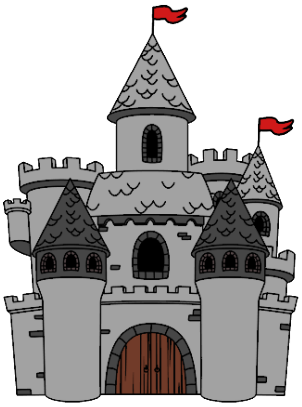
Mini-projet «Escape game »

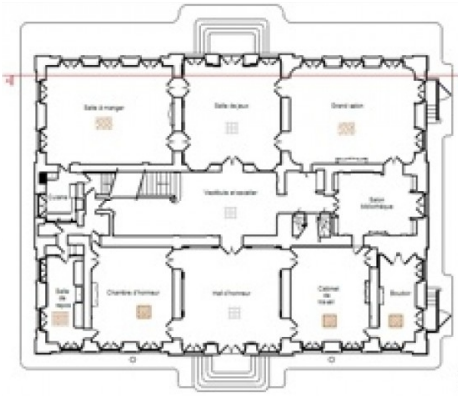


1. *Objectif*

- L'objectif de ce mini-projet est la création d'un jeu d'*escape game* simplifié.
- Un aventurier prisonnier d'un château doit trouver la sortie en résolvant des énigmes.

2. *Contexte*

- Un aventurier se réveille dans une pièce d'un château dont il ne possède pas le plan.



- Le château est constitué de pièces communiquant les unes avec les autres au moyen de passages menant soit dans 1 des 4 directions (nord, sud, est, ouest) soit vers le haut, soit vers le bas. Il possède au moins 3 niveaux.
- L'une des pièces au moins est une sortie.
- Les passages peuvent être
  - Verrouillés et éventuellement piégés.
  - Gardés par 1 sage (ou au autre chose) qui pose une énigme (Simple)
- Les pièces peuvent contenir :
  - Des objets : Potion de soin, arme ou parchemin de sort (à vous d'inventer)
  - Des ennemis : Monstres belliqueux (à vous d'inventer)
- Le joueur possède un nombre de points de vie de départ. Il en perd lorsqu'il est blessé par un ennemi ou qu'il déclenche un piège. Il en récupère avec des potions de soin. Optionnellement il peut en récupérer au bout d'un certain temps.
- Les ennemis aussi possèdent des points de vie.
- Lorsque l'aventurier rencontre un ennemi, un combat s'engage automatiquement et chacun attaque à tour de rôle.

- L'aventurier peut éventuellement tenter fuir par un des autres passages.
- Lorsque l'ennemi ou l'aventurier se touchent ils s'infligent des dommages de points de vie. Lorsque l'un d'eux atteint 0 point de vie ou moins il meurt.
- Il peut exister des ennemis plus ou moins résistants (Points de vie) , plus ou moins forts (Dommages sur l'aventurier) et plus ou moins habiles au combat (Chance de toucher l'aventurier) (Inventez !).

- Au départ l'aventurier se bat à mains nues et inflige peu de dommages aux ennemis, les armes qu'il peut trouver augmentent les dommages qu'il inflige.

- L'aventurier ne peut porter qu'une seule arme et aucun autre objet. Lorsqu'il trouve des potions, par exemple, il doit les utiliser immédiatement.

3. *Déroulé général*

- L'aventurier peut se déplacer : au démarrage et chaque fois qu'il entre dans une nouvelle pièce
  - un plan du niveau où il se trouve s'affiche (uniquement les pièces qu'il a déjà visité)
    - Pour simplifier les niveaux peuvent être carrés et les pièces de tailles identiques
  - Si un monstre belliqueux est présent il attaque l'aventurier et un combat se déroule. L'aventurier peut tenter de fuir.
  - Ensuite on lui indique les choix de déplacement possibles et les objets qui s'y trouvent
  - L'aventurier peut choisir
    - De ramasser une arme : s'il en a déjà une il l'échange avec la sienne
    - D'utiliser un objet sur place (potion etc …)
    - De sortir de la pièce par un des passages :
      - Si le passage n'est ni verrouillé ni gardé ni piégé il passe dans la pièce suivante,
      - si le passage est gardé, le sage pose l'énigme,
      - si le passage est verrouillé, l'aventurier peut essayer de défoncer ou crocheter la porte (1 essai de chaque seulement) à vous de voir comment gérer

- si le passage est piégé, l'aventurier prend des dommages,
- Le jeux se termine quand l'aventurier a trouvé la sortie où qu'il est décédé

4. *Résultat attendu*

- Vous devrez écrire un programme permettant de jouer à ce jeu,
- Votre programme devra être écrit en utilisant les concepts de la programmation orientée objet et le langage Java (Classes, encapsulation, composition, héritage, polymorphisme, exceptions et éventuellement interfaces et programmation générique). Au besoin vous utiliserez la récursivité.
- Vous afficherez les informations nécessaires pour que le joueur puisse suivre la partie et en fin de partie un score (à vous de voir lequel).
- ATTENTION : l'objectif principal de ce projet est le développement d'une application en Programmation Orientée Objet, ne pas perdre cet objectif de vue.

