Projet LU2IN002 - 2020-2021

Numéro du groupe de TD/TME : Groupe 2

Nom : QORAR	*Nom :SAHEL*	*Nom :MOHAMED*
Prénom :Rhizlène	*Prénom :Nada*	*Prénom :Djamila*
N° étudiant : 3801801	*N° étudiant :28600097*	*N° étudiant :29609129*

Thème choisi (en 2 lignes max.)

Nous avons choisi le thème du triathlon où trois athlètes (dont le joueur) participent à trois épreuves qui résultent en un classement.

Description des classes et de leur rôle dans le programme (2 lignes max par classe)

La classe Personne attribue aux athletes une vitesse initiale de 5 et contient la fonction getX qui repère où les athlètes sont situés.

La classe Athlete hérite de la classe Personne, définit tous les Athlètes, contient un clone d'Athlete, et développe toutes les fonctions attribuées aux Athlète durant l'épreuve.

La classe Spectateur hérite de la classe Personne, développe toutes les fontions attribuées aux Spectateurs qui se répercutent dur les Athlètes.

La classe Terrain crée un Terrain sur lequel se déroulent les épreuves.

La classe Piste hérite de la classe Terrain et définit la longueur et contient les fonctions nécessaires pour positionner les Spectateurs et déplacer les Athlètes.

La classe Piscine contient les fonctions nécessaires pour positionner les Spectateurs et déplacer les Athlètes. Elle diffère de PisteCourse et PisteCyclable.

La classe PisteCourse herite de la classe Piste.

La classe PisteCyclable hérite de la Classe Piste.

L'interface Action liste toutes les fonctions que peuvent utiliser les Personnes en action, ce qui peut se répercuter sur l'épreuve et le classement.

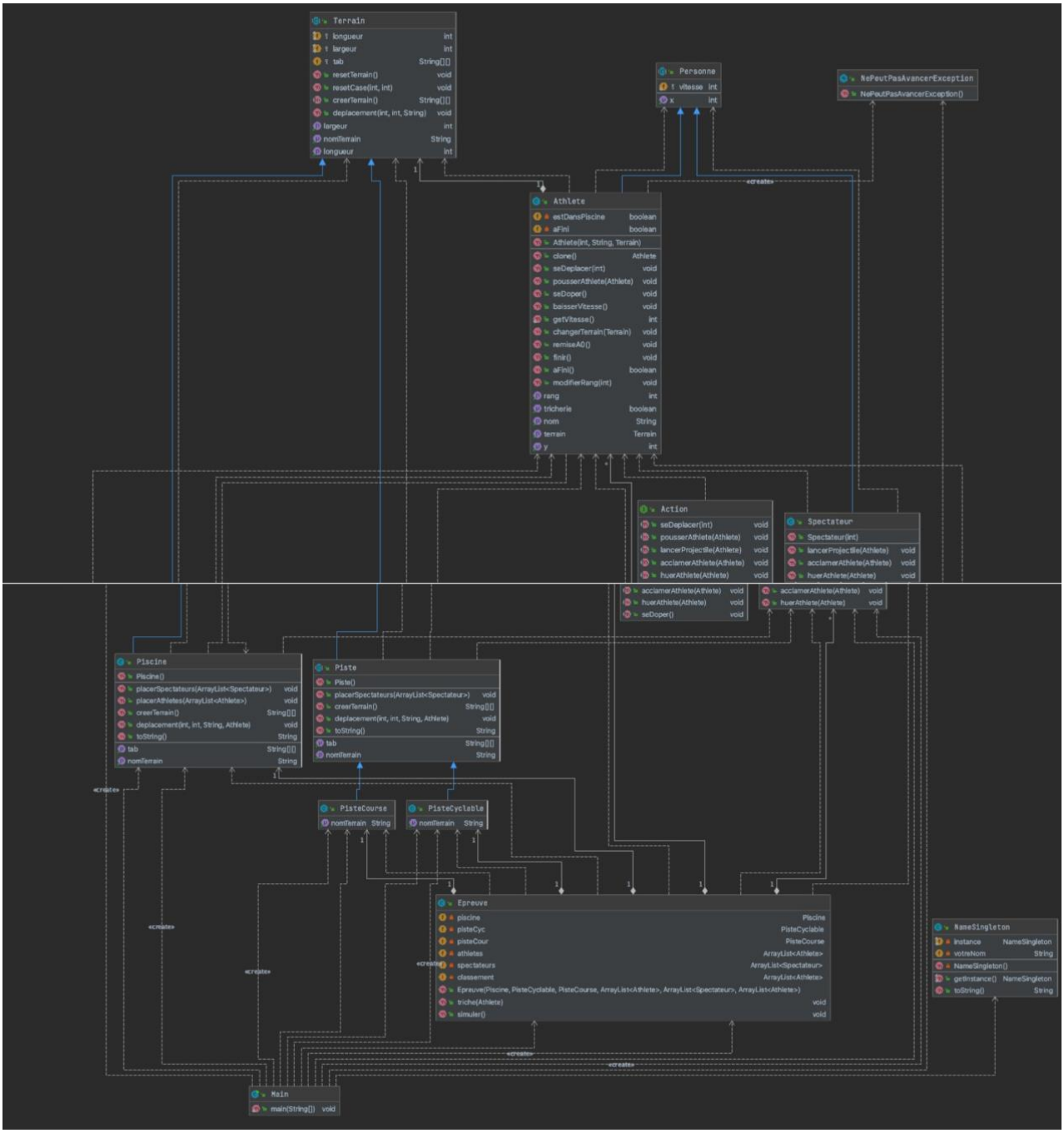
La classe Epreuve regroupe toutes les pistes, attribue chaque épreuve à son terrain et développe chaque action qui se répercute sur le jeu et le classement.

