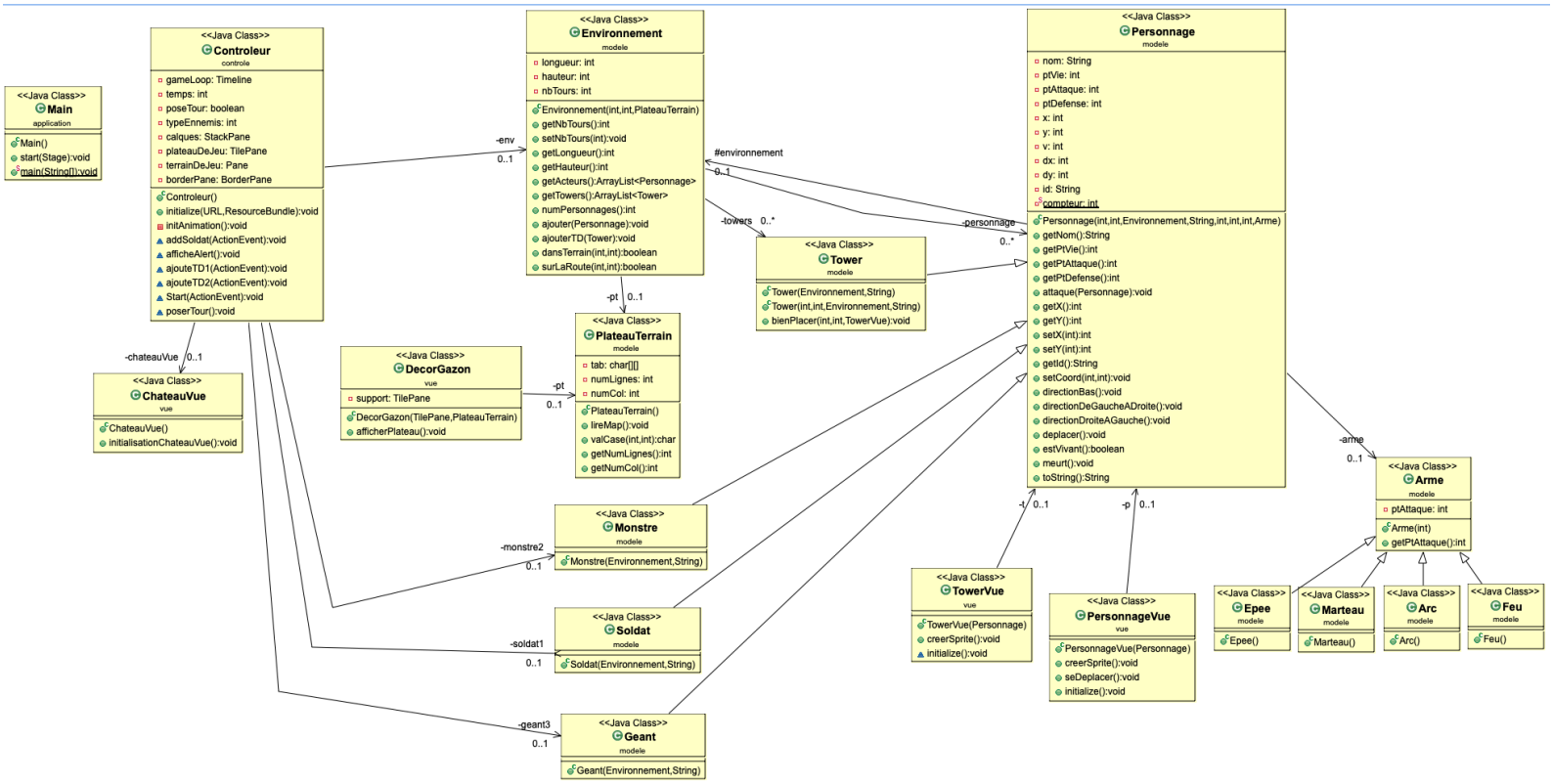
DOSSIER DU PROJET (19-20)

CHOUGUI Nadia
ALBARICO Marjorie

Architecture

Nous avons construit ce projet avec le modèle imposer MVC (modele, vue, contrôleur) on a ajouter un package image contenant toutes les images dont nous avons eu besoin et un package outils avec des méthodes qui parcours et affiche le tableau (terrain de jeu).



