Pierre Chevily Groupe 101 Année 2022-2023

Sofiane Bougamza Groupe 110

Compte rendu JAVA

Crazy Circus

Table des matières

[Introduction 2](#_Toc130157935)

[Bilan 3](#_Toc130157936)

[Diagramme 4](#_Toc130157937)

[Code source 5](#_Toc130157938)

[Tests 33](#_Toc130157939)

# Introduction

Crazy Circus est un jeu de société qui met à l'épreuve les joueurs en leur demandant de trouver la séquence la plus rapide pour exécuter des ordres pour trois animaux sur le podium bleu ou rouge. Ces ordres sont prédéfinis et permettent soit de faire bouger ces animaux d’un podium à l’autre, soit sur le même podium.

Le jeu se compose de 24 cartes, chacune présentant une situation initiale différente et une situation finale que les joueurs doivent atteindre en résolvant chaque situation à l'aide de leurs compétences de résolution de problèmes pour trouver la meilleure séquence d'ordres.

# Bilan

Ce projet fut assez difficile car ce fut le premier à développer de manière orientée objet. Néanmoins, il nous a permis de nous améliorer dans cette notion. Ainsi, le plus dur a été de concevoir dans nos têtes l’UML. Après avoir validé cette étape, nous avons pu aborder beaucoup plus facilement la programmation du projet.

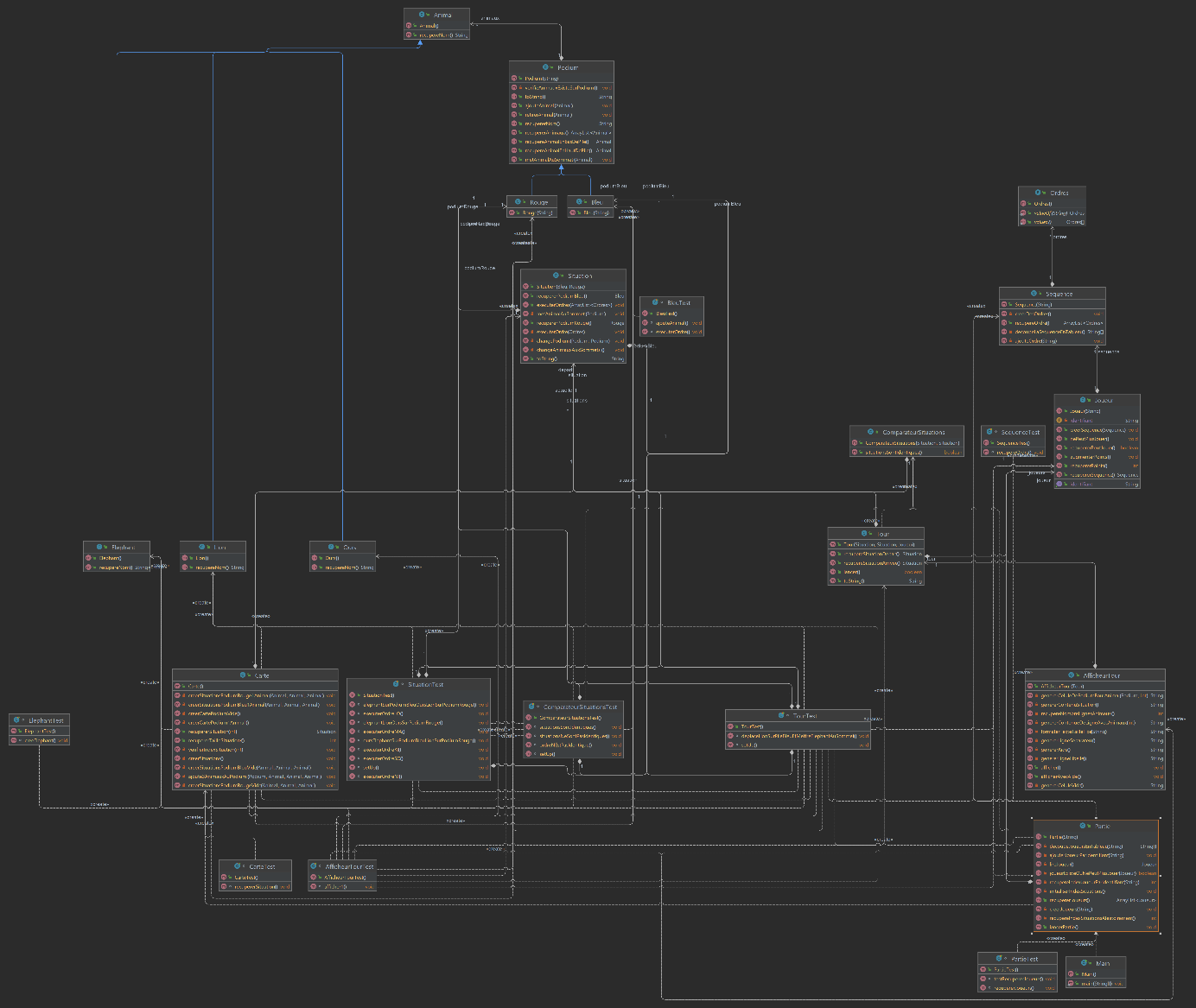
Aussi, nous avons dû apprendre comment manipuler l’héritage d’une classe, notamment pour les classes Podium et Animal.

Nous avons dans le même temps rencontré un problème au niveau de l’affichage de l’interface utilisateur. Cela fut long et fastidieux mais nous avons tout de même réussi à afficher correctement celle-ci. De même pour la création des situations/cartes qui fut assez redondante.

Au niveau des tests unitaires, nous avons rencontré des problèmes pour les tests des tours et des podiums

Tous les tests unitaires que vous retrouverez à la fin du document sont fonctionnels.

