Projet LU2IN002 - 2023-2024

Numéro du groupe de TD/TME : groupe 4

Nom : Mubiligi	*Nom :* Mnemoi
Prénom : Bettina	*Prénom :* Lina
N° étudiant : 21101302	*N° étudiant :* 21201897

Schéma UML des classes vision fournisseur (dessin "à la main" scanné ou photo acceptés)
voir fichier SchémaUML

Checklist des éléments utilisés:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une ArrayList	Agent4, Simulation
Classe avec membres et méthodes statiques	
Classe abstraite et méthode abstraite	Joyaux
Interface	Teleportable
Classe avec un constructeur par copie ou clone()	
Définition de classe étendant Exception	CoordonneesIncorrectesException CaseNonPleineException , DeplacementIncorrectException
Gestion des exceptions	Agent4, Simulation, GardienTeleportable
Utilisation du pattern singleton	

Copier / coller vos classes et interfaces à partir d'ici :

```
//JOYAUX

public abstract class Joyaux extends Contenu{
	private double prix;
	private String nom;

	public Joyaux(String nom, int qte){
		super(nom, qte);
```

```
            prix= Math.random()*((4000-1)+1);
            this.nom = nom;
      }

      public String toString(){
            return nom;
      }

      public double getPrix(){
            return prix;
      }
}

//DIAMANT
public class Diamant extends Joyaux{
      public Diamant(int qte){
            super("Diamant", qte);
      }

      public String toString(){
            return super.toString();
      }
}

//GARDIEN
public class Gardien extends Contenu{
      private double ptsVie;

      public Gardien(String nom, int qte){
            super(nom,qte);
            ptsVie = Math.random()*200;
      }

      public String toString(){
            return "Gardien";
      }
      public double getPtsVie(){
            return ptsVie;
      }
      public void setPtsVie(double t){
            ptsVie=t;
      }
}

//TELEPORTABLE
public interface Teleportable{
    public void seTeleporter();
}

//GARDIENTELEPORTABLE
public class GardienTeleportable extends Gardien implements Teleportable{
      private Grille grille;

    public GardienTeleportable(String nom, int qte, Grille grille){
        super(nom,qte);
            this.grille = grille;

    }
    public String toString(){
        return super.toString()+" Teleportable";
    }
    public void seTeleporter(){
```

```
        //l'agent se téléporte vers une case vide
        if(Math.random()<0.5){
            int fin = 0;
            for(int i = 0; i < grille.nbLignes; i++){
                for(int j = 0; j < grille.nbColonnes; j++){
                    try{
                    if(grille.caseEstVide(i, j)){
                        grille.setCase(i, j, this);
                        grille.videCase(i, j);
                        fin = 1;
                        System.out.println("le gardien présent dans la case
s'est téléporté avant que l'agent puisse le battre");
                        break;
                    }}catch (Exception e){System.out.println("Erreur:
"+e.getMessage());}
                }
                if (fin == 1) break;
            }
        }
        //affichage si le déplacement est impossible (grille pleine)
        else System.out.println("le gardien n'a pas pu à se téléporter, l'agent
l'a battu");

    }
}

//AGENT4
import java.util.ArrayList;

public class Agent4 {
      private int x;
      private int y;
      private ArrayList<Joyaux> sacJoyaux;
      private Grille grille;

      public Agent4(Grille grille){
            sacJoyaux = new ArrayList<Joyaux>();
            x = -1;
            y = -1;
            this.grille = grille;
      }

      public void seDeplacer(int xnew, int ynew){
            try {
                  if (grille.sontValides(xnew, ynew) == false) throw new
DeplacementIncorrectException("Déplacement impossible");
                  x = xnew;
                  y = ynew;
                  Contenu contenuCase = grille.getCase(x,y);

                  //l'agent ramasse le contenu de la case si c'est un joyau
                  if (contenuCase instanceof Joyaux){
                        sacJoyaux.add((Joyaux)contenuCase);
                        grille.videCase(x,y);
                        System.out.println("l'agent se déplace et ramasse un " +
contenuCase +" en position "+ x + ", " + y);
                  }
                  //l'agent perd le contenu de son sac si la case contenuCase
est un gardien
                  else if (contenuCase instanceof Gardien){
                        System.out.println("l'agent se déplace et perd le
contenu de son sac car il n'a pas de force");
                        sacJoyaux.clear();
```

```
                }
                else System.out.println("la case où est l'agent est vide");

            }catch (Exception e) {
                System.out.println("Erreur: "+e.getMessage());
            }
      }

      public  void seDeplacer(int xnew, int ynew, double f){
            try {
                        if (grille.sontValides(xnew, ynew) == false) throw new
DeplacementIncorrectException("Déplacement impossible");
                        x = xnew;
                        y = ynew;
                        //l'agent ramasse le contenu de la case si c'est un
joyau
                        Contenu contenuCase = grille.getCase(x,y);
                        if (contenuCase instanceof Joyaux){
                              sacJoyaux.add((Joyaux)contenuCase);
                              grille.videCase(x,y);
                              System.out.println("l'agent se déplace et ramasse
un " + contenuCase+" en position "+ x + ", " + y);
                        }

                        //simulation du combat face au gardien
                        else if (contenuCase instanceof Gardien){
                              //cas 1 : l'agent est plus fort que le gardien
                              if (((Gardien)contenuCase).getPtsVie() <= f){
                                    //le gardien tente de se téléporter s'il est
téléportable
                                    if(contenuCase instanceof
GardienTeleportable){

      ((GardienTeleportable)contenuCase).seTeleporter();
                                    }
                                    //le gardien est battu
                                    else{
                                          grille.videCase(x,y);
                                          System.out.println("l'agent se déplace
et bat le gardien grâce à sa force");
                                    }

                              }
                              //cas 2 : le gardien est plus fort que l'agent,
l'agent perd le contenu de son sac
                              else {
                                    sacJoyaux.clear();

      ((Gardien)contenuCase).setPtsVie(((Gardien)contenuCase).getPtsVie()-f);
                                    System.out.println("l'agent perd le contenu
de son sac malgré sa force");
                              }
                        }
                        else System.out.println("la case où est l'agent est
vide");
                }catch (Exception e) {
                        System.out.println("Erreur: "+e.getMessage());
                }
      }

      public double fortune(){
            double fortun = 0;
            for(Joyaux j : sacJoyaux){
```

```
                  fortun += j.getPrix();
            }
            return fortun;
      }

      public String contenuSac(){
            String s = "";
            for(Joyaux j : sacJoyaux){
                  s += j.toString()+", ";
            }
            return s;

      }
      public String toString(){
            return " Agent en position " +x+" , "+y+" avec fortune = "+fortune()
/* + " et "+contenuSac()+" joyaux."*/;
      }

}

//SIMULATION
import java.util.ArrayList;

public class Simulation{
    private Agent4 agent ;
    private Grille grille;
    private ArrayList<Contenu> tabContenu;

    public Simulation(Agent4 a, Grille g, ArrayList<Contenu> tab, int m){
        //initialisation des variables
        agent = a;
        grille = g;
        tabContenu=tab;

        //déclaration et initialisation de variables nécessaires
        int k = ((int) (Math.random()*grille.nbLignes));
        int l =((int) (Math.random()*grille.nbColonnes));
        int contenuMax = grille.nbLignes*grille.nbColonnes;

        //vérification de la validité de la valeur de m
        if(m>=contenuMax) m = contenuMax;
        if(m>=tab.size()) m = tab.size();
        System.out.println("ajout de  "+m+" contenus à la grille");

        //ajout du contenu à la grille
        try{
            for(int i = 0; i<m; i++){
                while (grille.caseEstVide(k,l) == false){
                    k = ((int) (Math.random()*grille.nbLignes));
                    l = ((int) (Math.random()*grille.nbColonnes));
                }
                grille.setCase(k,l,tab.get(i));
            }

       }catch (CoordonneesIncorrectesException e) {
            System.out.println("Erreur: "+e.getMessage());
        }
    }

    public String toString(){
        return grille.lesContenus() +"  "+ agent.toString();
    }
```

```
    public void lance(int nbEtapes){
            double force = Math.random()*(90)+10;
            int x = 0;
            int y = 0;
            int deplacementX;
            int deplacementY;

            for(int i=0; i<nbEtapes; i++){
                //calcul des coordonnées de la case vers laquelle l'agent se
déplace
                deplacementX = (int) (Math.random()*2)-1; // -1, 0, ou 1
                deplacementY = (int) (Math.random()*2)-1; // -1, 0, ou 1
                x += deplacementX;
                y += deplacementY;

                if (x>=grille.nbLignes) x = 0;
                if (x<0) x = grille.nbLignes -1;
                if (y>=grille.nbLignes) y = 0;
                if (y<0) y = grille.nbColonnes -1;

                //déplacement de l'agent
                if (Math.random() < 0.3){
                    System.out.print("Etape "+(i+1)+ " : " );
                    agent.seDeplacer(x, y, force);


                } else {
                    System.out.print("Etape "+(i+1)+ " : " );
                    agent.seDeplacer(x, y);
                }

            }


    }
}

//TESTSIMULATION

import java.util.ArrayList;

public class TestSimulation{
    public static void main(String[] args){
        //Simulation
        ArrayList<Contenu> c = new ArrayList<Contenu>();

        Grille g = new Grille(6,5);
        Agent4 agent =new Agent4(g);

        for(int i = 0; i < 5; i++){
            c.add(new Diamant(1));
        }

        for(int i = 0; i < 5; i++){
            c.add(new Gardien("Gardien", 1));
        }

        for(int i = 0; i < 20; i++){
            c.add(new GardienTeleportable("Gardien", 1, g));
        }

        Simulation s = new Simulation(agent, g, c, 30);
        System.out.println(s);
```

```
        s.lance(10);
        System.out.println(s);


    }

}

public class CaseNonPleineException extends Exception{

            public CaseNonPleineException(String message){
                        super(message);
            }

}

public class CoordonneesIncorrectesException extends Exception{

            public CoordonneesIncorrectesException(String message){
                        super(message);
            }

}

public class  DeplacementIncorrectException extends Exception{

            public  DeplacementIncorrectException(String message){
                        super(message);
            }

}
```

