Titre Du Jeu : 2184

(A3P 2018/2019 G6b) adresse du site web: https://perso.esiee.fr/~delattel

1.A) Auteur

DELATTE Laurent

1.B) Thème

Vous devez trouvez une combinaison de chiffres correspondant aux coordonnées géographique de la base Rebelle luttant contre le “Partis”. Pour cela, vous devez suivre les instructions fournis dans un petit livre noir sans vous faire arréter.

1.C) Résumé du scenario

Nous sommes en 2184 en Angleterre et plus précisement à Londre. Le pays est contrôlé par une dictature : “Le Partis” avec à sa tête Big Brother. Dans ce monde tout est sous haute surveillance, tout le monde est contrôlé, tout vos appareils sont sur écoute et la moindre recherche internet, le moindre comportement suspect peut vous envoyer en prison. C’est dans un batiment nommé “Le Ministère De La Vérité” que travaille Winston Smith. A chaques fois que le partis lui demande il doit “réécrire l’histoire” pour quelle convienne à leurs idées. Winston est un homme d’une trentaine d’année qui va un jour, lors des “2 minutes de la haines” (fête national qui vise à insulter et à injurier “la résistance”, qui est un groupe de personnes, basés on ne sait où et qui tente de faire renverser le partis en place en recrutant de nouveaux membres, effectuant des attentats pour déstabiliser le partis et changer les mentalités des gens afin qu’ils se révoltent) retrouver dans sa poche un petit calpin noir avec pour titre “A lire en lieu sûr”, ce calpin le guidera durant sa quête pour trouver les coordonnées géographique de la base rebelle tout en évitant de se faire arréter par le partis.

1.D) Plan

La pièce “Cellar” est une pièce caché et au sous sol de la pièce “Cafe” .

1.E) Scénario complet

(le scénario n’est pas encore terminé mais bien avancé)

Le jeu débutera dans la pièce “Home” donc chez Watson et automatiquement on affichera à l’ecran la première page du livre où il y aura marqué de se rendre dans la pièce “Cinema” pour pouvoir continuer à lire le livre. Une fois au cinéma un message vous affichera que vous êtes dans le noir et vous ne pouvez donc pas lire votre livre. Un second message vous dira d“aller acheter une lampe torche dans la”Third Street“. UNe fois rendu dans la pièce”Third street vous devrez aller voir le marchand qui est dans la rue (préciser dans un message affiché à lorsque