# Diagramme



# Code source

package CrazyCircus;

public class **Joueur** {

*// Initialiser les joueurs*

    private String identifiant = "";

    private int points = 0;

    private boolean peutJouer = true; *// Permet de déterminer si le joueur peut encore jouer pendant un tour ou non*

    private Sequence sequence; *// Séquence que le joueur aura saisi*

*// Constructeur d'un joueur, prenant en paramètre son identifiant*

    public **Joueur**(String identifiant) {

        this.identifiant = identifiant;

    }

*/\*\**

*\* @brief Getter de l'identifiant d'un joueur*

*\* @return l'identifiant du joueur*

*\*/*

    public String **getIdentifiant**() {

        return this.identifiant;

    }

*/\*\**

*\* @brief Getter de la possibilité pour le joueur de jouer*

*\* @return true si le joueur peut jouer, false sinon*

*\*/*

    public boolean **recupererPeutJouer**() {

        return peutJouer;

    }

*/\*\**

*\* @brief Setter de peutJouer*

*\* @param peutJouer : indique si le joueur peut jouer ou non*

*\*/*

    public void **nePeutPlusJouer** () {

        this.peutJouer = false;

    }

*/\*\**

*\* @brief Getter des points*

*\* @return les points du joueur*

*\*/*

    public int **recupererPoints**() {

        return points;

    }

*/\*\**

*\* @brief Ajoute un point au joueur*

*\*/*

    public void **augmenterPoints**() {

        this.points ++;

    }

*/\*\**

*\* @brief Crée une séquence à partir de celle reçue en paramètre*

*\* @param sequence : séquence que l'on veut créer*

*\*/*

    public void **creerSequence**(Sequence sequence) {

        this.sequence = sequence;

    }

*/\*\**

*\* @brief Getter de la séquence*

*\* @return la séquence du joueur*

*\*/*

    public Sequence **recupererSequence**() {

        return this.sequence;

    }

}

package CrazyCircus;

import java.lang.reflect.Array;

import java.util.\*;

public class **Partie** {

    private ArrayList<Joueur> joueurs = new ArrayList<>(); *// Tableau contenant tous les joueurs d'une partie*

    private static final int NOMBRE\_TOURS = 12;

    private ArrayList<Integer> indexSituations = new ArrayList<>();

*// Constructeur d'une partie, prenant en paramètres les joueurs de la partie*

    public **Partie**(String joueurs) {

        this.**creerJoueurs**(joueurs);

    }

*/\*\**

*\* @brief Crée les joueurs et les ajoute danss le tableau les contenant*

*\* @param joueurs : les joueurs à ajouter*

*\*/*

    private void **creerJoueurs**(String joueurs) {

        String[] listJoueurs = this.**decoupeJoueursEnTableau**(joueurs);

        for (int i = 0; i < listJoueurs.length; i++) {

            this.**ajouterJoueurParIdentifiant**(listJoueurs[i]);

        }

    }

*/\*\**

*\* @brief Decoupe la chaîne de caractères reçue en paramètres pour pouvoir ajouter les joueurs dans un tableau*

*\* @param joueurs : la chaîne de caractères à découper*

*\* @return le tableau de chaînes de caractères des joueurs*

*\*/*

    private String[] **decoupeJoueursEnTableau**(String joueurs) {

        return joueurs.**split**("\\s+");

    }

*/\*\**

*\* @brief Permet d'ajouter un joueur avec son identifiant*

*\* @param identifiant : l'identifiant du joueur à ajouter*

*\*/*

    private void **ajouterJoueurParIdentifiant**(String identifiant) {

        Joueur joueur = new **Joueur**(identifiant);

        this.joueurs.**add**(joueur);

    }

*/\*\**

*\* @brief Getter des joueurs*

*\* @return les joueurs de la partie*

*\*/*

    public ArrayList<Joueur> **recupererJoueurs**() {

        return this.joueurs;

    }

    private void **initialiserIndexSituations**() {

        for (int i = 0; i < NOMBRE\_TOURS\*2; i++)

            this.indexSituations.**add**(i);

    }

    private int **recupereIndexSituationsAleatoirement**() {

        Random r = new **Random**();

        int max = this.indexSituations.**size**() - 1;

        int index = r.**nextInt**(max);

        this.indexSituations.**remove**(index);

        return index;

    }

    public void **lancerPartie**() throws Exception {

        Carte carte = new **Carte**();

        this.**initialiserIndexSituations**();

        for (int i = 0; i < this.NOMBRE\_TOURS - 1; i++) {

            int indexSituationDepart = this.**recupereIndexSituationsAleatoirement**();

            Situation situationDepart = carte.**recupererSituation**(indexSituationDepart);

            int indexSituationArrivee = this.**recupereIndexSituationsAleatoirement**();

            Situation situationArrivee = carte.**recupererSituation**(indexSituationArrivee);

            for (int j = 0; j < this.joueurs.**size**(); j++) {

*// Demander à l'utilisateur de rentrer ses identifiants puis la séquence*

*// Vérifier que l'utilisateur existe et peut jouer.*

*// Si c'est le cas, lancer le tour*

                System.out.**println**(this.joueurs.**get**(j).**getIdentifiant**() + ", merci d'entrer votre commande");

                Joueur joueur = this.**lireJoueur**();

                Tour tour = new **Tour**(situationDepart,situationArrivee, joueur);

                AfficheurTour afficheurTour = new **AfficheurTour**(tour);

                afficheurTour.**afficherAvecAide**();

                tour.**lancer**();

                ComparateurSituations comparateur = new **ComparateurSituations**(tour.**recupereSituationDepart**(),tour.**recupereSituationArrivee**());

                if (comparateur.**situationsSontIdentiques**()) {

                    joueur.**augmenterPoints**();

                    System.out.**println**("Bonne combinaison, vous gagnez 1 point");

                    break;

                }

                AfficheurTour afficherApresCoup = new **AfficheurTour**(tour);

                afficherApresCoup.**afficher**();

            }

        }

    }

    private Joueur **lireJoueur**() throws Exception {

        Scanner scanner = new **Scanner**(System.in);

        String identifiantEtSequence = scanner.**nextLine**();

        String[] identifiantEtSequenceEnTableau = identifiantEtSequence.**split**("\\s+");

        String identifiant = identifiantEtSequenceEnTableau[0];

        String sequence = identifiantEtSequenceEnTableau[1];

        int indexJoueur = this.**recupereIndexJoueurParIdentifiant**(identifiant);

        Joueur joueur = this.joueurs.**get**(indexJoueur);

        joueur.**creerSequence**(new **Sequence**(sequence));

        return joueur;

    }

    private boolean **joueurExisteOuNePeutPlusJouer**(Joueur joueur) {

        for (Joueur joueurExistant : this.joueurs){

            if (joueur.**getIdentifiant**().**equals**(joueurExistant.**getIdentifiant**()) || joueur.**recupererPeutJouer**()) {

                return true;

            }

        }

        return false;

