# TP de programmation orientée objet.

L’objectif de ce TP est de créer un petit jeu d’aventure. Il consiste à incarner un personnage qui évolue sur une île (représenté par une carte quadrillée) à la recherche de sa princesse.

A chaque déplacement, il peut soit tomber sur un monstre, sur une case de soin (qui restaure la totalité de ses points de vie) ou sur le château caché où est retenue la princesse par le grand méchant Balbatrux.

## EXERCICE 1

Implémenter un programme simple effectuant un combat entre un joueur et un monstre.

Chacun a un nombre de points de vie et tape sur l’autre à tour de rôle jusqu’à la mort d’un des protagonistes.

Ne pas utiliser de classes.

## EXERCICE 2

### 1/Modélisation des joueurs et des monstres.

-Implémenter la classe Personnage dont hériterons nos monstres et nos joueurs :

Cette classe possède en entier représentant les Pv, un string représentant le nom ainsi qu’une méthode Bagarre qui va permettre d’infliger des dégâts à une cible.

Il peut être opportun de créer un booléen qui sera calculé pour passer à vrai quand les pv du personnage passent à 0.

-Implémenter la classe Monstre :

La classe Monstre hérite de Personnage et contient seulement un constructeur qui va initialiser un nombre de PV aléatoires.

-Implémenter la classe Joueur :

La classe Joueur hérite de personnage et contient un champ PvMax. Son constructeur a en paramètres un string pour le nom et la valeur des PvMax. Dans celui-ci, on va affecter les PvMax au champ PvMax mais aussi au champ Pv (pour que le joueur ne soit pas mort-né)

2/Implémentation du moteur de Partie

-Implémenter la classe Lieu qui représente une case.

La classe contient 3 booléens : EstFin, EstVisite, EstCaseSoin.

Aussi, elle contient une propriété Monstre qui permettra d’intégrer un Monstre à la case.

Enfin, il faut implémenter une méthode générer qui va permettre de générer aléatoirement la case (si c’est une case de soin, une case gardée par un montre ou une case vide) à partir du RandomHelper fournit.

-Implémenter la classe Carte

La carte représente le tableau de jeu.

Ce dernier présente 5 paramètres :

-une variable \_dimension qui représente la taille de la carte (On part du principe que la carte est carrée).

-un tableau à deux dimensions de lieux représentant la carte en elle-même.

- deux entiers représentant la position du joueur grâce à des coordonnées en X et en Y (représentant la position dans le tableau à deux dimensions).

- un champ calculé Position représentant le lieu courant (en effectuant une récupération dans notre tableau à deux dimensions.)

Le constructeur prend la dimension en paramètre et l’affecte à la variable \_dimension puis initialise \_carte à ces mêmes dimensions.

Coder une méthode Generer qui parcourt toutes les cases de la carte pour générer aléatoirement les cases (pour rappel la méthode Lieu.Generer existe). Aussi, on génère la position de la princesse, la position de départ. On peut ajouter un boss de fin sur la case si on le désire.

La dernière méthode « Deplacer » prend une direction en paramètre (point cardinaux). En fonction de cette direction, on va se déplacer sur la carte (c’est-à-dire dans le tableau). Attention aux bordures de carte !

-Implémenter la classe Partie.

La partie contient deux paramètres : la carte et le joueur.

La classe contient 6 méthodes :

Parametrer () :

Cette fonction effectue la création de la partie en fonction de paramètres choisis : difficulté et nom du joueur. La difficulté peut dicter le nombre de Pv du joueur ou la dimension de la carte.

Aussi, il faut générer la carte.

Lancer() :

Cette fonction effectue le déroulement de la partie (qui se déroule tant que le joueur est vivant et n’a pas sauvé la princesse.

Lors ce que l’on arrive sur une case, en fonction de leurs « caractéristiques », on va entrer dans les fonctions de gestion des cases.

Bagarre() : Combat avec un monstre

Soin() : Case de soin

DejaVisite() : cas ou une case a déjà été visitée. On va prévenir le joueur.

BagarreFinale() : Combat final contre le boss de fin

-Pour aller plus loin.

* ICombattant

Créer une interface ICombattant qui implémente une méthode Bagarre prenant en paramètre une cible de type Personnage

Faire dépendre les classes Monstre et Joueur de ICombattant et effectuer la refactorisation du code nécessaire.

* Affichage de la carte

Essayez de faire une représentation graphique de la carte dans la console. Il faut trouver un moyen de représenter l’intégralité de la carte et indiqué les cases sur lesquels le personnage est déjà passé.

Par exemple si c’est une case de soin afficher un +.