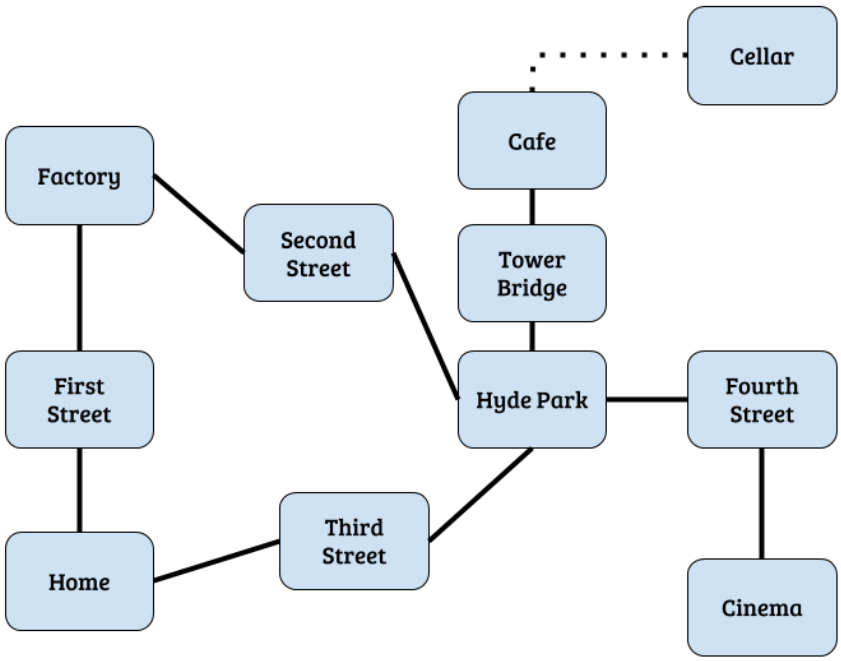

Figure 1: Plan2184

l'on rentre dans cette pièce). La lampe coutera 100 crédits mais comme vous aves déjà sur vous 500 crédits, vous pouvez l'acheter. Après être retourner dans le Cinema, vous pourrez lire le livre. Ce livre sera divisé en plusieurs chapitres: Qui sommes nous?, Ce que nous faisons? Pourquoi nous le faisons? et enfin Rejoin nous!. Les 3 premiers chapitres expliqueront rapidement le rôle de la "Résistance" mais le dernier chapitre donnera des indication assez précise pour trouver les coordonnées géographique. Le dernier chapitre explique qu'il faut se rendre dans la pièce "Cafe" pour plus d'informations. Mais vous ne pourrez pas accéder à cette pièce car le pont "Tower Bridge est relever et par conséquent vous ne pourrez pas passer. Heureusement il y aura dans la pièce un pnj qui vous dira de trouver la clé permettant d'activer le pont.

Après avoir trouvé la clé vous pourrez accéder à la pièce "Cafe" ou il y aura ensuite une pièce caché qui sera au sous sol, il faudra donc descendre pour s'y rendre.

Je n'ai pas la suite qu'il faut que je finisse d'inventer . Cependant je sais que durant sa recherche de clé vous trouverez en tout: 1 indice(la longitude, il ne restera plus qu'a trouver la lattitude) et 2 clé: 1 permettant d'ouvrir le pont et une autre(qui apparaitra après l'épreuve du pont) permettant d'accéder à la pièce caché "Cellar" qui est liée avec la pièce "Café" (cf plan). Dans "Cellar" il y aura une machine où vous rentrerez les coordonnées géographiques et cela vous retournera le nom de cette base rebelle: ESIEE Paris.

De plus dans "Home" il y aura un coffre où il pourra poser ses objets en trop (car vous aurez un nombre limité d'objets à porter), il y aura aussi un lit où vous pourrez dormir car vous aurez une barre de sommeil et enfin vous aurez aussi une barre de vie qui quand vous n'aurez pas mangé depuis assez longtemps elle diminuera (Pour manger vous pourrez acheter de la nourriture au marchands dans "Third Street".

1.F) Détail des lieux, items, personnages

Lieux	Items
Home	Lit + Coffre(Pour ranger ses objets en trop)
First Street	
Factory	Argent
Second Street	Indice longitude
Third Street	
Hyde Park	Première clé caché dans un bosquet
Fourth Street	Deuxième clé caché
Cinema	
Tower Bridge	
Cafe	porte caché menant à "Cellar"
Cellar	machine permettant de rentrer/vérifier les coordonnées géographiques

1.G) Situations gagnantes et perdantes

A chaques fois que l'on ouvre le livre en dehors de la pièce cinéma et sans avoir activer la lampe torche on se fait arréter, on va en prison, le jeu s'arrète et il faut tout recommencer. Si vous réussissez ces étapes vous réussissez cette partie du jeu. le Jeu s'arrète si vous n'avez plus de vie et/ou trop sommeil, si à la fin du jeu vous rentrez de mauvaise coordonnées géographiques. Vous gagnez la partie du jeu si vous arrivez à mettre les 2 clés au bon endroits. Vous gagnez la paries si vous rentrez les bonnes coordonnées géographiques à la fin du jeu.

1.H) Eventuellement énigmes, mini-jeux, combats, etc.