    }

    private int **recupereIndexJoueurParIdentifiant**(String identifiant) throws Exception {

        for(int i = 0; i < this.joueurs.**size**(); i++) {

            if (this.joueurs.**get**(i).**getIdentifiant**().**equals**(identifiant)) {

                return i;

            }

        }

        throw new **Exception**("Le joueur de cet identifiant n'existe pas");

    }

}

package CrazyCircus;

import java.util.ArrayList;

public class **Sequence** {

    private String texte;

    private ArrayList<Ordres> ordres = new ArrayList<>(); *// Ordres de la séquence*

    private static final int LONGUEUR\_ORDRE = 2; *// Un ordre a une longueur de 2 caractères*

*// Constructeur d'une séquence*

    public **Sequence**(String texte) {

        this.texte = texte;

        this.**creerDesOrdres**();

    }

*/\*\**

*\* @brief Crée des ordres et les ajoute dans le tableau correspondant*

*\*/*

    private void **creerDesOrdres**() {

        String[] ordres = this.**decoupeLaSequenceEnTableau**();

        for (int i = 0; i < ordres.length; i++) {

            this.**ajouteOrdre**(ordres[i]);

        }

    }

*/\*\**

*\* @brief Permet de découper la séquence en un tableau de chaînes de caractères*

*\* @return les ordres sous forme de tableau de chaînes de caractères*

*\*/*

    private String[] **decoupeLaSequenceEnTableau**() {

        return this.texte.**split**("(?<=\\G.{" + LONGUEUR\_ORDRE + "})");

    }

*/\*\**

*\* @brief Permet d'ajouter un ordre au tableau correspondant*

*\* @param ordre : l'ordre à ajouter*

*\*/*

    private void **ajouteOrdre**(String ordre) {

        if (ordre.**equals**("KI")) {

            this.ordres.**add**(Ordres.KI);

        } else if (ordre.**equals**("LO")) {

            this.ordres.**add**(Ordres.LO);

        }

        else if (ordre.**equals**("SO")) {

            this.ordres.**add**(Ordres.SO);

        }

        else if (ordre.**equals**("NI")) {

            this.ordres.**add**(Ordres.NI);

        }

        else if (ordre.**equals**("MA")) {

            this.ordres.**add**(Ordres.MA);

        }

    }

*/\*\**

*\* @brief Getter des ordres*

*\* @return les ordres de la séquence*

*\*/*

    public ArrayList<Ordres> **recupereOrdre**() {

        return this.ordres;

    }

}

package CrazyCircus;

import CrazyCircus.Animal.Animal;

import CrazyCircus.Animal.Elephant;

import CrazyCircus.Animal.Lion;

import CrazyCircus.Animal.Ours;

import CrazyCircus.Podium.Bleu;

import CrazyCircus.Podium.Podium;

import CrazyCircus.Podium.Rouge;

import java.util.ArrayList;

public class **Carte** {

    private ArrayList<Situation> situations = new ArrayList<>();

*// Constructeur de la classe Carte*

    public **Carte**() {

        this.**creerSituations**();

    }

*/\*\**

*\* @brief Getter d'une situation*

*\* @param index : index de la situation que l'on veut choisir*

*\* @pre On vérifie que l'index est bien correct*

*\* @return la situation à la position index*

*\*/*

    public Situation **recupererSituation**(int index) throws Exception {

        this.**verifierIndexSituation**(index);

        return this.situations.**get**(index);

    }

*/\*\**

*\* @brief Getter du nombre de situations*

*\* @return le nombre de situations*

*\*/*

    public int **recupererTailleSituations**() {

        return this.situations.**size**();

    }

*/\*\**

*\* @brief Sert à vérifier que l'index donné d'une situation est bien valide*

*\* @param index : position de la situation que l'on veut vérifier*

*\*/*

    private void **verifierIndexSituation**(int index) throws Exception {

        if (index < 0 || index > this.situations.**size**()) {

            throw new **Exception**("index invalide. Il doit être compris entre 0 et " + this.situations.**size**());

        }

    }

*/\*\**

*\* @brief Crée toutes les situations possibles*

*\*/*

    private void **creerSituations**() {

        this.**creerCartePodiumVide**();

        this.**creerCartePodium1Animal**();

    }

*/\*\**

*\* @brief Crée toutes les situations possibles avec un podium vide et 3 animaux sur l'autre podium*

*\*/*

    private void **creerCartePodiumVide**() {

        this.**creerSituationsPodiumBleuVide**(new **Elephant**(), new **Ours**(),      new **Lion**());

        this.**creerSituationsPodiumBleuVide**(new **Elephant**(), new **Lion**(),      new **Ours**());

        this.**creerSituationsPodiumBleuVide**(new **Ours**(),     new **Lion**(),      new **Elephant**());

        this.**creerSituationsPodiumBleuVide**(new **Ours**(),     new **Elephant**(),  new **Lion**());

        this.**creerSituationsPodiumBleuVide**(new **Lion**(),     new **Ours**(),      new **Elephant**());

        this.**creerSituationsPodiumBleuVide**(new **Lion**(),     new **Elephant**(),  new **Ours**());

        this.**creerSituationsPodiumRougeVide**(new **Elephant**(), new **Ours**(),      new **Lion**());

        this.**creerSituationsPodiumRougeVide**(new **Elephant**(), new **Lion**(),      new **Ours**());

        this.**creerSituationsPodiumRougeVide**(new **Ours**(),     new **Lion**(),      new **Elephant**());

        this.**creerSituationsPodiumRougeVide**(new **Ours**(),     new **Elephant**(),  new **Lion**());

        this.**creerSituationsPodiumRougeVide**(new **Lion**(),     new **Ours**(),      new **Elephant**());

        this.**creerSituationsPodiumRougeVide**(new **Lion**(),     new **Elephant**(),  new **Ours**());

    }

*/\*\**

*\* @brief Crée toutes les situations possibles avec un podium contenant 1 animal et 2 animaux sur l'autre podium*