Modele	Controleur	Vue
Environnement.java	Controleur.java	ChateauVue.java
PlateauTerrain.java		DecorGazon.java
Personnage.java		PersonnageVue.java
Soldat.java		TowerVue.java
Monstre.java		map.csv
Geant.java		plateau.fxml
Epee.java		
Arme.java		
Feu.java		
Arc.java		

Détails : diagrammes de classe

La classe mère s'appelle Personnage. Nous avons 4 différents types de personnages (soldats monstre géants towers defence) ce sont tous des sous classe qui extends la classe mere. Nous avons fait le choix de construire nos personnages comme cela car c'est ce qui nous paraissaient le plus logique étant donné qu'il ont tous des attributs en commun . Les seules différence seront des valeurs passer en dur dans le super (exemple: les pas de vie ne sont pas les memes pour le soldat et le monstre). La classe Personnage elle a 12 attributs (nom, ptVie, ptAttaque, ptDefense, arme, x, y, v, dx, dy, Environnement, id). Elle a 18 méthodes (sans compter le constructeur) dont 10 guetteurs ou setters pour tout ce qui est ptVie, attaque, défense, id , coordonnées x, y ….)
Parmi les autres méthodes nous en avons 4 qui gère les déplacement, les directions des déplacement (directionBas() ,directionDroiteGauche(), directionGaucheDroite(), déplacer()

sont tous des méthodes void qui ne prennent rien en paramètres). Nous avons une méthodes boolean estVivant() vérifie si il reste des points de vie au personnage. Une autre méthode meurt():void qui fait mourrir nos personnage en mettant leurs PV a 0.

Programmation de la classe Environnement :

Dans la classe Environnement nous allons avoir un environnement dans lequel le personnage se trouve. Elle a 6 attributs (longueur, hauteur, nbTours, plateauTerrain et 2 arraylist une de personnage et une autre de Towers), et 11 méthodes sans compter le constructeur. La plupart sont des guetteurs et des setters (pour les X, Y, car il servent comme coordonnées pour placer le personnage) et pour le nombre de tours. Il y a une méthodes ajouter(personnage):void qui gère une arraylist de personnage pour les ajouter dans l'environnement. Ensuite nous avons coder la meme méthodes pour les towers défense (ajouterTD(Tower):void) qui a la même fonction aliter elle gère une arraylist de Towers et les ajoute dans l'environnement. Nous avons également coder deux méthodes booléenne qui s'occupe de prévenir si il y a un un dépassement de bornes de l'environnement.

La méthode surLaRoute()

C'est une méthode booléenne. La méthode surLaRoute (int, int) elle regarde si le personnage qu'on va placer ne sortira pas de la route (le chemin de pierre dans notre jeu). Cette méthode prendra les coordonnées de la tuiles a la position du perso (grace a la méthode valCase) et vérifia si a cette endroit on est sur du gazon (coder par 0 dans la map) ou sur le chemin (1 dans la map). Elle renverra false si on est sur des case 0 et true si on est sur 1.

La méthode dansTerrain()

C'est une méthode booléenne qui vérifie si on est bien dans les bornes de l'environnement (grâce au attributs longueur, hauteur). La méthode dansTerrain(int, int); renvoie false dès que le personnage sort de l'environnement sinon true.

Diagramme de séquence:

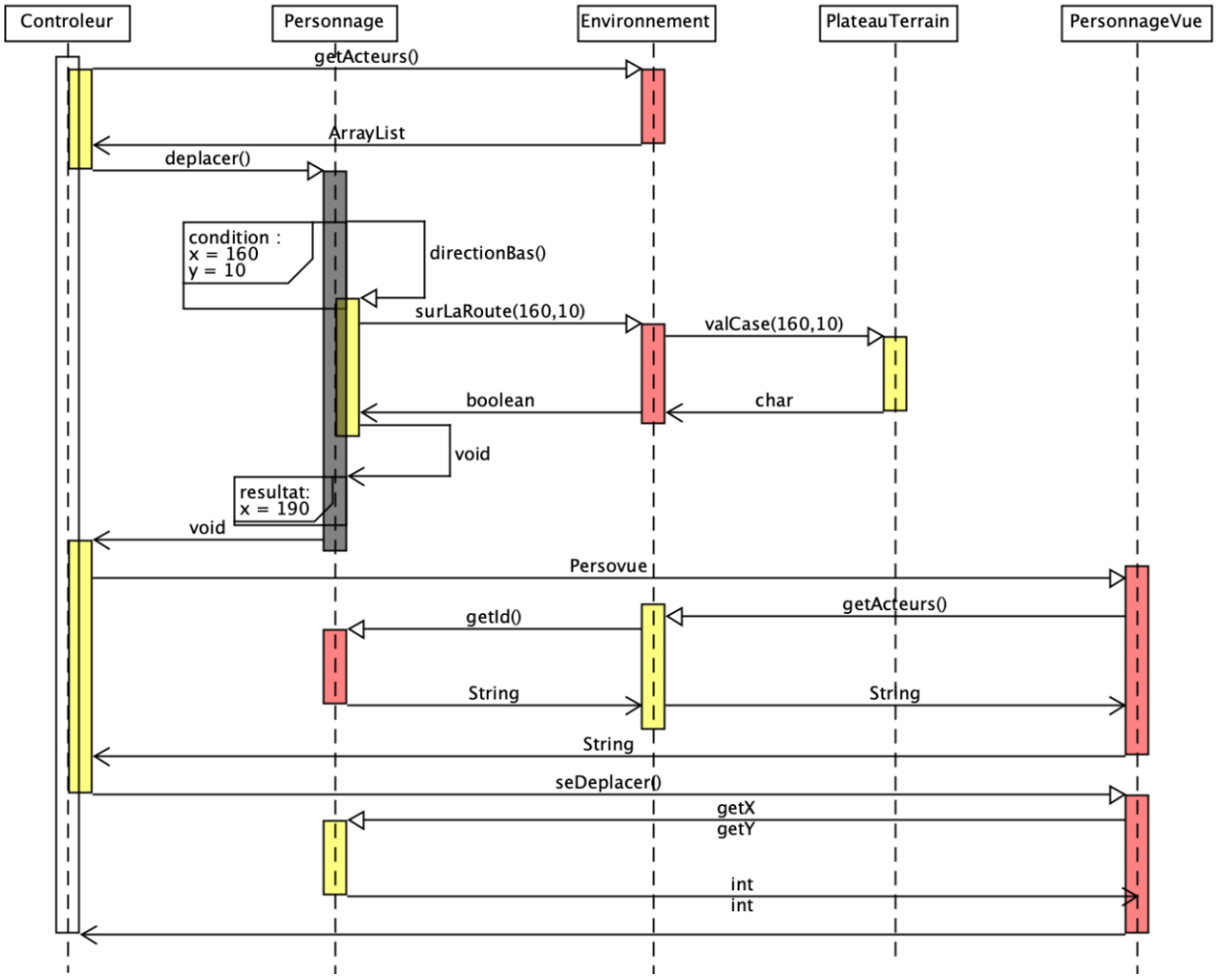


Diagramme de sequence

Pour déplacer la personnage. Dans la class controleur, il va prendre l'arraylist des personnage sur la class environnement et chaque personnage appel la méthode deplacer(). (cf. Exemple sur diagram) par exemple les coordonnées de cette personnage sont x=160 et y=10. Sur la méthode deplacer, il y a des conditions ifelse qui va verifier les coordonnées pour appeler un autre méthode, dans cet exemple, il appel la méthode directionBas qui va aussi appeler la méthode surLaRoute avec les coordonnées qui passeront par le paramètre, cette méthode retourne un boolean, true si la personnage est sur le chemin mais pour le vérifier, il va appeler la méthode valCase(*coordonnées*) de la classe PlateauTerrain qui retourne la valeur de la case (0 ou 1). Si il retourne 1, la méthode suLaRoute retourne true et va changer la valeur de x. Après, dans la controleur, il va appeler la class PersonnageVue pour prendre l'arraylist des personnages sur la class environnement et chaque personnage obtient un id, il va appeler la méthode getId de la classe personnage pour chercher quelle personnage doit gérer, dès qu'il a trouvé la classe controleur va appeler la méthode seDeplacer de la classe PersonnageVue qui va aussi appeler les méthodes getX et getY pour afficher les nouvelles coordonnées de la personnage.

