
**Sorbonne université**

**Faculté des sciences et de l'ingénierie**

**Master Science et Technologie Logiciel (STL)**

# Rapport de Projet Module PAF

**Tabellout Salim 21307533**
**Tabellout Yanis 21307532**

**9 mai 2024**

# TABLE DES MATIÈRES

# TABLE DES FIGURES

CHAPITRE 1

# SIMCITY

## 1. Strcuture général du code

Notre code a été sélectionné de manière hiérarchique, ce qui nous permet de percevoir notre jeu comme un ensemble de petits composants qui se développent et évoluent progressivement au fil du temps. À ce stade, nous pouvons déjà discerner et classifier plusieurs modules distincts, dans deux répertoires différents, **src** et **test**.

### 1.1. Source

1. **Forme**
2. **Zone**
3. **Batiment**
4. **Types**
5. **Citoyen**
6. **Occupation**
7. **Maladies**
8. **Vehicules**
9. **Produit**
10. **Prefecture**
11. **Parking**
12. **Entreprise**
13. **Ville**

### 1.2. Structure générale des Test

Contient Tous les Tests de chaque Module, sous le format **NomModuleSpec**, et chacun des test est lancé dans le module **Spec**. plus de **200 Tests menés** sur nos modules pour garantir le bon fonctionnement de notre programme.

# 2. Strcuture des modules

## 2.1. Module Forme

Avant de définir une Forme, on doit commencer par définir la notion de cordoonées qui est plutot standard qui se composent de **Cx et Cy**, et donc pour définir une forme on peut distinguer trois types.

- **Segement Horizental** : les cordonnées '**Cy** restent les mêmes, en rajoute une longeur **L** avec laquelle on peut facilement trouver le point le plus à l'est (Cx +L).
- **Segement Vertical** : les cordonnées '**Cx** restent les mêmes, en rajoute une longeur **L** avec laquelle on peut facilement trouver le point le plus au sud (Cy+L).
- **Rectangle** : à partir des points le plus à l'est et le plus le Nord, et à partir de la **Longeur** et la **Hauteur** du rectangle on peut en déduire le point le plus à l'est (Cx +Longeur) et sud (Cy + Hauteur)

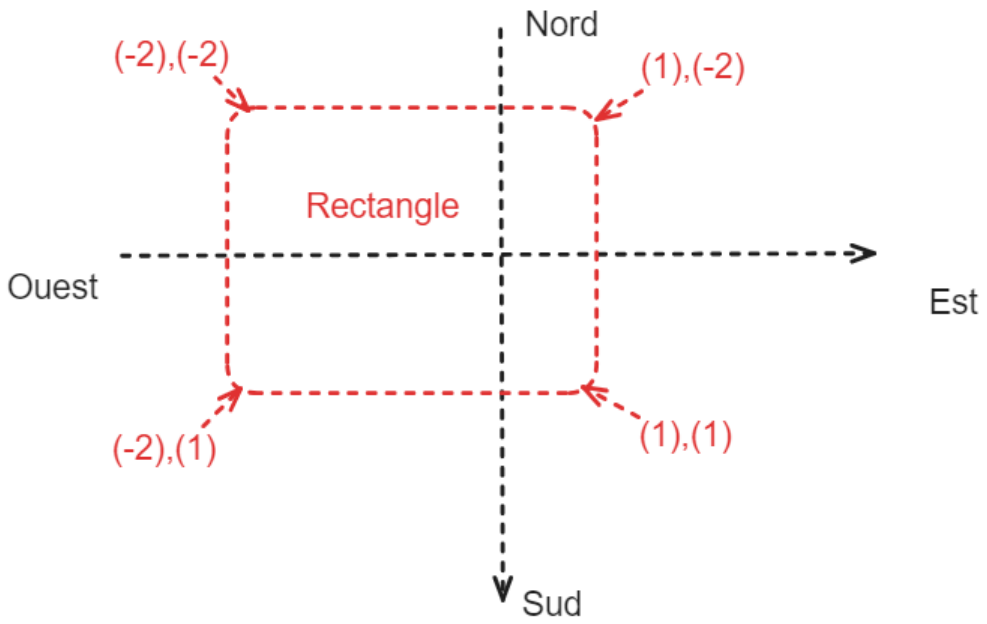



FIGURE 1.1 – Example d'un rectangle

On doit noter que le Nord contient les adresses les plus petites pour garder une cohérence avec les coordonnées de l'est, par conséquent pour trouver les points le plus au sud on doit rajouter la hauteur a notre point nord. Par conséquent, pour vérifier si un point appartient a une forme il suffit juste de vérifier qu'il est entre les limites de la forme.

