



[TITRE DU DOCUMENT]

[Sous-titre du document]



[DATE]

[NOM DE LA SOCIETE]

[Adresse de la société]

Table des matières

I) PREMIERE PARTIE	2
A) Auteur	2
B) Thème (phrase-thème validée)	2
C) Résumé du scénario	2
D) Plan	3
E) Scénario détaillé	4
F) Détail des lieux, items, personnages	4
1- Les lieux	4
2- Les personnages	8
3- Les items.....	8
G) Situation gagnantes et perdantes	8
H) Enigmes, mini-jeux, combats, etc...	8
I) Commentaires	9
II) Réponses aux exercices	9
7.5) PrintLocationInfo	9
7.6) getExit.....	9
7.7) getExitString	10
7.8) HashMap, setExit.....	11
7.8.1) Ajoute d'un déplacement vertical dans le scénario	12
7.9) KeySet.....	12
7.11) getLongDescription	13
7.14) Look	13
7.15) Eat.....	14
7.16) showAll, showCommands	15
7.18) getCommandList	15
7.18.1) Zuul-better.....	16
7.18.3) chercher des images.....	16
7.18.4) Titre du jeu	16
7.18.6) Zuul-with-images.....	16
7.18.8) Ajout d'un bouton	17
7.19) Mettre les images dans un dossier.....	19
7.20) Item	19
7.22) Item avec une hashmap	21

Rapport

Titre : [pas encore de titre]

I) PREMIERE PARTIE

A) Auteur

Auteur : Jolan Roustant E1Grp10

B) Thème (phrase-thème validée)

Ma phrase thème est : « *Un soldat dans les tranchées lors de la première guerre mondiale doit transmettre un message* ».

C) Résumé du scénario

Un soldat français se réveille dans les tranchées durant la première guerre mondiale, un camarade lui annonce que son supérieur a quelque chose d'important à lui dire. Il découvre qu'il doit transmettre à temps un message de la plus haute importance à un autre bataillon, en effet ce bataillon va se faire attaquer, il doit donc prévenir ce bataillon à temps. On doit donc suivre les indications des soldats sur notre chemin et éviter les pièges (mines, éboulements...) afin de trouver ce bataillon.

