Méthodes Agiles – PROJET

Version : 1.4

LIGUE 1 CONFORAMA

Binôme :

Khellaf DJEBBAR

Massinissa BOULKARIA

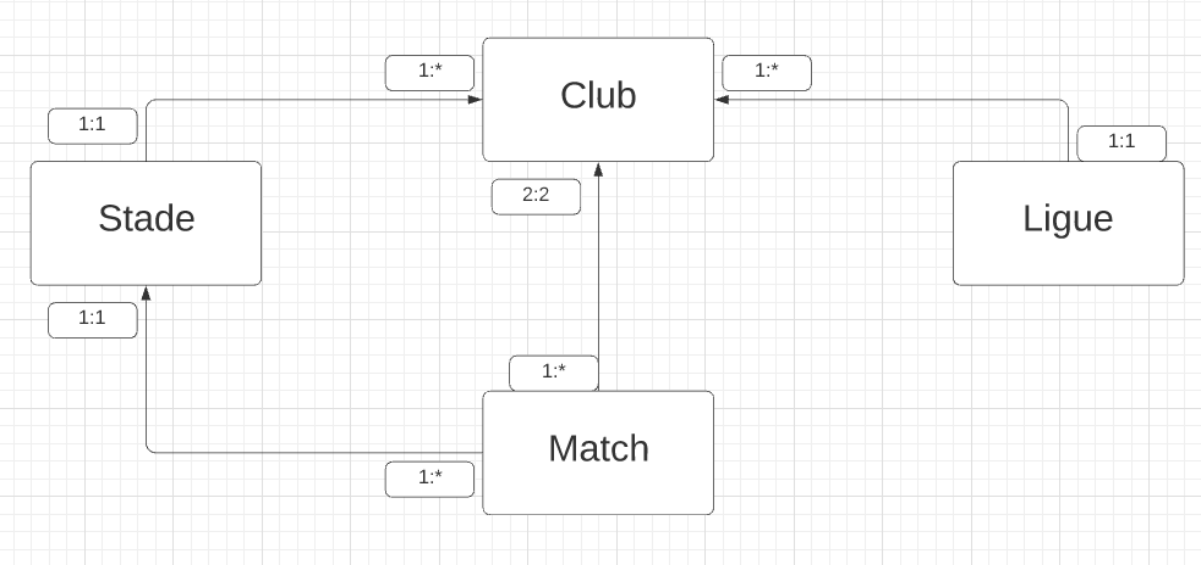
Tuteur :

M. Michel ZAM

- ## Présentation du modèle :

Le projet ligue 1 Conforama consiste à organiser des matchs entre clubs évoluant dans ce championnat, tout en tenant compte des stades où ces rencontres se dérouleront.

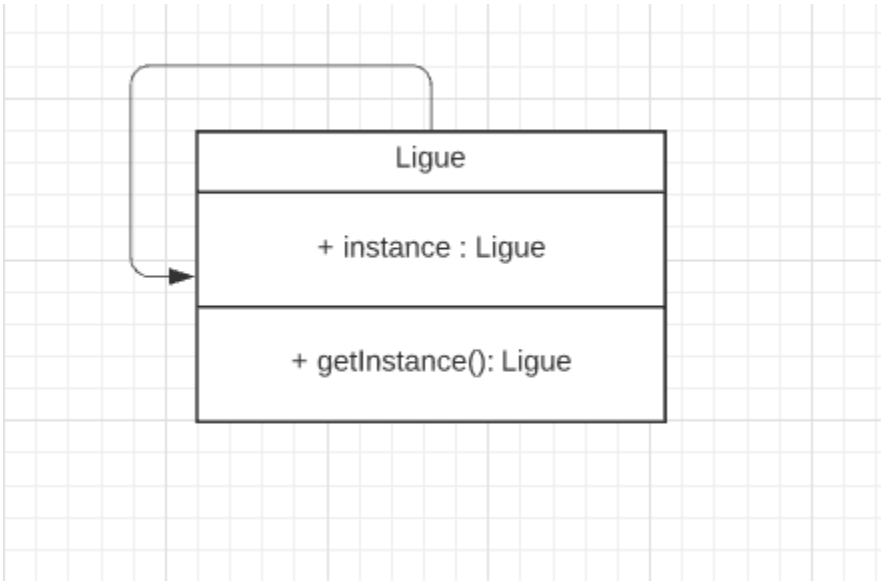
Pour ce faire, nous avons créé 4 classes, représentées dans le diagramme suivant :



- ## Design Pattern :

  - ### Design Pattern Singleton :

Nous avons créé la classe “Ligue” pour donner un sens à notre modèle, où nous pouvons organiser des matchs dans un cadre bien établie par la “Ligue 1 Conforama”, les matchs serons jouer dans le cadre d’un championnat qui englobera tous les clubs, donc une autre ligue ne peut pas être crée. D’où l’utilisation d’un Singleton qui empêchera la création d’une autre ligue que la ligue 1.