*\*/*

    private void **creerCartePodium1Animal**() {

        this.**creerSituationsPodiumBleu1Animal**(new **Elephant**(), new **Ours**(), new **Lion**());

        this.**creerSituationsPodiumBleu1Animal**(new **Elephant**(), new **Lion**(), new **Ours**());

        this.**creerSituationsPodiumBleu1Animal**(new **Lion**(), new **Elephant**(), new **Ours**());

        this.**creerSituationsPodiumBleu1Animal**(new **Lion**(), new **Ours**(), new **Elephant**());

        this.**creerSituationsPodiumBleu1Animal**(new **Ours**(), new **Lion**(), new **Elephant**());

        this.**creerSituationsPodiumBleu1Animal**(new **Ours**(), new **Elephant**(), new **Lion**());

        this.**creerSituationsPodiumRouge1Animal**(new **Elephant**(), new **Ours**(), new **Lion**());

        this.**creerSituationsPodiumRouge1Animal**(new **Elephant**(), new **Lion**(), new **Ours**());

        this.**creerSituationsPodiumRouge1Animal**(new **Lion**(), new **Elephant**(), new **Ours**());

        this.**creerSituationsPodiumRouge1Animal**(new **Lion**(), new **Ours**(), new **Elephant**());

        this.**creerSituationsPodiumRouge1Animal**(new **Ours**(), new **Lion**(), new **Elephant**());

        this.**creerSituationsPodiumRouge1Animal**(new **Ours**(), new **Elephant**(), new **Lion**());

    }

*/\*\**

*\* Crée une situation avec le podium bleu vide*

*\* @param animal1 : animal en haut du podium*

*\* @param animal2 : animal au milieu du podium*

*\* @param animal3 : animal en bas du podium*

*\* @see creerCartePodiumVide pour son utilisation*

*\*/*

    private void **creerSituationsPodiumBleuVide**(Animal animal1, Animal animal2, Animal animal3) {

        Bleu podiumBleu = new **Bleu**("BLEU");

        Rouge podiumRouge = new **Rouge**("ROUGE");

        this.**ajoute3AnimauxAuPodium**(podiumRouge,animal1, animal2, animal3);

        this.situations.**add**(new **Situation**(podiumBleu, podiumRouge));

    }

*/\*\**

*\* Crée une situation avec le podium rouge vide*

*\* @param animal1 : animal en haut du podium*

*\* @param animal2 : animal au milieu du podium*

*\* @param animal3 : animal en bas du podium*

*\* @see creerCartePodiumVide pour son utilisation*

*\*/*

    private void **creerSituationsPodiumRougeVide**(Animal animal1, Animal animal2, Animal animal3) {

        Bleu podiumBleu = new **Bleu**("BLEU");

        Rouge podiumRouge = new **Rouge**("ROUGE");

        this.**ajoute3AnimauxAuPodium**(podiumBleu,animal1, animal2, animal3);

        this.situations.**add**(new **Situation**(podiumBleu, podiumRouge));

    }

*/\*\**

*\* @brief Permet d'ajouter 3 animaux sur un podium donné en paramètre*

*\* @param podium : Podium qui contiendra les 3 animaux reçus en paramètres*

*\* @param animal1 : animal en haut du podium*

*\* @param animal2 : animal au milieu du podium*

*\* @param animal3 : animal en bas du podium*

*\*/*

    private void **ajoute3AnimauxAuPodium**(Podium podium, Animal animal1, Animal animal2, Animal animal3) {

        podium.**ajouteAnimal**(animal1);

        podium.**ajouteAnimal**(animal2);

        podium.**ajouteAnimal**(animal3);

    }

*/\*\**

*\* @brief Crée une situation avec 1 animal sur le podium bleu*

*\* @param animalPodiumBleu : animal sur le podium bleu*

*\* @param animalPodiumRouge1 : animal au sommet du podium rouge*

*\* @param animalPodiumRouge2: animal en bas du podium rouge*

*\*/*

    private void **creerSituationsPodiumBleu1Animal**(Animal animalPodiumBleu, Animal animalPodiumRouge1, Animal animalPodiumRouge2) {

        Bleu podiumBleu = new **Bleu**("BLEU");

        podiumBleu.**ajouteAnimal**(animalPodiumBleu);

        Rouge podiumRouge = new **Rouge**("ROUGE");

        podiumRouge.**ajouteAnimal**(animalPodiumRouge1);

        podiumRouge.**ajouteAnimal**(animalPodiumRouge2);

        this.situations.**add**(new **Situation**(podiumBleu, podiumRouge));

    }

*/\*\**

*\* @brief Crée une situation avec 1 animal sur le podium rouge*

*\* @param animalPodiumRouge : animal sur le podium rouge*

*\* @param animalPodiumBleu1 : animal au sommet du podium bleu*

*\* @param animalPodiumBleu2: animal en bas du podium bleu*

*\*/*

    private void **creerSituationsPodiumRouge1Animal**(Animal animalPodiumRouge, Animal animalPodiumBleu1, Animal animalPodiumBleu2) {

        Rouge podiumRouge = new **Rouge**("ROUGE");

        podiumRouge.**ajouteAnimal**(animalPodiumRouge);

        Bleu podiumBleu = new **Bleu**("BLEU");

        podiumBleu.**ajouteAnimal**(animalPodiumBleu1);

        podiumBleu.**ajouteAnimal**(animalPodiumBleu2);

        this.situations.**add**(new **Situation**(podiumBleu, podiumRouge));

    }

}

package CrazyCircus;

import CrazyCircus.Podium.Bleu;

import CrazyCircus.Podium.Podium;

import CrazyCircus.Podium.Rouge;

import CrazyCircus.Animal.Animal;

import java.util.ArrayList;

public class **Situation** {

    private Bleu podiumBleu;

    private Rouge podiumRouge;

*// Constructeur de la situation*

    public **Situation**(Bleu podiumBleu, Rouge podiumRouge) {

        this.podiumBleu = podiumBleu;

        this.podiumRouge = podiumRouge;

    }

*/\*\**

*\* @brief Getter du podium bleu*

*\* @return le podium bleu*

*\*/*

    public Bleu **recupererPodiumBleu**() {

        return this.podiumBleu;

    }

*/\*\**

*\* @brief Getter du podium rouge*

*\* @return le podium rouge*

*\*/*

    public Rouge **recupererPodiumRouge**() {

        return this.podiumRouge;

    }

*/\*\**

*\* @brief Lit les ordres et appelle la fonction permettant de les exécuter*

*\* @param ordres : ordres à exécuter*

*\*/*

    public void **executerOrdres**(ArrayList<Ordres> ordres) {

        for (Ordres ordre : ordres) {

            this.**executerOrdre**(ordre);

