Projet LU2IN002 - 2020-2021

Numéro du groupe de TD/TME : 6

Nom : Aboudaoud

Prénom : Yousri Kacem

N° étudiant : 28712992

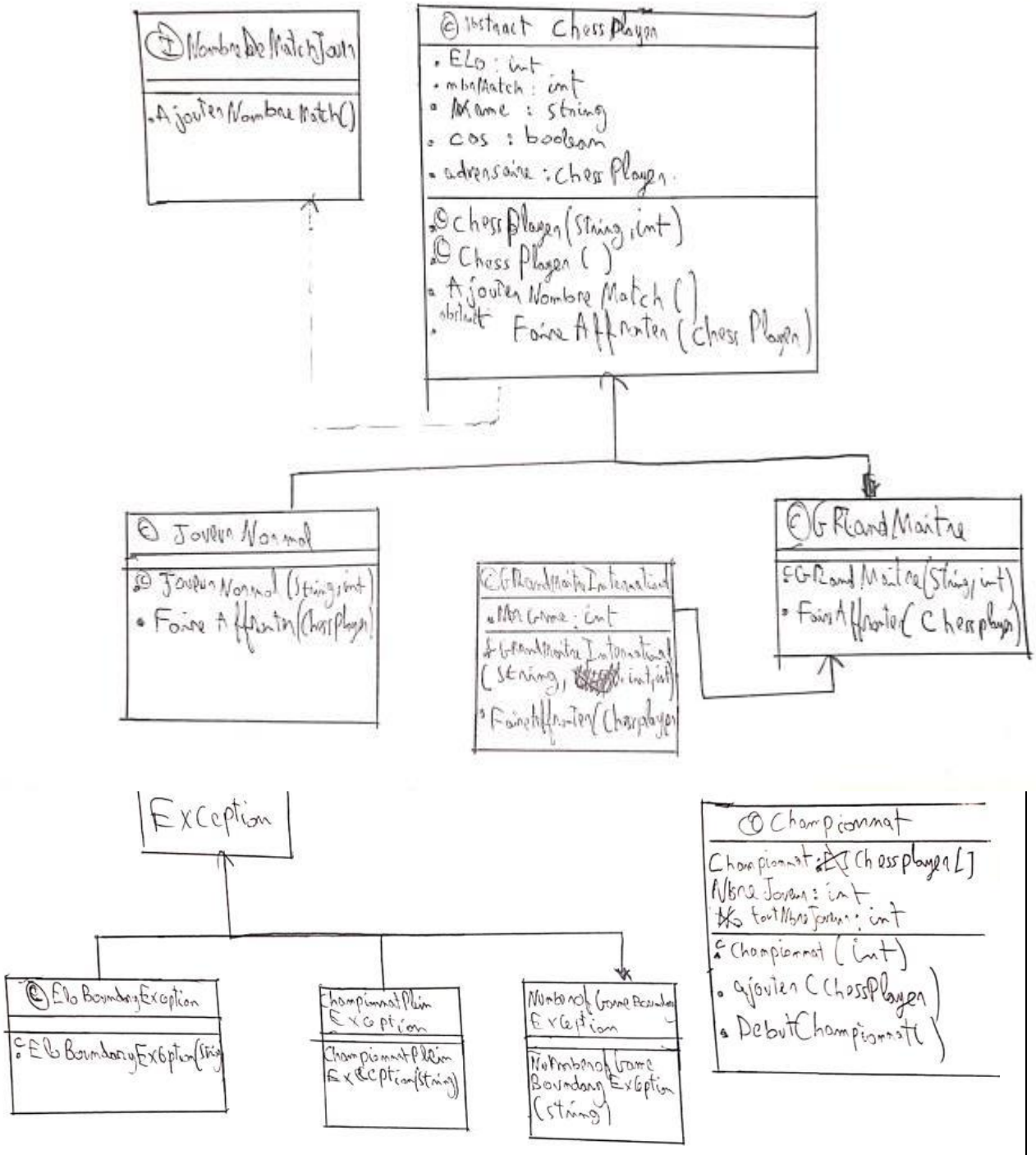
Thème choisi (en 2 lignes max.)

