><td rowspan="4">Pusher sur Git
Code
Rapport</td></tr>
<tr><td>A rendre :
Comandes Git</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Temps estimé :
3h</td><td colspan="2">Temps à consacrer : 9h</td><td colspan="2">Temps à consacrer : 9h</td></tr>
</table>

Figure 1: emploi du temps

Date rendu "rapport command git": 25 septembre 23h59
Date rendu "première version": 11 octobre 23h59
Date rendu "final": 25 octobre 23h59

1.2 Mise en contexte

Votre projet vise la réalisation d'un simulateur d'environnement IOT.

Vous devrez modéliser un écosystème de capteurs ayant pour but de monitorer l'environnement intérieur d'un sous-marin en plongée.
Votre écosystème sera basé sur quatre types de capteurs différents, devant monitorer la température, la lumière, l'humidité et la pression dans le sous-marin. Ces capteurs devront ensuite communiquer avec un serveur sur lequel seront affiché et/ou stocké les données transmises.

1.3 Outils de développement

Pour ce projet, vous devrez suivre les contraintes suivantes:

- Concernant les IDE, il vous est imposé de travailler sur **Eclipse**.
- Si vous ne souhaitez pas utiliser d'IDE, vous pouvez mettre en place un makefile ou un Cmake.

Concernant Eclipse, si vous en avez besoin, vous trouverez un tuto à l'adresse suivante:

Windows: http://urlr.me/26jDt

Linux: http://urlr.me/CybqR

1.4 Notation

Voici la notation pour le projet, sachant que la note éliminatoire au tp est de 12/20:

- Rendre le rapport sur le tuto git et ses commandes: **1 points**
- Push un rendu intermédiaire fonctionnant: **10 points**
- Push un rendu final fonctionnant: **9 points**

2 Première séance

Premier programme et mise en place de l'outil de suivi de code, de versioning & de travail collaboratif.

2.1 Votre premier programme

Pour votre premier programme, vous allez coder le classique "Hello World !", **en C++**. Une fois ce dernier écrit, vous devrez le compiler, puis l'exécuter.

Si vous n'utilisez pas d'IDE, vous utiliserez le compilateur GNU pour le langage C++, "g++". Vous devrez dans un premier temps renommé votre fichier compilé "HELLOWORLD" (avec l'option de compilation qui convient), puis vous devrez afficher les Warnings.
Pour savoir comment rédiger la commande g++, vous utiliserez la documentation Linux présente dans la console, le "man".

Une fois le travail effectué, vous le ferez constater par un professeur.

2.2 Mise en place du repo Git

Nous allons maintenant nous pencher sur le développement de notre simulateur. Comme expliqué précédemment, notre simulateur est composé d'un serveur ainsi que d'un ensemble de capteurs.

Cependant, avant de commencer à développer votre première classe, vous allez mettre en place un dépôt GitHub. GitHub est une plate-forme web permettant d'héberger votre travail, de mettre en place du versioning et de pouvoir travailler à plusieurs sur un projet de développement. C'est notamment grâce à ce dépôt, que votre travail sera rendu en milieu de semestre, afin d'être évalué. Vous devrez donc vous créer un compte, puis cloner un dépôt distant sur lequel vous allez travailler. Une fois votre compte créé, vous irez voir le prof qui vous ajoutera en tant que contributeur sur son dépôt.

Vous trouverez à cette adresse, le tuto officiel en ligne de Git

https://git-scm.com/docs

Vous allez commencer par vérifier que Git est bien installé sur votre ordinateur. Pour cela vous exécuterez la commande suivante:

git –version

qui devra vous retourner les lignes suivantes:

git version X.X.X

Maj. August 13, 2021 Le système d'authentification par mot de passe ayant été supprimé, vous devez maintenant générer un "personal access token" qui vous servira de mot de passe. Pour cela, suivez le tuto sur le lien suivant:

http://urlr.me/rw2Q9

Vous allez commencer par cloner le repo fourni par votre prof, **à l'emplacement que vous souhaitez**

git clone https://github.com/Aldarme/TPAP4A.git

Ensuite, vous devrez créer une nouvelle branche

git branch MABRANCH

Puis, vous vous déplacerez sur votre nouvelle Branch, sur laquelle vous allez pouvoir héberger votre code

git checkout MABRANCH

Enfin, vous pousserez votre nouvelle branch sur le serveur distant :

git push - -set-upstream origin MABRANCH

Cela fait, vous disposer maintenant d'un espace de travail, sur un repo distant, qui vous est dédié. Vous allez maintenant copier votre code dans le dossier du repo que vous venez de cloner.
Nous allons maintenant placer notre fichier sous "suivis de version", ce qui va permettre de traquer les changements qui surviendront dans le code. On dit que notre code va être ajouté à l'index.