D) Plan

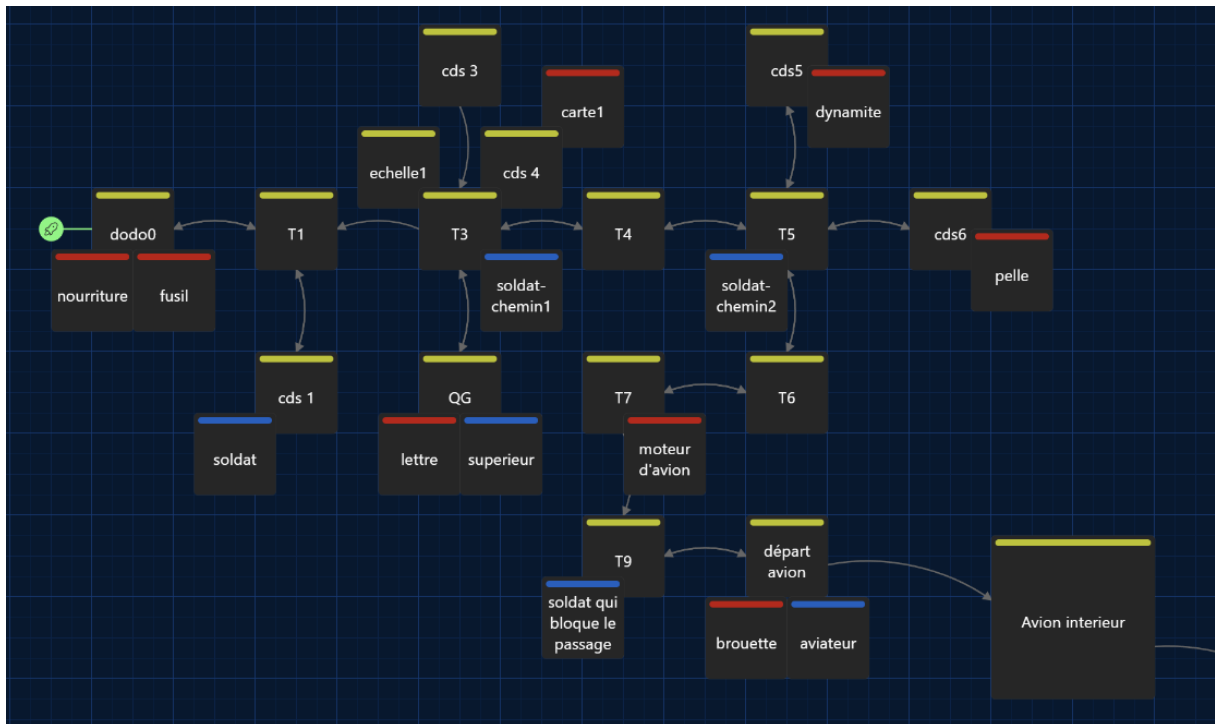


Figure 1 - Première partie du plan

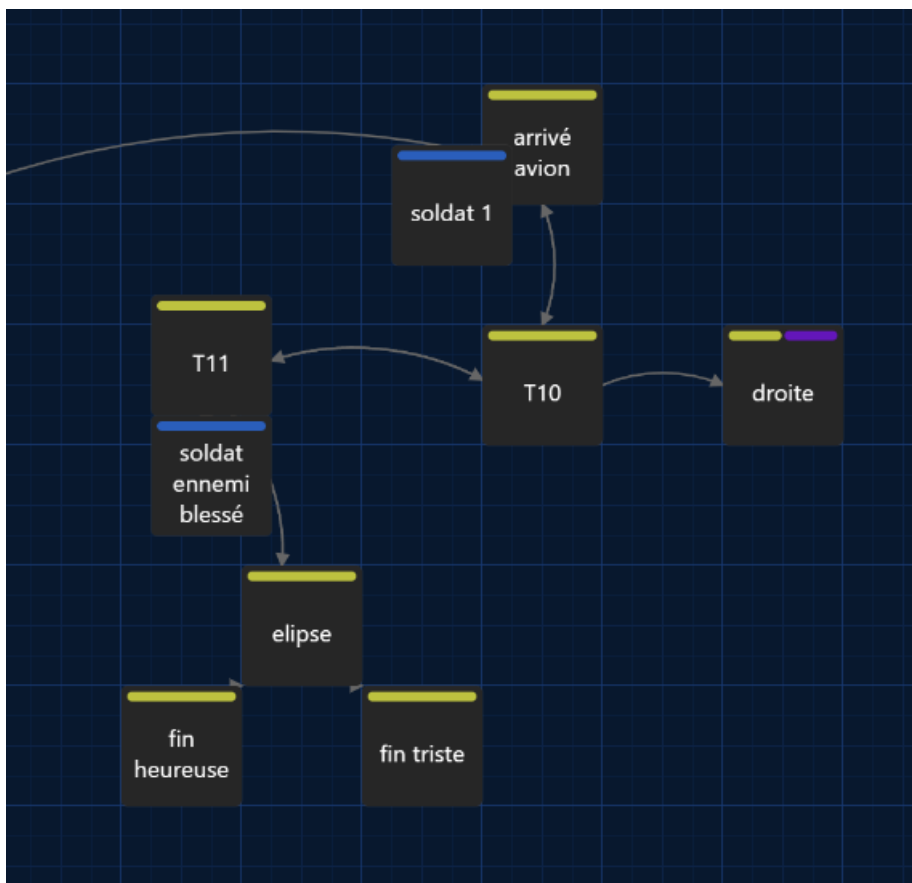


Figure 2- seconde partie du plan

Le plan est fini mais peut encore évoluer selon mes envies. Les lieux sont en jaune, les personnages en bleu et les objets en rouge.

E) Scénario détaillé



À son réveil, un camarade lui dit de le suivre pour entendre ce que son supérieur a à dire. Le supérieur donne au soldat une lettre à remettre pour sauver un bataillon en danger. Le soldat doit alors choisir d'accepter ou de refuser la mission. S'il accepte, il doit se rendre à l'endroit indiqué par le supérieur mais se heurte à un glissement de terrain qui bloque le chemin. Il doit trouver une pelle pour dégager le chemin. Plus tard il va rencontrer un aviateur qui est prêt à l'aider mais dont l'avion est endommagé et doit être réparé. Le joueur doit alors l'aider à réparer l'avion. Pendant le vol il peut choisir de parler de sa mission ce qui aura comme conséquence de faire échouer la mission (mais le joueur ne le sais qu'à la fin). Arrivé sur la terre ferme il va continuer son chemin en suivant les indications des soldats, mais s'il se trompe il peut tomber sur des mines. Enfin après avoir beaucoup marché il va arriver au bataillon allié, s'il s'est confié au pilote, le bataillon a malheureusement subi une attaque car le pilote était un traître. Dans le cas contraire, il réussit sa mission, sauve le bataillon, qui va se préparer à lutter et est considéré comme un héros.