        }

    }

*/\*\**

*\* @brief Permet d'afficher la situation avec les podiums contenus dedans*

*\* @return la chaîne de caractères représentant les podiums*

*\*/*

    public String **toString**() {

        return this.podiumBleu.**toString**() + "\n" + this.podiumRouge.**toString**();

    }

*/\*\**

*\* @brief Permet d'exécuter un ordre.*

*\* @param ordre : ordre à exécuter*

*\*/*

    private void **executerOrdre** (Ordres ordre) {

        {

            try {

                if (ordre == Ordres.KI) {

*// L’animal se trouvant en haut de la pile du podium bleu saute pour rejoindre le sommet de la pile du podium rouge*

                    this.**changePodium**(this.podiumBleu, this.podiumRouge);

                } else if (ordre == Ordres.LO) {

*// L’animal se trouvant en haut de la pile du podium rouge saute pour rejoindre le sommet de la pile du podium bleu*

                    this.**changePodium**(this.podiumRouge, this.podiumBleu);

                } else if (ordre == Ordres.SO) {

*// Les deux animaux se trouvant au sommet des piles des deux podiums échangent leur place.*

                    this.**changeAnimauxAuxSommets**();

                } else if (ordre == Ordres.NI) {

*// L’animal se trouvant en bas de la pile du podium bleu monte et se place en haut de la pile de ce même podium*

                    this.**metAnimalAuSommet**(this.podiumBleu);

                } else if (ordre == Ordres.MA) {

*// L’animal se trouvant en bas de la pile du podium rouge monte et se place en haut de la pile de ce même podium*

                    this.**metAnimalAuSommet**(this.podiumRouge);

                }

            } catch (Exception exception) {

                System.out.**println**(exception.**getMessage**());

            }

        }

    }

*/\*\**

*\* @brief Fait sauter un animal d'un podium à un autre*

*\* @param podiumDepart : le podium sur lequel était l'animal*

*\* @param podiumArrive : le podium sur lequel l'animal va sauter*

*\*/*

    private void **changePodium**(Podium podiumDepart, Podium podiumArrive) throws Exception {

        Animal AnimalEnHautDePile = podiumDepart.**recupereAnimalEnHautDePile**();

        podiumDepart.**retirerAnimal**(AnimalEnHautDePile);

        podiumArrive.**metAnimalAuSommet**(AnimalEnHautDePile);

    }

*/\*\**

*\* @brief Permet de changer les animaux aux sommets des podiums*

*\*/*

    private void **changeAnimauxAuxSommets**() throws Exception {

        Animal AnimalEnHautDePileDepart = this.podiumBleu.**recupereAnimalEnHautDePile**();

        Animal AnimalEnHautDePileArrivee = this.podiumRouge.**recupereAnimalEnHautDePile**();

        this.podiumBleu.**retirerAnimal**(AnimalEnHautDePileDepart);

        this.podiumRouge.**retirerAnimal**(AnimalEnHautDePileArrivee);

        this.podiumRouge.**metAnimalAuSommet**(AnimalEnHautDePileDepart);

        this.podiumBleu.**metAnimalAuSommet**(AnimalEnHautDePileArrivee);

    }

*/\*\**

*\* @brief Permet de mettre un animal sur le podium reçu en paramètre*

*\* @param podium : podium sur lequel l'animal va aller*

*\*/*

    private void **metAnimalAuSommet**(Podium podium) throws Exception {

        Animal animal = podium.**recupereAnimalEnBasDePile**();

        podium.**retirerAnimal**(animal);

        podium.**metAnimalAuSommet**(animal);

    }

}

package CrazyCircus;

*// Enumération de tous les ordres possibles*

public enum **Ordres** {KI, LO, SO, NI, MA}

package CrazyCircus;

import java.util.ArrayList;

import java.util.concurrent.ThreadLocalRandom;

public class **Tour** {

    private Situation depart;

    private Situation arrivee;

    private Joueur joueur;

    public **Tour**(Situation depart, Situation arrivee, Joueur joueur) {

        this.depart = depart;

        this.arrivee = arrivee;

        this.joueur = joueur;

    }

    public Situation **recupereSituationDepart**() {

        return this.depart;

    }

    public Situation **recupereSituationArrivee**() {

        return this.arrivee;

    }

    public String **toString**() {

        return "Joueur : " + this.joueur.**getIdentifiant**()

                + "\nSituation depart : \n" + this.depart.**toString**()

                + "\nSituation arrivee :\n" + this.arrivee.**toString**();

    }

    public boolean **lancer**() throws Exception {

        ArrayList<Ordres> ordres = this.joueur.**recupererSequence**().**recupereOrdre**();

        Situation situationPourJouer = this.depart;

        this.depart.**executerOrdres**(ordres);

        ComparateurSituations comparateur = new **ComparateurSituations**(situationPourJouer, this.arrivee);

        return comparateur.**situationsSontIdentiques**();

    }

}

package CrazyCircus;

import CrazyCircus.Animal.Animal;

import java.util.ArrayList;

public class **ComparateurSituations** {

    private Situation actuelle;

    private Situation souhaitee;

*// Constructeur d'un comparateur de situations*

    public **ComparateurSituations**(Situation actuelle, Situation souhaitee) {

        this.actuelle = actuelle;

        this.souhaitee = souhaitee;

    }

*/\*\**

*\* @brief Permet de comparer 2 situations*

*\* @return true si les situations sont identiques, false sinon*

*\*/*

    public boolean **situationsSontIdentiques**() {

        return this.actuelle.**recupererPodiumBleu**().**toString**().**equals**(this.souhaitee.**recupererPodiumBleu**().**toString**())

            && this.actuelle.**recupererPodiumRouge**().**toString**().**equals**(this.souhaitee.**recupererPodiumRouge**().**toString**());

    }

}

package CrazyCircus;

import java.util.List;

import java.util.ArrayList;

import java.util.Collections;

import java.lang.Math;

import CrazyCircus.Podium.Podium;

public class **AfficheurTour** {

    private final int TAILLE\_CELLULE = 10;

    private final String SEPARATEUR\_LIGNE = "-----";

    private final String SIGNE\_VERS = "==>";

    private Tour tour;

    public **AfficheurTour**(Tour tour)

    {

        this.tour = tour;

    }

    public void **afficher**()

    {

        String contenu = this.**genererContenuSituation**();

        System.out.**println**(contenu);

