RAPPORT DE PROJET

PROJET
« DSP calculator »

Projet réalisé par :

Mouhab ZITOUNI
Nour El Houda GALAI

Projet encadré par :
Pierre FIOULHOUX
John CHAUSSARD
Xiaotong QIAN

REMERCIEMENTS

Nous tenons à remercier et à témoigner toute notre reconnaissance aux personnes suivantes pour avoir assuré un cours complet de Java et des TP bien dirigés, qui nous ont aidé à former une base solide en langage Java pour pouvoir réaliser ce projet :

M. FIOULHOUX : Pour avoir assurer un cours bien clair.
M. CHAUSSARD : Pour avoir assurer des Tds bien expliqués.
Mme. QIAN : Pour avoir assurer la bonne réalisation des Tps.

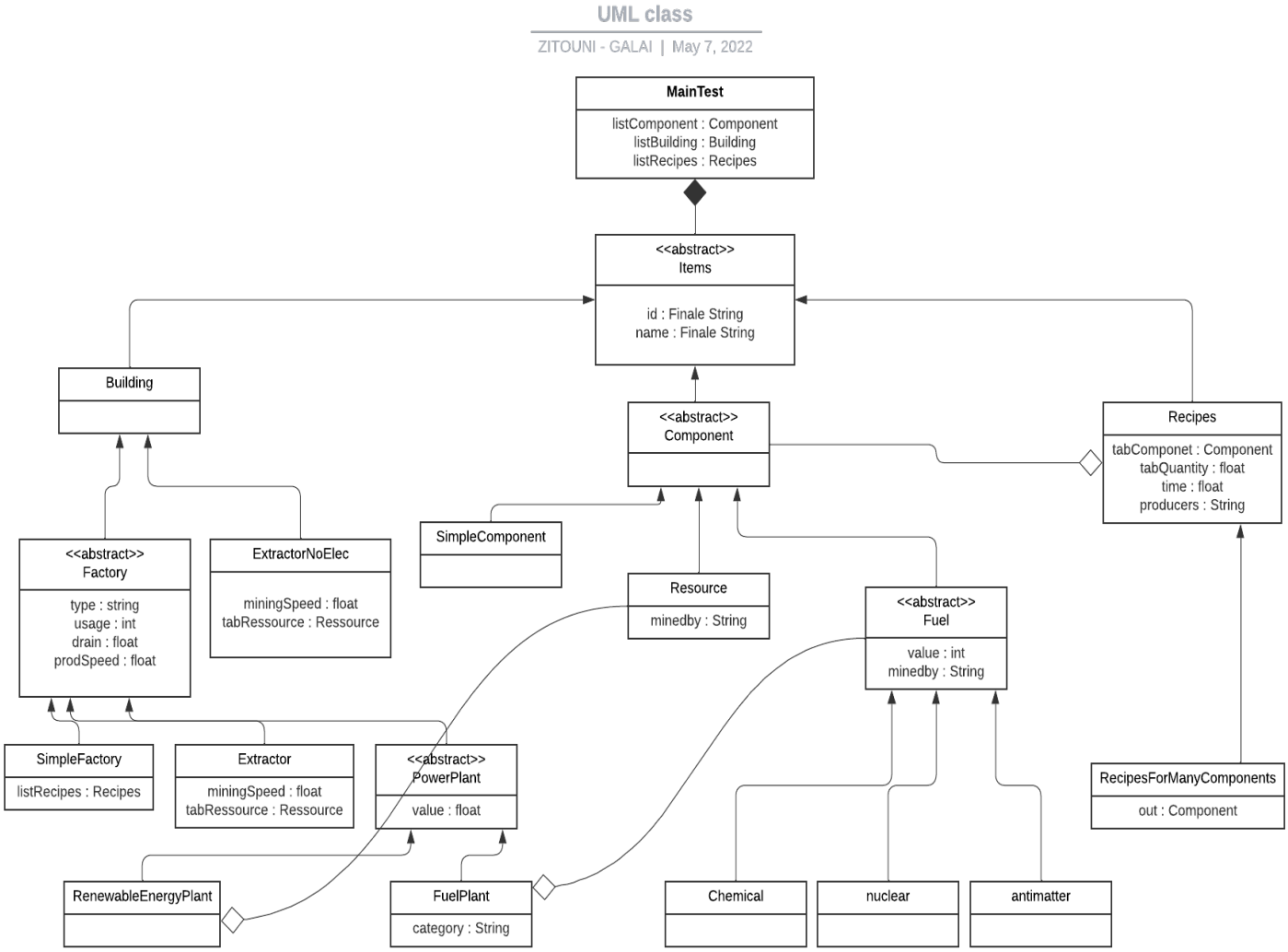
SOMMAIRE

I. INTRODUCTION

Dans le cadre de notre première année du cycle ingénieurs en Informatique à Sup Galilée, il nous est proposé un projet nous permettant de mettre en pratique nos connaissances et nos compétences en langage Java et en programmation orienté objet.