### 2.1.1. Sous-fonctions

- **Appartient** : a partir d'un point et une forme, on décide si le point appartient à une forme.
- **Adjacent** : si un point est adjacent a une forme sans être en collision.

#### 2.1.2. Fonctions principales

- **collision** : On prend chaque Combinaison (WN, WS, EN, ES) de la première forme, et on vérifie si elle appartient a la deuxième forme.
- **adjacentes** : On prend chaque Combinaison (WN, WS, EN, ES) de la première forme et on vérifie si elles sont adjacentes à deuxième forme.

### 2.2. Module Zone

#### 2.2.1. Type de zone

1. Eau.
2. Route.
3. Zone Residentielle.
4. Zone Industrielle.
5. Zone Commerciale.
6. Administration.

#### 2.2.2. Fonctionnalitées

Dans ce module on peut :

- renvoyer la forme d'une zone.
- Verifier si une Zone est valide.
- Vérifier que deux zones sont disjointes, si ces dernières ne sont pas en collision.
- Savoir si une zone donnée est une zone routière.

### 2.3. Module Types

Ce module a été crée pour éviter les imports circulaire entre modules, il contient les **ID** des Zones, Bâtiments et citoyens, prefectures,vehicules, entreprises,parkings,banque. Contient aussi quelques données importantes

- **Nationalité** : Etranger ou français
- **Diplome** : Obtenu ou en cours
- **Personne** : Décris une personne, qui est composée de ID du citoyen, ses coordonnées,son occupation actuelle, ses crimes, sa nationalité et quels maladies est-t-il atteint
- **Vie** : Décris une Vie, qui est elle même composée de la somme d'argent en banque, son niveau de santé, son niveau de faim et de fatigue.
- **Vie Personnelle** : Composée de la maison, du travail, les courses ainsi que la liste des véhicules.
- **Entreprise** : Composée d'une liste de batiments, des employés qui occupe des postes, du capital et vehicules de services.
- **Parking** : Soit un un parking d'une maison ou bien d'un immeuble, chacun a une capacité et un nombre de véhicules dedans.
- **Propriaitaire du véhicule** : C'est soit une entreprise, soit une citoyen

## 2.4. Module Batiment

On choisi d'avoir plusieurs types de batiments, la majorité ont une forme, une zone Id, une entrée une capacité et une liste de citoyen, mais il y a quelques différences mineures entre les batiments.

1. **Cabane**
2. **Atelier** : a une liste d'employées.
3. **Epicerie** : a une liste de clients ainsi qu'un stock de produit.
4. **Commissariat** : a un héliport.
5. **Ecole** : a une liste d'élèves.
6. **Hopital** : a un héliport ainsi qu'une liste de patients.
7. **Cinema** : a une liste de spectateur
8. **Restaurant** : a une liste de client.
9. **Banque** : a un capital total et une liste de clients.
10. **Prefecture**
11. **Maison** : a une liste d'habitants
12. **Caserne de pompier** : a un nombre total de pompiers, un nombre de camions et un heliport.
13. **Entreprise**

On tient a noter que les fonctions d'ajouts de capacité, d'ajout ou suprresion de citoyen d'une liste...etc sont plutot standard et on recommand vivement de voir leur code sources car elles ne seront pas traités ici.

### 2.4.1. Fonctions Importantes

#### 2.4.1.1. Banque

1. **Virement Entre deux banques** : on enlève le montant s'il est disponible dans la banque qui émet le virement.
2. **Virement Entre deux citoyens** : un simple virement.
3. **Virement Bancaire** : en utilisant les fonctionnalités précédentes, on vérifie d'abord que l'emtteur est client de la banque de laquelle il envoie (idem pour le recepteur), en revanche, si la banque qui envoie c'est elle qui reçoit il n'y a pas de transactions au sein de la même banque vu qu'on retire de l'argent puis on le rajoute.