Il y a le scénario existe sous la forme d'un fichier Twine.







F) Détail des lieux, items, personnages

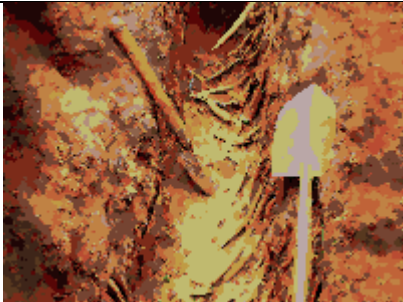




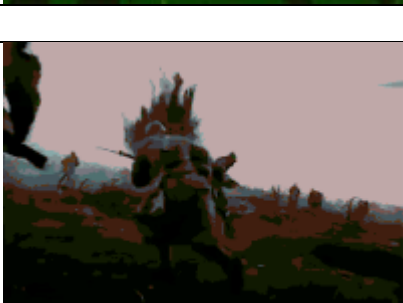
1- Les lieux

Les lieux ne sont pas dans un ordre précis.

	Nom	Description	Image
1	Dodo0	C'est ta chambre, c'est un peu petit et tu dois la partager avec 7 autres soldats.	
2	T1	Tu vois la lumière du jour, le soleil t'aveugle et la boue te glace les pieds. Tu dois suivre ton compagnon.	

3	T3	Le QG est au sud, c'est pas si loin.\nIl y a un soldat:\nSoldat inconnu: 'Salut, au sud se trouve les quartiers des superieurs, au nord tu trouveras l'infirmierie et les tranchées se poursuivent à l'est.'	
4	T4	Une tranchée vide.	
5	T5		
6	T6	Il y a eu un éboulement, il faut quelque chose pour débloquer le chemin.	
7	T7		
8	T9		

9	T10		
10	T11		
11	CDS1	Tu ne peux pas aller plus loin au risque de désertir.	
12	CDS3	C'est l'infirmerie.	
13	CDS4	Tu es dans une tranchée secondaire, en dessous de la surface. Il y a un soldat seul.	
14	CDS5		

15	CDS6		
16	QG	Tu es dans les quartiers des supérieurs, ton chef t'attend.	
17	DepartAvion		
18	ArriveAvion		
19	InterrieurAvion		
20	Droite		
21	Echelle1		

2- Les personnages

Pas de personnages pour l'instant.

3- Les items

	Nom	Lieu	Description	Poids
1	pelle	CDS6	une pelle qui creuse	2310
2	dynamique	CDS5	un bâton de dynamite	1000
3	moteur	T7	Une pièce de moteur qui a l'air cassée	62000
4	lettre	QG	Une lettre importante	10
5	fusil	Dodo0	Votre arme	3000
6	Boîte de singe	Dodo0	De la nourriture, ça peut vous rendre plus fort	200
7	Carte	CDS4	Une carte qui affiche une partie des tranchées	10
8	Brouette	DepartAvion	Une brouette qui augmente la taille de l'inventaire. ('utiliser brouette' pour activer son effet)	5000

G) Situation gagnantes et perdantes

Situation gagnante :

- Arrivé au bataillon sans avoir parlé de sa mission au pilote.

Situation perdante :

- Arriver au bataillon en ayant parlé de sa mission au pilote.
- Tomber sur des mines.
- Refuser la mission
- Revenir au QG avant d'avoir donné la lettre car on est considéré comme un traître.
- Monter au combat

H) Enigmes, mini-jeux, combats, etc...

Rien de tout cela.

I) Commentaires

II) Réponses aux exercices

7.5) PrintLocationInfo

L'intérieur de la fonction printLocationInfo ressemble à ça :

```
    this.aCurrentRoom = this.aRoom;  
    System.out.println("You are "+this.aCurrentRoom.getDescription());  
    System.out.print("Exits: ");  
    if(this.aCurrentRoom.aNorthExit!=null)  
    {  
        System.out.print("North ");  
    }if(this.aCurrentRoom.aEastExit!=null)  
    {  
        System.out.print("East ");  
    }if(this.aCurrentRoom.aSouthExit!=null)  
    {  
        System.out.print("South ");  
    }if(this.aCurrentRoom.aWestExit!=null)  
    {  
        System.out.print("West ");  
    }  
    System.out.println(" ");
```

On va l'appeler dans les méthodes printWelcome et goRoom de la classe game.