    }

    public void **afficherAvecAide**()

    {

        String contenu = this.**genererContenuSituation**();

        contenu+="\n";

        contenu+=this.**genererAide**();

        System.out.**println**(contenu);

    }

    private String **genererContenuSituation**(){

        int nbLignesAnimaux = this.**recupereNombreLignesAnimaux**();

        String contenu = "";

        for (int i = 0; i < nbLignesAnimaux; i++) {

            contenu+= this.**genererContenueDeLigneAvecAnimaux**(i);

            contenu+="\n";

        }

        contenu+=this.**genererLigneSeparateur**();

        contenu+="\n";

        contenu+=this.**genererLigneLibelle**();

        return contenu;

    }

    private String **genererAide**() {

        String contenu = "";

        contenu+= "----------------------------------------------\n";

        contenu+= "KI : BLEU --> ROUGE\t\tNI : BLEU  ˆ\n";

        contenu+= "LO : BLEU <-- ROUGE\t\tMA : ROUGE ˆ\n";

        contenu+= "SO : BLEU <-> ROUGE";

        return contenu;

    }

    private int **recupereNombreLignesAnimaux**() {

        ArrayList<Integer> nbAnimaux = new ArrayList<Integer>();

        nbAnimaux.**add**(this.tour.**recupereSituationDepart**().**recupererPodiumBleu**().**recupererAnimaux**().**size**());

        nbAnimaux.**add**(this.tour.**recupereSituationDepart**().**recupererPodiumRouge**().**recupererAnimaux**().**size**());

        nbAnimaux.**add**(this.tour.**recupereSituationArrivee**().**recupererPodiumBleu**().**recupererAnimaux**().**size**());

        nbAnimaux.**add**(this.tour.**recupereSituationArrivee**().**recupererPodiumRouge**().**recupererAnimaux**().**size**());

        return Collections.**max**(nbAnimaux);

    }

    private String **genererContenueDeLigneAvecAnimaux**(int indexAnimal)

    {

        String contenu = "";

        contenu+=this.**genererCeluleDePodiumPourAnimal**(this.tour.**recupereSituationDepart**().**recupererPodiumBleu**(), indexAnimal);

        contenu+=this.**genererCeluleDePodiumPourAnimal**(this.tour.**recupereSituationDepart**().**recupererPodiumRouge**(), indexAnimal);

        contenu+=this.**genererCeluleVide**();

        contenu+=this.**genererCeluleDePodiumPourAnimal**(this.tour.**recupereSituationArrivee**().**recupererPodiumBleu**(), indexAnimal);

        contenu+=this.**genererCeluleDePodiumPourAnimal**(this.tour.**recupereSituationArrivee**().**recupererPodiumRouge**(), indexAnimal);

        return contenu;

    }

    private String **genererLigneSeparateur**()

    {

        String contenu = "";

        contenu+=this.**formaterTexteTailleFixe**(this.SEPARATEUR\_LIGNE);

        contenu+=this.**formaterTexteTailleFixe**(this.SEPARATEUR\_LIGNE);

        contenu+=this.**formaterTexteTailleFixe**(this.SIGNE\_VERS);

        contenu+=this.**formaterTexteTailleFixe**(this.SEPARATEUR\_LIGNE);

        contenu+=this.**formaterTexteTailleFixe**(this.SEPARATEUR\_LIGNE);

        return contenu;

    }

    private String **genererLigneLibelle**()

    {

        String contenu = "";

        contenu+=this.**formaterTexteTailleFixe**(this.tour.**recupereSituationDepart**().**recupererPodiumBleu**().**recupererNom**());

        contenu+=this.**formaterTexteTailleFixe**(this.tour.**recupereSituationDepart**().**recupererPodiumRouge**().**recupererNom**());

        contenu+=this.**genererCeluleVide**();

        contenu+=this.**formaterTexteTailleFixe**(this.tour.**recupereSituationArrivee**().**recupererPodiumBleu**().**recupererNom**());

        contenu+=this.**formaterTexteTailleFixe**(this.tour.**recupereSituationArrivee**().**recupererPodiumRouge**().**recupererNom**());

        return contenu;

    }

    private String **genererCeluleDePodiumPourAnimal**(Podium podium, int indexAnimal)

    {

        if (indexAnimal < podium.**recupererAnimaux**().**size**()) {

            return this.**formaterTexteTailleFixe**(podium.**recupererAnimaux**().**get**(indexAnimal).**recupereNom**());

        }

        return this.**genererCeluleVide**();

    }

    private String **genererCeluleVide**()

    {

        return this.**formaterTexteTailleFixe**("");

    }

    private String **formaterTexteTailleFixe**(String texte)

    {

        return String.**format**("%1$-" + this.TAILLE\_CELLULE + "s", texte);

    }

}

package CrazyCircus;

import java.util.ArrayList;

public class **Main** {

    public static void **main**(String[] args) throws Exception {

        String identifiantsJoueurs = "";

        for (int i = 0; i < args.length; i++) {

            identifiantsJoueurs += args[i] + " ";

        }

        Partie partie = new **Partie**(identifiantsJoueurs);

        partie.**lancerPartie**();

    }

}

package CrazyCircus.Podium;

import CrazyCircus.Animal.Animal;

import java.io.StringReader;

import java.util.ArrayList;

public class **Podium** {

    private String nom; *// Nom du podium : soit bleu, soit rouge*

    private ArrayList<Animal> animaux = new ArrayList<>(); *// Tableau des animaux du podium*

*// Constructeur de Podium*

    public **Podium**(String nom) {

        this.nom = nom;

    }

*/\*\**

*\* @brief Getter pour le nom*

*\* @return le nom du podium*

*\*/*

    public String **recupererNom**() {

        return this.nom;

    }

*/\*\**

*\* @brief Getter les animaux du podium*

*\* @return les animaux du podium*

*\*/*

    public ArrayList<Animal> **recupererAnimaux**() {

        return this.animaux;

    }

*/\*\**

*\* @brief Textualise le tableau d'animaux*

*\* @return la chaîne de caractères contenant les animaux d'un podium*

*\*/*

    public String **toString**() {

        String libelle = "Podium " + this.nom + " :\n";

        for (int i = 0; i < this.animaux.**size**(); i++) {

            libelle+=this.animaux.**get**(i).**recupereNom**() + "\n";

        }

        return libelle;