#### 2.4.1.2. Hospitalisation

On ajoute un patient seulement si :

- L'hopital n'est pas plein.
- Le citoyen n'est pas déjà hospitalisé.

Et puis si le patient guéri on peut le filtrer de la liste des patients.

#### 2.4.1.3. Vente de produit

Si le produit est disponible dans le stock.

#### 2.4.2. Invariants

Que le nombre d'habitant/employés/patients...etc ne dépasse pas la capacité et que la capacité est toujours positive. (se trouve dans le module Ville )

#### 2.4.3. Remarque

: les pre/post conditions des bâtiments se trouvent dans le module Ville, celà est du au fait qu'on a besoin d'une zone pour construire le bâtiment.

### 2.5. Module Citoyen

Un citoyen dans city est un des trois types suivants :

1. **Immigrant** : Une **personne** (2.3.) ayant une **vie** (2.3.)
2. **Habitant** : Une **personne** ayant une **vie** et une **Vie Personnelle** (2.3.)
3. **Emigrant** : C'est juste une personne2.3.

#### 2.5.1. Fonctions Importantes

- **Transformer un immigrant en habitant** : il suffit juste qu'il une vie personnelle.
- **Rajouter les années de séjour** : en effet, pour les étrangers on doit compter le nombre d'année présent sur le territoire français.
- **Changer l'état d'une diplome** En cours -> Obtenu
- **manger** : un produit consommable, diminiue le niveau de faim a zero.
- **cuisiner** : un produit consomable et le produit devient cuit, mais rajoute de 5 le niveau de fatigue
- **mourrir** : le joueur disparait du jeu
- **Dormir** : un certain nombe d'heures, rend le niveau de fatigue a 0 et rend le niveau de santé a 100%.
- **Se reveiller** : après avoir dormi un certain nombre d'heure, le citoyen se reveil.
- **Se deplacer** : se deplacer vers des coordonnées précis
- **Revenir A la maison** : si le niveau de fatigue est supérieure a 90 et qu'il est entrain de travailler, on lui rajoute la somme journalière de son travail, on le fait déplacer à la maison et celà diminue son niveau de fatigue de 5 .
- **Guerrir d'une maladie** : si la maladie n'est pas chronique ou mortelle, il peut guérir.
- **Acheter un véhicule**

#### 2.5.2. Propriétés

- **Nationalité** : Propriétés sur la nationalité, si un étudiant est un étranger, il doit avoir un diplome avec un an d'étude au minimum, ou si le diplome est en cours il faut prévoir au moins un an d'études, et les années de résidences pour les salariés étranger doivent être supérieure a zero.
- **Interval de valeur** : que nos données sont entre 0 et 100 (faim, fatigue, et santé).
- **Argent positif** : vérifier que l'argent en banque positif ou nul.

#### 2.5.3. Invariant

Que l'argent est positif, que son niveau de santé, faim et fatigue sont entre 0 et 100...etc

### 2.6. Module Occupation

Un citoyen a un instant **t** peut être occupé par une des occupations suivantes :

1. **Travailler** : avec un salaire journalier.
2. **Dormir** : un certain nombre d'heure.
3. **Faire des courses** : et le prix des courses.
4. **Se deplacer** : vers des coordonnées.
5. **Manger**
6. **Cuisiner**

### 2.7. Module Maladies

Une maladie est définie par son nom, ses symptomes, son traitement, par ailleurs, il existe trois types de maladie :

1. **Infectieuse** : le malade peut guérir.
2. **Chronique** : Le malade ne pas pas guérir.
3. **Mortelle** : le citoyen attend sa mort.

### 2.8. Module Vehicules

Un vehicule est défini par on Id, son immatriculation, son propriétaire s'il en a, ses passagers, son prix et sa capacité ainsi que son type. En effet il existe quelques types de vehicules

1. **Voiture**
2. **Moto**
3. **Camion**
4. **Bus**
5. **Helicoptère** : on note qu'un helicoptère peut atterir et décoller, et ne peut pas rouler sur des routes commes les autres véhicules.

On tient à noter que les véhicules appartenant aux entreprises ne peuvent être revendu qu'aux entreprises et pas les citoyens.