A ce propos, et à partir d'un fichier XML où on trouve tous les données antérieures d'un jeu de construction appelé « Dyson Sphere Program »; il est demandé de construire une structure de classes afin de proposer certaines fonctionnalités.

II. Diagramme de classe



Concernant le diagramme de classe, on a remarqué que : composants qui lui même contient bâtiment et ressource ; et aussi recette sont caractérisés par un Id et un nom (d'après le fichier XML) ; ce qui justifie la classe items caractérisée par Id et nom qui est la classe mère de tous ce qui est mentionné.

III. Classe Console Menu

III.1. Compilation et exécution

Le code se compile avec « javac MainTest.java » er s ‘execute avec « java MainTest ».

Dès le programme se lance un menu va s’afficher comme suit :

III.2. Menu et exception

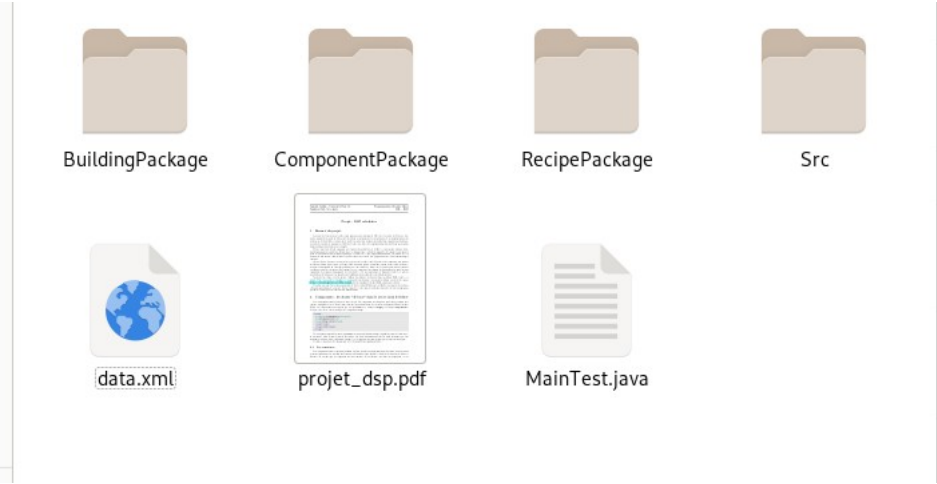
```
54    public void printMenu(){
55
56      System.out.println("----------Welcome to DSP Calculator----------\n");
57      System.out.println("Choose an option");
58      System.out.println("1: Show all components ");
59      System.out.println("2: Show all buildings");
60      System.out.println("3: Show all recipies");
61      System.out.println("0: Exit");
62
63    }
64
```

On a réussi à construire un Menu solide qui fonctionne qu'avec les chiffres affichés comme choix (1,2,3, ou 0). D'autres chiffres ou symboles ne vont être pas acceptés, dans le cas échéant, une exception va être générée grâce à une boucle comme indiquée ci dessous :

```
27    public void Run(){
28      while(true){
29        printMenu();
30        optionName = scanner.nextLine();
31        try{
32
33          option = Integer.parseInt(optionName);
34          switch (option){
35            case 1: test1(listComponent,""); break;
36            case 2: test2(listBuilding,""); break;
37            case 3: test3(listRecipes,""); break;
38            case 0: exit(0);
39            default:  System.out.print("\033\143");  System.out.print("Unvalid choice : Type valid numbers only\n\n"); Run();
40          }
41          break;
42        }catch(NumberFormatException e){
43          System.out.print("\033\143");
44          System.out.print("Unvalid choice : Type valid numbers only\n\n");
45        }
46      }
47    }
48
49
50
```

IV. Packages

On a organisé notre projet avec des répertoires et des sous-répertoires tout compatible avec l'hiérarchie du diagramme de classe, comme indiqué ci dessous :



Le MainTest contient le main du programme, et contient aussi la fonction de stockage de données qui s'appelle « StoreData » après laquelle arrive la fonction du « Menu » avec « new » et la méthode de « Run ».

MainTest.java

```java
import java.util.ArrayList;
import Src.*;
import RecipePackage.*;
import ComponentPackage.*;
import BuildingPackage.*;

public class MainTest {



  public static void main(String[] args)
  {

    ArrayList<Component> listComponent = new ArrayList<>();
    ArrayList<Building> listBuilding = new ArrayList<>();
    ArrayList<Recipes> listRecipes = new ArrayList<>();

    StoreData.StoreDataFromXml(listComponent, listBuilding,listRecipes);

    ConsoleMenu menu = new ConsoleMenu(listComponent, listBuilding,listRecipes);
    //System.out.print("\033\143");

    menu.Run();

  }

}

```

Comme demandé, la génération des exceptions est utilisées dans la classe « StoreData » au fur et à mesure du stockage des données.

```java
try{
  if(!category3.equals("nuclear") && !category3.equals("chemical") && !category3.equals("antimatter"))
   throw new FuelCategoryException(id, category3,"Fuel Plant");
  listBuilding.add(new FuelPlant(id, name, type, value, prodSpeed,category3 ));
}
catch(FuelCategoryException e){
   System.err.println(e);
}
```