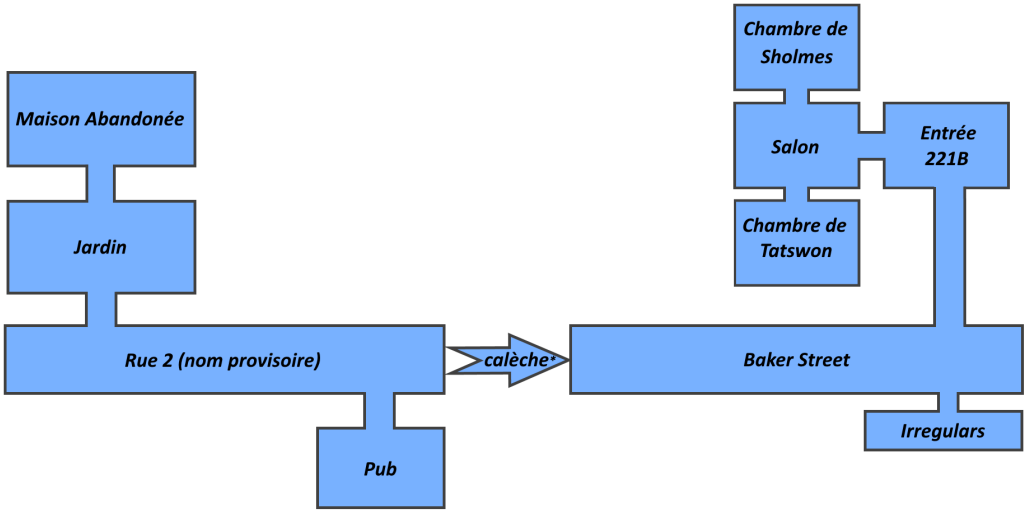
I. Herlock Sholmes : The Crimson Alibi (Titre provisoire)

I.A) Alexian Plancke
I.B) Résoudre les affaires rencontrées et découvrir l'identité du baron du crime
I.C) Incarner le célèbre détective **Herlock Sholmes**, et son acolyte le **Docteur Tatswon,** afin de résoudre les crimes organisés par un mystérieux baron du crime.



** Il faudra prendre une calèche pour changer de rue*

I.D)
I.E) <u>Partie 1 :</u>

<u>A Study in S:</u> Dans une maison abandonnée à Londres, un homme est retrouvé mort. A-t-il été assassiné? Son corps ne présente aucune blessure. Sur le mur, un mystérieux mot est découvert… Afin d'aider l'**Inspecteur Larstède**, vous explorez les alentours et interrogez les 3 individus suspectés par celui-çi.
Sur les lieux du crime, vous trouvez une bague, et identifiez les différentes traces de pas présentes dans le jardin. Vous interrogez les 3 suspects, puis rentrez chez vous, sous l'œil étonné de votre ami le **Docteur Tatswon**.

I.F) <u>Lieux</u> :
- **Le jardin :** traces de pas plus ou moins profondes, de tailles différentes, correspondant au meurtrier et aux policiers. Aucun item pour l'instant.
- Maison abandonnée: Lieu du crime, corps de la victime gisant par terre, un mystérieux message apparaît en utilisant une fiole de luminol. Une bague est trouvée près du corps
- Rue 1 *(Pas de nom pour l'instant)* : Rue de la scène du crime, au bout de celle-çi ce trouve le "Pub" du coin. Aucun items pour l'instant.
- Pub *(Pas de nom pour l'instant)* : Pub assez connu dans le coin, la victime y passait beaucoup de temps, dont le soir où il a été assassiné. Aucun items pour l'instant.
- Baker Street : Rue de votre appartement, au bout de celle-çi, le repère des "Irregulars" (nom donné aux enfants de la rue)
- "Base" des Irregulars : Repère des enfants de votre rue, nommés les "**Irregulars"**, ils pourront vous aider, si vous avez besoin d'informations, ou d'aide pour les énigmes.
- Hall d'entrée (221B) : Hall d'entrée de votre appartement, vous pourrez interagir avec **Mme Dushons**.

