Trade Infinity (A3P 2021/2022) J4

Léo Roullois

25 octobre 2021

Table des matières

1 Description du projet **2**
- 1.1 Auteur . . . 2
- 1.2 Thème (phrase-thème validée) . . . 2
- 1.3 Résumé du scénario (complet). . . . 2
- 1.4 Plan (complet, avec indication de la partie "réduit" si exercice 7.3.3) . . . 3
- 1.5 Scénario détaillé (complet, avec indication de la partie "réduit" si exercice 7.3.3) . . . 4
- 1.6 Détail des lieux, items, personnages . . . 4
- 1.7 Situations gagnantes et perdantes . . . 4
- 1.8 Eventuellement énigmes, mini-jeux, combats, etc. . . . 4
- 1.9 Commentaires (ce qui manque, reste à faire, ...) . . . 4

2 Réponses aux exercices (à partir de l'exercicie 7.5 inclus). **5**

3 Mode d'emploi (si nécessaire, instructions d'installation ou pour démarrer le jeu) **10**

4 Déclaration obligatoire anti-plagiat. **11**

Chapitre 1

Description du projet

1.1 Auteur

Léo Roullois

1.2 Thème (phrase-thème validée)

Au World Trade Center, Laylow un jeune trader doit multiplier son capitale par 1000 grâce aux cryptomonnaies pour gagner le jeu.

1.3 Résumé du scénario (complet).

Un trader a un capital de départ : 10k$.
Pour gagner le jeu il doit multiplier son capital par 100 à l'aide des cryptomonnaies.
Pour cela, Laylow sera confronté à différentes salles :

- BTC (Bitcoin)
- ETH (Ethereum)
- DEFI (BSC, blockchains ethereum...)
- NFTs (Non Fungible Token)
- ShitCoins (Dogecoin, Shiba INU, SafeMoon, ...
- Trading (analyse fondamentale, technique, ...)
- Minage (PoW, PoS, ...)
- Hack (Scam, manipulation du marché, attaque à 51%) : une fois la salle débloquée, potentiel gain très grand mais 10
- ICOs

Dans chaque salle il y aura des énigmes et des quêtes concernant les thèmes en question. Pour un total de 9 salles, le joueur devra en visiter au moins 7 avant de finir le jeu.

Le trader commence dans la salle **BTC** avec 10k$. Une énigme lui sera posée (à définir). S'il réponds correctement, il peut choisir de placer un certain pourcentage de son capital dans le BTC : de 0% à 30%(investissement qui sera nécéssairement rentable par la suite, mais le joueur ne le sais pas encore).

Dans la salle **ETH**, il y aura également des énigmes (à propos des smarts contracts, initiation à le finance décentralisée...), qui permettront en fonction des réponses de débloquer un ou plusieurs investissements en ETH. Dans la suite du jeu, le joueur devra repassé dans la salle ETH pour trouver la clé qui ouvrira la salle DEFI (en complétant tous les challenge lié à la finance décentralisée).

Dans la salle **DEFI**, il y aura 2 niveaux :

— Etage BSC (Binance Smart Chain) : Le joueur peut se créer des revenus passifs en participant à des pools de liquidité ou en farmant des tokens sur la Binance Smart Chain (gains moyens, frais peu élevés, investissement plutôt sures).
— Etage ETH (Ethereum) : De même, sauf que la blockchain Ethereum propose des gains plus élevés mais également des frais plus élevés (donc rentable sur le long terme...).

Dans la salle **Minage**, le joueur aura la possibilité de miner des cryptomonnaies et donc d'être rémunérer s'il valide des blocs (probabilité + ou - élevée en fonction de la puissance de calcul qu'il possède).

Dans la salle **NFT** le joueur peut créer des NFT et tenter de les vendre sur les marchés. Plus le NFT sera considéré comme jolie, plus le joueur pourra le vendre cher.

Dans la salle **ICO**, le joueur peut participer à des ICO, la majorité d'entres elles vont perdre de la valeur lors du lancement, mais au contraire, si elle prends de la valeur, le gain est potentiellement énorme.

Dans la salle **Trading**, le joueur va pouvoir suivre des analyses fondamentales et répondre à des quiz qui vont lui permettre d'apprendre les bases en trading et ensuite d'investir dans certains coins qu'il trouve intéressant.