7.6) getExit

L'accesseur ressemble à ça avec "pDirection" au lieu de "direction", "aNorthExit" au lieu de "northExit" (pareil pour les autres attributs). (Cette partie du rapport a été faite après avoir modifier le code):

```
// méthodes existantes non modifiées

public Room getExit(String direction)
{
    if(direction.equals("nord")) {
        return northExit;
    }
    if(direction.equals("est")) {
        return eastExit;
    }
    if(direction.equals("sud")) {
        return southExit;
    }
    if(direction.equals("ouest")) {
        return westExit;
    }
    return null;
}
```

On appelle cette fonction dans goRoom()

7.7) getExitString

Le code ressemble à ça et on appelle cette fonction dans printLocationInfo() :

```
public String getExitString()
{
    String vExitsList = "Les sorties visibles ici sont : ";
    if(this.getExit("nord")!=null)
    {
        vExitsList += "nord ";
    }
    if(this.getExit("est")!=null)
    {
        vExitsList += "est ";
    }
    if(this.getExit("sud")!=null)
    {
        vExitsList += "sud ";
    }
    if(this.getExit("ouest")!=null)
    {
        vExitsList += "ouest ";
    }
    return vExitsList;
}
```

On peut donc modifier la méthode printLocationInfo() qui doit appeler getExitString()

```
private void printLocationInfo()
{
    System.out.println("Tu es "+this.aCurrentRoom.getDescription());
    System.out.println(this.aCurrentRoom.getExitString());
}
```

Comprenez-vous pourquoi il est logique de demander à Room de produire les informations sur ses sorties (et ne pas lui demander de les afficher), et pourquoi il est logique de demander à Game d'afficher ces informations (et ne pas lui demander de les produire) ?

- C'est logique car Room possède les attributs sur les sorties alors que grâce à l'accesseur, Game peut les afficher mais pas les modifier donc c'est pour ça qu'elle ne les produit pas mais les affiche justes.

7.8) HashMap, setExit

Le but de l'exercice est de remplacer les 4 attributs, par une HashMap, l'avantage est de pouvoir ajouter de nouvelles directions facilement.

Il faut importer la HashMap :

```
import java.util.HashMap;
```

Il faut créer un nouvel attribut qui contiendra les sorties sous la forme

<String (représentant le nom de la sortie), Room (la pièce)>

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

Il faut maintenant compléter le constructeur pour initialiser la HashMap.

```

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

```

Modification de la méthode setExits() et getExits():

```

public void setExits(final String pDirection, final Room pNeighbor)
{
    this.exits.put(pDirection, pNeighbor);
}

public Room getExit(String pDirection)
{
    return exits.get(pDirection);
}

```

Nouvelle façon de mettre les sorties des salles :

```

vDodo0.setExits("est", vT1);
vT1.setExits("est", vT2);
vT1.setExits("sud", vCDS1);
vT1.setExits("ouest", vDodo0);
vT2.setExits("est", vT3);
vT2.setExits("ouest", vT1);

```

7.8.1) Ajoute d'un déplacement vertical dans le scénario

Scénario modifié.

7.9) KeySet

Il faut refaire fonctionner la fonction getExitString()

```

public String getExitString()
{
    String vExitsList = "Les sorties visibles ici sont : ";
    Set<String> vKeys = this.aExits.keySet();
    for(String exit : vKeys)
    {
        vExitsList += " " + exit;
    }
    return vExitsList;
}

```

7.11) getLongDescription

Voilà la nouvelle fonction.

```

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

```

On peut donc modifier printLocationInfo() pour juste afficher ce que nous retourne cette nouvelle fonction.

La classe Room n'a plus que 2 attributs car on utilise une hashmap.

7.14) Look

On modifie la classe CommandWords afin que ce soit plus simple d'ajouter de nouvelles commandes en ajoutant un tableau constant. On supprime aussi l'intérieur du constructeur.

```
private final String[] aValidCommands = {"aller", "quitter", "aide",  
"regarder"};
```

On ajoute la fonction look dans la classe Game.

```
private void look()  
{  
    System.out.println(this.aCurrentRoom.getLongDescription());  
}
```

On ajoute ensuite cette condition dans la fonction processCommand :

```
else if (pCommand.getCommandWord().equals("regarder"))  
{  
    this.look();  
}
```

7.15) Eat

On ajoute la commande « manger » dans le tableau de la classe CommandWords.