Les coordonnées géographiques seront donné à travers 2 énigmes qu'il faudra résoudre. Il y aura une sorte de mini jeux dans la pièce Factory qui permettra de gagner de l'argent pour survivre. (Pas encore décider du mini jeux exactement)

1.I) Commentaires

Ce qu'il manque/Reste à faire: - Finir le scénario entièrement et créer le mini jeux (déjà sur papier) - Finir tout les exercices en retard (finis durant ces vacances)

II. Réponses aux exercices

- **Exercice 7.5** : Dans cet exercice on nous demande de corriger nos duplication de code. Ici nous avons 2 méthodes: `PrintWelcome` et `GoRoom` qui affichent les informations propres à la nouvelle situation courante. Pour remédier à ce problème nous allons créer une nouvelle méthode `PrintLocationInfo` qui sera appelé par les 2 méthodes précédentes quand elles ont besoin d'afficher ces informations.
- **Exercice 7.6** : Dans cet exercice j'ai résolu les problèmes de couplage(les sortis sont énumérées beaucoup trop de fois dans notre code). Pour remédier à ce problème j'ai créé créer une Hash Map (dans les exercices suivants(**Pas encore faits**)) mais avant toutes choses j' ai d'abord découpler les classes `Room` et `Game` en appliquant le principe d'encapsulation. J'ai donc créer une méthode `getExit` qui peut être accessible depuis une autre classe par un accesseur, c'est ce que j'ai fais pour la classe `Game`.
- **Exercice 7.7** : Dans cet exercice j'ai évité le couplage entre la classe `Game` et `Room` comme réalisé dans l'exercice précédent. Il est maintenant possible de modifier la manière de représenter les attributs de `Room` sans modifier `Game`.
- **Exercice 7.8** : Dans cet exercice j'ai changer la gestion des salles voisines avec quelque chose de plus dynamique (qui me permettait d'ajouter autant de direction que je le voulais). Pour cela j'ai ajouter à mon programme une Hashmap.

- **Exercice 7.9** : ***A rédiger***
- **Exercice 7.10** : ***A rédiger***
- **Exercice 7.11** : Dans cet exercice j'ai une nouvelle fois réduit le couplage en mettant la méthode permettant la description des rooms et leurs sortit dans la classe `Room`. Il ne me reste plus que dans la classe `Game` la procédure `printLocationInfo` qui va appeler la méthode `getLongDescription` pour donner la description de la room et de ses sortit.
- **Exercice 7.12 (optionnel)** : ***A faire, très rapide, juste faire une liste des objets qui s'active au démarrage du jeux***
- **Exercice 7.13 (optionnel)** : ***A rédiger***
- **Exercice 7.14** : Encore une fois dans cet exercice je vais une nouvelle fois réduire le couplage en m'occupant du couplage implicite et plus particulièrement de la localisation des modifications, pour cela j'ai ajouté dans le tableau des méthodes connus, la commande `look`. Puis nous ajouterons cette commande dans la méthode `process command` pour quelle soit reconnu par le compilateur. La commande `look` va nous permettre de "voir" ce qui se trouve autour de nous donc pour le moment les sortit de la pièce dans laquelle on se situe et sa description, exactement comme la méthode `printLocationInfo` .

Exercice bonus fait mais beug cf photo prise méthode `affichStringDeuMot` demander soit à Mr.Bureau

