ENSEEIHT 1SN
Déroulement du module Technologie Objet
Java, UML, etc.

18 février 2024

Table des matières

Ce document décrit le déroulement chronologique du cours de Technologie Objet dispensé à l'ENSEEIHT en 1ière année Sciences du Numérique (1SN).

Les sujets sont disponibles au format PDF (pdf).

Les programmes donnés ont été testés en utilisant le JDK (version 8 ou 11). Nous nous limiterons cependant aux concepts présents dans la version 1.7 du langage Java (Java 7).

Vous avez à votre disposition une version de ce document sous la forme d'un seul fichier HTML [1] et une version découpée en parties.

Ce document et les documents associés (sujets, transparents, etc.) contiennent encore certainement des erreurs et/ou des imprécisions. Merci de signaler vos remarques, vos suggestions ou vos interrogations à Xavier Crégut <Prenom.Nom@enseeiht.fr> [2].

Ce module est composé de :

— Cours asynchrones (pas de présentiel)
— Séances
— Projets
— Ressources

1. `to-1sn-2023-deroulement-corrige.html`
2. `mailto:Xavier.Cregut@enseeiht.fr`

Chapitre 1

Technologie Objet

1.1 Cours asynchrones

Voici les supports de cours qui sont à lire avant les TD/TP. Ces consignes peuvent être retrouvées dans le fichier cours-consignes-main.pdf

- Présentation de l'UE.
- Classes (environ 90') et les quelques questions sur le cours : avant le TD 1 (23 janvier) :
  1. Modularite : Classe (22)
  2. Constructeurs et destructeur (46)
  3. Masquage d'information (61)
  4. Membres de classe : 82
  5. Programmation impérative : 92 (le petit questionnaire au début est là pour vérifier que vous avez compris les points essentiels)
  6. Tests unitaires avec JUnit : 156 (utilisés au TP 2)
  7. Fabriques statiques (207, abordées au TD 1), ellipse (211)...
  8. Il est bien sûr conseillé de lire tous les transparents du support même s'ils ne sont pas mentionnés explicitement dans la liste ci-dessus.
- Relations entre classes (environ 30') : avant le TD 2 (6 février).
- Interface et Généricité (environ 90') : avant le TD 2 et TP 4 (6 février). Ces deux notions sont indépendantes même si elles sont regroupées sur le même support. La partie interface sera un peu utilisée sur le mini-projet (PR01).
- Héritage – Classes abstraites – Réutilisation (120') : avant le TP 5 (11 février) et le TD 3 (11 février).
- Exceptions (environ 1h30) : avant le TD 4 (20 février).
- (révisions) Relations entre classes (environ 30') : avant le TD 5 (27 février).
- UML diagramme de cas d'utilisation, diagrammes de séquence, diagramme d'états et diagrammes d'activité [1] (environ 1h45) : avant le TD 6 (5 mars).

1. https://www.eyrolles.com/Chapitres/9782212133448/Chap-6_Roques.pdf

— Patrons, Structures de données et Collections (environ 2h30) : avant le TD 8 (11 mars) et le TP 12 (12 mars).
— Interfaces graphiques avec Java/Swing (environ 1h45) : avant le TD 9 (19 mars) et le TP 14 (26 mars). Il est conseillé de lire et travailler ce cours après le TD 8 « Encore les segments et les points » (séance TD 7, 13 mars).
Pour valider la compréhension de ce cours, il est conseillé de faire les exercices suivants avec comme point de départ la classe ComprendreSwing.
Solution : Voici un corrigé et les fichiers correspondants.

1.2 Séances

1.2.1 Séance Présentation 1 (16/01/2024, 16 :15)

— 1SN : 16/01/2024, 16 :15 à 17 :00, B00, Cregut Xavier

Introduction 1 : Présentation de l'UE

Voir Moodle...

1.2.2 Séance CM 1 (22/01/2024, 08 :00)

— 1SN : 22/01/2024, 08 :00 à 09 :45, B00 (389), Cregut Xavier

CM 1 : Abstraction et modularité : classes

Le cours présente la classe comme un moyen d'abstraction et de modularité. L'objectif est de savoir faire en objet ce que vous savez déjà faire en programmation impérative.

1.2.3 Séance TD 1 (22/01/2024, 14 :00 au 24/01/2024, 14 :00)

— 1SNA, 1SNB : 22/01/2024, 14 :00 à 15 :45, B006 (27), Cregut Xavier
— 1SNC, 1SND : 22/01/2024, 14 :00 à 15 :45, B208 (42), Bedouet Judicael
— 1SNE, 1SNF : 22/01/2024, 14 :00 à 15 :45, A302 (64), Ouederni Meriem
— 1SNG, 1SNH : 24/01/2024, 14 :00 à 15 :45, B007 (28), Cregut Xavier
— 1SNI, 1SNJ : 23/01/2024, 16 :15 à 18 :00, B208 (42), Bedouet Judicael
— 1SNK, 1SNL : 24/01/2024, 14 :00 à 15 :45, A301 (40), Dupont Guillaume

TD 1 : Spécification et implantation

Le TD 1 concerne la spécification et l'implantation d'une classe. Il permet de démontrer l'intérêt de déclarer les attributs privés et explique le principe de l'accès uniforme et la règle de protection en écriture des attributs.
Solution : Voici un corrigé du TD 1 possible et les fichiers associés :

— la solution des exercices 1 à 5, donc le point cartésien et les deux programmes de test ;
— la version polaire du point et le premier programme de test ;
— la version mixte du point et le second programme de test.
— la La classe Point avec les fabriques statiques pour créer des points à partir des coordonnées cartésiennes ou des coordonnées polaires

Remarque : Cette solution est partielle puisque les modifications pour les exercices 4, 5 et 6 ne sont pas données.

1.2.4 Séance TP 1 (23/01/2024, 08 :00 au 24/01/2024, 16 :15)

— 1SNA : 23/01/2024, 08 :00 à 09 :45, C203 (32), Bonnet Louis
— 1SNB : 23/01/2024, 08 :00 à 09 :45, C205 (32), Singh Neeraj
— 1SNC : 23/01/2024, 08 :00 à 09 :45, C201 (18), Charles William
— 1SND : 23/01/2024, 08 :00 à 09 :45, C202 (18), Gasparini Simone
— 1SNE : 23/01/2024, 08 :00 à 09 :45, C204 (32), Ouederni Meriem
— 1SNF : 23/01/2024, 08 :00 à 09 :45, C206 (32), Lebegue Jean-Claude
— 1SNG : 24/01/2024, 16 :15 à 18 :00, C216a, Chigot Estelle
— 1SNH : 24/01/2024, 16 :15 à 18 :00, C203 (32), Superman 1SN
— 1SNI : 24/01/2024, 16 :15 à 18 :00, C201 (18), Pelissier-Combescure Marie
— 1SNJ : 24/01/2024, 16 :15 à 18 :00, C202 (18), Megna Anael
— 1SNK : 24/01/2024, 16 :15 à 18 :00, C204 (32), Dupont Guillaume
— 1SNL : 24/01/2024, 16 :15 à 18 :00, C206 (32), Riviere Peter

TP 1 : Points et segments

Le TP 1 propose d'utiliser les principaux outils du JDK (javac, java et javadoc) en ligne de commande, de vérifier certains aspects du langage qui ont été présentés en cours, de compléter une classe Segment et d'écrire un programme de test.

