JAVA DOCUMENT

NGUYEN Xuan Huy p1419771

HUYNH Cong Lap p1419778

App.java:

Le méthode main() du jeu

Field.java:

La boucle du jeu

Renvoyer les entrées des utilisateurs

Afficher les personnages, les balles, les obstacles, les murs, les explosions

Character.java

La classe abstrait

L'implémentation de l'interface Hitable

Les méthodes:

- Charater: le constructeur
- display: l'affichage du personnage
- rotate: le traitement de la rotation de la flèche du personnage
- spriteAnimate: l'affichage de l'animation du personnage
- boost: la réalisation du méthode spriteAnimate
- getBall: le retour du vecteur du balle du personnage
- isHitted: la méthode surchargé, vérifier la collision entre le personnage et la balle
- getx: le retour de la coordonnée X du personnage
- gety: le retour de la coordonnée Y du personnage
- moveLeft: la méthode abstrait, le personnage se déplace à gauche
- moveRight: la méthode abstrait, le personnage se déplace à droit
- turnLeft: la méthode abstrait, tourner la flèche à gauche
- turnRight: la méthode abstrait, tourner la flèche à droit
- shoot: la méthode abstrait, tirer les balles
- autoMoveTurnShoot: la méthode abstrait, déplacer, tourner la flèche et tirer aléatoirement

- isDead: la méthode abstrait , marquer l'état du personnage: mourir

- dead: la méthode abstrait , afficher l'animation du personnage frappé

CharacterFactory.java

L'interface

Les méthodes:

- createCharacter: la méthode abstrait, créer le personnage (Player ou Computer)

Cloud.java

L'implémentation de l'interface Hitable

Les méthodes:

- Cloud: le constructeur

- isHitted: la méthode surchargé, vérifier la collision entre la nuage et la balle

- printXY: afficher la trace de la nuage

- move: exécuter le déplacement aléatoire de la nuage

Command.java

L'interface

Les méthodes

- execute: la méthode abstrait, exécuter l'action du personnage

Computer.java

La classe dérivée de la classe abstrait Character

Les méthodes:

- Computer: le constructeur

- moveLeft: la méthode surchargé, le Computer se déplace à gauche

- moveRight: la méthode surchargé, le Computer se déplace à droit

- turnLeft: la méthode surchargé, tourner la flèche à gauche

- turnRight: la méthode surchargé, tourner la flèche à droit

- shoot: la méthode surchargé, tirer les balles

- autoMoveTurnShoot: la méthode surchargé, déplacer, tourner la flèche et tirer aléatoirement

ComputerFactory.java

L'implémentation de l'interface CharacterFactory

Les méthodes:

- createCharacter: créer un objet Computer

InputHandler.java

La gestion des entrées de l'utilisateur

Les méthodes:

- InputHandler: le constructeur

- handleInput: créer les commandes (correspondant aux actions du personnage) comme moveleft, moveright, etc

MoveLeftCommand.java

L'implémentation de l'interface Command

Les méthodes:

- execute: exécuter la méthode moveLeft() du personnage.

MoveRightCommand.java

L'implémentation de l'interface Command

Les méthodes:

- execute: exécuter la méthode moveRight() du personnage

Player.java

La classe dérivée de la classe abstrait Character

Les méthodes:

- Player: le constructeur

- moveLeft: la méthode surchargé, le Player se déplace à gauche

- moveRight: la méthode surchargé, le Player se déplace à droit

- turnLeft: la méthode surchargé, tourner la flèche à gauche

- turnRight: la méthode surchargé, tourner la flèche à droit

- shoot: la méthode surchargé, tirer les balles

- getPlayer: retourner un objet Player

- isDead: la méthode abstrait , marquer l'état du Player: mourir

- dead: la méthode abstrait , afficher l'animation du Player frappé

PlayerFactory.java

L'implémentation de l'interface CharacterFactory

Les méthodes:

- createCharacter: créer un objet Player

Projectile.java

L'implémentation de l'interface Hitable

Les méthodes:

- Projectile: le constructeur

- display: afficher la balle

- fly: traiter les coordonnées de la balle quand elle est tiré

- setStop: stopper le movement de la balle

- getx: retourner la coordonnée X de la balle

- gety: retourner la coordonnée Y de la balle

- getWidthImage: retourner la largeur de l'image "ball.png"

- getHeightImage: retourner la hauteur de l'image "ball.png"

- isHitted: la méthode surchargé, vérifier la collision entre la balle et le personnage ou l'obstacle ou le murs

ShootCommand.java

L'implémentation de l'interface Command

Les méthodes:

- execute: exécuter la méthode shoot() du personnage

TurnLeftCommand.java

L'implémentation de l'interface Command

Les méthodes:

- execute: exécuter la méthode turnLeft() du personnage

TurnRightCommand.java

L'implémentation de l'interface Command

Les méthodes:

- execute: exécuter la méthode turnRight() du personnage

AutoCommand.java

L'implémentation de l'interface Command

Les méthodes:

- execute: exécuter la méthode autoMoveTurnShoot() du personnage

Hittable.java

L'interface

Les méthodes:

- isHitted: vérifier la collision

SpriteAnimation.java

Traiter l'animation de l'explosion

Les méthodes:

- SpriteAnimation: le constructeur, traiter les types de timeline de l'animation de l'explosion
- playContinuously: afficher l'animation de l'explosion
- setFrame: démarrer l'explosion
- removeImage: stopper l'explosion

Sprite.java

Traiter l'animation du personnage

Les méthodes:

- Sprite: le constructeur, traiter le timeline de l'animation du personnage
- playContinuously: afficher l'animation de la déplacement du personnage
- playShoot: afficher l'animation du tirs du personnage
- playDead: afficher l'animation lorsque le personnage est détruit

Wall.java

Les méthodes:

- Walls: le constructeur, créer les murs du jeu
- isHitted: vérifier la collision entre les murs et les balles
- getRect: retourner un vecteur de rectangle

Ball_Obstacle_CollisionTest.java

Tester la collision entre les balles et les obstacles

Les méthodes:

- init: créer un obstacle et un balle
- testNormalCollision: le cas de la collision
- testNotCollision: le cas de la non-collision

JavaFXThreadingRule.java

Créer un thread pour tester

Source: http://andrewtill.blogspot.fr/2012/10/junit-rule-for-javafx-controller-testing.html

Player_vs_Ball_CollisionTest.java

Tester la collision entre les balles et les Players

Les méthodes:

- init: créer un obstacle et un balle
- testNormalCollision: le cas de la collision
- testNotCollision: le cas de la non-collision

Wall_Ball_CollisionTest.java

Tester la collision entre les murs et les balles

Les méthodes:

- init: créer un mur et un balle
- testNormalCollision: le cas de la collision
- testNotCollision: le cas de la non-collision