    }

*/\*\**

*\* @brief ajoute un animal à un podium*

*\* @param animalAAjouter : l'animal que l'on veut ajouter*

*\*/*

    public void **ajouteAnimal**(Animal animalAAjouter) {

*//        for (Animal animal : animaux){*

*//            if (animaux.contains(animal))*

*//                return;*

*//        }*

        this.animaux.**add**(animalAAjouter);

    }

*/\*\**

*\* @brief Donne l'animal au sommet du podium*

*\* @pre On vérifie qu'il y ait au moins un animal sur le podium*

*\* @return l'animal au sommet du podium*

*\*/*

    public Animal **recupereAnimalEnHautDePile**() throws Exception {

        this.**verifieAnimauxExisteSurPodium**();

        return this.animaux.**get**(0);

    }

*/\*\**

*\* @brief Donne l'animal en bas du podium*

*\* @pre On vérifie qu'il y ait au moins un animal sur le podium*

*\* @return L'animal en bas du podium*

*\*/*

    public Animal **recupereAnimalEnBasDePile**() throws Exception {

        this.**verifieAnimauxExisteSurPodium**();

        int indexAnimalEnBasDePile = this.animaux.**size**() - 1;

        return this.animaux.**get**(indexAnimalEnBasDePile);

    }

*/\*\**

*\* @brief Retir un animal du podium*

*\* @param animal : l'animal que l'on veut retirer*

*\*/*

    public void **retirerAnimal**(Animal animal)

    {

        for (int i=0; i < this.animaux.**size**(); i++) {

            if (this.animaux.**get**(i).**recupereNom**() == animal.**recupereNom**()) {

                this.animaux.**remove**(i);

            }

        }

    }

*/\*\**

*\* @brief Met l'animal donné en paramètre au sommet du podium*

*\* @param animal : l'animal que l'on veut mettre au sommet du podium*

*\*/*

    public void **metAnimalAuSommet**(Animal animal) {

        this.animaux.**add**(0, animal);

    }

*/\*\**

*\* @brief Verifie qu'il y ait au moins un animal sur le podium*

*\*/*

    private void **verifieAnimauxExisteSurPodium**() throws Exception {

        if (this.animaux.**size**() == 0) {

            throw new **Exception**("Le podium " + this.nom + " est vide");

        }

    }

}

package CrazyCircus.Podium;

*// Classe enfant de Podium, servant à créer un Podium Bleu*

public class **Bleu** extends **Podium** {

*// Constructeur du podium Bleu*

    public **Bleu**(String nom) {

        super("BLEU");

    }

}

package CrazyCircus.Podium;

import CrazyCircus.Ordres;

*// Classe enfant de Podium, servant à créer un Podium Rogue*

public class **Rouge** extends **Podium** {

*// Constructeur du podium Rouge*

    public **Rouge**(String nom) {

        super("ROUGE");

    }

}

package CrazyCircus.Animal;

public class **Animal** {

    protected String nom = ""; *// Nom de l'animal*

*/\*\**

*\* @brief Getter du nom de l'animal*

*\* @return le nom de l'animal*

*\*/*

    public String **recupereNom**() {

        return this.nom;

    }

}

package CrazyCircus.Animal;

*// Classe enfant de la classe Animal*

public class **Lion** extends **Animal**{

    protected String nom = "LION";

*/\*\**

*\* @brief Getter du nom de l'animal*

*\* @return le nom de l'animal*

*\*/*

    public String **recupereNom**(){

        return this.nom;

    }

}

package CrazyCircus.Animal;

*// Classe enfant de la classe Animal*

public class **Ours** extends **Animal**{

    protected String nom = "OURS";

*/\*\**

*\* @brief Getter du nom de l'animal*

*\* @return le nom de l'animal*

*\*/*

    public String **recupereNom**(){

        return this.nom;

    }

}

package CrazyCircus.Animal;

*// Classe enfant de la classe Animal*

public class **Elephant** extends **Animal**{

    protected String nom = "ELEPHANT";

*/\*\**

*\* @brief Getter du nom de l'animal*

*\* @return le nom de l'animal*

*\*/*

    public String **recupereNom**(){

        return this.nom;

    }

}

# Tests

package CrazyCircus.tests;

import CrazyCircus.AfficheurTour;

import CrazyCircus.Animal.Elephant;

import CrazyCircus.Animal.Lion;

import CrazyCircus.Animal.Ours;

import CrazyCircus.Joueur;

import CrazyCircus.Podium.Bleu;

import CrazyCircus.Podium.Rouge;

import CrazyCircus.Situation;

import CrazyCircus.Tour;

import org.junit.jupiter.api.Test;

import java.util.ArrayList;

import static org.junit.jupiter.api.Assertions.\*;

class **AfficheurTourTest** {

    @Test

    void **afficher1**() {

        Bleu podiumDepartBleu = new **Bleu**("BLEU");

        podiumDepartBleu.**ajouteAnimal**(new **Ours**());

        podiumDepartBleu.**ajouteAnimal**(new **Lion**());

        Rouge podiumDepartRouge = new **Rouge**("ROUGE");

        podiumDepartRouge.**ajouteAnimal**(new **Elephant**());

        Situation situationDepart = new **Situation**(podiumDepartBleu, podiumDepartRouge);

        Bleu podiumArriveBleu = new **Bleu**("BLEU");

        podiumArriveBleu.**ajouteAnimal**(new **Lion**());

        podiumArriveBleu.**ajouteAnimal**(new **Elephant**());

        Rouge podiumArriveRouge = new **Rouge**("ROUGE");

        podiumArriveRouge.**ajouteAnimal**(new **Ours**());

        Situation situationArrivee = new **Situation**(podiumArriveBleu, podiumArriveRouge);

        Joueur joueur = new **Joueur**("PC");

        Tour tour = new **Tour**(situationDepart, situationArrivee, joueur);

        AfficheurTour afficheur = new **AfficheurTour**(tour);

        afficheur.**afficherAvecAide**();

**assertTrue**(true);

    }

}

package CrazyCircus.tests;

import CrazyCircus.Animal.Elephant;

import CrazyCircus.Animal.Lion;

import CrazyCircus.Animal.Ours;

import CrazyCircus.Joueur;

import CrazyCircus.Podium.Bleu;

import CrazyCircus.Podium.Rouge;

import CrazyCircus.Sequence;

import CrazyCircus.Situation;

import CrazyCircus.Tour;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import java.util.ArrayList;

import static org.junit.jupiter.api.Assertions.\*;

class **TourTest** {

    Bleu podiumBleu;