- Salon (221B) : Salon de votre appartement, vous pourrez fumer votre pipe.
- **Chambre d' Herlock** : Chambre du détective, très désordonnée, sur le bureau se trouve de nombreux instruments de chimie, ainsi que de nombreux produits expérimentaux inventés par celui-çi, comme la fiole de luminol.
- **Chambre de Tatswon** : Chambre du médecin. Aucun items pour l'instant

Items:
- Fiole de luminol : Fiole contenant le produit qui met en surbrillance les endroits où il y a eu du sang.
- Bague : Indice permettant de deviner le véritable coupable
- Pipe : Pipe en bois, pas d'usage pour l'instant (quelques idées, comme regagner des PV par exemple)
- Canne-Épée : Arme de prédilection d' Herlock
- Revolver : Arme de prédilection de Tatswon
- Carte de Visite : Aucune utilité (Exo 7.22.2)

Personnages:
- Herlock Sholmes : Grand détective, il n'est pourtant connu que des forces de police, et pas encore du public. Il possède des capacités de déduction hors normes, et de nombreux savoirs lui permettant de voir ce que les autres ne peuvent voir.
- Dr Tatswon : Ancien médecin de guerre, il loge depuis peu chez **Sholmes**.
- Mme Deshons : Logeuse d' **Herlock Sholmes**, au 221B Baker Street.
- Inspecteur Larstède : Inspecteur de Scotland Yard (d'origine Anglaise-Française), il demande de l'aide à **Sholmes** lors d'enquêtes particulièrement coriace.
- Les Irregulars : Nom donné aux enfants du quartier. Bien que ce ne sont encore que des enfants, ils peuvent se montrer très utiles.
- Le Baron du crime (Pr Rymiator) : Comte qui prodigue des conseils et des moyens aux coupables pour permettre de se venger, en ne laissant que des "crimes parfaits", ou des "meurtre en chambre close".

I.G) En cours de réflexion
I.H) Combat final contre le baron du crime, énigmes en cours de création

II. Exercices

Ex 7.4:
Tout d'abord, j'ai ajouté les créations des pièces afin de compléter le plan nécessaire à l'exécution du jeu.
J'ai transféré les 5 classes du module v1 à la racine du projet. Ensuite, j'ai compilé chaque classe, puis vérifié que les dépendances étaient bien affichées, et enfin exécuter le programme pour contrôler qu'il fonctionne correctement.

Ex 7.5:
J'ai créé la méthode privée printLocationInfo afin d'éviter de la duplication de code:
Cette méthode est déjà appelée à deux reprises, lors du démarrage du jeu, puis à chaque déplacement. Afin d'éviter un crash lorsque la commande tapée n'est pas reconnue, un test est effectué avant de traiter la commande.

Ex 7.6:
J'ai modifié la classe Room afin qu'elle soit "loosely couple" avec la classe Game:
omme il y avait un problème d'ambiguïté par rapport au message d'erreur, une "fake room" a été créée pour retourner une salle incorrecte différente de null et afficher le bon message d'erreur

Ex 7.7:
Afin que le code de Game soit moins dépendant de l'implémentation de Room, la logique contenue dans la méthode printLocationInfo à été déportée dans la classe Room.
Pour cela, un nouvel accesseur à été créé pour obtenir le nom de chaque direction, une fonction private getExitString a été créé pour simplifier l'implémentation.

Ex 7.8:
Les Rooms nord, sud, ouest et est, qui étaient des propriétés de chaque pièce, ont été remplacées par une HashMap. La pièce a donc maintenant une liste de sorties nommées.
Le reste du code de la classe Room a été adapté pour utiliser la HashMap à la place des anciennes propriétés qu'elle remplace. Du fait de l'encapsulation réalisée précédemment, les autres classes n'ont pas été impactées.

Ex 7.9:
Il est possible de simplifier l'implémentation de la méthode getExitString en parcourant les sorties dans la liste plutôt qu'en les testant une par une. Cela nécessite de parcourir l'ensemble des clés plutôt que l'ensemble des valeurs.
Pour ajouter une cave, il se pose le problème de modifier la fonction setExits afin de rajouter un escalier pour monter et un escalier pour descendre. Mais cela ne semble pas être une solution qui permettra d'ajouter facilement de nouvelles sorties à l'avenir et il semble préférable de créer une méthode "setter" qui ajoute une sortie quelconque à une salle.
Une fois la méthode setExits créée, il devient très simple d'ajouter une cave.
Un des problèmes rencontrés est que l'ancienne fonction était appelée 1 fois pour chaque direction, ce qui a fait que les directions étaient appelées 4 fois à cause de la nouvelle implémentation avec les boucles.

Ex 7.10:
Explication de la méthode getExits: Cette méthode va créer un tableau qui contient l'ensemble des clés de la HashMap. Il suffit ensuite de parcourir ce tableau pour avoir accès à toutes les directions dans lesquelles il y a une sortie et surtout leur nom, pour pouvoir les concaténer dans une chaîne.

Ex 7.11:
Ajout d'une méthode qui retourne la description détaillée d'une pièce. Cela permet de respecter la règle qui veut qu'une classe soit responsable de la manipulation de ses données.