- **Exercice 7.15** : Dans cet exercice j'ai ajouté à la liste des commandes valides la commande `eat` qui une fois exécuté nous affichera le message suivant : > You have eaten now and you are not hungry any more
- **Exercice 7.16** : Dans cet exercice j'ai résolu un problème de couplage pour que lorsqu'on ajoute une commande valide cela l'affiche dans la méthode `help`. Pour cela j'ai créé une méthode `showAll` qui sera ensuite appellé par la méthode que j'ai créé dans la classe `Parser` qui permet de sélectionner seulement les commandes valides qui seront ensuite affiché dans la méthode `printHelp` qui nous affichera.
- **Exercice 7.17** : Dorénavant lorsque nous ajoutons une nouvelle commandes je vais devoir aussi toucher à la classe `Game` car on va devoir ajouter cette nouvelle commande à la méthode `processCommand`(qui liste les commandes que nous pouvons écrire dans le jeu) et son action. Il faudra de même créé une méthode dans la classe `Game` si nous voulons que cette commande retourne quelque chose. Enfin il faudra l'ajouter dans la méthode `sValidCommand` qui se trouve dans la classe `CommandWords` ce qui nous permet de ne pas rajouter la commande en question dans la méthode `printHelp`.
- **Exercice 7.18** : Dans cet exercice nous avons remplacer la méthode `showAll` par `getCommandList` qui lui est une String ce qui nous permet

de créer la liste directement dans la classe `CommandWords`. Cela est plus pratique et ça nous évite de créer une multitude de méthode.

- **Exercice 7.18.1** : J'ai bien comparé mon code avec le zuul better et tout fonctionne bien, il n'y a pas de Warnning ni d'erreurs dans l'exection du jeux.
- **Exercice 7.18.2 (optionnel)** : ***A faire***
- **Exercice 7.18.3** : Les images ont bien été trouver et recadré à la même taille (cf.les liens dans le chapitre IV).
- **Exercice 7.18.4** : Le titre du jeu est *"2184"* et il a bien été ajouté à mon code et à mon site internet.
- **Exercice 7.18.5** : Dans cet exercice j'ai créé une nouvelle Hashmap qui me permet d'associer à chaques `Room` une image, nous verrons dans l'exercice suivant comment implémenter dans le programme des images.
- **Exercice 7.18.6** : J'ai ici complété mon programme en le comparant avec le projet `zuul-with-images...`. Grâçe à cet ajout mon jeu possède maintenant des images.
- **Exercice 7.18.7 (optionnel)** :
- **Exercice 7.18.8** : Dans cet exercice j'ai d'abord créé 4 boutons correspondant aux directions nord, sud, est, ouest (ici un seul représenté):

```
private JButton    aButtonNorth;
```

je les aient ensuite initialisés:

```
this.aButtonNorth = new JButton ("north");
```

j'ai ensuite créé un panel pour mes directions

```
JPanel vPanelDirection = new JPanel();
```

Puis j'ai initialisé le panel non utilisé à l'ouest du panel principale avec mon nouveau panel de direction :

initialisation du panel à l'ouest du panel principale

```
vPanel.add(vPanelDirection, BorderLayout.WEST);
```

et dans ce nouveau panel j'ai attrbué mes directionspar rapport à mes bouttons :

```
vPanelDirection.setLayout(new BorderLayout());
vPanelDirection.add( aButtonNorth, BorderLayout.NORTH);
```

puis pour chaques commandes on lui associe la méthode actionListener qui permet de savoir si oui ou non on à appuyé sur le bouton en question :

```
this.aButtonNorth.addActionListener ( this );
```

Enfin dans la méthode `actionPerformed`j'ai indiqué comment chaques commandes doivent être interprétés par le compilateur:

```
Object vSource = pE.getSource();

if ( vSource == aButtonNorth ){
    aEngine.interpretCommand("go north");
}
```

- **Exercice 7.19.2** : Toutes les images ont bien été mises dans un fichier image à la racine du projet.
- **Exercice 7.20** : Dans cet exercice j'ai créé une nouvelle classe `Item` où je déclare toutes les composante à mon objet donc un attribut: `aDescription`, `aPoids`,`aPrix` et `aNom` J'ai ensuite créé des accesseurs pour chaques attributs.
- **Exercice 7.21** : Toutes les informations relatives à un objet doivent être produite par une méthode qui renverras ces informations. La classe qui doit produire la chaine décrivant l'objet est la classe `Item`. La classe qui devrait l'afficher est la classe `Room` car c'est elle qui affiche toute les informations en lien avec une salles, or une salle va contenir tout nos objets. Pour éviter des problèmes de couplages c'est bien la classe `Room` qui affichera cette méthode. J'ai donc modifier mon code et j'ai rajouté une méthode dans ma classe `Item`:

```
public String getItemInformations()
{
    return this.aDescription + " qui a un poids de " + this.aPoids;
}//accesseur pour obtenir les informations relative
```