//Quelques logs :

Log1 :
ajout de 10 contenus à la grille
Grille de 4x4 cases: il y a 10 cases occupées
[Gardien Teleportable, Gardien, Gardien, Gardien, Gardien Teleportable, Diamant, Diamant, Diamant, Diamant, Diamant] Agent en position -1 , -1 avec fortune = 0.0
Etape 1 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 2 : l'agent se déplace et ramasse un Diamant en position 3, 3
Etape 3 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 4 : l'agent perd le contenu de son sac malgré sa force
Etape 5 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 6 : la case où est l'agent est vide
Etape 7 : la case où est l'agent est vide
Etape 8 : la case où est l'agent est vide
Etape 9 : la case où est l'agent est vide
Etape 10 : l'agent se déplace et ramasse un Diamant en position 3, 0
[Gardien Teleportable, Gardien, Gardien, Gardien, Gardien Teleportable, Diamant, Diamant, Diamant] Agent en position 3 , 0 avec fortune = 3511.0283253197495

Log2 :
ajout de 15 contenus à la grille
Grille de 5x4 cases: il y a 15 cases occupées.
[Diamant, Diamant, Diamant, Gardien, Diamant, Diamant, Diamant, Diamant, Gardien

Teleportable, Diamant, Gardien, Gardien, Gardien Teleportable, Diamant, Diamant] Agent en position -1 , -1 avec fortune = 0.0
Etape 1 : l'agent se déplace et ramasse un Diamant en position 0, 0
Etape 2 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 3 : la case où est l'agent est vide
Etape 4 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 5 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 6 : l'agent se déplace et ramasse un Diamant en position 2, 2
Etape 7 : l'agent se déplace et ramasse un Diamant en position 2, 1
Etape 8 : l'agent se déplace et ramasse un Diamant en position 1, 1
Etape 9 : l'agent se déplace et ramasse un Diamant en position 0, 1
Etape 10 : la case où est l'agent est vide
[Diamant, Gardien, Diamant, Gardien Teleportable, Diamant, Gardien, Gardien, Gardien Teleportable, Diamant, Diamant] Agent en position 0 , 1 avec fortune = 12875.717087557103

Log3 :
ajout de 30 contenus à la grille
[Gardien, Gardien Teleportable, Gardien Teleportable, Diamant, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien, Gardien Teleportable, Gardien, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Diamant, Diamant, Gardien Teleportable, Gardien Teleportable, Gardien, Gardien Teleportable, Diamant, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien, Diamant] Agent en position -1 , -1 avec fortune = 0.0
Etape 1 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 2 : l'agent perd le contenu de son sac malgré sa force
Etape 3 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 4 : l'agent se déplace et ramasse un Diamant en position 5, 4
Etape 5 : le gardien présent dans la case s'est téléporté avant que l'agent puisse le battre
Etape 6 : l'agent se déplace et perd le contenu de son sac car il n'a pas de force
Etape 7 : le gardien présent dans la case s'est téléporté avant que l'agent puisse le battre
Etape 8 : l'agent se déplace et ramasse un Diamant en position 3, 3
Etape 9 : la case où est l'agent est vide
Etape 10 : la case où est l'agent est vide
[Gardien, Gardien Teleportable, Gardien Teleportable, Diamant, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien, Gardien Teleportable, Gardien, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Diamant, Gardien Teleportable, Gardien Teleportable, Gardien, Gardien Teleportable, Diamant, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien Teleportable, Gardien] Agent en position 3 , 3 avec fortune = 2972.320895372682