Ce TP peut aussi être fait avec `https://repl.it/` en utilisant la console qui se trouve à droite de l'éditeur pour taper les commandes. Il est conseillé de s'inscrire de manière à pouvoir conserver ses fichiers d'une connexion à l'autre. Dans la console, vous pouvez aussi utiliser les commandes de subversion...

Vous partirez des classes fournies.

Solution : Voici un corrigé du TP 1 possible et les fichiers associés.

1.2.5 Séance Projet 1 (23/01/2024, 10 :15 au 25/01/2024, 08 :00)

— 1SNA, 1SNB : 23/01/2024, 10 :15 à 12 :00, C214a+C214b, Bonnet Louis
— 1SNC, 1SND : 23/01/2024, 10 :15 à 12 :00, C201 (18)+C202 (18), Gasparini Simone
— 1SNE, 1SNF : 23/01/2024, 10 :15 à 12 :00, C204 (32)+C206 (32), Ouederni Meriem
— 1SNG, 1SNH : 25/01/2024, 08 :00 à 09 :45, C216a+C216b, Chigot Estelle
— 1SNI, 1SNJ : 25/01/2024, 08 :00 à 09 :45, C214a+C214b, Megna Anael
— 1SNK, 1SNL : 25/01/2024, 08 :00 à 09 :45, C204 (32)+C206 (32), Riviere Peter

TP 2 : Tester la classe Point avec JUnit

Le TP 2 propose de tester la classe Point en utilisant le framework JUnit [2].

Attention, JUnit 4 est installé dans `/mnt/n7fs/ens/tp_cregut/junit4.jar` Il faut donc faire :

export CLASSPATH=/mnt/n7fs/ens/tp_cregut/junit4.jar :.

ou en csh/tcsh :

setenv CLASSPATH /mnt/n7fs/ens/tp_cregut/junit4.jar :.

Si vous avez déjà une définition de CLASSPATH dans votre ~/.bashrc, il suffit de la compléter pour ajouter le nouveau chemin d'accès : `/mnt/n7fs/ens/tp_cregut/junit4.jar`.

Voici les fichiers fournis.

Solution : Voici une solution possible.

TP 3 : Introduction à Eclipse

Le TP 3 vous permet de prendre en main cette plateforme de développement et de vous familiariser avec ses possiblités.

PR 1 : Mini-projet

La description est donnée dans la rubrique Projets / Mini-projet.

1.2.6 Séance TP 2 (05/02/2024, 14 :00 au 05/02/2024, 16 :15)

- 1SNA : 05/02/2024, 16 :15 à 18 :00, C203 (32), Bonnet Louis
- 1SNB : 05/02/2024, 16 :15 à 18 :00, C205 (32), Pennec Galann
- 1SNC : 05/02/2024, 16 :15 à 18 :00, C201 (18), Charles William
- 1SND : 05/02/2024, 16 :15 à 18 :00, C202 (18), Gasparini Simone
- 1SNE : 05/02/2024, 16 :15 à 18 :00, C301(18), Ouederni Meriem
- 1SNF : 05/02/2024, 16 :15 à 18 :00, C304 (32), Lebegue Jean-Claude
- 1SNG : 05/02/2024, 14 :00 à 15 :45, C216a, Chigot Estelle
- 1SNH : 05/02/2024, 14 :00 à 15 :45, C216b, Gasparini Simone
- 1SNI : 05/02/2024, 14 :00 à 15 :45, C201 (18), Pelissier-Combescure Marie
- 1SNJ : 05/02/2024, 14 :00 à 15 :45, C202 (18), Megna Anael
- 1SNK : 05/02/2024, 14 :00 à 15 :45, C204 (32), Ouederni Meriem
- 1SNL : 05/02/2024, 14 :00 à 15 :45, C206 (32), Riviere Peter

TP 4 : Afficher graphiquement les points et les segments

Le TP 4 propose de se familiariser avec un afficheur graphique dont la documentation est donnée puis de l'utiliser pour afficher graphiquement les points et les segments.

2. `http://junit.org`

Sous Eclipse (ou équivalent), on utilisera le fichier d'archive afficheur.jar qu'il faudra ajouter comme « External archive file » au « Build path » du projet Java.

Voici les fichiers fournis.

Solution : Voici un corrigé du TP 4 possible et les fichiers associés.

1.2.7 Séance Projet 2 (06/02/2024, 08 :00 au 07/02/2024, 14 :00)

- 1SNA, 1SNB : 06/02/2024, 08 :00 à 09 :45, C214a+C214b, Cregut Xavier
- 1SNC, 1SND : 06/02/2024, 08 :00 à 09 :45, C201 (18)+C202 (18), Charles William
- 1SNE, 1SNF : 06/02/2024, 08 :00 à 09 :45, C204 (32)+C206 (32), Ouederni Meriem
- 1SNG, 1SNH : 07/02/2024, 14 :00 à 15 :45, C216a+C216b, Gasparini Simone
- 1SNI, 1SNJ : 07/02/2024, 14 :00 à 15 :45, C214a+C214b, Pelissier-Combescure Marie
- 1SNK, 1SNL : 07/02/2024, 14 :00 à 15 :45, C204 (32)+C206 (32), Dupont Guillaume

PR 1 : Mini-projet (fin)

Continuation du thème PR 1 de la séance Séance Projet 1

1.2.8 Séance TD 2 (06/02/2024, 10 :15 au 07/02/2024, 16 :15)

- 1SNA, 1SNB : 06/02/2024, 10 :15 à 12 :00, B006 (27), Cregut Xavier
- 1SNC, 1SND : 06/02/2024, 10 :15 à 12 :00, C101 (90), Bedouet Judicael
- 1SNE, 1SNF : 06/02/2024, 10 :15 à 12 :00, A302 (64), Ouederni Meriem
- 1SNG, 1SNH : 07/02/2024, 16 :15 à 18 :00, B007 (28), Cregut Xavier
- 1SNI, 1SNJ : 07/02/2024, 16 :15 à 18 :00, A301 (40), Bedouet Judicael
- 1SNK, 1SNL : 07/02/2024, 16 :15 à 18 :00, A303 (64), Dupont Guillaume

TD 2 : Ensemble ordonné

Le TD 2 propose de modéliser un ensemble ordonné utilisé dans la résolution (naïve) du crible d'Ératosthène.