#### 2.8.1. Fonctions Générales

- **Achat d'un véhicule** : on distingue les deux cas cités en dessus.
- **Ajouter des passagers** : en vérifiant que le nombre de passagers de dépasse pas la capacité du véhicule.
- **Faire rouler un véhicule** : on vérifiant que le véhicule a des passagers, que le type de véhicule peut rouler.

## 2.9. Module Produit

Un produit est défini par son Id, son nom, son prix, le type de produit (Alimentaire, Electronique...etc) ainsi que le type de production (Local, Importation (avec le pays d'importation)). On note aussi que les produits alimentaire ont un type (viande..etc) et leur état (cuit, périmé ou Frais). On a aussi la notion de **stock**, qui pour chaque produit a la quantité disponible.

### 2.9.1. Fonctionalités Générales

- **Achat d'un produit**
- **Gérer le stock de produit**
- **Enlever un produit du stock** : c'est dans le cas d'achat de produit.
  1. **Pre** : Il existe au moins un produit
  2. **Post** : le produit a été enlevé et sa quantité dans le stock a été diminué de 1.
- **invariant** : que le stock du produit n'est pas négatif.

## 2.10. Module Préfecture

Le but principale de la prefecture est de naturaliser les étrangers, Voici les trois règles principales pour naturaliser un citoyen.

1. Ne doit pas avoir de crimes.
2. Avoir une situation stable (être un habitant).
3. Présence dans le territoire français :
   - **Etudiant** : 2 Ans de présence avec un diplome obtenu.
   - **Salarié** : 5 Ans de présence au territoire français.

**l'invariant** de la prefecture est qu'elle ne doit pas gérer un citoyen qui est déjà dans une autre prefecture. Pour la naturalisation :

1. **Pre-conditions** : Le citoyen est un habitant, ayant vérifier les conditions citées dessus.
2. **Post-conditions** : Le citoyen est français.

## 2.11. Module Parking

On distingue deux types de parking, celui des immeubles et ceux des maisons, chacun d'eux a une capacité, une liste de véhicules ainsi que l'ID des vehicules.

### 2.11.1. Fonctionalité Général

- **Ajout d'un vehicule** : si il existe une place disponible.
- **Enlèvement d'un vehicule** : si le véhicule est dans le parking.
- **invariant** : que il n'y a pas plus de véhicules que la capacité maximum, et que le parking maison peut avoir au maximum 6 véhicules.
- **Pre et post conditions** : Preconditions que le type du véhicule on peut le garer, et la poste conditions que le véhicule est vraiment ajouté.

## 2.12. Module Entreprise

Une entreprise est définie par son identifiant, une liste de ses bâtiments **( Minimum 1 bâtiment )**, une liste des employés avec leurs postes (au moins un qui est le **CEO**), son capital ainsi que ses véhicules de services. Par ailleurs, comme mentionné dans le partie **2.8** que les véhicules des entreprises ne peuvent être revendu qu'aux entreprises.

### 2.12.1. Fonctionnalité Général

- **Recruter ou virer un employé** : On note qu'on ne peut pas virer le CEO si il est seul dans son entreprise.
- **Augmenter et diminuer le capital** : à travers les différentes dépenses.
- **Enlever un batiment** : en s'assurant qu'il reste au moins un batiment.
- **Achat d'un véhicule**
- **Invariant & pré et post conditions**
  1. L'entreprise n'est pas en faillite.
  2. L'entreprise a au moins un employé (CEO).
  3. L'entreprise a au moins un batiment.
  4. L'entreprise n'a pas plus d'un seul CEO.

## 2.13. Module Ville

Une ville est définie par un ensemble de zone et de citoyens, et un ensemble de batiments.

### 2.13.1. Fonctionalité Générale

- **Construction** : d'un batiment dans une zone, une zone dans une ville
- **Suppression** : d'un batiment, et d'une zone.
- **Rouler un véhicule**
- **Atterir et décoller un hélicoptère**
- **invariants Ville**
  1. **Les Zones disjointes** : vérifier que toutes les zones sont disjointes.
  2. **Adjacence à une route** : vérifier que toutes les toutes les zones sont adjacentes a au moins une route.
  3. **Les routes sont connexes** : Vérifier qu'il y a toujours une route d'un point **A** vers un point **B**
  4. **Invariants des batiments** : que chaque batiments est dans une seule zone (ID).