- **Exercice 7.22** : Dans cet exercice j'ai crée une Hashmap dans la classe `Room` pour associer chaques objets que j'ai initialisé dans la méthode de `creatRooms` de la classe `GameEngine` qui était autrefois la classe `Game`. Une fois terminé j'ai ajouté dans ma classe `Room` une nouvelle méthode qui va renvoyé une String avec le nom de tout les objets disponibles dans la pièces dans laquelle on se situe. J'appelle cette méthode dans `getLongDescription` pour que quand je tape la commande `look` elle renvoie tout les objets disponibles dans la pièce.
- **Exercice 7.22.2** : Un item est bien placé dans ma pièce principale et un pièce contient bien plusieurs items
- **Exercice 7.23** : Dans cet exercice j'ai créé une nouvelle méthode `back` qui permet de revenir dans la salle précisement visité. De plus si l'on tape un second mot après `back` cela nous renvoi un message d'erreurs.
- **Exercice 7.26** : J'ai bien ajouté une stack que j'ai nommé `aRoomStack`, tout fonctionne parfaitement.
- **Exercice 7.26.0** :

  - *Stack* : permet d'ampiler des objet et donc de les garder en mémoire
  - *push()* : méthode du JDK qui nous permet de mettre un objet au dessu de la stack (de la pile)
  - *pop()* : méthode du JDK qui supprime l'objet en haut de la stack et renvoie cet objet en tant que valeur de cette fonction.
  - *empty()* : méthode du JDK qui permet de savoir si la stack est vide ou non
  - *peek()* : méthode du JDK va renvoyer l'objet en haut de la stack sans le supprimé

- **Exercice 7.26.1** : Les 2 JavaDoc on bien été mise à jour.

- **Exercice 7.28.1** : Dans cette exercie j'ai créé une nouvelle commande `test` qui permet d'executer du code présent dans un fichier text présent dans le répertoire du jeu. Pour cela j'ai créé une nouvelle méthode `test` dans la classe `GameEngine`. Dans cet méthode pour lire un fichier on fait:

```
try {
    Scanner vSc = new Scanner ( new File( pNomFichier ) );
    while (vSc.hasNextLine()){
        this.interpretCommand(vSc.nextLine());
    } //while qui test si il y a une ligne après
}//essai du code

catch ( final Exception pE){
    this.aGui.println("file error");
}
```

puis on ajoute cette commande aux commandes valides. J'ai ensuite vérifié cette commande en créant un fichier texte puis je l'ai executé dans mon jeu.

- **Exercice 7.29** : Dans cet exercice j'ai créé une classe `Player`, avec à l'intèrieur plusieurs méthodes permettant de découpler celle-ci avec la classe `GameEngine`. Désormais, toutes les informations concernant le joueur sont contenus dans cette classe. Ainsi, on peut avoir la possibilité d'avoir plusieurs joueurs avec leurs propres progression, poids actuellement transport, lieux courant.

III. Mode d'emploi

- Pour lancer le jeu: Ouvrir BlueJ, faites clique droit sur la classe `Game`, sélectionner `newGame`, puis refaite un clique droit sur `newGame` cette fois ci et sélectionner `void play`.

IV. Déclaration obligatoire anti-plagiat

Pour les images(liens):

- https://thespaces.com/delphi-lux-berlin-cinema/

- https://www.imagenesmy.com/imagenes/black-white-park-13.html
- http://rosiemoetuesday.co.uk/category/blog/
- https://mapio.net/pic/p-45943449/
- https://www.pinterest.fr/pin/434175220310481536/