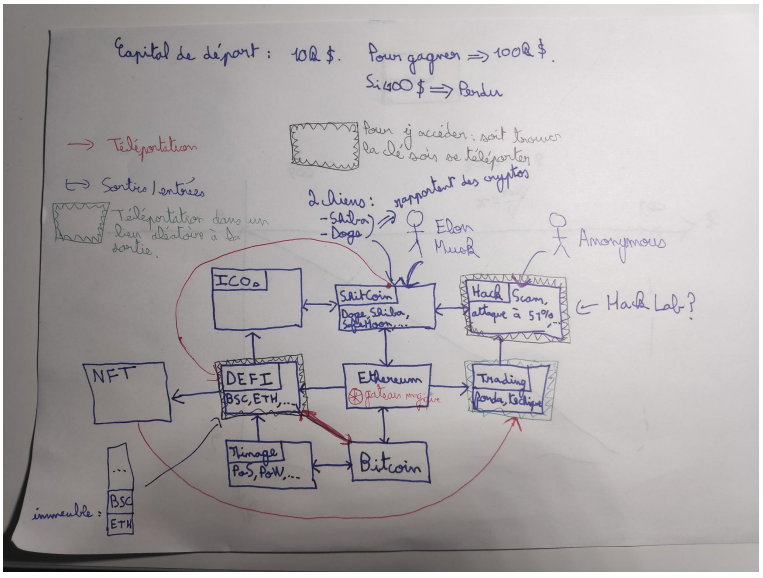
Dans la salle **Hack** le joueur aura la possibilité d'organiser des attaques sur des blockchains : si l'attaque est validée, le joueur gagne instantanément la partie, même s'il n'a pas visiter le nombre de salles requis.

Dans la salle**Shitcoin** le joueur peut investir dans des shitcoins qui ont peut de chances de prendre de la valeur, mais si tel est le cas, alors le shitcoin en question prendrais minimum + 1000% de valeur en l'espace de quelques jours.

Enfin, le joueur pourra adopter des chiens, acheter des ordinateurs ou cartes graphiques (pour augmenter sa puissance de calcul) afin d'effectuer des attaques dans la salle **Hack** ou bien pour miner des cryptomonnaies dans la salle **Minage**. Le joueur pourra parler à des célébrités tel qu'Elon Musk afin qu'il l'aide ses investissements sur le shitcoin DogeCoin. Il pourra acheter un wallet Ledger afin de sécurisé toute ses transactions, dans le cas contraire il y a un risque que ses transactions échoues. Il pourra également manger un gateau magique qui va accélérer le temps et faire prendre de la valeur a certaines cryptomonnaies (définie aléatoirement).

1.4 Plan (complet, avec indication de la partie "réduit" si exercice 7.3.3)

(Plan encore un peu brouillon, version propre et épurée à venir. J'espère que vous allez quand même réussir à me comprendre.)



1.5 Scénario détaillé (complet, avec indication de la partie "réduit" si exercice 7.3.3)

A venir

1.6 Détail des lieux, items, personnages

A venir

1.7 Situations gagnantes et perdantes

A venir

1.8 Eventuellement énigmes, mini-jeux, combats, etc.

A venir

1.9 Commentaires (ce qui manque, reste à faire, ...)

Beaucoup de chose ;)

Chapitre 2

Réponses aux exercices (à partir de l'exercicie 7.5 inclus).

Exercice 7.5 (printLocationInfo).

```
public void printLocationInfo() {
    System.out.println("You are " + this.aCurrentRoom.getDescription());
    String vOutput = "";
    if (!(this.aCurrentRoom.aNorthExit == null)) {
        vOutput = vOutput + "north ";
    }
    if (!(this.aCurrentRoom.aEastExit == null)) {
        vOutput = vOutput + "east ";
    }
    if (!(this.aCurrentRoom.aSouthExit == null)) {
        vOutput = vOutput + "south ";
    }
    if (!(this.aCurrentRoom.aWestExit == null)) {
        vOutput = vOutput + "west";
    }
    System.out.println("Exits: " + vOutput);
}
```

Exercice 7.6 (getExit).

1) Car un attribut non initialisé par le constructeur prends la valeur null.

2) et 3) J'ai créer cette fonction dans la classe Room :

```
public Room getExit(final String direction) {
    if (direction.equals("north")) {
        return this.aNorthExit;
    }
    if (direction.equals("east")) {
        return this.aEastExit;
    }
    if (direction.equals("south")) {
```

```
            return this.aSouthExit;
        }
        if(direction.equals("west")) {
            return this.aWestExit;
        }
        return this;
    }
```

Puis j'ai modifié ceci dans la procédure goRoom de Game :

```
    String vDirection = pCommand.getSecondWord();
        Room vNextRoom=aCurrentRoom.getExit(vDirection);
        if (this.aCurrentRoom==vNextRoom) {
            System.out.println("Unknown direction !");
        }else if (vNextRoom == null) {
            System.out.println("There is no door !");
            return;
        } else {
            this.aCurrentRoom = vNextRoom;
            this.printLocationInfo();
        }
```

Exercice 7.7 (getExitString).