Ex 7.14:
Ajout d'une méthode pour obtenir la description détaillée d'une salle, branchée sur la commande **look**

Ex 7.15:
Ajout d'une méthode **smoke**, qui permet à Herlock de sortir sa pipe et de se mettre à la fumer tandis qu'il recherche des indices.

Ex 7.16:
Ajout d'une nouvelle méthode pour afficher l'ensemble des commandes. Logiquement, celle-ci se trouve dans la classe commande.

Ex 7.18:
J'ai déplacé l'affichage des commandes de la classe Command dans la classe Game, ce qui présente l'intérêt que la classe Command n'ait plus de dépendance avec l'interface (elle est donc moins "couplée".

Ex 7.18.1:
J'ai comparé mon projet avec le fichier zuul-better, et effectué quelques modifications minimes.

Ex 7.18.3:
Recherche des différentes images du jeu, et redimensionnée.

Ex 7.18.8:
Dans le "Border Layout Est", je rajoute un bouton avec l'aide de JButton.
Ensuite, j'ajoute le listener pour capturer les évènements. Le 1er bouton est un bouton "Quit" qui prend toute la partie droite de l'écran.
Comme c'est très pratique et que cela me fait gagner du temps, je cherche comment ajouter plusieurs bouton, et dans le layout je rajoute un autre layout de type Y_AXIS pour que les boutons soient empilés les uns sur les autres

Ex 7.20:
Je crée une classe Item avec une description, un poids et une description détaillée.
Lors de l'initialisation du jeu, j'ajoute un item dans différentes pièces et j'affiche l'item présent quand on rentre dans la pièce

Ex 7.21:
Mise en place d'une description pour chaque Item créé dans l'exercice précédent, ce qui nécessite quelque changement pour rajouter le paramètre pDescription

Ex 7.22:
Ajout des Items imaginés pour le scénario, avec un item (Carte de Visite) qui se trouve dans la pièce où le joueur démarre, même si il n'a aucune influence sur le jeu, et plusieurs items dans le Salon (Pipe et Loupe)

Ex 7.23:
Ajout d'une commande Back, qui renvoie le joueur dans la dernière salle visitée (ce qui a été ensuite changé dans l' ***Ex 7.25***, afin que la succession de l'appel de back fasse revenir le joueur à sa position initiale)

Ex 7.26:
Empiler les "Rooms" au fur et à mesure du parcours, et modification de la commande Back pour dépiler la dernière pièce visitée. (Ajout d'un test pour gérer le cas où la pile est vide)

Ex 7.27:
What sort of baseline functionality tests might we wish to establish on the current version of the game ?
On peut préparer une liste de commandes pour tester toutes les capacités du jeu : parcourir les différentes salles, revenir en arrière, afficher l'aide, etc…
Ensuite, cette liste de commande sera exécutée afin de vérifier qu'elle ne provoque aucune erreur.

Ex 7.28.1:
Ajout d'un fichier texte afin de pouvoir exécuter une série de test prédéfinis rapidement

Ex 7.28.2:
Création de deux nouveaux fichiers commande

Ex 7.29:
Ajout d'une classe player, transfert du code qui concerne le joueur de la classe GameEngine à la classe player. Transformation de la liste des salles en HashMaps de manière à pouvoir manipuler les salles par leurs nom, car la classe player n'a plus accès à la liste des salles.

Ex 7.30:
Modification de la classe player pour prendre et déposer un objet de la salle courante

Ex 7.31:
Modification de la classe player pour stocker les items dans une Hashmap, modification de take et drop pour prendre et déposer les items par leurs noms

Ex 7.31.1:
Création d'une nouvelle classe ItemList qui permet de stocker une liste d'objets.
Modification de la classe player et Room pour stocker leurs objets dans la nouvelle classe ItemList

Ex 7.32:
Ajout de différentes caractéristiques du joueur dont la force, qui est tiré de manière aléatoire lors de la création de ce dernier.
Le poids max est fixé à 250% de sa force

Ex 7.33:
Implémentation d'une nouvelle commande Items qui affiche l'inventaire des objets du joueur, ajout de celle çi dans la liste des commandes (Et ajout d'un bouton).
Modification de la classe ItemList et Player

Ex 7.34:
Ajout d'un "Magic Cookie", ajout de la possibilité de manger avec eat, il est impossible de manger autre chose que le cookie, ajout d'une enclume trop lourde pour tester l'effet du cookie, ajout d'une caractéristique du player extraWeight qui permet de porter un poid supérieur à ce que sa force normale lui permettrait

Ex 7.34.1:
Ajouts des commandes récentes dans le fichier test parcours.txt, exécution du fichier et correction des bugs

Ex 7.34.2:
Régénération des 2 javadocs, et correction des erreures