git add MONFICHIER

Ensuite, nous allons valider les modifications apportées à l'index

git commit –m "Description du commit"

Enfin, nous finissons en envoyant notre code sur le serveur distant

git push

Finissez par push le code du Helloworld sur votre branche du repo.
Une fois le travail effectué, vous le ferez constater par un professeur.

2.3 Travail à rendre

Pour vous former et explorer les possibilités qu'offre l'outil git, vous devez faire le tutoriel fourni à l'adresse suivante:

lien cliquable "Learn Git branching".

Une page de réussite de toutes les sections est disponible à la fin du tutoriel.

De plus, vous utiliserez l'outil "lien cliquable Visualizing Git", pour tester et comprendre les commandes suivantes:

- Git config
- Git init
- Git status
- Git add
- Git push
- Git merge
- Git diff
- Git blame

Vous devez rendre un court rapport contenant le certificat de réussite du tuto git et l'explication des commandes git ci-dessus.

3 Le Projet

3.1 Première classe

Vous pouvez maintenant vous attaquer à votre première tâche, qui sera de réaliser la classe "Server". Cette classe doit vous permettre de recevoir les données des capteurs et de les visualiser dans la console et/ou de les stocker dans des fichiers de logs.

Pour cela, vous allez commencer par implémenter la forme canonique de Coplien de la classe. Une fois cela fait, vous écrirez deux fonctions:

- "consoleWrite" afin de visualiser les données arrivantes dans la console.
- "fileWrite" pour stocker les données des capteurs dans des fichiers de logs (chaque capteur devra disposer de son fichier de logs).

3.2 Architecture

Vous allez maintenant mettre en place l'architecture de votre projet. Pour cela, vous trouverez sur le teams de AP4A, dans Fichier/SupportTP, une vidéo intitulé "Projet_Architecture" qui va vous présenter et expliquer l'architecture attendue pour votre projet. De plus, vous trouverez à la fin de ce sujet, le diagramme de classes du projet.

Vous allez maintenant créer les deux nouvelles classes indiquées sur votre diagramme de classes; Le diagramme de classes vous indiquant les fonctions essentielles à implémenter.

3.3 Sensor

La classe "Sensor" doit vous permettre de générer les données qui devront être remontées via le Scheduler, au Server.

Dans un premier temps, pour le **rendu Intermédiaire**, tous vos capteurs vous retournerons le même type de donnée.

Puis, pour le **rendu final**, vous devrez pouvoir générer des données de plusieurs types différents, en fonction des capteurs.

- Temperature, Humidity : FLOAT
- Pression : INTEGER
- Light : BOOLEAN

Attention, les données générées devront être cohérentes avec celles pouvant être mesurées dans un sous-marin.

3.4 Scheduler

La classe Scheduler est le coeur de votre simulateur.
C'est lui qui va définir la fréquence à laquelle les données des capteurs seront récupérées pour être transmise au serveur.
La contrainte finale sera que la donnée de chaque capteur pourra être récupérée et transmise au serveur à des intervalles de temps différents.

3.5 Server

La classe Server doit permettre de recevoir et/ou logger les données provenant des capteurs du sous-marin. Chaque type de capteur aura son fichier de log dédié.
De plus, vous devrez pouvoir activer ou non l'affichage et le log des données.
Le code de cette classe est une évolution du code écrit pour "Première classe".

4 Guide de Programmation

Vous trouverez sur le teams de AP4A, dans Fichier/SupportTP, un coding standard que vous aurez à respecter pour la programmation de votre TP projet.