On crée la fonction eat() dans la classe Game :

```
private void eat()  
{  
    System.out.println("Tu viens de manger tu n'as pas faim");  
}
```

On ajoute aussi la même condition que look dans la fonction processCommand()

7.16) showAll, showCommands

Le but de showAll est de lister les commandes disponibles pour le joueur. On va donc parcourir le tableau avec une boucle for.

```
public void showAll()
{
    for(String command : this.aValidCommands)
    {
        System.out.print(command + " ");
    }
    System.out.println();
}
```

On va aussi ajouter une méthode showCommands dans la classe Parser afin d'y accéder plus facilement.

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

Dans la classe Game on n'est plus obligé de lister toutes les commandes avec un S.o.p car on peut appeler la méthode showCommands.

7.18) getCommandList

On ne veut plus que la classe CommandWords affiche des choses. Il faut donc faire en sorte que les méthodes ne print rien mais retourne des chaînes de caractères.


```
public String getCommandList()
{
    String vList = "";
    for(String command : this.aValidCommands)
    {
        vList += command + " ";
    }
    return vList;
}
```

On modifie showCommands dans la classe Parser pour qu'il appelle notre nouvelle méthode getCommandList().

7.18.1) Zuul-better

Rien à modifier.

7.18.3) chercher des images

Les images sont listées plus haut.

7.18.4) Titre du jeu

Le titre est "WWW : Wild Wide War" qui est une référence au film Wild Wild West ainsi qu'à Word Wide Web.

7.18.6) Zuul-with-images

Le but est d'ajouter les images à notre jeu. On n'a pas besoin de modifier Command et CommandWords. On doit modifier la classe Room pour qu'elle gère les images. On ajoute un attribut String almageName pour stocker l'url de l'image. Voici le nouveau constructeur :

```

public Room(final String pDescriptionLieu, final String pImage)
{
    this.aDescription = pDescriptionLieu;
    this.aExits = new HashMap<String, Room>();
    this.aImageName = pImage;
}

```

On ajoute aussi un simple getter afin de pouvoir facilement récupérer le nom de l'image.

On doit ajouter 2 classes, UserInterface et GameEngine, la première est donnée, la seconde reprend une partie de la classe game. On a donc une Classe Game plus courte :

```

public class Game
{
    private UserInterface aGui;
    private GameEngine aEngine;
    public Game()
    {
        this.aEngine = new GameEngine();
        this.aGui = new UserInterface( this.aEngine );
        this.aEngine.setGUI( this.aGui );
    }
}

```

On a initialisé une Interface utilisateur. Elle va permettre d'afficher plus proprement le jeu.

On doit remplacer chaque `System.out.println()` par `this.aGui.println()`

La classe Parser est fortement modifiée car on ne lit plus la commande de la même façon (on utilise plus de Scanner).

7.18.8) Ajout d'un bouton

J'ai décidé d'ajouter 2 boutons pour prendre en main les nouvelles fonctionnalités (Jpanel).

```
private JButton aButtonMap;  
private JButton aButtonHelp;
```

On initialise les boutons comme attributs de type JButton. Puis on les initialise :

```
this.aButtonMap = new JButton("map");  
this.aButtonHelp = new JButton("Help");
```

On crée un Panel et on y ajoute les boutons afin de grouper les boutons au même endroit.

```
JPanel vPanelButton = new JPanel();  
vPanelButton.setLayout( new BorderLayout() );  
vPanelButton.add(this.aButtonMap, BorderLayout.CENTER);  
vPanelButton.add(this.aButtonHelp, BorderLayout.SOUTH);
```

On ajoute notre panel de boutons au panel principal.

```
vPanel.add(vPanelButton, BorderLayout.EAST);
```

On ajoute un actionListener pour pouvoir savoir quand le bouton est cliqué.

```
this.aButtonMap.addActionListener(this);  
this.aButtonHelp.addActionListener(this);
```

On ajoute les actions à faire au début de la méthode actionPerformed() à l'aide de condition

```
if(pE.getActionCommand() == "aide")
{
    this.aEngine.interpretCommand(pE.getActionCommand());
}
else if(pE.getActionCommand() == "map")
{
    this.aEngine.interpretCommand(pE.getActionCommand());
}
```

7.19) Mettre les images dans un dossier

J'avais déjà pris l'initiative de le faire. Les images ne sont plus appelées « image.png » mais « img/image.png ».