Structures de données avancées :

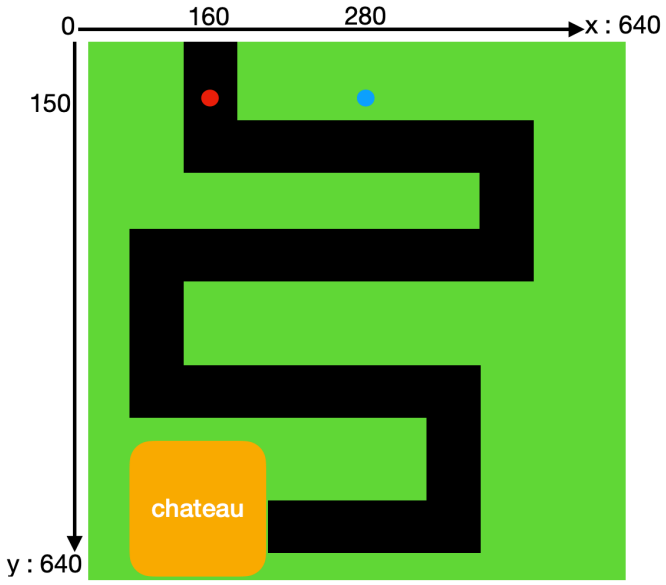
Exception :

Utilisation maîtrisée d'algorithmes intéressants :

Programmation de la classe Personnage

La position des personnages

Les personnages (les soldats, les towers, etc.) sont placées dans un même environnement. Les soldats, monstre et géants sont placées sur le chemin (cf. Circle rouge) et les TDs sont placées hors chemin (cf. Circle bleu). L'environnement a une largeur et une hauteur en pixel et qui fait référence à un map csv (taille: 20x20 et une case fait 32 pixels). La position du personnage dans cet environnement est donnée par les coordonnées en attribut x et y.



Le déplacement des personnage

La vitesse

L'attribut v représente la vitesse de la personnage, le nombre de pixels ajoutés ou enlevés lors d'un déplacement.

La direction

Les attributs dx et dy représentent la direction du déplacement en x et y.

La méthode void direction…()

On la code pour chaque direction. On attribuera a dx et dy un math.round des coordonnées x, y divisées par 32 (pour que ce soit en pixels). C'est a dire un nombre choisis aléatoirement, tout en regardant si on est toujours dans l'environnement et sur le chemin (avec la méthode surLaRoute()).
Ensuite selon la direction que notre personnage prends on ajoute au x (si c'est un déplacement horizontale exemple de gauche a droite) ou au y (si c'est un déplacement verticale exemple de haut en bas) la vitesse pour le faire avancer .

La méthode void déplacer()

Cette méthode est adapter au chemin de notre map. Elle suit le chemin c'est à dire qu' on prends les coordonnées x y (du début du chemin par exemple notre chemin commence a la 4 tuile et fini à la 6 alors notre x sera égal à 4 et 6 tandis que notre Y ira de 1 à 3) que l'on multiplie par 32 pour qu'elle soit adapter au tuiles de la map et ensuite pour donner la direction dans laquelle on ira on appelle les méthodes direction..().
On garde le meme principe dans des conditions else if pour representer toutes les situations possible sur notre chemin (virages, descentes…) on changera juste les coordonnées x,y selon ou on est sur la map et l'appel de méthode direction selon la direction qu'on voudra prendre.

Junits :

Nos tests Junit concernent les classes Environnement et plateauTerrain.

Documents pour Gestion de projet

1.Document utilisateur :

Le jeu se déroules dans un univers fantastique le but du jeu est de défendre la princesse dans le chateau des attaques des ennemis (monstres, soldats, géants) qui veulent la capturer. Pour cela on peut installer des towers défense tout au long du chemin qui mène vers le chateau. Il y'a différents types d'ennemis qui agissent différemment et il ya différents type de towers qui agissent elles aussi différemment.

Descriptions tours et ennemis:

Personnage	Arme	ptVie	ptAttaque	ptDefense	ptDArme
Soldat	Epee	80	15	15	5
Monstre	Feu	80	10	15	8
Geant	Marteau	80	10	15	10
Tower	Arc	100	20	20	5

Scoring :

Il y’a des vagues d’ennemis (5 par 5) Si les towers defense arrive a tuer les 5 ennemis sans qu’il n’arrive au chateau alors le joueur gagne sinon les ennemis capture la princesse et le jeu se termine