Solution : Voici un corrigé du TD 2 possible. Les ensembles seront repris en TP et sur le cours sur les collections.

1.2.9 Séance TP 3 (12/02/2024, 08 :00 au 14/02/2024, 14 :00)

- 1SNA : 13/02/2024, 10 :15 à 12 :00, C216b, Bonnet Louis
- 1SNB : 13/02/2024, 10 :15 à 12 :00, C216a, Pennec Galann
- 1SNC : 13/02/2024, 08 :00 à 09 :45, C201 (18), Charles William
- 1SND : 13/02/2024, 08 :00 à 09 :45, C202 (18), Gasparini Simone
- 1SNE : 13/02/2024, 08 :00 à 09 :45, C204 (32), Superman 1SN
- 1SNF : 13/02/2024, 08 :00 à 09 :45, C206 (32), Lebegue Jean-Claude
- 1SNG : 14/02/2024, 14 :00 à 15 :45, C216a, Chigot Estelle

— 1SNH : 14/02/2024, 14 :00 à 15 :45, C216b, Gasparini Simone
— 1SNI : 14/02/2024, 14 :00 à 15 :45, C201 (18), Pelissier-Combescure Marie
— 1SNJ : 14/02/2024, 14 :00 à 15 :45, C202 (18), Megna Anael
— 1SNK : 12/02/2024, 08 :00 à 09 :45, C204 (32), Dupont Guillaume
— 1SNL : 12/02/2024, 08 :00 à 09 :45, C206 (32), Riviere Peter

TP 5 : Héritage comme spécialisation

Le TP 5 propose de compléter l'éditeur de schémas mathématiques en donnant la possibilité de nommer les points.

Sont fournies les classes de l'application et toujours le paquetage contenant les classes de l'afficheur et sa documentation.

Solution : Voici un corrigé du TP 5 possible et les fichiers associés.

TP 6 : Héritage comme généralisation

Le TP 6 propose de formaliser la notion de schéma mathématique.

Sont fournies les classes de l'application et toujours le paquetage contenant les classes de l'afficheur et sa documentation.

Solution : Voici un corrigé du TP 6 possible et les fichiers associés.

1.2.10 Séance TD 3 (13/02/2024, 08 :00 au 14/02/2024, 16 :15)

— 1SNA, 1SNB : 13/02/2024, 08 :00 à 09 :45, B006 (27), Cregut Xavier
— 1SNC, 1SND : 13/02/2024, 10 :15 à 12 :00, B306 (49), Bedouet Judicael
— 1SNE, 1SNF : 13/02/2024, 10 :15 à 12 :00, A302 (64), Dupont Guillaume
— 1SNG, 1SNH : 13/02/2024, 10 :15 à 12 :00, B006 (27), Cregut Xavier
— 1SNI, 1SNJ : 14/02/2024, 16 :15 à 18 :00, A301 (40), Bedouet Judicael
— 1SNK, 1SNL : 14/02/2024, 14 :00 à 15 :45, A303 (64), Dupont Guillaume

TD 3 : Comptes bancaires : une vue simplifiée

Le TD 3 propose de modéliser des comptes bancaires et une banque. Il permet de revoir la plupart des concepts objets, en particulier l'encapsulation, l'héritage, la redéfinition et la liaison dynamique, mais aussi les attributs et méthodes de classe.

1.2.11 Séance TP 4 (19/02/2024, 10 :15 au 19/02/2024, 16 :15)

— 1SNA : 19/02/2024, 16 :15 à 18 :00, C203 (32), Bonnet Louis
— 1SNB : 19/02/2024, 16 :15 à 18 :00, C205 (32), Pennec Galann
— 1SNC : 19/02/2024, 16 :15 à 18 :00, C201 (18), Charles William
— 1SND : 19/02/2024, 16 :15 à 18 :00, C202 (18), Gasparini Simone
— 1SNE : 19/02/2024, 16 :15 à 18 :00, C301(18), Ouederni Meriem

— 1SNF : 19/02/2024, 16 :15 à 18 :00, C206 (32), Lebegue Jean-Claude
— 1SNG : 19/02/2024, 10 :15 à 12 :00, C216a, Chigot Estelle
— 1SNH : 19/02/2024, 10 :15 à 12 :00, C216b, Gasparini Simone
— 1SNI : 19/02/2024, 10 :15 à 12 :00, C201 (18), Pelissier-Combescure Marie
— 1SNJ : 19/02/2024, 10 :15 à 12 :00, C202 (18), Megna Anael
— 1SNK : 19/02/2024, 14 :00 à 15 :45, C203 (32), Dupont Guillaume
— 1SNL : 19/02/2024, 14 :00 à 15 :45, C206 (32), Riviere Peter

TP 7 : Outils associés à Java Modeling Language (JML)

Attention : OpenJML nécessite une version 8 de Java. La version installée est la version 11. Il serait possible d'installer une version 8 et vous faire jongler entre Java 8 et Java 11. Cependant, et parce que vous avez vu l'intérêt de la programmation par contrat en PIM avec Ada et en Modélisation, **nous ne ferons pas ce TP** !

Le TP 7 présente les outils associés à JML que nous utilisons dans le cadre de la programmation par contrat.
Attention : Nous allons utiliser OpenJML [3] qui implante JML [4]. Il est malheureusement très peu robuste !

Voici les classes fournies.

TP 8 : Ensemble et ensemble chaîné

Le TP 8 propose d'implanter des ensembles et ensembles ordonnés d'entiers en s'appuyant sur des structures chaînées, puis de les généraliser grâce à la généricité.

Sont fournies les classes de l'application.

1.2.12 Séance TD 4 (20/02/2024, 08 :00 au 21/02/2024, 14 :00)

— 1SNA, 1SNB : 20/02/2024, 08 :00 à 09 :45, B006 (27), Cregut Xavier
— 1SNC, 1SND : 20/02/2024, 08 :00 à 09 :45, B208 (42), Bedouet Judicael
— 1SNE, 1SNF : 20/02/2024, 08 :00 à 09 :45, A302 (64), Ouederni Meriem
— 1SNG, 1SNH : 21/02/2024, 14 :00 à 15 :45, C101 (90), Cregut Xavier
— 1SNI, 1SNJ : 21/02/2024, 14 :00 à 15 :45, C103 (88), Bedouet Judicael
— 1SNK, 1SNL : 21/02/2024, 14 :00 à 15 :45, A303 (64), Dupont Guillaume

TD 4 : Exceptions

Le TD 4 propose de manipuler les exceptions sur un programme simple qui calcule les somme des arguments de la ligne de commande.

Voici les fichiers fournis.