La classe NameSingleton demande au joueur le nom de son Athlète.

La classe NePeutPasAvancerException crée une exception qui passe l'athlète à l'étape suivante s'il est arrivé au bout du terrain.

La classe Main lance la simulation du triathlon, crée les spectateurs et athlètes qu'elle place ensuite sur le terrain selon les contraintes.

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



Checklist des contraintes prises en compte:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une ArrayList	Epreuve Piste Piscine
Classe avec membres et méthodes statiques	Personne NameSingleton Athlete
Classe abstraite et méthode abstraite	Terrain Piste Personne

Interface	Action
Classe avec un constructeur par copie ou clone()	Athlete
Définition de classe étendant Exception	NePeutPasAvancerException
Gestion des exceptions	Epreuve seDeplacer() deplacement()
Utilisation du pattern singleton	NameSingleton

Présentation de votre projet (max. 2 pages) : texte libre expliquant en quoi consiste votre projet.

Pour notre projet nous avons choisi le thème du triathlon.

Le triathlon est une discipline sportive constituée de trois épreuves d'endurance enchaînées : natation, cyclisme et course à pied. Sa forme moderne apparaît officiellement aux États-Unis en 1974 et se développe depuis dans le monde entier.

Dans notre simulation, le joueur incarne un athlète qui rivalise avec deux autres concurrents pour gagner le triathlon. Chacun commence avec la même vitesse qui peut être altérée par divers procédés. Un athlète peut, au hasard, tricher en se dopant, ce qui augment sa vitesse, pousser ou être poussé par un concurrent si ce dernier est proche de lui (cela détériore la vitesse du sportif poussé), il faut cependant ne pas tricher près d'un spectateur ou cela risque de se retourner contre l'athlète.

Les spectateurs peuvent également avoir un impact sur les sportifs: ils peuvent, au hasard, en acclamer un, ce qui augmente ses chances de réussite, en huer un autre ce qui réduit ses chances de réussite, ou lancer des projectiles à un athlète qui est surpris à tricher.

Il faut également noter que certaines actions dépendent du terrain où se trouvent les personnes.

Selon les évènements de la compétition, un classement est rendu à la fin de l'épreuve, montrant qui est le vainqueur du triathlon.

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

```java
public interface Action{
  public void seDeplacer(int vitesse) throws NePeutPasAvancerException;
  public void pousserAthlete(Athlete a);
  public void lancerProjectile(Athlete a);
  public void acclamerAthlete(Athlete a);
  public void huerAthlete(Athlete a);
  public void seDoper();
```

```
}


public abstract class Terrain {
  protected static final int longueur = 30;
  protected static final int largeur = 7;
  protected String[][] tab;


  public void resetTerrain() {
    for (int i = 0; i < longueur; i++) {
      for (int j = 1; j < largeur; j++) {
        tab[0][j] = "\n";
        if ((j == 1) || (j == largeur - 1)) {
          tab[i][j] = "_";
        }
      }
    }
  }


  public void resetCase(int x, int y) {
    tab[x][y] = " ";
  }


  public abstract String[][] creerTerrain();


  public void deplacement(int x, int y, String s) {
    tab[x][y] = s;
  }


  public int getLongueur() {
    return longueur;
  }


  public int getLargeur() {
    return largeur;
  }


  public String getNomTerrain() {
    return "terrain";
  }
}
```

```java
import java.util.ArrayList;


public class Piscine extends Terrain {


  public Piscine() {
    tab = new String[longueur][largeur];
    for (int i = 0; i < longueur; i++) {
      for (int j = 0; j < largeur; j++) {
        tab[i][j] = " ";
      }
    }
  }


  public void placerSpectateurs(ArrayList<Spectateur> ps) {
    for (int i = 0; i < ps.size(); i++) {
      tab[(ps.get(i)).getX()][0] = "S";
    }
  }


  public void placerAthletes(ArrayList<Athlete> pa) {
    for (int i = 0; i < pa.size(); i++) {
      tab[(pa.get(i)).getX()][(pa.get(i)).getY()] = (pa.get(i)).getNom();
    }
  }


  public String[][] creerTerrain() {
    for (int i = 0; i < longueur; i++) {
      for (int j = 1; j < largeur; j++) {
        tab[0][j] = "\n";
        if ((j == 1) || (j == largeur - 1)) {
          tab[i][j] = "_";
        }
      }
    }
    return tab;
  }


  public void deplacement(int x, int y, String s, Athlete a) {
    try {
      tab[x][y] = s;
      a.remiseA0();
    } catch (IndexOutOfBoundsException e) {
      System.out.println(e.getMessage());
```

```
    }
  }


  public String toString() {
    String res = "";
    for (int j = 0; j < largeur; j++) {
      for (int i = 0; i < longueur; i++) {
        res += tab[i][j];
      }
    }
    return res;
  }


  public String[][] getTab() {
    return tab;
  }


  public String getNomTerrain() {
    return "piscine";
  }
}


import java.util.ArrayList;


public abstract class Piste extends Terrain {


    public Piste() {
        tab = new String[longueur][largeur];
        for (int i = 0; i < longueur; i++) {
            for (int j = 0; j < largeur; j++) {
                tab[i][j] = " ";
            }
        }
    }


    public void placerSpectateurs(ArrayList<Spectateur> ps) {
        for (int i = 0; i < ps.size(); i++) {
            tab[(ps.get(i)).getX()][0] = "S";
        }
    }


    public String[][] creerTerrain() {
        for (int i = 0; i < longueur; i++) {
```

```
            for (int j = 1; j < largeur; j++) {
                tab[0][j] = "\n";
                if ((j == 1) || (j == largeur - 1)) {
                    tab[i][j] = "_";
                }
            }
        }
        return tab;
    }


    public void deplacement(int x, int y, String s, Athlete a) {
        try {
            tab[x][y] = s;
            a.remiseA0();
        } catch (IndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
        }
    }


    public String toString() {
        String res = "";
        for (int j = 0; j < largeur; j++) {
            for (int i = 0; i < longueur; i++) {
                res += tab[i][j];
            }
        }
        return res;
    }


    public String[][] getTab() {
        return tab;
    }


    abstract public String getNomTerrain();
}




public class PisteCyclable extends Piste {


  public String getNomTerrain() {
    return "pisteCyclable";
  }
}
```

```java
public class PisteCourse extends Piste {


  public String getNomTerrain() {
    return "pisteCourse";
  }
}



public abstract class Personne {
    protected int x;
    protected static int vitesse = 5;


    public int getX() {
        return x;
    }
}


public class Spectateur extends Personne {
    public Spectateur(int x) {
        this.x=x;
    }


    public void lancerProjectile(Athlete a) {
      if ((a.getTricherie())&&((a.x==x)||(a.x==x+1)||(a.x==x-1))) {
        a.baisserVitesse();
        System.out.print(a.getNom());