J'ai modifié la procédure suivante dans Game :

```
    public void printLocationInfo() {
        System.out.println("You are " + this.aCurrentRoom.getDescription());
        System.out.println(this.aCurrentRoom.getExitString());
    }
```

Et créer cette fonction dans Room :

```
    public String getExitString() {
        String vOutput = "";
        if (this.aNorthExit != null) {
            vOutput = vOutput + "north ";
        }
        if (this.aEastExit != null) {
            vOutput = vOutput + "east ";
        }
        if (this.aSouthExit != null) {
            vOutput = vOutput + "south ";
        }
        if (this.aWestExit != null) {
            vOutput = vOutput + "west";
        }
        return ("Exits: " + vOutput);
    }
```

Exercice 7.8 (HashMap, setExit)

Voici le nouvel attribut qui remplace les 4 attributs de salle précédents :

```
private HashMap<String,Room> aExits;
```

Et les modifications dans la classe room :

```
public Room(final String pDescription) {
    this.aDescription = pDescription;
    this.aExits = new HashMap<String,Room>();
}
```

et j'ai supprimé la procédure setExits pour la remplacée par :

```
public void setExit(String pDirection, Room pNeighbor) {
    if (pDirection != null) {
        this.aExits.put(pDirection, pNeighbor);
    }
}
```

puis la fonction get Exit :

```
public Room getExit(final String direction) {
    return this.aExits.get(direction);
}
```

et enfin getExitString :

```
public String getExitString() {
    String vOutput = "Exits : ";
    Set<String> allKeys = this.aExits.keySet();
    for (String vKey : allKeys) {
        vOutput+=vKey+" ";
    }
    return vOutput;
}
```

Des modifications ont également été faites dans Game pour coller aux nouvelles fonctions.

Exercice 7.9 (keySet).

Modifications déjà effectuées lors de l'exercice précédent.

Exercice 7.10 (getExitString CCM ?).

Afin de lister toutes les sorties possibles d'une salle à partir de la HashMap de cette même salle, il faut commencé par créer une collection de String qui contiendra toutes les clés de cette HashMap. Une fois cette collection crée, nous allons itérer sur cette collection grâce à la syntaxe for vue précédemment. A chaque itération, nous ajouterons la valeur de la clé à une variable, qu'on retournera à la sortie de cette boucle for.

Exercice 7.11 (getLongDescription)

Dans la classe Game :

```
public void printLocationInfo() {
    System.out.println(this.aCurrentRoom.getLongDescription());
}
```

Et dans la classe Room :

```
public String getLongDescription() {
    return "You are "+this.aDescription+".\n"+this.getExitString();
}
```

Exercice 7.14

Dans la classe Game :

```
private void look(final Command pCommand) {
    if (pCommand.hasSecondWord()) {
        System.out.println("I don't know how to look at something in particular yet.");
    } else {
        System.out.println(this.aCurrentRoom.getLongDescription());
    }
}
private boolean processCommand(final Command pCommand) {
    if (pCommand.isUnknown()) {
        System.out.println("I don't know what you mean...");
        return false;
    } else {
        if (pCommand.getCommandWord().equals("go")) {
            this.goRoom(pCommand);
        } else if (pCommand.getCommandWord().equals("help")) {
            this.printHelp();
        } else if (pCommand.getCommandWord().equals("look")) {
            this.look(pCommand);
        }

        if (pCommand.getCommandWord().equals("quit")) {
            return quit(pCommand);
        } else {
            return false;
        }
    }
}
```

Et dans CommandWords :

```
private static final String[] VALID_COMMANDS = {
    "go","quit","help","look"
};
```

Ainsi que des modifications mineures pour accéder au tableau VALID_COMMANDS.

Exercice 7.15.

J'ai ajouté la commande « buy » qui permet d'acheter des cryptomonnaies (bitcoin, ethereum...).

Exercice 7.16

Dans la classe CommandWords :

```java
/**
 * Affiche toute les commandes du jeu
 */
public void showAll() {
    for (String pCommand : VALID_COMMANDS) {
        System.out.print(pCommand+" ");
    }
    System.out.println();
}
```

Dans la classe Parser :

```java
public void showCommands() {
    aValidCommands.showAll();
}
```

Et dans la classe Game :

```java
/**
 * Affiche un message d'aide
 */
 private void printHelp() {
    System.out.println("You are lost. You are alone.");
    System.out.println("Your command words are:");
    aParser.showCommands();
    System.out.println("go quit help");
}
```

Chapitre 3

Mode d'emploi (si nécessaire, instructions d'installation ou pour démarrer le jeu)

A venir

Chapitre 4

Déclaration obligatoire anti-plagiat.

Je n'ai recopié aucune ligne de code. Par contre, je me suis inspiré des idées et réponses proposées dans certains exercices du projet.