3. `http://www.openjml.org/`
4. `http://www.jmlspecs.org/`

1.2.13 Séance Projet 3 (20/02/2024, 10 :15 au 21/02/2024, 16 :15)

— 1SNA, 1SNB : 20/02/2024, 10 :15 à 12 :00, C214a+C214b, Cregut Xavier
— 1SNC, 1SND : 20/02/2024, 10 :15 à 12 :00, C201 (18)+C202 (18), Charles William
— 1SNE, 1SNF : 20/02/2024, 10 :15 à 12 :00, C204 (32)+C206 (32), Lebegue Jean-Claude
— 1SNG, 1SNH : 21/02/2024, 16 :15 à 18 :00, C216a+C216b, Chigot Estelle
— 1SNI, 1SNJ : 21/02/2024, 16 :15 à 18 :00, C214a+C214b, Pelissier-Combescure Marie
— 1SNK, 1SNL : 21/02/2024, 16 :15 à 18 :00, C204 (32)+C206 (32), Dupont Guillaume

PR 2 : Projet court

Voir la rubrique correspondante ci-dessous pour plus d'information sur le projet court.

1.2.14 Séance TP 5 (27/02/2024, 08 :00 au 28/02/2024, 16 :15)

— 1SNA : 27/02/2024, 08 :00 à 09 :45, C203 (32), Bonnet Louis
— 1SNB : 27/02/2024, 08 :00 à 09 :45, C205 (32), Cregut Xavier
— 1SNC : 27/02/2024, 08 :00 à 09 :45, C306 (32), Charles William
— 1SND : 27/02/2024, 08 :00 à 09 :45, C304 (32), Gasparini Simone
— 1SNE : 27/02/2024, 08 :00 à 09 :45, C301(18), Ouederni Meriem
— 1SNF : 27/02/2024, 08 :00 à 09 :45, C302 (18), Lebegue Jean-Claude
— 1SNG : 28/02/2024, 16 :15 à 18 :00, C216a, Chigot Estelle
— 1SNH : 28/02/2024, 16 :15 à 18 :00, C216b, Gasparini Simone
— 1SNI : 28/02/2024, 16 :15 à 18 :00, C201 (18), Pelissier-Combescure Marie
— 1SNJ : 28/02/2024, 16 :15 à 18 :00, C202 (18), Megna Anael
— 1SNK : 28/02/2024, 16 :15 à 18 :00, C204 (32), Dupont Guillaume
— 1SNL : 28/02/2024, 16 :15 à 18 :00, C206 (32), Riviere Peter

TP 9 : Exception : Agendas hiérarchiques

Le TP 9 propose de manipuler les exceptions au travers de la modélisation d'agendas et de groupes d'agenda.

Voici les fichiers fournis.

1.2.15 Séance TD 5 (27/02/2024, 10 :15 au 28/02/2024, 14 :00)

— 1SNA, 1SNB : 27/02/2024, 10 :15 à 12 :00, B006 (27), Cregut Xavier
— 1SNC, 1SND : 27/02/2024, 10 :15 à 12 :00, B306 (49), Bedouet Judicael
— 1SNE, 1SNF : 27/02/2024, 10 :15 à 12 :00, A302 (64), Ouederni Meriem
— 1SNG, 1SNH : 28/02/2024, 14 :00 à 15 :45, A301 (40), Cregut Xavier
— 1SNI, 1SNJ : 28/02/2024, 14 :00 à 15 :45, C002 (78), Bedouet Judicael
— 1SNK, 1SNL : 28/02/2024, 14 :00 à 15 :45, A303 (64), Dupont Guillaume

TD 5 : Un éditeur orienté ligne

Le TD 5 constitue un exercice de synthèse qui a pour but d'illustrer les atouts de l'approche objet. Il propose de programmer un (mini-)éditeur orienté ligne piloté par un menu textuel.

1.2.16 Séance Examen 1 (13/05/2024, 10 :00)

— 1SNA, 1SNB, 1SNC, 1SND, 1SNE : 13/05/2024, 10 :00 à 11 :30, C101 (90)+C103 (88), Riviere Peter, Singh Neeraj
— 1SNF, 1SNG, 1SNH, 1SNI, 1SNJ, 1SNK, 1SNL : 13/05/2024, 10 :00 à 11 :30, B00 (389), Alcouffe Remy, Bedouet Judicael, Ouederni Meriem, Pelissier-Combescure Marie

Examen 1 : Examen

1.2.17 Séance Oral 1 (28/05/2024, 10 :15 au 28/05/2024, 16 :15)

— 1SNA, 1SNB : 28/05/2024, 14 :00 à 15 :45, C103 (88), Cregut Xavier, Duffort Olivier
— 1SNC, 1SND : 28/05/2024, 14 :00 à 15 :45, C101 (90), Bedouet Judicael, Francois Gilles
— 1SNE, 1SNF : 28/05/2024, 16 :15 à 18 :00, C103 (88), Duffort Olivier, Ouederni Meriem
— 1SNG, 1SNH : 28/05/2024, 16 :15 à 18 :00, C101 (90), Cregut Xavier, Francois Gilles
— 1SNI, 1SNJ : 28/05/2024, 10 :15 à 12 :00, C103 (88), Bedouet Judicael, Duffort Olivier
— 1SNK, 1SNL : 28/05/2024, 10 :15 à 12 :00, C101 (90), Dupont Guillaume, Francois Gilles

1.3 Projets

1.3.1 Mini-projet

Le sujet

Le mini-projet propose d'écrire une classe Cercle (y compris son programme de test et sa documentation !).
Organisation : Un 'git pull' depuis votre projet git des TP vous fera apparaître le dossier mini-projet. C'est dans ce dossier qu'il faut travailler et rendre les deux versions de votre mini-projet. Vous recevrez aussi des retours via Git.
Vérification des normes de programmation avec checkstyle : checkstyle est un outil pour automatiser la vérification des standards de programmation. Il permettra par exemple de vérifier le nom des identifiants (mais pas leur sens !), les commentaires de documentation, l'utilisation des espaces, etc.

Pour lancer checkstyle, faire depuis une machine Linux de l'N7 :

```
java -jar /mnt/n7fs/ens/tp_cregut/checkstyle.jar -c checkstyle.xml Cercle
```

Le fichier checkstyle.xml est le fichier de configuration à utiliser.
checkstyle ne sera utilisé que sur Cercle.java.

Jalons

- **23/01/2024 :** mise en ligne du sujet.
- **10/02/2024, en fait 08/02 :** début des retours automatiques (ils peuvent commencer avant).
- **10/02/2024, décalée au 13/02/2024 :** date limite pour la remise de la première version complète.

  Attention : Cette première version doit être une version complète de votre mini-projet. Même si vous aurez un retour et la possibilité de l'améliorer, ne considérez pas que la seule date qui compte est la date de deuxième rendu !

  Pénalités : Vous aurez des pénalités lors de la première évaluation si le travail fourni n'est pas jugé suffisant. En particulier, les pénalités suivantes s'appliqueront :
  - 3 points : pas de rendu sur Gitlab. Vous devez réussir à rendre votre travail sur Gitlab. Le travail rendu par courriel ne sera pas évalué.

