Test Driven Developpement

LOUIS RANDRIAMORA ANDRIANTSIORY
KOLAWOLE ABDOULAYE
RUDY ITUNIME MASALA

ECAM 2019-2020

Test Driven Developpement

Introduction :

Le but du travail est de concevoir les étapes de la réalisation d'un projet informatique dans le but d'écrire des tests pour un programme de gestion d'une ludothèque préalablement développé et de faire le refactoring de celui-ci.

Caractéristique :

Langage de programmation : Java
Interface de développement : Intellij IDEA
Outil de communication : UML (diagramme de classe), GitHub, FB Messenger

Objectif du projet :

Gestion d'une ludothèque

Description du projet :

Une ludothèque est composée de jeux et est gérée, visitée par des personnes. On aura donc une classe ***GameLibrary*** qui sera composée d'une classe ***Games*** et aura une relation d'agrégation avec une classe ***Person.***

Un jeu a donc un nom, un statut selon qu'il est disponible ou pas, un fabricant et sera soit un jeu vidéo, un jeu de société ou un jouet. La classe ***Games*** aura donc ces différents attributs et sera héritée donc par les classes ***VideoGame***, ***BoadGame***, ***Toy***.

Un jeu vidéo a une plateforme, un jeu de société à un nombre de joueurs et un jouet est fait avec un certain matériau.

Comme dit plutôt la ludothèque donc sera utilisé par des personnes qui ont des noms et une fonction qui est donc soit un gérant soit des adhérents.

La classe ***Person*** sera donc hérité par les classes ***Manager*** et ***Adherent*** sachant qu'un gérant peut être aussi un adhérent.

Le gérant s'identifiera à l'application et pourra gérer la ludothèque (ajouter des jeux, faire des recherches, obtenir la liste des prêts, etc…).

Les adhérents quant à eux emprunteront donc des jeux vidéo avec une date d'emprunt et une date de retour et pourra faire aussi des recherches de jeux pour entre autres voir quels jeux sont disponibles.

Il y a donc une classe ***Borrowing*** qui géra les emprunts avec les dates d'emprunts et de retour des jeux.

Critères :

- Maintenabilité

Utilisation de la programmation orienté objet

- Intégrité

Pratiquer de tests unitaires

- Simplicité

Facilité à comprendre la logique de programmation
Design pattern singleton

- Extensibilité

Emploi de l'Héritage
Emploi de l'encapsulation
Emploi du design pattern Factory , du design pattern Observer , du design pattern façade

Convention de nommage :

Packages
Tout en minuscule
Com.sun.eng

Constants
Tout en majuscule
static final int MIN_WIDTH = 4;

Classes et interfaces
1ère lettre du mot en majuscule et le reste en minuscule
class ImageSprite;
interface RasterDelegate;

Variables
1ère lettre en minuscule ensuite 1ère lettre du mot en majuscule et le reste en minuscule
float myWidth;

Lien github :

https://github.com/TestDrivenDeveloppement/Ludotheque.git

Diagramme de classe: