Projet LU2IN002 - 2020-2021

Numéro du groupe de TD/TME :

Nom : Sylla
Prénom : Amy
N° étudiant : 28618173

Thème choisi (en 2 lignes max.)
Le thème choisi est la gestion d un club de sport automobile et de cyclisme.

Description des classes et de leur rôle dans le programme (2 lignes max par classe)
-Classe Membre : c'est le point d'entrée du programme et décrit les members du club sportif. -Classe Sportif : étend la classe Membre et donne une description générale des sportifs membres du club. -Classe Cycliste : étend la classe Sportif et décrit les sportifs de la discipline "cyclisme". -Classe Automobiliste : étend la classe Sportif et décrit les sportifs de la discipline "course automobile". -Classe Velo : le vélo est le materiel de base du Cycliste. -Classe VoitureDeSport : la VoitureDeSport est le materiel de base de l'Automobiliste. -Classe Parking : Il s'agit de l'unique parking du club constitué de velos et de voitures de sport -Classe Fonctionnel : interface implementee par Velo et VoitureDeSport contenant les méthodes communes à ces deux classes. -Classe VehiculeEnBonEtatException : vérifie si une voiture de sport est en bon état. -Classe SportifMaladeException : permet d'interrompre l'exécution si le poids d'un sportif est irrégulier. -Classe SportInconnuException : elle permet d'interrompre le calcul si l'utilisateur propose un sport autre que la course automobile ou le cyclisme. -Classe GestionSportif : comme son nom l'indique elle gère les sportifs du club (grâce au mécanisme de capture d'exceptions). -Classe GestionVehicule : elle permet de gérer les voitures de sport (grâce au mécanisme de capture d'exceptions). -Classe OrganisationCourse : c'est une classe qui propose une simulation d'une course de voiture

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

Checklist des contraintes prises en compte:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une ArrayList	-Parking -GestionVehicule -OrganisationCourse
Classe avec membres et méthodes statiques	Parking
Classe abstraite et méthode abstraite	Membre
Interface	Fonctionnel
Classe avec un constructeur par copie ou clone()	Sportif
Définition de classe étendant Exception	-VehiculeEnBonEtatException - SportifMaladeException - SportInconnuException
Gestion des exceptions	-GestionSportif -GestionVehicule
Utilisation du pattern singleton	Parking

Présentation de votre projet (max. 2 pages) : texte libre expliquant en quoi consiste votre projet.
Le thème de ce projet final est le sport. J'ai opté pour la gestion d'un club de cyclisme et de course automobile. Mon projet s'articule autour de trois grands axes: une partie ressources humaines(classes Membre, Sportif, Automobiliste, Cycliste, SportifMaladeException,SportInconnuException), une partie ressources matérielles (classes Velo, VoitureDeSport, Fonctionnel,VehiculeEnBonEtatException,Parking) et enfin une dernière partie pour tester les fonctions écrites(classes GestionSportif, GestionVehicule, OrganisationCourse). I) Ressources humaines: La classe Membre est le "point d'entrée" de mon programme. Elle permet de repertorier comme son nom l'indique tous les membres du club sportif. La classe Sportif étend la classe Membre et

permet la gestion pure des sportifs du club. elle comporte deux constructeurs, dont un par copie; elle comporte aussi des méthodes pour fatiguer et entrainer les sportifs, leur permettant ainsi d'augmenter leur niveau de qualification. On trouve aussi dans la classe Sportif les accesseurs aux différents attributs de classe; par ailleurs j'ai profité de l'accesseur du poids pour contrôler l'état de santé des sportifs grâce à SportifMaladeException (qui se déclenche lorsque le poids du sportif est inférieur à 55kg). L'accesseur au sport pratiqué me permet aussi de vérifier que les sportifs du club sont bien des cyclistes et des automobilistes grâce à l'exception SportInconnuException(qui se déclenche si le sport pratiqué est différent de "cyclisme" et de "course automobile"). La classe Sportif est étendue par deux classes:
-la classe Cycliste: pour la gestion des cyclistes. Elle prend en charge la vitesse moyenne du cycliste (accesseur et setteur) ainsi que son identifiant.
-la classe Automobiliste: pour gérer les automobilistes. Elle permet d'attribuer un identifiant à chaque automobiliste.
NB: je tiens à préciser que j'ai fait en sorte qu'un cycliste ait sa propre vitesse moyenne et que la vitesse d'un automobiliste soit assimilée à celle de sa voiture de sport.