Organisation d'un tournoi de jeux d'échecs selon le classement ELO

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

- ChessPlayer : Classe qui représente les joueurs d'échecs.
- GrandMaitre : Classe fille de ChessPlayer dont les objets ont un ELO obligatoirement supérieur à 2500 et qui fait affronter plusieurs joueurs.
- GrandMaitreInternational : Classe fille de GrandMaitre dont les objets ont un nombre de match qui doit être supérieur à 9.
- JoueurNormal : Classe fille de ChessPlayer dont les objets ont un ELO obligatoirement supérieur à 1000 et inférieur à 2500.
- Championnat : Classe qui gère un tableau de joueur d'échecs.
- TestChampionnat : Classe qui teste nos autres classes.
- Arbitre : Classe du singleton.
- ChampionnatPleinException : Classe étendant Exception.
- EloBoundaryException : Classe étendant Exception sert pour lever l'exception de dépasser les limites des ELO
- NumberOfGamesBoundaryException : Classe étendant Exception sert pour lever l'exception de sortir de la limite de 9 matches dans GrandMaitreInternational.

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


NombreDeMatchJouer
+ AjouterNombreMath()
Abstract ChessPlayer
- ELo : int
- nbMatch : int
- Name : string
- cas : boolean
- adresseIne : ChessPlayer
+ ChessPlayer(String, int)
+ ChessPlayer()
+ AjouterNombreMatch()
abstract FaireAffronter(ChessPlayer)
JoueurNormal
+ JoueurNormal(String, int)
+ FaireAffronter(ChessPlayer)
GMRandMaitreInternational
- NbreGrosse : int
+ GMRandMaitreInternational(String, int)
+ FaireAffronter(ChessPlayer)
GMRandMaitre
+ GMRandMaitre(String, int)
+ FaireAffronter(ChessPlayer)
Exception
EloBoundaryException
+ EloBoundaryException(String)
ChampionnatPleinException
ChampionnatPleinException(String)
NombreGrosseBoundaryException
NombreGrosseBoundaryException(String)
Championnat
Championnat : ChessPlayer[]
NbreJoueur : int
tourNbreJoueur : int
+ Championnat(int)
+ ajouter(ChessPlayer)
+ DebutChampionnat()

Checklist des contraintes prises en compte:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une ArrayList	Championnat
Classe avec membres et méthodes statiques	Championnat
Classe abstraite et méthode abstraite	ChessPlayer, FaireAffronter
Interface	NombreDeMatchJouer
Classe avec un constructeur par copie ou clone()	ChessPlayer
Définition de classe étendant Exception	ChampionnatPleinException, NumberOfGamesBoundaryException, EloBoundaryException
Gestion des exceptions	ChessPlayer, TestChampionnat, Championnat
Utilisation du pattern singleton	Arbitre

Présentation de votre projet (max. 2 pages) : texte libre expliquant en quoi consiste votre projet.
Le projet consiste en une organisation d'un tournoi de jeux d'échecs. Chaque joueur possède un classement ELO représente son niveau de jeu, dans notre programme un GrandMaitre doit avoir un Elo supérieur à 2500 ; ce type de joueur peut affronter un autre joueur ce qui rends ce dernier son « adversaire ». En ce qui concerne ce « affrontement » c'est la même chose pour les autres classes, JoueurNormal diffère de GrandMaitre sur la condition de l'ELO, GrandMaitreInternational ajoute une autre condition de construction celle des nombres de matchs. Ainsi Chaque Championnat consiste à organiser l'affrontement de divers joueurs.

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

```
public class Arbitre{
    /** Constructeur privé */
    private Arbitre()
    {}


    private static Arbitre INSTANCE = null;

    /** Point d'accès pour l'instance unique du singleton */
    public static Arbitre getInstance()
    {
        if (INSTANCE == null)
```

```
        {   INSTANCE = new Arbitre();
        }
        return INSTANCE;
    }
}
public class Championnat  {

      ChessPlayer[] championnat;  //cette classe gére un tableau de joueur d'échecs
      public int NbreJoueur; // nbre de joueur du actuel tournoi
      public static int ToutNbreJoueur; //nbre de joueur de tout les championnats


      public Championnat(int NbrJoueur) {
            championnat = new ChessPlayer[NbrJoueur];
      }

      public void ajouter(ChessPlayer P) throws
ChampionnatPleinException,EloBoundaryException {
            if (P.ELO <1000)
                  throw new EloBoundaryException("Ce joueur n'a pas sa place dans ce
championnat");
            if (NbreJoueur <championnat.length) {
                  championnat[NbreJoueur]=P;
                  NbreJoueur++;
                  ToutNbreJoueur++;
            }
            else {
                  throw new ChampionnatPleinException("Le championnat ne peut plus
accepter plus de joueur");
            }



}

      public static void DebutChampionnat() {         // message de départ
            System.out.println("Début de l'organisation du championnat");

            {

            }
      }


}

public class ChampionnatPleinException extends Exception {


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

}

public abstract class ChessPlayer  implements NombreDeMatchJouer{

      protected int ELO;  // le fameux classement ELO
      protected int nbrMatch; //nombre de match joués par un joueur
      protected String Name;      //le nom du joueur
      protected boolean cas;  //le cas représente le fait que le joueur ait un
```

```java
adversaire ou non
	protected ChessPlayer adversaire;


	public ChessPlayer(String name, int elo) throws EloBoundaryException {
		if (elo<=0)
			throw new EloBoundaryException ("Constructeur : Elo inexistant ");
//Un ELO ne peut être inférieur ou égal à zero
		this.ELO=elo;
		this.Name=name;
	}
	public ChessPlayer() { //constructeur d'un joueur lambda
		this.ELO=1500;
		this.Name="joueur lambda";
	}

	@Override
	public void AjouterNombreMatch() {
		this.nbrMatch++;
	}

	public abstract void FaireAffronter(ChessPlayer p);


}
public class EloBoundaryException extends Exception {

	public EloBoundaryException(String message) {
		super(message);

	}
}

public class GrandMaitre extends ChessPlayer  {




	public GrandMaitre(String name, int elo) throws EloBoundaryException {
		super(name, elo);

		if (elo<2500)
			throw new EloBoundaryException ("Pas un Elo d'un grand maitre");


	}

	public void FaireAffronter(ChessPlayer p) {  // méthode qui fait affronter deux
joueurs par le biais de la vérification de  l'atribut cas
		if(this.cas==true|| p.cas==true) {
			System.out.println("Ces deux joueurs "+this.Name+" et "+p.Name+ "
ne peuvent s'affronter");
		}
		else {
			this.cas=true ;  // Nous changeons la valeur de l'attribut cas
			p.cas=true;
			this.adversaire=p;
			p.AjouterNombreMatch();
			this.AjouterNombreMatch();
			System.out.println(this.Name+" , vas donc affronter "+p.Name);
		}
```

```
	}


}

public class GrandMaitreInternational extends GrandMaitre{

	private final int NbrGame; // cet attribut représente le nbr de match avant le championnat

	public GrandMaitreInternational(String name, int elo,int NbrGame) throws EloBoundaryException,NumberOfGamesBoundaryException {
		super(name, elo);
		this.NbrGame=NbrGame;
		if (elo<2500)
			throw new EloBoundaryException ("Pas un Elo d'un grand maitre");
		if (NbrGame<9)
			throw new NumberOfGamesBoundaryException("Pas suffisamment de match pour être un Grand Maitre International ");
	}



	public void FaireAffronter(ChessPlayer p) {
		if(this.cas==true|| p.cas==true) {
			System.out.println("Ces deux joueurs "+this.Name+" et "+p.Name+ " ne peuvent s'affronter");
		}
		else {
			this.cas=true ;
			p.cas=true;
			System.out.println(this.Name+" , vas donc affronter "+p.Name);
		}
	}



}

public class JoueurNormal extends ChessPlayer {

	public JoueurNormal(String name, int elo) throws EloBoundaryException {
		super(name, elo);

		if (elo>2500||elo<1000)
			throw new EloBoundaryException ("Ce joueur doit être normal ! ");
	}




		public void FaireAffronter(ChessPlayer p) {
			if(this.cas==true|| p.cas==true) {
				System.out.println("Ces deux joueurs "+this.Name+" et "+p.Name+ " ne peuvent s'affronter");
			}
			else {
				this.cas=true ;
				p.cas=true;
				System.out.println(this.Name+" , vas donc affronter "+p.Name);
```

```
                }
            }
        }
public interface NombreDeMatchJouer {

        public void AjouterNombreMatch();

}
public class NumberOfGamesBoundaryException extends Exception{

        public NumberOfGamesBoundaryException(String message) {
            super(message);

        }
}

public class TestChampionnat {

        public static void main(String[] args) {
            Championnat.DebutChampionnat();

            Championnat Ch=new Championnat (5);

            try {
                ChessPlayer p= new GrandMaitre("Nakamura",2880);
                ChessPlayer p2= new GrandMaitre("Carlsen",2780);
                ChessPlayer p3= new GrandMaitre("Kasparov",2680);

                Ch.ajouter(p3);
                Ch.ajouter(p2);
                Ch.ajouter(p);

                p.FaireAffronter(p2);


                System.out.println(" "+Championnat.ToutNbreJoueur);

            } catch (EloBoundaryException e) {
                System.out.println(" "+e.toString());

            } catch (ChampionnatPleinException e) {
                System.out.println(" "+e.toString());
            }


        }

}
```