Voici la classe “Ligue” :

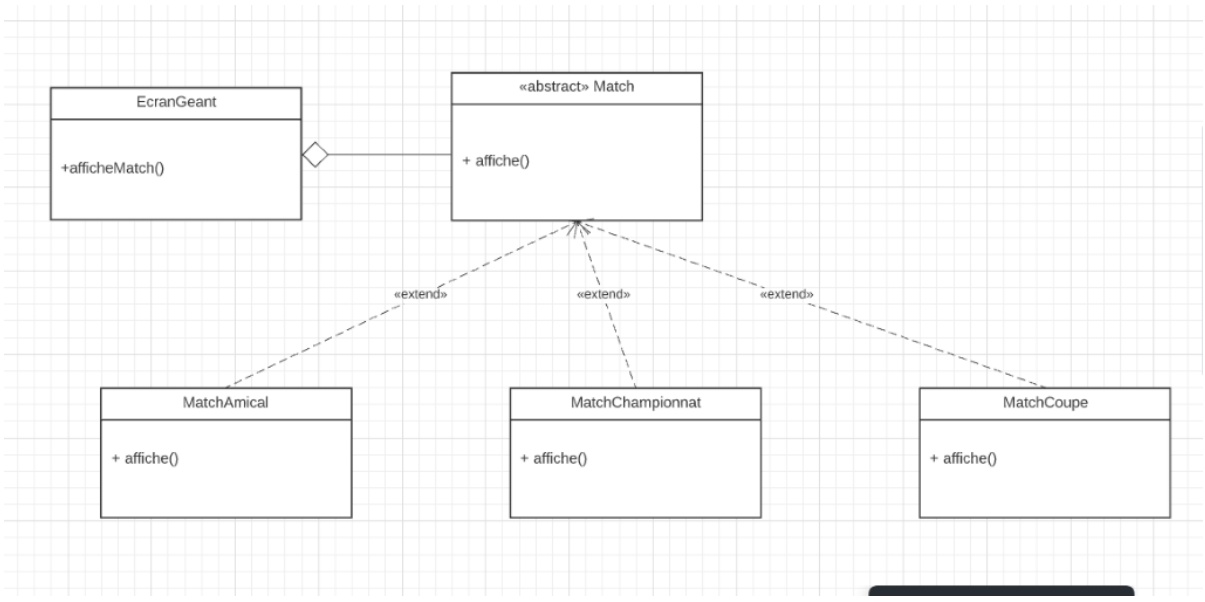
```
public class Ligue{
    private static Ligue instance;
    private static String nom;

    public Ligue() {

    }

    public static Ligue getInstance() {
        if(instance == null) {
            instance = new Ligue();
            nom= "Ligue1";
        }
        return instance;
    }

    public String getNom() {
        return nom;
    }

}
```

Testons maintenant la classe “Ligue” : la barre verte et bien afficher :

- **Design Pattern Strategy :**

Dans le cadre de la reprise du championnat la “Ligue 1 Conforama” a décidé de diffuser quelques informations sur des Ecrans géant, installés à cause de l’absence des supporters dans les stades afin de suivre les matchs.

Le diagramme de classe suivant illustre notre Design Pattern Strategy :



Nous avons mis en place ce Design Pattern Strategy afin de diffuser 3 types de match autorisé par la ligue qui sont “MatchAmical”, “MatchChampionnat” et “MatchCoupe”, plus tard nous pourrons utiliser model pour élargir le type de matchs difusés et/ou utiliser d’autre contexte que celui de “EcranGeant”.

La classe “match” devient donc abstraite comme suit :

```
import java.util.ArrayList; // import the ArrayList class


public abstract class Match {

    protected Club club;
    protected Club club_adverse;
    protected String date;
    protected ArrayList<Integer> listPlaces;
    protected ArrayList<Integer> listPlacesReserves;

    abstract public int nbChangement();
    abstract public String affiche();
```

Classe “MatchAmical” : nous avons implémenté la classe match amical qui nous permettra de récupérer les matchs amicaux dans la classe match :

```
public class MatchAmical extends Match{

    @Override
    public String affiche() {
        return "ce match est un match amical";
    }

    @Override
    public int nbChangement() {
        return 10;
    }

}
```