5 Tool Boxes

Voici la liste des logiciels que vous avez pu mettre en place pour votre projet:

- Editeur Architecture logiciel (UML): le seul et unique "ASTAH"
- Man
- Editeur de code: VIM, EMACS, Gedit, atom, nano, imax, sublimeTex
- IDE: Eclipse
- Compilateur: g++
- Debugger: gdb
- Makefile (compilation automatise)
- Valgrind (detection fuite memoire)
- Git
- GitHub
- Site entrainement: Coding Game, Fizz Buzz

6 Annexe

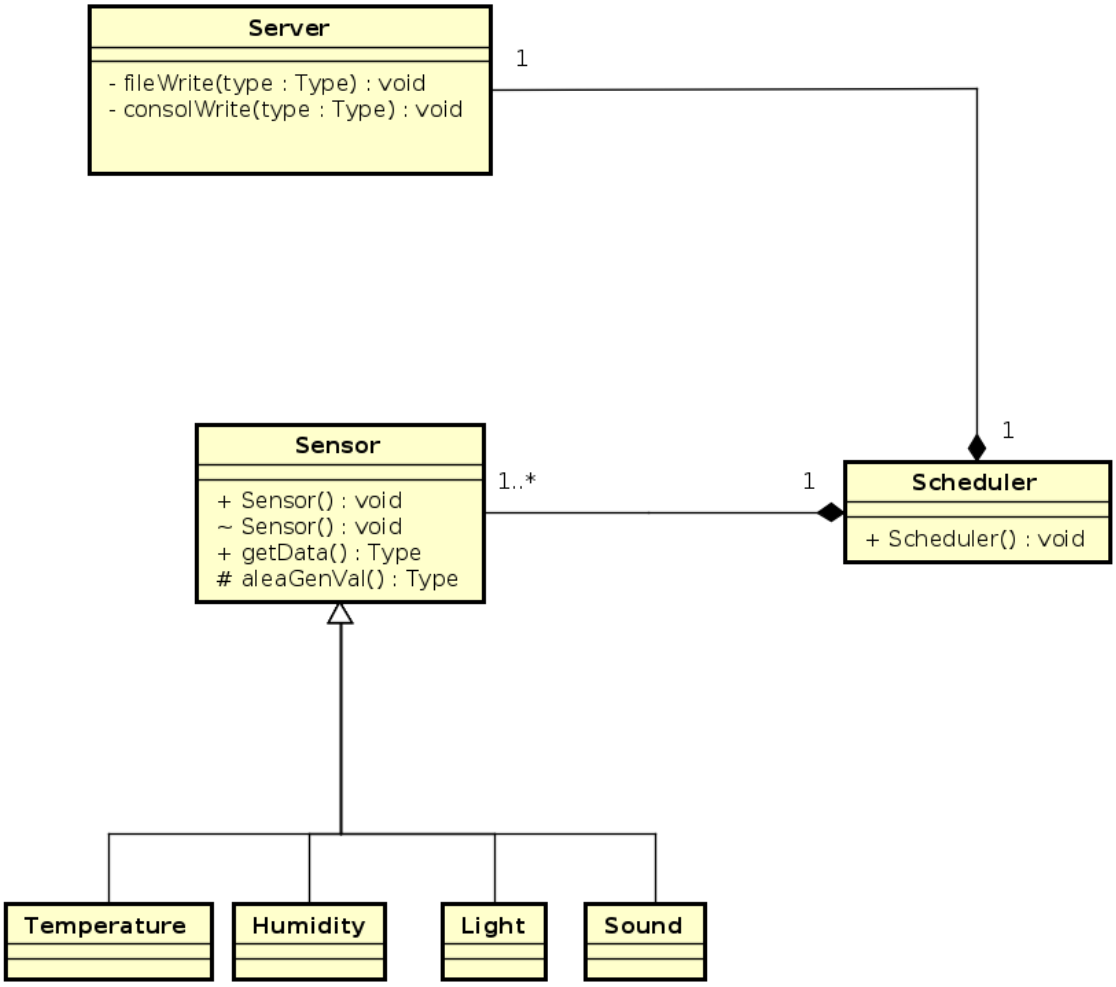

Diagram0.png

Figure 2: Diag. de classe