7.20) Item

On ajoute les objets comme montré ci-dessous dans la méthode createRooms() de la classe GameEngine.

```
Item vIntegrale = new Item(" je suis une intégrale", 12);
vT1.setItem(vIntegrale);
```

Il faut donc une classe item avec deux attributs (une description et un poids) et deux accesseurs.

```
public class Item
{
    private int aWeight;
    private String aDescription;

    public Item(final String pDescription, final int pWeight)
    {
        this.aDescription = pDescription;
        this.aWeight = pWeight;
    }

    public String getDescription(){return this.aDescription;}

    public int getWeight(){return this.aWeight;}
}
```

On doit ajouter un item dans une pièce (la classe Room) pour pouvoir stocker l'item.

On ajoute un setter pour récupérer l'objet. Il faut maintenant pouvoir récupérer la description de l'objet.

```
public String getItemDescription()
{
    if(this.aItem != null)
    {
        return this.aItem.getDescription() + " qui pèse " +
this.aItem.getWeight();
    }
    else
    {
        return "Il n'y a pas d'objet ici...";
    }
}
```

On peut modifier la méthode getLongDescription :

```
return this.aDescription + "\n" + this.getExitString() + "\n" +
this.getItemDescription();
```

7.22) Item avec une hashmap

On veut pouvoir stocker plusieurs Item par pièce donc on va remplacer l'attribut de type Item par un attribut de type HashMap<String, Item> (on va l'appeler aItemList). On doit donc modifier le constructeur, transformer les setter en méthode pouvant ajouter un élément à la HashMap avec la fonction put().

On modifie bien la fonction getItemDescription()

```
public String getItemDescription()
{
    String vItemDescription = "Les objets sont : \n";
    Set<String> vKeys = this.aItemList.keySet();
    for(String vItem : vKeys)
    {
        vItemDescription += " " + vItem + "\n";
    }//for
    return vItemDescription;
}
```

On ajoute donc les objets de cette façon désormais :

```
vT1.addItem("intégrale", vIntegrale);
```

7.22.2) Intégrer les objets (items)

Objets intégrés.

7.23) Back

On ajoute la commande dans la liste des commandes de la classe CommandWords. On ajoute un attribut de type de Room dans la classe GameEngine pour stocker la pièce précédente. On ajoute la condition (comme pour toutes les nouvelles commandes) dans processCommand(). Dans la fonction goRoom() on y ajoute une ligne qui modifie notre pièce précédente :

```
this.aPreviousRoom = this.aCurrentRoom;
```

Enfin on code notre fonction back :

```
private void back()
{
    if(this.aPreviousRoom == null)
    {
        this.aGui.println("Tu n'as pas avancé, tu ne peux pas retourner sur tes pas");
    }
    else
    {
        this.aCurrentRoom = this.aPreviousRoom;
        this.aGui.println( this.aCurrentRoom.getLongDescription() );
        if ( this.aCurrentRoom.getImageName() != null )
            this.aGui.showImage( this.aCurrentRoom.getImageName() );
    }
}
```

7.26) Back stack

On souhaite pouvoir revenir en arrière de plusieurs pièces, pour cela il nous faut un historique de nos pièces où on est allé. On va donc utiliser la classe Stack.

On importe java.util.Stack dans GameEngine

On crée la Stack dans le constructeur.

```
this.aPreviousRooms = new Stack<Room>();
```

Il suffit ensuite de changer 2 lignes :

```
- if(this.aPreviousRoom == null)
+ if(this.aPreviousRooms.empty())
```

Et :

```
this.aCurrentRoom = this.aPreviousRoom;  
this.aCurrentRoom = this.aPreviousRooms.pop();
```

7.28.1) Test

Création d'une nouvelle méthode test.

```
private void test(final String pFileName)  
{  
    Scanner vScan;  
    try  
    {  
        vScan = new Scanner(new File("alltests/" + pFileName + ".txt"));  
  
        while(vScan.hasNextLine())  
        {  
            String vLine = vScan.nextLine();  
            interpretCommand(vLine);  
        }  
    }  
    catch(final FileNotFoundException pFNFE)  
    {  
    }  
}
```

Images :

Dodo0 :

- <https://www.lunion.fr/id357546/article/2022-04-02/au-musee-de-la-grande-guerre-de-meaux-la-tranchee-expose-ses-verites>
- Photo de figurine de soldat

T1 :

- Image de dall-e
- Soldat de la bande-annonce d'un film (source perdu)

T2

- Midjourney

T3

-