II) Ressources Matérielles:
Les principaux matériaux de mon club sportif sont les vélos(pour les cyclistes) et les voitures de sport (pour les automobilistes).Les classes Velo et VoitureDeSport implémentent l'interface Fonctionnel et sont munies de l'attribut 'etat' qui me permet d'effectuer un contrôle technique des véhicules du club, grâce à VehiculeEnBonEtatException. Elles comportent aussi les accesseurs et setteurs convenables pour les différents attributs de classe. Pour plus d'informations voir la documentation des classes). La classe Parking illustre une utilisation du pattern Singleton. Il s'agit ici de l'unique local où seront emplacés l'ensemble des vélos et voitures de sport du club.

III) Tests des fonctions
-GestionSportif: me permet de tester les fonctions écrites dans la partie "Ressources humaines"
-GestionVehicule: me permet de tester les fonctions écrites dans la partie Ressources matérielles
-OrganisationCourse: j'y ai effectué une petite simulation d'une course de voiture.

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

#Membre

```
package javaproject;


/**
 *
 * @author Amy Sylla
 *Cette classe permet de gérer les membres du club
 */


public abstract class Membre{

	protected String nom;
	protected int age;


	/**
	 * Constructeur
```

```java
	 * @param nom le nom du membre
	 * @param age l'age du membre
	 */
	public Membre(String nom, int age){
		this.nom = nom;
		this.age = age;
	}


	/**
	 * accesseur de nom
	 * @return le nom du membre
	 */
	public String getNom(){
		return nom;
	}

	/**
	 * accessseur de age
	 * @return l'age du membre
	 */
	public int getAge(){
		return age;
	}


	/**
	 *affichage du membre
	 * @return le nom et l'age du membre
	 */
	public String toString(){
		return "Nom: " + nom + ", age: " + age;
	}


	/**
	 * méthode pour entrainer les membres du club
	 */
	public abstract void sEntrainer();
}


#Sportif

package javaproject;


/**
 *
 * @author Amy Sylla
 *Cette classe permet de gérer les sportifs du club
 */


public class Sportif extends Membre{

	protected String sportPratique;
	protected double energie;
	protected int niveau;
	protected double poids;
```

```java
	/**
	 * Constructeur
	 * @param nom le nom du sportif
	 * @param age l'age du sportif
	 * @param sport le sport pratique par le sportif
	 * @param poids le poids du sportif
	 */
	public Sportif(String nom, int age, String sport, double poids){
		super(nom,age);
		sportPratique=sport;
		energie=100;  //energie initial d'un sportif
		niveau=0;    //niveau initial d'un sportif
		this.poids = poids;
	}


	/**
	 * Constructeur par copie
	 * @param s sportif à copier
	 */
	public Sportif(Sportif s){
		super(s.nom, s.age);
		sportPratique= s.sportPratique;
		energie= s.energie;
		niveau=s.niveau;
		poids=s.poids;
	}


	/**
	 * accesseur de poids
	 * @return le poids du sportif
	 * @throws SportifMaladeException
	 */
	public double getPoids() throws SportifMaladeException{
		if (poids < 55.0){
			throw new SportifMaladeException("Ce sportif devrait consulter un
medecin.");
		}
		return poids;
	}


	/**
	 * permet de passer à un niveau supérieur
	 */
	public void sEntrainer(){
		if (Math.random() > 0.3){  //dans 30% des cas les sportifs qui
s'entrainent passe au niveau superieur
			niveau++;
		}
	}


	/**
	 * permet de modifier l'energie
	 */
	public void fatiguer(){
		if (Math.random()<0.1){
			energie += -1;
		}
	}
```

```
	/**
	 * affichage du sportif
	 * @return le nom, le sport pratiqué et le niveau
	 */
	public String toString(){
		return "Nom: " + nom +", sport pratique: " + sportPratique + ", niveau: "
+ niveau;
	}



	/**
	 * accesseur de sportPratique
	 * @return le sport pratiqué par le sportif
	 * @throws SportInconnuException
	 */
	public String getSportPratique() throws SportInconnuException{
		if (sportPratique.compareTo("cyclisme")!=0 &&
sportPratique.compareTo("course automobile")!=0){
			throw new SportInconnuException("Sport inconnu.");
		}
		return sportPratique;
	}



	/**
	 * accesseur de energie
	 * @return
	 */
	public double getEnergie(){
		return energie;
	}



	/**
	 * accesseur de niveau
	 * @return le niveau du sportif
	 */
	public int getNiveau(){
		return niveau;
	}
}
```

#Cycliste

```
package javaproject;

/**
 *
 * @author Amy Sylla
 * Cette classe décrit les sportifs de la discipline cyclisme
 */


public class Cycliste extends Sportif {

	private static int identifiant = 0;
```

```
	private double vitesse_moyenne;


	/**
	 * Constructeur
	 * @param nom le nom du cycliste
	 * @param age l'age du cycliste
	 * @param vitesse_moyenne la vitesse du cycliste
	 * @param poids le poids du cycliste
	 */

	public Cycliste(String nom, int age, double vitesse_moyenne, double poids){
		super(nom,age,"cyclisme",poids);
		identifiant++;
		vitesse_moyenne = vitesse_moyenne;

	}



	/**
	 * permet de modifier la vitesse courante du cycliste
	 * @param v la nouvelle vitesse
	 */
	public void setVitesseMoyenne(double v){
		vitesse_moyenne += v;
	}



	/**
	 * accesseur de identifiant
	 * @return l'identifiant du cycliste
	 */
	public static int getIdentifiant(){
		return identifiant;
	}



	/**
	 * accesseur de vitesse_moyenne
	 * @return la vitesse moyenne du cycliste
	 */
	public double getVitesseMoyenne(){
		return vitesse_moyenne;
	}



	/**
	 * affichage du cycliste
	 * @returnl'identifiant, le nom, le sport pratiqué et le niveau
	 */
	public String toString(){
		return "Numero: " + identifiant + " " + super.toString();
	}

}
```

#Automobiliste

```java
package javaproject;

/**
 *
 * @author Amy Sylla
 *Cette classe décrit les sportifs de la discipline course de sport
 */



public class Automobiliste extends Sportif{

	private static int identifiant = 0;


	/**
	 * Constructeur
	 * @param nom  le nom de l'automobiliste
	 * @param age l'age de l'automobiliste
	 * @param poids le poids de l'automobiliste
	 */
	public Automobiliste(String nom, int age, double poids){
		super(nom,age,"course automobile",poids);
		identifiant++;
	}


	/**
	 * accesseur de identifiant
	 * @return l'identifiant de l'automobiliste
	 */
	public int getIdentifiant(){
		return identifiant;
	}


	/**
	 * affichage de l'automobiliste
	 * @return l'identifiant, le nom, le sport pratiqué et le niveau
	 */
	public String toString(){
		return "Numero: " + identifiant + " " + super.toString();
	}

}
```

#Velo

```java
package javaproject;

/**
 *
 * @author Amy Sylla
 *Cette classe permet de gerer les velos du club sportif
 */



public class Velo implements Fonctionnel{
```

```java
	private int etat;
	private String marque;
	private double distance_parcourue;


	/**
	 * Constructeur
	 * @param marque la marque du velo
	 */
	public Velo(String marque){
		this.marque = marque;
		etat = (int)(Math.random()%11);
		distance_parcourue = 0.0;
	}


	/**
	 * permet de controler l'etat des velos et voitures de sport
	 * @return true si le velo ou la voiture de sport est en bon etat, false sinon
	 * @throws VehiculeEnBonEtatException declenche l'exception si le velo n'est en bon etat
	 */
	public boolean enEtat() throws VehiculeEnBonEtatException{
		if (etat<7){
			throw new VehiculeEnBonEtatException("Ce velo doit Ãªtre remplace.");
		}
		return true;
	}


	/**
	 * permet de modifier distance_parcourue
	 * @param distance la nouvelle distance parcourue
	 */
	public void rouler(double distance){
		distance_parcourue +=  distance;
	}


	/**
	 * accesseeur de etat
	 * @return l'etat du velo
	 */
	public int getEtat(){
		return etat;
	}


	/**
	 * accesseur de marque
	 * @return la marque du velo
	 */
	public String getMarque(){
		return marque;
	}


	/**
	 * accesseur de distance_parcourue
	 * @return la distance parcourue par le velo
	 */
```

```
	public double getDistanceParcourue(){
		return distance_parcourue;
	}


	/**
	 * affichage du velo
	 * @return la marque du velo
	 */
	public String toString(){
		return "Velo de la marque " + marque +", etat:" + etat + ", distance
parcourue:" + distance_parcourue;
	}


}
```

#VoitureDeSport

```
package javaproject;

/**
 *
 * @author Amy Sylla
 *Cette classe permet de gerer les voitures du club sportif
 */

public class VoitureDeSport implements Fonctionnel{

	private int etat;
	private String marque;
	private double distance_parcourue;
	private double vitesse_moyenne;


	/**
	 * Constructeur
	 * @param marque la marque de la voiture
	 */
	public VoitureDeSport(String marque){
		this.marque = marque;
		etat = (int)(Math.random()*11);
		distance_parcourue = 0.0;
		vitesse_moyenne = Math.random()*100 + 100;
	}


	/**
	 * permet de controler l'etat de la voiture
	 * @return l'etat de la voiture
	 * @throws VehiculeEnBonEtatException declenche l'exception si la voiture n'est
en bon etat
	 */
	public boolean enEtat() throws VehiculeEnBonEtatException{
		if (etat<7){
			throw new VehiculeEnBonEtatException("Ce velo doit etre remplace.");
		}
		return true;
	}
```

```java
/**
 * permet de modifier la vitesse de la voiture
 * @param v la nouvelle vitesse de la voiture
 */
public void setVitesseMoyenne( double v){
	vitesse_moyenne += v;
}


/**
 * permet de modifier distance_parcourue
 * @param d la nouvelle distance
 */
public void setDistanceParcourue(double d){
	distance_parcourue = d;
}


/**
 * permet d'ajouter une certaine distance parcourue
 * @param distance la distance à ajouter
 */
public void rouler(double distance){
	distance_parcourue += distance;
}


/**
 * accesseur de etat
 * @return l'etat de la voiture
 */
public int getEtat(){
	return etat;
}


/**
 * accesseur de marque
 * @return la marque de la voiture
 */
public String getMarque(){
	return marque;
}

/**
 * accesseur de distance_parcourue
 * @return la distance parcourue par la voiture
 */
public double getDistanceParcourue(){
	return distance_parcourue;
}


/**
 * accesseur de vitesse_moyenne
 * @return la vitesse de la voiture
 */
public double getVitesseMoyenne(){
	return vitesse_moyenne;
}
```

```
	/**
	 * affichage de la voiture
	 * @return la marque, l'etat, la distance parcourue et la vitesse moyenne
	 */
	public String toString(){
		return "Voiture de sport de la marque " + marque + ", etat:" + etat + ",
distance_parcourue:" + distance_parcourue + " ,vitesse_moyenne:"+ vitesse_moyenne;
	}
}
```

#Fonctionnel

```
package javaproject;

/**
 *
 * @author Amy Sylla
 *Cette interface est implémentée par les classes Velo et VoitureDeSport
 */

public interface Fonctionnel{


	/**
	 * permet de controler l'etat des velos et voitures de sport
	 * @return true si le velo ou la voiture de sport est en bon etat, false sinon
	 * @throws VehiculeEnBonEtatException
	 */
	public boolean enEtat() throws VehiculeEnBonEtatException;

	/**
	 * permet de modifier distance_parcourue
	 * @param distance la nouvelle distance parcourue
	 */
	public void rouler(double distance);
}
```

#Parking

```
package javaproject;

/**
 *
 * @author Amy Sylla
 *Cette classe permet de gerer l'ensemble des velos et voituresdu club sportif
 */


public class Parking{

	private static Parking INSTANCE = null;
	private static Velo[] tabVelos = new Velo[50];
	private static VoitureDeSport[] tabVoitures = new VoitureDeSport[50];


	/**
	 * Constructeur
	 */
	private Parking(){
		for (int i=0; i< tabVelos.length; i++){
```

```
                tabVelos[i] = null;
            }

            for (int i=0; i< tabVoitures.length; i++){
                tabVoitures[i] = null;
            }
        }

        /**
         * permet de creer une instance unique de Parking
         */
        private synchronized static void creerInstance(){
            if (INSTANCE == null){
                INSTANCE = new Parking();
            }
        }


        /**
         * permet d'acceder à l'unique instance de Parking
         * @return l'unique instance de Parking
         */
        public static Parking getInstance(){
            if (INSTANCE == null){
                creerInstance();
            }
            return INSTANCE;
        }


        /**
         * permet d'ajouter si possible un velo dans le parking
         * @param v le velo à ajouter
         */
        public void ajouterVelo(Velo v){
            int i=0;
            while ( i < tabVelos.length && tabVelos[i]!= null){
                i++;
            }
            if (i == tabVelos.length - 1){
                System.out.println("L'espace velo est complet. Ajout impossible.");
            }else{
                tabVelos[i] = v;
            }
        }


        /**
         * permet d'ajouter si possible une voiture dans le parking
         * @param v la voiture à ajouter
         */
        public void ajouterVoiture(VoitureDeSport v){
            int i=0;
            while ( i < tabVoitures.length && tabVoitures[i]!= null){
                i++;
            }
            if (i == tabVoitures.length - 1){
                System.out.println("L'espace voiture est complet. Ajout
impossible.");
            }else{
                tabVoitures[i] = v;
            }
```

```
	}

}
```

#SportifMaladeException

```
package javaproject;

/**
 *
 * @author Amy Sylla
 * Cette classe permet de gerer l'etat de sante des sportifs du club
 */


public class SportifMaladeException extends Exception{


	/**
	 * Constructeur
	 * @param s le message à afficher
	 */
	public SportifMaladeException(String s){
		super(s);
	}


	/**
	 * affichage du message
	 */
	public String toString(){
		return "Ce sportif doit consulter un medecin.";
	}

}
```

#VehiculeEnBonEtatException

```
package javaproject;

/**
 *
 * @author Amy Sylla
 *Cette classe permet de verifier l'etat des velos et voitures du club
 */


public class VehiculeEnBonEtatException extends Exception{


	/**
	 * Constructeur
	 * @param s le message à afficher
	 */
	public VehiculeEnBonEtatException( String s){
		super(s);
	}
```

```
	/**
	 * affichage du message
	 */
	public String toString(){
		return "Ce vehicule doit etre remplace.";
	}


}
```

#SportInconnuException

```
package javaproject;

/**
 *
 * @author Amy Sylla
 * Cette classe permet de verifier les disciplines sportives du club
 *
 */



public class SportInconnuException extends Exception{


	/**
	 * Constructeur
	 * @param s le message à afficher
	 */
	public SportInconnuException(String s){
		super(s);
	}


	/**
	 * affichage du message
	 */
	public String toString(){
		return "Ce club ne gere pas ce sport.";
	}
}
```