4. *Outils*

1. Simuler le hasard

- Vous aurez besoin d'introduire du *hasard* notamment pour gérer les combats. Vous pourrez utiliser la méthode Math.random() :

```
public static double random()

Renvoie une valeur double avec un signe positif, supérieur
ou égal à 0,0 et inférieur à 1,0. Les valeurs renvoyées
sont choisies de façon pseudo-aléatoire avec une
distribution (approximativement) uniforme à partir de cette
plage.
```

- Voici la méthode genererUnEntier() qui utilise la méthode Math.random() pour générer pseudo-aléatoirement un entier compris entre 1 et un maximum. Vous pouvez l'utiliser ou vous en inspirer dans votre programme :

```
public static int genererUnEntier(int max){

        return (int) (Math.random() * max + 1);
    }
```

4. *Éléments de méthode*

- Pour traiter ce problème vous pouvez le décomposer en vous posant les questions suivantes :
  - *De quelles fonctionnalités générales le programme doit-il disposer ?*
  - *De quelles données le programme a-t-il besoin pour mettre en œuvre les fonctionnalités ? Par quelles classes peut-on représenter ces données ?*
  - *Quelles actions élémentaires le programme doit-il être capable de réaliser afin de disposer des fonctionnalités identifiées ? A quels objets ses actions se rattachent-elles.*

- Quelques suggestions :
  - Pour construire vos classes
    - Le château est composé de pièces qui sont situé sur un niveau,
    - Une pièce peut contenir des objets (de différents types),
    - Une pièce peut être occupée par un ennemi,
    - Une pièce comporte des passages qui donnent sur des pièces,
    - Un passage peut être gardé par une énigme (et un sage),
    - Un passage peut contenir un piège,
    - Une énigme est constituée d'une question, des réponses possibles et d'une bonne réponse,
    - Un piège est d'un type et inflige des dommages
    - …
  - Quelques fonctionnalités possibles des classes
    - un niveau peut être affiché
    - une énigme peut être poser
    - un occupant peut combattre
    - un aventurier peut se déplacer, utiliser un objet ...
- Synthétiser le problème

- Vous avez les outils pour traduire le problème sous forme d'un algorithme en français basé sur un enchaînement d'actions élémentaires.

- Traduisez ces actions élémentaires en code informatique
  - Écrivez les classes nécessaires pour traiter le problème ?
    - Pensez « objets » et non méthodes ⇒ par exemple une énigme contiendra la ou les méthodes qui permettent de la poser
    - réfléchissez aux interactions entre les objets, comme relier pièces et niveaux, les pièces entre elles etc ...
  - Dotez les classes des méthodes nécessaires à la réalisation des actions élémentaires.
  - Peut-être certaines de ces actions sont-elles généralistes et peuvent donner lieu à l'écriture de méthodes de classe.
  - Simplifier vos méthodes en écrivant des « sous-méthodes » quand cela est utile. Rappel : une méthode doit se concentrer sur la réalisation d'une tâche précise.

5. *Consignes générales*

- Faites simple : si un code est trop compliqué, simplifiez-le en le découpant (en méthodes par exemple). Remplacez les if … else par des switch quand c'est possible, utilisez des tableaux quand cela est pertinent.
- Utilisez ce que vous connaissez (les notions vues en cours) : n'essayez pas d'intégrer des morceaux de codes que vous ne comprenez pas. L'objectif est de mettre en oeuvre ce que vous avez appris.
- Accompagnez votre utilisateur : affichez des messages clairs. Contrôlez les saisies pour éviter les valeurs erronées.
- Soignez la présentation des affichages : Passez des lignes quand c'est nécessaire, introduisez des séparateurs, mettez des majuscules aux initiales des mots et surtout faites attention à l'orthographe.
- Écrivez un code lisible : Nommez vos variables intelligemment, respecter les conventions de nommage, commentez, indentez et aérez !
- Testez et fiabilisez votre code :
  - Assurez -vous que chacune de vos classes et méthodes fonctionnent indépendamment du reste du code.
  - Si certaines parties de votre programme ne fonctionnent pas ou ne sont pas terminées au moment de le rendre, signalez le dans le dossier de programmation.

Dans ce cas arrangez-vous pour rendre un source compilable sans erreur, par exemple en laissant le corps d'une méthode vide, en renvoyant une constante, en commentant les parties contenant des erreurs ...

7. *Attendus*

- Vous fournirez l'ensemble du code source produit.
- Vous accompagnerez votre code source d'un dossier de programmation succinct (Au maximum 4 à 5 pages sans les éventuels exemples) dans lequel vous détaillerez notamment :
  - La méthode (au sens façon de faire) générale utilisée,
  - Les choix des types de données pour les données principales,
  - Pour chaque classe , son utilité et son fonctionnement (Cela ne vous empêche pas de commenter le code), la signification des champs et l'utilité des méthodes.
  - Vous pouvez éventuellement joindre des exemples de résultats d'exécution.

  Remarque : Le dossier de programmation sera réalisé au moyen d'un traitement de texte.