        System.out.println(" s'est pris un projectile !");
      }
    }


    public void acclamerAthlete(Athlete a){
      System.out.println(a.getNom()+" se fait acclamer !");
      double rand = Math.random();
      if (rand<0.4){
         a.vitesse++;
         System.out.println(a.getNom()+" est confiant !");
       }
    }
    public void huerAthlete(Athlete a){
      System.out.println(a.getNom()+" se fait huer !");
```

```
      double rand = Math.random();
      if (rand<0.4){
        a.vitesse--;
        System.out.println(a.getNom()+" n'est pas confiant...");
    }
  }
}


public class Athlete extends Personne {
  private int y;
  private String nom;
  private boolean estTricheur = false;
  private boolean estDansPiscine = true; // les athlètes commencent dans une
piscine
  private Terrain terr;
  private boolean aFini = false;
  private int rang;


  public Athlete(int y, String nom, Terrain terr) {
    this.y = y;
    this.nom = nom;
    this.terr = terr;
  }


  public Athlete clone() { // clone
    return new Athlete(y, nom, terr);
  }


  public void seDeplacer(int vitesse) throws NePeutPasAvancerException {
    if (vitesse>=0){
      if (x<terr.getLongueur()){
        terr.resetCase(x, y);
      } else {
        throw new NePeutPasAvancerException();
      }
      x+=vitesse;
      estTricheur = false;
        if (x<terr.getLongueur()){
          terr.deplacement(x, y, nom);
        } else {
          throw new NePeutPasAvancerException();
        }
    } else {
      System.out.println(nom+" a du mal à avancer !");
      if (Math.random()<0.8) seDoper();
    }
  }
```

```
  public void pousserAthlete(Athlete a) {
    if ((y == a.y + 1) || (y == a.y - 1) || !(terr instanceof Piscine)) {
      a.vitesse -= 1;
      estTricheur = true;
      System.out.print(nom);
      System.out.print(" a poussé ");
      System.out.print(a.nom);
      System.out.println(" !");
    } else {
      String message = String.format("%s ne peut pas pousser %s car %s\n", nom,
a.getNom(), terr.getNomTerrain());
      System.out.println(message);
    }
  }


  public void seDoper() {
    vitesse+=3;
    estTricheur = true;
    System.out.print(nom);
    System.out.println(" s'est dopé !");
  }


  public void baisserVitesse() {
    vitesse--;
  }


  public static int getVitesse() {
    return vitesse;
  }


  public int getY() {
    return y;
  }


  public String getNom() {
    return nom;
  }


  public Terrain getTerrain(){
    return terr;
  }
```

```
  public void changerTerrain(Terrain terr) {
    this.terr = terr;
  }

  public boolean getTricherie() {
    return estTricheur;
  }

  public void remiseA0(){
    x=0;
    vitesse=5;
  }

  public void finir(){
    aFini = true;
  }

  public boolean aFini(){
    return aFini;
  }

  public void modifierRang(int rang){
    this.rang = rang;
  }

  public int getRang(){
    return rang;
  }
}

import java.util.Scanner;



public class NameSingleton{

  private static final NameSingleton instance = new NameSingleton();
  private String votreNom;

  private NameSingleton(){
    // Athlete du joueur
```

```
    Scanner toScan = new Scanner(System.in);
    System.out.println("Entrez votre nom (un seul caractère):");
    votreNom = toScan.nextLine();
  }


  public static final NameSingleton getInstance(){
    return instance;
  }


  public String toString(){
    return votreNom;
  }



}
public class NePeutPasAvancerException extends Exception{
  public NePeutPasAvancerException(){
    super("L'athlète a atteint la fin de ce terrain.");
  }
}


import java.util.ArrayList;


public class Main {
  public static void main(String[] args) {


    ArrayList<Athlete> classement = new ArrayList<Athlete>();
    ArrayList<Athlete> athletes = new ArrayList<Athlete>();
    ArrayList<Spectateur> spectateurs = new ArrayList<Spectateur>();


    Piscine piscine = new Piscine();
    piscine.creerTerrain();


    PisteCyclable pisteCyc = new PisteCyclable();
    pisteCyc.creerTerrain();


    PisteCourse pisteCour = new PisteCourse();
    pisteCour.creerTerrain();


    // spectateurs placés au hasard sur la première ligne
    Spectateur s1 = new Spectateur((int) Math.random() * 10);
    Spectateur s2 = new Spectateur((int) Math.random() * 20 + 10);
```

```
    // on les ajoute dans la liste des spectateurs
    spectateurs.add(s1);
    spectateurs.add(s2);


    // on crée l'athlète de l'utilisateur et les deux autres
    NameSingleton votreNom = NameSingleton.getInstance();
    Athlete vous = new Athlete(3, votreNom.toString(), piscine);
    Athlete a2 = new Athlete(4, "B", piscine);
    Athlete a3 = new Athlete(5, "A", piscine);


    // on les ajoute dans la liste d'athlètes
    athletes.add(vous);
    athletes.add(a2);
    athletes.add(a3);


    // on place les athlètes et les spectateurs sur le premier terrain
    piscine.placerSpectateurs(spectateurs);
    piscine.placerAthletes(athletes);
    pisteCyc.placerSpectateurs(spectateurs);
    pisteCour.placerSpectateurs(spectateurs);


    Epreuve triathlon = new Epreuve(piscine, pisteCyc, pisteCour, athletes,
spectateurs, classement);
    triathlon.simuler();
    for (int i = 0; i < athletes.size(); i++) {
      Athlete athleteTemp = athletes.get(i);
      if (athleteTemp.aFini()) {
        int temp = athleteTemp.getRang() + 1;
        System.out.println(athleteTemp.getNom() + " est n°" + temp);
      }
    }
  }
}


import java.util.ArrayList;


public class Epreuve {
    private Piscine piscine;
    private PisteCyclable pisteCyc;
    private PisteCourse pisteCour;
    private ArrayList<Athlete> athletes;
    private ArrayList<Spectateur> spectateurs;
    private ArrayList<Athlete> classement;
```

```
    public Epreuve(Piscine piscine, PisteCyclable pisteCyc, PisteCourse pisteCour,
ArrayList<Athlete> athletes, ArrayList<Spectateur> spectateurs, ArrayList<Athlete>
classement) {
      this.piscine = piscine;
      this.pisteCyc = pisteCyc;
      this.pisteCour = pisteCour;
      this.athletes = athletes;
      this.spectateurs = spectateurs;
      this.classement = classement;
    }


    public void triche(Athlete a){
      if (Math.random()<0.5)
a.pousserAthlete(athletes.get((int)(Math.random()*2+1)));
      if (Math.random()<0.5) a.seDoper();


      if (Math.random()<0.5)
(spectateurs.get((int)(Math.random()*(spectateurs.size()))).lancerProjectile(athle
tes.get((int)(Math.random()*2)));
      if (Math.random()<0.5)
(spectateurs.get((int)(Math.random()*(spectateurs.size()))).acclamerAthlete(athlet
es.get((int)(Math.random()*2)));
      if (Math.random()<0.5)
(spectateurs.get((int)(Math.random()*(spectateurs.size()))).huerAthlete(athletes.g
et((int)(Math.random()*2)));
    }


    public void simuler(){
      int rang = 0;
      while (rang<=2){
        for (int i=0; i<athletes.size(); i++){
          Athlete athleteTemp = athletes.get(i);
          if (classement.contains(athleteTemp)) continue;
          //Exceptions
          if (athleteTemp.getTerrain() instanceof Piscine){
            try { // piscine
              athleteTemp.seDeplacer(athleteTemp.getVitesse());
              triche(athleteTemp);
            } catch (NePeutPasAvancerException e){
                  athleteTemp.changerTerrain(pisteCyc); // l'athlete passe dans
la piste cyclable
                  pisteCyc.deplacement(0, athleteTemp.getY(),
athleteTemp.getNom(), athleteTemp); // il se place au début de la piste cyclable
              }
            }
          else if (athleteTemp.getTerrain() instanceof PisteCyclable){
```

```
            try { // piste cyclable
              athleteTemp.seDeplacer(athleteTemp.getVitesse());
              triche(athleteTemp);
            } catch (NePeutPasAvancerException e2){
                athleteTemp.changerTerrain(pisteCour);
                pisteCour.deplacement(0, athleteTemp.getY(),
athleteTemp.getNom(), athleteTemp); // il se place au début de la piste de course
            }
          }
          else if (athleteTemp.getTerrain() instanceof PisteCourse){
            try { // piste course
              athleteTemp.seDeplacer(athleteTemp.getVitesse());
              triche(athleteTemp);
          } catch (NePeutPasAvancerException e3){
                athleteTemp.finir();
                String message = String.format("%s a fini le triathlon !\n",
athleteTemp.getNom());
                System.out.println(message);
                athleteTemp.modifierRang(rang);
                classement.add(athleteTemp);
                rang++;
            }
          }
          System.out.println("\n      PISCINE");
          piscine.resetTerrain();
          System.out.println(piscine);
          System.out.println("      PISTE CYCLABLE");
          pisteCyc.resetTerrain();
          System.out.println(pisteCyc);
          System.out.println("      PISTE DE COURSE");
          pisteCour.resetTerrain();
          System.out.println(pisteCour+"\n");


          try {
              Thread.sleep(1000);
          } catch (InterruptedException ex) {
              Thread.currentThread().interrupt();
          }
        }
      }
    }
  }
```