    Attention : Il ne faut pas déplacer les fichiers sinon les tests automatiques ne fonctionneront pas.
  - 2 points : la classe Cercle ne compile pas.
  - 1 point : la classe SujetCercleTest ne compile pas.
  - 1 point : la classe CercleTest ne compile pas (ou est vide).
  - 0.5 point : la classe CercleTest s'exécute avec des erreurs.
  - 1 point : pas de commentaire de documentation.
- **semaine du 12/02/2024 :** évaluation de la première version par les enseignants de TP.
- **24/02/2024 :** date limite pour la remise de la deuxième et dernière version (toujours sur Gitlab).

Évaluation automatique

Le résultat des tests automatiques est disponible dans votre projet Git, sur la branche "evaluations", dans le fichier `evaluations/evaluation1.txt` (et dans le dossier `evaluations/evaluation1`). Les vérifications faites sont détaillées dans les paragraphes suivants.

Le plus simple est de consulter ces résultats via Gitlab et en sélectionnant la branche "evaluations" au lieu de "main". Vous pouvez aussi les consulter via eclipse en changeant de branche ou depuis un terminal (git switch evaluations). Attention, cependant, vous ne pouvez pas pousser de modifications sur la branche evaluations. Il faudra donc revenir à la branche main pour modifier votre code. Si vous voulez avoir les remarques en locale, le plus simplet et le plus sûr est certainement de faire une autre copie de votre projet.

Notez bien que cette évaluation automatique constituent une évaluation de votre mini-projet. Vous n'avez pas accès au code source de certains de ces tests et vous ne pouvez donc pas les lancer vous-même. Cependant, les tests sont lancés de temps en temps. Aussi si vous faites un commit, lors du prochain lancement des tests, les résultats seront disponibles sur votre projet (récupérables via un pull).

L'évaluation automatique est lancée de temps en temps sur le code qui a été poussé sur Gitlab.

Attention, les tests fournis ne sont pas exhaustifs et de nouveaux tests pourront être ajoutés pour l'évaluation finale.

La classe RobustesseCercleTest permet de vérifier l'instrumentation des préconditions avec assert . Votre classe Cercle doit réussir ces tests (voir evaluation1 . txt).

Attention, pour RobustesseCercleTest , il faut activer la vérification des assertions en utilisant −ea en ligne de commande ou, sous Eclipse, faire Run As -> RUN Configuration -> Arguments ajouter −ea dans **VM Arguments**.

Les classes FormeCercleTest et Forme2CercleTest permettent de faire des vérifications sur la manière dont votre programme est écrit. Nous avons décidé de ne pas vous donner ces classes. Vous ne pourrez donc pas rejouer ces tests quand vous le souhaitez.

Vous ne pouvez pas non plus rejouer les tests qui concernent votre classe CercleTest. En effet, pour tester cette classe, nous nous servons de classes Cercle contenant des erreurs (dites mutantes) et vérifions si votre classe de test détecte ces erreurs. Ainsi, les tests CercleMutant# permettent de tester la classe CercleTest. Le principe est le suivant : chaque classe CercleMutant# est la classe Cercle avec une erreur. Cette erreur ne concerne bien sûr que E12, E13 ou E14 (voir la contrainte C6 du sujet). Si votre classe CercleTest ne détecte pas l'erreur, c'est qu'elle est incomplète : les tests qu'elle fait ne sont pas suffisants. C'est à vous de voir comment la compléter.

La classe CercleMini est une classe Cercle sans erreur. La classe CercleTest ne doit donc pas signaler d'erreurs sur cette classe. Pour vérifier que vous ne testez que les éléments de E12, E13 et E14, certaines méthodes ont été « désactivées » : elles lèvent une exception.

Nous ne pouvons pas vous donner les classes CercleMini et CercleMutant#. ce serait trop facile ! Quand on écrit un programme de test, on ne connait pas les erreurs qui ont été ou seront commises lors de l'écriture du code à tester.

Détection de similitudes : Pour vous aider à identifier les redondances dans votre classe Cercle, nous avons exécuté sim_java dessus (voir fichier evaluation1 . txt).

Attention, les portions de code qui sont indiquées comme étant similaires ne correspondent pas toujours à de la redondance de code !

Première évaluation

Pénalités sur le premier rendu : Les pénalités automatiques sur le premier rendu ont été calculées. Elles sont disponibles sur votre dépôt dans le fichier evaluations/penalites1.txt (et le détail dans le dossier evaluations/penalites1).

Retours sur la première version : Voici les retours en html et en (pdf).

Il s'agit d'une compilation des erreurs ou maladresses rencontrées. Vous devez donc les lire et, si elles s'appliquent à votre mini-projet, en tenir compte pour la version finale. L'intérêt de cette approche est que chacun, quelque soit son niveau d'avancement dans le mini-projet, a accès à toutes les remarques.

Bien sûr, votre intervenant de TP peut vous faire des remarques spécifiques si vous avez commis une maladresse très particulière qui n'a donc pas d'intérêt pour les autres étudiants.

Les tests automatiques constituent aussi une évaluation de votre mini-projet.

FAQ

Comment seront comptabilisées les pénalités ? Les pénalités limitent la note que vous pourrez avoir. Par exemple, si vous avez eu 4 points de pénalité, donc 2 points sur 20, la note finale sera la plus petite de la note obtenue sur la version V2 et de 18.

La formule est donc la suivante :

```
note_finale = Min(note_V2, 20 - pénalités_V1 / 2)
```

1.3.2 Projet court

Sujet

Le projet court doit être fait à partir des éléments fournis dans votre projet Git (dossier 'projet-court') et rendu via ce même projet Git sur Gitlab.

Il ne faut pas déplacer les fichiers fournis !

Jalons

Voir le sujet...

Tests en boîte noire

Pour que vous sachiez comment votre programme sera testé, nous vous fournissons quelques tests ainsi qu'un script qui permet de les lancer (testeur.sh). Ils sont sur votre projet Git dans le dossier tests/.

Chaque test est défini par deux fichiers. Le premier a l'extension .run et est le test à lancer (par exemple en faisant : sh testeur.sh leTest.run). Le second a l'extension .expected. Il contient le résultat attendu, ce que le programme doit afficher dans le terminal.

Le fichier testeur.sh est le testeur. Il prend en argument les tests à exécuter (fichier avec l'extension .run).

```
# Lancer le seul test exempleTricheurSujet.run
sh testeur.sh tests/exempleTricheurSujet.run

# Lancer tous les tests du dossier courant
sh testeur.sh tests/*.run
```

Il lance chaque test (fichier .run) et enregistre son résultat avec l'extension .computed. La commande diff permet de calculer la différence entre les deux fichiers (.expected et .computed). Les options -Bbw de diff sont utilisées pour ignorer les lignes vides et les blancs. Le verdict est enfin affiché, 'ok' ou 'ERREUR'. Dans le cas d'une erreur, le contenu du fichier diff est affiché.