### 2.13.2. Préconditions et Postconditions

1. **Construction d'une zone :**
   - **Precondition** : d'une construction d'une zone, on doit vérifier que la zone est valide.
   - **Postcondition** : Que l'invariant est vrai et que la taille de la ville avant ajout est inférieure de 1 à la taille de la ville après ajout.
2. **Construction d'un bâtiment :**

- **Precondition** : Zone de construction est correcte et que le batiment (Id) n'est pas présent déjà dans la ville.
- **Postcondition** : Le batiment est présent dans la zone dédié et qu'il est présent.

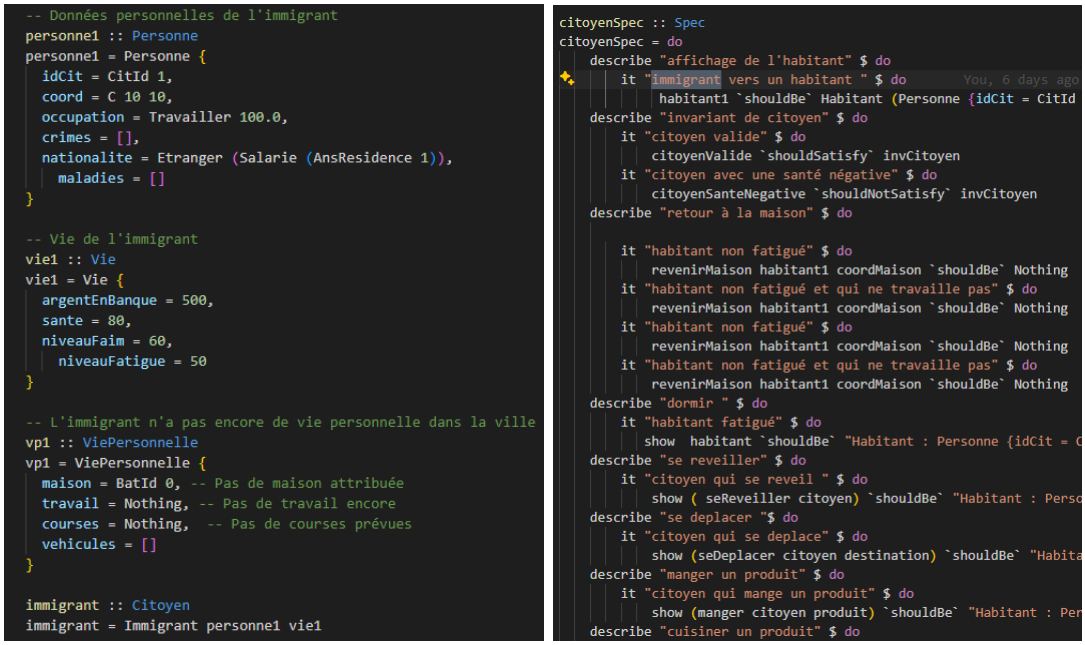
3. **Suppression d'un bâtiment :**
   - **Precondition** : Le batiment qu'on veut supprimer existe et que la zone est une zone résidentielle/commerciel/industrielle.
   - **Postcondition** : que le batiment n'est plus dans la zone.

## 3. Structure des Tests

Nous avons essayé de faire le maximum de test possible sur nos code, en effectuant plus de 200 Tests sur nos fonctions. Chaque fichier de test est situé dans le répertoire **"/test"** et chacun des test est importé dans le fichier **Spec**. On a fait en sorte de tester tous les cas d'erreurs ainsi les cas normaux pour couvrir le maximum de cas de test.


FIGURE 1.2 – Tests sur le projet

Pour Simplifier la lecture des tests, on a proposé que au lieux d'avoir des tests directement, on crée des variables locale pour rendre nos test plus lisible, compréhensible et surtout **maintenable**.



```haskell
-- Données personnelles de l'immigrant
personne1 :: Personne
personne1 = Personne {
  idCit = CitId 1,
  coord = C 10 10,
  occupation = Travailler 100.0,
  crimes = [],
  nationalite = Etranger (Salarie (AnsResidence 1)),
    maladies = []
}

-- Vie de l'immigrant
vie1 :: Vie
vie1 = Vie {
  argentEnBanque = 500,
  sante = 80,
  niveauFaim = 60,
    niveauFatigue = 50
}

-- L'immigrant n'a pas encore de vie personnelle dans la ville
vp1 :: ViePersonnelle
vp1 = ViePersonnelle {
  maison = BatId 0, -- Pas de maison attribuée
  travail = Nothing, -- Pas de travail encore
  courses = Nothing,  -- Pas de courses prévues
  vehicules = []
}

immigrant :: Citoyen
immigrant = Immigrant personne1 vie1
```

```haskell
citoyenSpec :: Spec
citoyenSpec = do
    describe "affichage de l'habitant" $ do
        it "immigrant vers un habitant " $ do        You, 6 days ago
            habitant1 `shouldBe` Habitant (Personne {idCit = CitId
    describe "invariant de citoyen" $ do
        it "citoyen valide" $ do
            citoyenValide `shouldSatisfy` invCitoyen
        it "citoyen avec une santé négative" $ do
            citoyenSanteNegative `shouldNotSatisfy` invCitoyen
    describe "retour à la maison" $ do

        it "habitant non fatigué" $ do
            revenirMaison habitant1 coordMaison `shouldBe` Nothing
        it "habitant non fatigué et qui ne travaille pas" $ do
            revenirMaison habitant1 coordMaison `shouldBe` Nothing
        it "habitant non fatigué" $ do
            revenirMaison habitant1 coordMaison `shouldBe` Nothing
        it "habitant non fatigué et qui ne travaille pas" $ do
            revenirMaison habitant1 coordMaison `shouldBe` Nothing
    describe "dormir " $ do
        it "habitant fatigué" $ do
          show  habitant `shouldBe` "Habitant : Personne {idCit = C
    describe "se reveiller" $ do
        it "citoyen qui se reveil " $ do
            show ( seReveiller citoyen) `shouldBe` "Habitant : Perso
    describe "se deplacer "$ do
        it "citoyen qui se deplace" $ do
            show (seDeplacer citoyen destination) `shouldBe` "Habita
    describe "manger un produit" $ do
        it "citoyen qui mange un produit" $ do
            show (manger citoyen produit) `shouldBe` "Habitant : Per
    describe "cuisiner un produit" $ do
```

FIGURE 1.3 – Exemple de Test sur le citoyen

- **Note** : Tous les tests on été fait manuellement, un par un, et non pas par ChatGPT ou tout autre IA.

## 4. Bilan Général

Les extensions proposées dans l'énoncé n'ont pas été implementées cependant ont été remplacées par nos propres extensions, qui visent a avoir un jeu qui est plus proche a une vie réel en france, avec plusueirus services, des entreprises, des hopitaux, des produits achetés et à consommer, gérer la vie d'un citoyen, pour résumer, un jeu de notre propre conception a été developpé pour avoir un jeu qui se rapproche du vrai jeu **Simcity.**

### 4.1. Contraintes liée a l'OS Windows

Vu que des contraintes lié à notre **OS Windows**, après une semaine de tentative on n'a pas pu installer SDL2 sous nos machines, on a essayé sous WSL, WSL2 et même sans , c'est à cause de celà qu'on n'a pas pu avoir un affichage graphique mais celà est hors de capacité mais ce n'est pas du au fait qu'on ne veut pas.

### 4.2. Difficultés rencontrés

— Description très abstraite du projet

— Manque de documentation sur Haskell et

### 4.3. Résumé de notre travail

- **Projet de base** : ✓
- **Invariants** : ✓

- **Pre et post conditions** : ✓
- **Classe de Types (Either, Maybe)** : ✓
- **Utilisation des classes algébriques (Composition..etc)** : ✓
- **Transport** : ✓
- **Commerce et Gestion de Stock** : ✓
- **Services supplémentaire (ecoles, hopitaux, pompiers...)** : ✓
- **Extensions** : ✓
- **Inv/Pre/Post des Extensions** : ✓
- **Maintenabilité du code** : ✓
- **Utilisation du design pattern Composite** : ✓
- **Utilisation du principe SOLID** : ✓
- **Tests des fonctions et de propriétés** : ✓