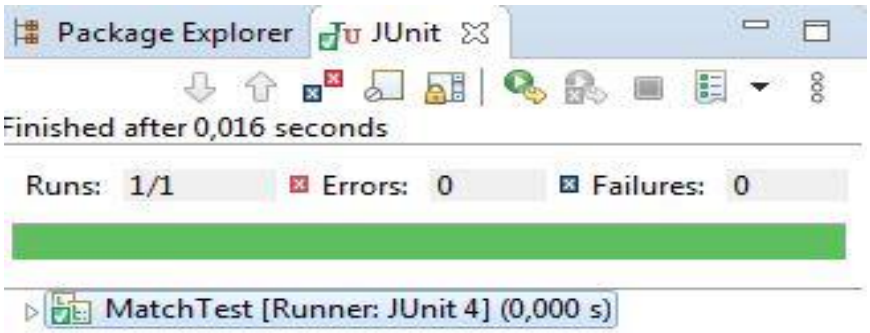
Classe “MatchChampionnat” :

```
import java.util.ArrayList;

public class MatchChampionnat extends Match{

    public MatchChampionnat() {

    }

    public MatchChampionnat(Club club, Club club_adverse, String date) {
        super();
        this.club = club;
        this.club_adverse = club_adverse;
        this.date = date;
        this.listPlaces= new ArrayList<Integer>();
        this.listPlacesReserves= new ArrayList<Integer>();
        for(int i = 0; i< this.club.getStade().getNbrPlace()+1;i++) {
            this.listPlaces.add(1);
        }
```

Classe “MatchCoupe” : nous avons aussi créer la classe MatchCoupe pour générer les matchs de la coupe à partir de la classe match :

```
public class MatchCoupe extends Match{

    @Override
    public String affiche() {
        return "ce match est un match de coupe";
    }

    @Override
    public int nbChangement() {
        return 4;
    }
```

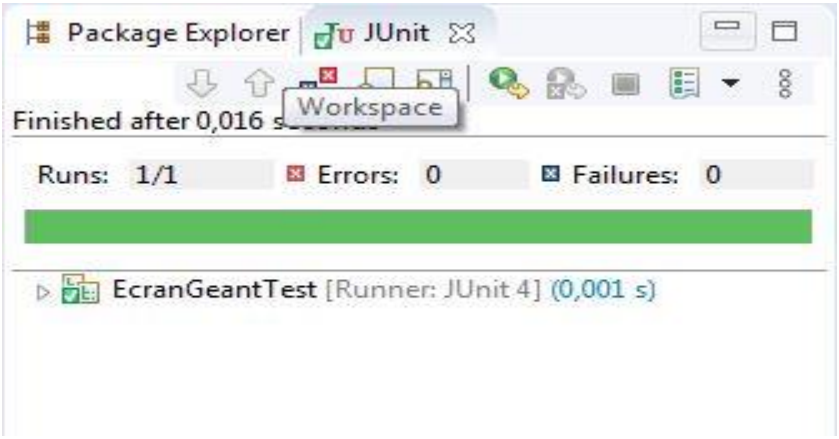
Test sur la Classe Macth :



Utilisation du contexte de l'écran geant :

```
public class EcranGeant {
    private Match match;

    public String affichageMatch() {
        return match.affiche();
    }

    public void setMatch(Match match) {
        this.match = match;
    }

}
```

Test sur la classe "EcranGeant":



- **User Stories :**

```
Feature: Demo feature
  Scenario: organisation d un match de championnat
    Given Un match necissite deux clubs
    When Un match de championnat se joue dans une ligue
    Then les deux clubs sont dans la meme ligue
```

Notre classe SetDifinition est :

ClubTest.java | demo.feature | SetDifiniti... | Club.java | MatchTest.java

```
import cucumber.api.java.en.When;
import java.util.ArrayList;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.core.IsEqual.equalTo;

public class SetDifinition {

    private Club c1;
    private Club c2;
    private Match match;


    @Given("^un match necissite deux clubs$")
    public void un_match_necissite_deux_clubs() throws Exception {
        // Write code here that turns the phrase above into concrete actions
        c1= new Club("PSG", new Stade("ParcDesPrinces",48500));
        c2= new Club("ManUnited", new Stade("OldTraford",60500));
    }

    @When("^un match de championnat se joue dans une ligue")
    public void un_match_de_championnat_se_joue_dans_une_ligue() throws Exception {
        // Write code here that turns the phrase above into concrete actions
        un_match_necissite_deux_clubs();
        match = new MatchChampionnat(c1,c2,"25/05/2020");
    }

    @Then("^les deux clubs sont dans la meme ligue")
    public void les_deux_clubs_sont_dans_la_meme_ligue() throws Exception {
        // Write code here that turns the phrase above into concrete actions
        un_match_de_championnat_se_joue_dans_une_ligue();
        assertThat(1,is(c1.getLigue().equals(c2.getLigue())));
    }
}
```