Les tests peuvent être lancés depuis le dossier 'projet-court' ou le dossier 'projet-court/src'.

Attention à ne pas ajouter les fichiers .diff ou .computed sur Git car ce sont des fichiers engendrés. On peut ajouter les lignes suivantes dans le fichier .gitignore (à la racine de votre projet) :

```
*.diff
*.computed
```

Réponses à vos questions

Recommencer une partie Doit on pouvoir recommencer une partie une fois l'actuelle terminée ? (avec alternance du départ entre joueur 1 et joueur 2 ?)

Non. Une seule partie.

NoSuchElementException. j'ai aujourd'hui effectué les tests boites noires que vous nous avez fournis.

Je n'ai qu'une seule erreur que je n'arrive pas à comprendre : il semblerait que dans exemplePresqueSujet, vous testiez avec les entrées suivantes : 2 - X - 5 - 3 - 1.

Sous Eclipse, comme sur un terminal, je peux tout à fait lancer une partie avec Toto@humain et Titi@rapide puis saisir ces 5 arguments au fur et à mesure sans avoir d'erreur et ainsi finir la partie. Cependant, avec la commande sh testeur.sh tests/*.run, l'exemplePresqueSujet me lève une erreur "NoSuchElementException" dans le Scanner de ma classe Humain. Il semblerait que ce soit à la saisie du deuxème caractère (X).

Il semblerait également pour en avoir discuté avec mes camarades que je ne sois pas le seul à avoir cette exception. Ne peut-elle pas venir du test ?

NoSuchElementException se produit quand il n'y a plus de données en entrée. Vous pouvez le reproduire en tapant un CTRL-D au clavier (qui ferme l'entrée standard).

Le problème vient de la manière dont vous utilisez Scanner. Vous créez un nouveau Scanner à chaque getPrise(). Le Scanner précédent a consommé les caractères (lecture en avant). Ils ne sont donc plus disponibles pour les lectures suivantes.

Vous pouvez le vérifier en ajoutant, juste avant la création du Scanner et juste après une saisie :

```
try {
    System.out.println(" available ␣:␣" + System.in.available());
} catch (java.io.IOException e) {
    System.out.println(e);
}
```

qui affiche le nombre de caractères disponibles sur l'entrée standard.

Vous verrez que s'il y a 10 caractères disponibles avant la création du Scanner, dès qu'une lecture est faite, il n'y a plus de caractères disponibles sur l'entrée standard. D'où le NoSuchElementException.

Ce qu'il faut faire, c'est ne créer qu'un seul Scanner pour toute l'application.

Tests fournis. Que fait l'utilisateur ? Vous nous avez fourni un fichier contenant des exemples de tests.

Lors de l'exemple "exemplePresqueSujet.expected", ligne 12, je ne suis pas sûre de ce que fait l'utilisateur. Est-ce bien un " " ?

Pour savoir ce que fait l'utilisateur il faut regarder le fichier en .run. Ce que fournit l'utilisateur n'apparaît pas sur la sortie (et donc pas non plus dans .expected) car ce n'est pas saisi au clavier (donc pas d'écho dans le terminal) mais récupéré grâce à la redirection de l'entrée standard avec '«EOF'. Ce qui est fourni au programme est donc entre les deux EOF dans le .run. L'utilisateur rentre donc successivement :

```
2
X
5
3
1
```

Problème de compilation. J'ai voulu compiler (juste compiler) la classe Jouer sans rien modifier et elle ne compile pas car la classe ConfigurationException est introuvable alors qu'elle est dans le même paquetage.

Il faut être dans le dossier qui contient le dossier "allumettes" (le plus simple)... Ou définir le CLASSPATH ou utiliser l'option '-cp' de javac et java.

```
cd ...../projet-court/src
javac allumettes/Jouer.java
java allumettes.Jouer
```

Utiliser Eclipse est certainement une bonne idée ! Il est important de savoir utiliser ce type d'outil qui peut, par bien des aspects, vous faciliter la gestion du code Java (ou d'autres langages).

Mon programme boucle avec le message "vous devez donner un entier" avec exemplePresqueSujet J'ai un problème dans le projet court, c'est que j'ai exécuté les tests fournis et ils marchent très bien mais quand j´exécute le test avec exemplePresqueSujet je trouve dans le fichier.computed une boucle infinie avec le message "vous devez donner un entier" après la saisie de 'X' alors que dans eclipse ça marche très bien avec les mêmes entrées.

Lors d'une saisie incorrecte, il faut consommer les caractères qui posent problèmes, ici "X". On peut utiliser la méthode nextLine() de Scanner pour le faire.

1.3.3 Projet long

Nous vous proposons de définir vous-mêmes le sujet du projet long sur lequel vous allez travailler. Vous trouverez ici les principales consignes et les jalons de ce projet.

Rappel : Ce projet sert de support à la matière Gestion de projet / Méthodes agiles (SCRUM) de l'UE SHS et sera évalué aussi bien par TOB (Projet Long) que par Gestion de projet. Vous aurez donc 2 notes.

Constitution des équipes

Une équipe (un groupe de projet) doit être constituée au sein d'un même groupe de TD (tous les membres d'une équipe doivent appartenir au même groupe de TD).

Il y aura 4 équipes par groupe de TD. Ceci fera donc des équipes de 7 personnes, plus ou moins une.

Vous ne pouvez pas faire partie de la même équipe de TOB si vous étiez déjà dans une même équipe en PIM.

Chaque équipe sera suivie par un enseignant qui est son enseignant de TD TOB :

— AB : Xavier Crégut <prenom.nom@toulouse-inp.fr>
— CD : Judicael Bedouet <prenom.nom@onera.fr>
— EF : Meriem Ouederni <prenom.nom@toulouse-inp.fr>
— GH : Xavier Crégut <prenom.nom@toulouse-inp.fr>
— IJ : Judicael Bedouet <prenom.nom@onera.fr>
— KL : Guillaume Dupont <prenom.nom@toulouse-inp.fr>

Pour la partie Méthodes agiles (SCRUM), vous pouvez poser vos questions à Gilles FRANCOIS <prénom.nom.e@thalesdigital.io> (quand le premier TD de la matière aura eu lieu).

Cette année les équipes de projet seront choisies au hasard... Sauf si les groupes de TD arrivent à former des 4 équipes avec comme règle « deux personnes dans la même équipe en PIM ne peuvent pas être dans la même équipe en TOB ».