#GestionSportif

```
package javaproject;

//cette classe va nous permettre de tester les differentes fonctionse©crites et de
simuler l'organisation du club


public class GestionSportif{

	public static void main (){

		//Gestion des sportifs

		Automobiliste s1 = new Automobiliste("Bryant",36,78.6);
```

```
		Cycliste s2 = new Cycliste("anna", 24, 50.0, 90.0);
		Sportif s3 = new Sportif(s2);
		Automobiliste s4 = new Automobiliste("julien",25, 58.0);


		try{

			//verification du sport pratique
			System.out.println("Verification du sport pratique:");
			System.out.println(s1.getSportPratique());
			System.out.println(s2.getSportPratique());
			System.out.println(s3.getSportPratique());
			System.out.println(s4.getSportPratique());

			//verification de l'etat de sante
			System.out.println("verification de l'etat de sante:");
			System.out.println(s1.getPoids());
			System.out.println(s2.getPoids());
			System.out.println(s3.getPoids());
			System.out.println(s4.getPoids());

		}catch(SportInconnuException e){
			System.out.println(e.toString());

		}catch(SportifMaladeException e){
			System.out.println(e.toString());

		}finally{
			for (int i = 0; i<10; i++){
				s1.sEntrainer();
				s2.sEntrainer();
				s3.sEntrainer();
				s4.sEntrainer();

				s1.fatiguer();
				s2.fatiguer();
				s3.fatiguer();
				s4.fatiguer();
			}

			System.out.println(s1);
			System.out.println(s2);
			System.out.println(s3);
			System.out.println(s4);
		}

	}
}
```

#GestionVehicule

```
package javaproject;

import java.util.ArrayList;

public class GestionVehicule{

	public static void main (String[] args){

		Parking p;
```

```
		int capaciteVelos;
		int capaciteVoitures;
		ArrayList<Object> tabVelos = new ArrayList<Object>();
		ArrayList<Object> tabVoitures = new ArrayList<Object>();

			// on va supposera que les entrees en ligne de commande sont
correctes

			p = Parking.getInstance();
			capaciteVelos = Integer.parseInt(args[0]);
			capaciteVoitures = Integer.parseInt(args[1]);


			//initialisation du tableau de velos
			for (int i=2;i<capaciteVelos+2; i++){
				tabVelos.add(new Velo(args[i]));
			}

			//initialisation du tableau de voitures
			for (int j=2; j<capaciteVoitures+2; j++){
				tabVoitures.add(new VoitureDeSport(args[j]));
			}

			//entretien technique des velos
			try{
				for (Object o: tabVelos){
				System.out.println(((Velo)o).enEtat());
				}

				for (Object o: tabVoitures){
				System.out.println(((VoitureDeSport)o).enEtat());
				}

			}catch(VehiculeEnBonEtatException v){
				System.out.println(v.toString());

			}finally{
				for (Object o: tabVelos){
				System.out.println((Velo)o);
				}

				for (Object o: tabVoitures){
				System.out.println((VoitureDeSport)o);
				}
			}
	}
}
```

#OrganisationCourse

```
package javaproject;

public class OrganisationCourse{

	public static void main(String[] args){

		double distanceAparcourir = 1000.0;
		double distFstStep =0.0;  //distance parcourue Ã  la premiere etape
		double tempsMis = 0.0;
```

```
            //l'utilisateur donne en ligne de commande le nombre de voitures en course
et la marque unique
            //on suppose que les entrees sont correctes
            VoitureDeSport[] tab = new VoitureDeSport[Integer.parseInt(args[0])];

            for (int i=0; i<tab.length; i++){
                  tab[i]= new VoitureDeSport(args[1]);
            }

            //presentation des voitures
            System.out.println("Les voitures en course sont:");
            try{
                  for (int i=0; i<tab.length; i++){
                  System.out.println(tab[i]);
                  Thread.sleep(3000);
                  }
            }catch(InterruptedException i){
                  System.out.println("Execution interrompue");
            }

            //lancement de la course
            //chaque voiture parcoure une certaine distance avec sa vitesse initiale
puis le reste avec une autre vitesse

            System.out.println("Les voitures sont lancees.");
            for (int i=0; i<tab.length; i++){
                  distFstStep = Math.random()*1000;
                  tab[i].rouler(distFstStep);
                  tempsMis = (tab[i].getDistanceParcourue()) *
(tab[i].getVitesseMoyenne());
                  tab[i].setVitesseMoyenne(Math.random()*50);
                  tempsMis += (distanceAparcourir - distFstStep)*
(tab[i].getVitesseMoyenne());
                  tab[i].setDistanceParcourue(distanceAparcourir);
                  System.out.println(tab[i] + " a mis "+ tempsMis + " secondes");
            }


      }
}
```