Notation du projet

— Chaque étape du projet (décrite ci-après) sera prise en compte dans la note du projet.
— À la fin du projet, une démonstration, une présentation orale et un compte-rendu concernant l'organisation du projet seront faits. **Trois personnes seront tirées au sort**, chacune pour présenter l'un des aspects. Les autres membres de l'équipe répondront aux questions.
— Une partie de la note sera définie par l'équipe elle-même.
  Chaque membre de l'équipe répartira un nombre de points égal à 1 de moins que la taille de son équipe entre ses coéquipiers. Chaque coéquipier aura un nombre de points compris entre -1 et 3. Par exemple, pour une équipe de 6 personnes, chaque équipier pourra répartir 5 points sur ses coéquipiers en donnant à chacun un nombre de points entre -1 et 3.
  La note d'un étudiant sera donc : (note de l'enseignant) + (moyenne des points attribuées par ses coéquipiers - 1).

2024/01/26 : Jalon « Équipes constituées » (étudiants... Ou enseignants).

2024/02/05 : Jalon « Création projets Git » (Xavier Crégut).

Création des projets Git qui seront utilisés pour le projet. Les projets ne peuvent être créés qu'une fois les équipes constituées.

2024/02/10, décalé 2024/02/12 : Jalon « Liste des sujets envisagés » (équipes).

Chaque équipe déposera sur le projet Git dans le dossier livrables un document sujets.pdf qui contiendra au moins 4 sujets de projet envisagés, classés par ordre de préférence décroissante de l'équipe (le premier sujet est le sujet préféré de l'équipe).

Les sujets proposés doivent être décrits dans le document sujets.pdf (tout en minuscules) qui doit être déposé dans le dossier « livrables »

Chaque sujet doit être décrit sur une page environ dans le but de convaincre de la pertinence et de l'intérêt de votre projet. Vous devez considérer que ce document s'adresse à des clients potentiels, à des business angels, à votre supérieur hiérarchique...

Il doit y avoir suffisamment d'information sur le projet pour que votre enseignant puisse le comprendre et évaluer le travail envisagé. Ce document sera le point de départ pour discuter de votre projet avec votre enseignant. C'est lui qui validera le sujet de votre projet au final.

En aucun cas il ne faut parler des solutions informatiques qui sera apportées pour réaliser ce projet. Il ne s'agit pas de décrire comment ce sujet sera traité d'un point de vue programmation mais ce qu'il apportera à ses utilisateurs.

Quelques recommandations :

1. Le projet doit être suffisamment ambitieux puisque les équipes sont de grande taille ! Vous ne devez pas pouvoir terminer le projet dans le temps imparti. C'est le but.
2. L'application développée devra être pourvue d'une interface graphique.
3. Vous ne ferez ni une application concurrente, ni une application distribuée, ni une application client/serveur (ces notions seront vues en 2A).

Attention, vous ne pouvez pas prendre un sujet qui serait du type « boîte à jeux » où chaque membre de l'équipe s'occuperait d'un des jeux !

L'enseignant évaluera les sujets et donnera le feu vert à l'équipe pour traiter l'un des sujets (ou vous aidera à construire un sujet adapté aux attentes).

Exemples de sujets proposés et choisis les années passées

— Assistant d'aide aux raffinages.
— Application de proposition de repas
— Gestion et maintenance d'une maison intelligente
— Jeu guidé pour apprendre la programmation
— Manuscript Manager (aide à l'écriture d'un roman)
— Réalisation d'un RayTracer
— Simulation de la propagation d'un virus
— Escape game
— Application de gestion des crèches
— Programme d'apprentissage musical

2024/02/16 : Jalon « Sélection du sujet de projet » (enseignant responsable de l'équipe de projet).

L'enseignant responsable de l'équipe de projet communiquera le projet retenu.

2024/02/24 : Jalon « Fonctionnalités de l'application » (équipe).

Il s'agit de décrire les fonctionnalités de l'application que vous allez développer dans un document appelé fonctionnalites.pdf (nom à respecter impérativement, nom tout en minuscules, sans accents), à déposer dans le dossier « livrables ». Il doit contenir :

1. une page de garde qui rappelle le numéro de l'équipe, ses membres, le titre du projet, l'objet du document...
2. l'objectif général du projet
3. une description des fonctionnalités (attendues) de l'application qui sera développée. Il s'agit de décrire le service rendu (besoin) et non la manière dont ce service sera rendu (comment).
4. les interfaces utilisateur envisagées (esquisses réalisées à main levée ou avec un outil de dessin).
5. plusieurs cas d'usage (scénarios) pour comprendre les fonctionnalités (et, plus tard, valider l'application développée).
6. (facultatif) En annexe, lister les points durs, les points qui vous paraissent difficiles dans le projet proposé.

Il ne faut pas hésiter à mettre beaucoup de fonctionnalités. Les étapes suivantes consisteront à les organiser, définir des priorités, etc. Tout ne sera pas à développer lors du projet long !
Attention : Ici, on ne doit pas parler de programmation objet, d'UML, de Java, etc. On ne veut pas savoir comment sera réalisée l'application mais bien qu'elles sont les les possibilités qu'elle offrira à ses utilisateurs.

Ce document pourrait donc servir de trame pour le futur manuel utilisateur. Il restera à ajouter quelques captures d'écrans pour remplacer les esquisses et les modes opératoires tels qu'ils ont été finalement réalisés.

Semaines 13 à 21 : Réalisation du projet en 3 itérations.

Le projet sera organisé en trois itérations de 2 semaines (précédées d'une itération 0).

Attention : à chaque fin d'itération, chaque membre de l'équipe doit déposer un très court rapport INDIVIDUEL qui décrit ce qu'il a fait lors de l'itération. Ce rapport sera nommé `rapport#-$USER.pdf` (tout en minuscules) où `$USER` est votre identifiant LDAP (votre login) et # est le numéro de l'itération (1, 2 ou 3). Ce rapport doit être court et précis et lister les activités réalisées (en indiquant le degré d'avancement) sur une page maximum. Les détails de la réalisation seront donnés dans le rapport général.

À la fin de chaque itération, il faut aussi rendre le code complet de l'application dans un fichier source#. jar , le manuel utilisateur manuel− utilisateur #. pdf et le rapport associé rapport #. pdf, # est le numéro de l'itération (1, 2 ou 3).

Le rapport documente votre projet. Il doit inclure les éléments élaborés suite aux séances gestion de projet mais aussi les éléments de conception (diagrammes UML). Les versions intermédiaires de ce rapport ne seront pas évaluées. Le rapport à la dernière itération constituera le rapport final.

Le manuel utilisateur explique comment utiliser le programme livré. Comme pour le rapport, les version intermédiaires ne seront pas évaluées.

Attention à bien vérifier que les .java sont présents dans l'archive).

On peut construire un jar depuis Eclipse (il faut penser à sélectionner l'inclusion des sources) ou depuis la ligne de commande. Les options de la commande jar sont globalement les mêmes que tar.

Semaine 13 : 2024/03/25 au 2024/03/30, Itération 0

.

Semaines 14 & 17 : 2024/04/02 au 2024/04/27, Itération 1

.

2024/04/27, Remise des livrables Itération 1 (rapports individuels + code + manuel utilisateur + rapport général).

Semaines 18 & 19 : 2024/04/29 au 2024/05/13, Itération 2

.

2024/05/13, Fin Itération 2 : rapports individuels + code + manuel utilisateur + rapport général.

Semaines 20 & 21 : 2024/05/14 au 2024/05/25, Itération 3

.

2024/05/25, Fin Itération 3 : rapports individuels + code + manuel utilisateur + rapport général).

Vous devez rendre dans le dossier livrables de votre dépôt SVN :

1. Le rapport (rapport3.pdf) doit inclure :
   (a) Un rappel très succinct du sujet. Les détails seront dans le manuel utilisateur.
   (b) Les principales fonctionnalités (user stories ou epics) identifiées en précisant leur état de réalisation (et dans quelle itération elles ont été faites pour celles qui sont terminées ou commencées).
   (c) Le découpage de l'application en sous-systèmes (en paquetages).
   (d) Les diagrammes de classe qui donnent l'architecture de votre application. On pourra soit faire un diagramme de classe global, soit un diagramme de classe par fonctionnalité.

On ne fera apparaître sur ces diagrammes que les informations utiles pour comprendre l'architecture. En particulier, il est inutile de faire apparaître toutes les méthodes de toutes les classes.

(e) La dynamique du système, toujours en s'appuyant sur les diagrammes UML.

(f) Les principaux choix de conception et réalisation, les problèmes rencontrés et les solutions apportées, etc.

(g) L'organisation de l'équipe et la mise en œuvre des méthodes agiles.

Il n'y a pas de nombre de pages à respecter. Il faut être concis et précis. Sachez que ce rapport est le point d'entrée sur votre code. Il doit permettre au lecteur de comprendre comment vous avez traité votre sujet et donc de savoir ce que vous avez fait sans avoir à consulter le code. Si le lecteur est intéressé par un aspect particulier, la lecture du rapport devrait l'éclairer sur ce qui a été fait, comment ça a été fait et les sources à lire pour avoir tous les détails d'implantation.

2. le manuel utilisateur : manuel-utilisateur3.pdf
3. les sources de votre projet : sources3.jar.

2024/05/29, 8h : Remise du support pour la présentation orale

Le support de présentation utilisé pour l'oral du projet : presentation.pdf (voir ci-dessous pour les consignes) doit être déposé dans le dossier livrables.

2024/05/29, 8h : Points d'équipe

Modalité à venir...

2024/05/29 : Présentation orale.

Pour la **présentation orale**, le déroulement sera le suivant :

1. 14 minutes (maximum à ne pas dépasser !) à utiliser pour :
   — une démonstration de l'application réalisée
   — une présentation technique du travail réalisé (architecture, choix de conception et réalisation, etc.). UML est certainement un outil utile pour cette présentation...
   — une présentation de l'organisation de l'équipe et la mise en œuvre des méthodes agiles.

   Vous devriez passer de 3 à 6 minutes sur chacune de ces trois parties.

   Attention : Les trois orateurs seront tirés au sort. Chacun doit donc être prêt à présenter chacune des trois parties.

   Vous êtes libres d'organiser ces trois présentations dans l'ordre que vous souhaitez (même s'il paraît logique de commencer par la démonstration car tous les membres du jury ne connaitront pas votre sujet).
2. 10-15 minutes de questions/discussions.

Il peut être intéressant de commencer par la démonstration car c'est l'occasion de rappeler le périmètre du projet et comment il a été traité du point de vue des utilisateurs. La présentation orale se concentrera sur les aspects techniques et organisationnels du projet.

La **présentation technique** doit être technique ! Il n'est pas utile de nous détailler le sujet (la démonstration l'a fait). Ce qu'il faut faire, c'est nous expliquer comment vous l'avez traité : quels sont les choix faits ? Quels sont les problèmes rencontrés et les solutions apportées. Nous vous rappelons qu'UML est un bon outil pour communiquer !

La **gestion de projet** doit expliquer comment l'équipe s'est organisée, les outils utilisés en accord avec ce qui a été présenté en gestion de projet.

Horaires de passage pour l'oral

Toutes les équipes assistent à toutes les présentations de leur groupe de TD.

L'ordre de passage des équipes sera choisi au fur et à mesure.

1.4 Ressources

En suivant les liens, vous trouverez :

— l'archive afficheur.jar et sa documentation.

— Une version de junit4 : junit4.jar.

— quelques annales

— Le carnet de bord Agile fourni par Gille FRANÇOIS, son supplément (en particulier la charte produit) et le lien sur la vidéo « La Gestion de Projet Agile en deux mots » [5] ainsi que le support utilisé pour le TD 2.

1.4.1 Réponses à quelques questions fréquentes

Comment s'identifier sur les QCM de TD ?

Pour vous identifier, nous utilisons votre numéro Apogée que vous trouvez sur planète dans Mon dossier Web ou Mes notes. C'est un numéro a 8 chiffres qui commence par leur année de première inscription. On utilisera les 5 derniers chiffres et on codera un chiffre par ligne en haut du QCM. Le chiffre des unités est sur la dernière ligne : le nombre se lit de haut en bas.

Par exemple, si votre numéro Apogée est 20230471, les 5 derniers chiffres sont 30471. Il faut donc cocher le 3 sur la première ligne, le 0 sur la deuxième, le 4 sur la troisième, le 7 sur la quatrième et le 1 sur la dernière.

Il est aussi important de mettre votre nom (en majuscules !) et votre prénom dans le cadre prévu à cet effet.

Comment cocher une case sur un QCM ?

Il faut utiliser un stylo noir ou bleu, pas de rouge, pas de vert, pas de crayon à papier.

Il faut mettre suffisamment de matière dans la case (croix large, case noirci).

Comment corriger une case cochée par erreur sur un QCM ?

5. `https://www.youtube.com/watch?v=3qMpB-UH9kA`

Il faut utiliser du blanc pour effacer la case et son contenu et, surtout, il ne faut pas redessiner la case. Ceci conduirait certainement à la considérée comme cochée.

Comment vérifier que j'ai réussi à mettre mes fichiers sur le Gitlab ?

Plusieurs solutions sont possibles :

1. Vous pouvez taper la commande (à la racine de votre dépôt) : `git status`
2. Vous pouvez utiliser un navigateur pour aller sur Gitlab, sur votre projet et vérifier que tous les fichiers y sont, avec vos dernières modifications.
3. Vous pouvez faire une nouvelle copie locale de votre dépôt (`git clone`) dans un nouveau dossier (par exemple /tmp) et voir ce qui remonte. Vous pouvez en profiter pour vérifier que le projet compile, les tests passent, etc.
4. ...