    Rouge podiumRouge;

    Situation situation;

    @BeforeEach

    void **setUp**() {

        podiumBleu = new **Bleu**("BLEU");

        podiumBleu.**ajouteAnimal**(new **Lion**());

        podiumBleu.**ajouteAnimal**(new **Ours**());

        podiumRouge = new **Rouge**("ROUGE");

        podiumRouge.**ajouteAnimal**(new **Elephant**());

        situation = new **Situation**(podiumBleu, podiumRouge);

    }

*/\**

*@Test*

*void deplaceElephantSurPileBleu() {*

*Rouge podiumRougeAttendu = new Rouge("ROUGE");*

*Bleu podiumBleuAttendu = new Bleu("BLEU");*

*podiumBleuAttendu.ajouteAnimal(new Elephant());*

*podiumBleuAttendu.ajouteAnimal(new Lion());*

*podiumBleuAttendu.ajouteAnimal(new Ours());*

*Situation situationAttendue = new Situation(podiumBleuAttendu, podiumRougeAttendu);*

*Joueur joueur = new Joueur("PC");*

*joueur.creerSequence(new Sequence("LO"));*

*Tour tour = new Tour(situation, situationAttendue, joueur);*

*boolean joueurAreussi = false;*

*try {*

*joueurAreussi = tour.lancer();*

*} catch (Exception exception) {*

*System.out.println(exception.getMessage());*

*}*

*assertTrue(joueurAreussi);*

*}*

*\*/*

    @Test

    void **deplaceLionSurPileBleuEtMettreElephantAuSommet**() {

        Rouge podiumRougeAttendu = new **Rouge**("ROUGE");

        Bleu podiumBleuAttendu = new **Bleu**("BLEU");

        podiumRougeAttendu.**ajouteAnimal**(new **Elephant**());

        podiumRougeAttendu.**ajouteAnimal**(new **Lion**());

        podiumBleuAttendu.**ajouteAnimal**(new **Ours**());

        Situation situationAttendue = new **Situation**(podiumBleuAttendu, podiumRougeAttendu);

        Joueur joueur = new **Joueur**("PC");

        joueur.**creerSequence**(new **Sequence**("KIMA"));

        Tour tour = new **Tour**(situation, situationAttendue, joueur);

*//        System.out.println(tour);*

        try {

            System.out.**println**(tour);

            tour.**lancer**();

*//System.out.println(tour);*

        } catch (Exception exception) {

            System.out.**println**(exception.**getMessage**());

        }

**assertEquals**(1, joueur.**recupererPoints**());

    }

}

package CrazyCircus.tests;

import CrazyCircus.Animal.Animal;

import CrazyCircus.Animal.Lion;

import CrazyCircus.Animal.Ours;

import CrazyCircus.Animal.Elephant;

import CrazyCircus.Ordres;

import CrazyCircus.Podium.Bleu;

import CrazyCircus.Podium.Rouge;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import CrazyCircus.Situation;

import java.util.ArrayList;

class **SituationTest** {

    Bleu podiumBleu;

    Rouge podiumRouge;

    Situation situation;

    @BeforeEach

    void **setUp**() {

        Lion lion = new **Lion**();

        Ours ours = new **Ours**();

        Elephant elephant = new **Elephant**();

        podiumBleu = new **Bleu**("BLEU");

        podiumBleu.**ajouteAnimal**(lion);

        podiumBleu.**ajouteAnimal**(ours);

        podiumRouge = new **Rouge**("ROUGE");

        podiumRouge.**ajouteAnimal**(elephant);

        situation = new **Situation**(podiumBleu, podiumRouge);

    }

    @Test

    void **executerOrdreKI**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.KI);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**("OURS", animauxSurPodiumBleu.**get**(0).**recupereNom**());

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

**assertEquals**("LION", animauxSurPodiumRouge.**get**(0).**recupereNom**());

**assertEquals**("ELEPHANT", animauxSurPodiumRouge.**get**(1).**recupereNom**());

    }

    @Test

    void **executerOrdreLO**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.LO);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**("ELEPHANT", animauxSurPodiumBleu.**get**(0).**recupereNom**());

**assertEquals**("LION", animauxSurPodiumBleu.**get**(1).**recupereNom**());

**assertEquals**("OURS", animauxSurPodiumBleu.**get**(2).**recupereNom**());

    }

    @Test

    void **executerOrdreSO**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.SO);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**("ELEPHANT", animauxSurPodiumBleu.**get**(0).**recupereNom**());

**assertEquals**("OURS", animauxSurPodiumBleu.**get**(1).**recupereNom**());

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

**assertEquals**("LION", animauxSurPodiumRouge.**get**(0).**recupereNom**());

    }

    @Test

    void **executerOrdreNI**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.NI);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**("OURS", animauxSurPodiumBleu.**get**(0).**recupereNom**());

**assertEquals**("LION", animauxSurPodiumBleu.**get**(1).**recupereNom**());

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

**assertEquals**("ELEPHANT", animauxSurPodiumRouge.**get**(0).**recupereNom**());

    }

    @Test

    void **executerOrdreMA**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.MA);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**("LION", animauxSurPodiumBleu.**get**(0).**recupereNom**());

**assertEquals**("OURS", animauxSurPodiumBleu.**get**(1).**recupereNom**());

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

**assertEquals**("ELEPHANT", animauxSurPodiumRouge.**get**(0).**recupereNom**());

    }

    @Test

    void **elephantSurPodiumBleuOursLionSurPodiumRouge**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.KI);

        ordres.**add**(Ordres.KI);

        ordres.**add**(Ordres.MA);

        ordres.**add**(Ordres.LO);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**(1, animauxSurPodiumBleu.**size**());

**assertEquals**("ELEPHANT", animauxSurPodiumBleu.**get**(0).**recupereNom**());

**assertEquals**(2, animauxSurPodiumRouge.**size**());

**assertEquals**("OURS", animauxSurPodiumRouge.**get**(0).**recupereNom**());

**assertEquals**("LION", animauxSurPodiumRouge.**get**(1).**recupereNom**());

    }

    @Test

    void **oursElephantSurPodiumBleuLionSurPodiumRouge**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.SO);

        ordres.**add**(Ordres.NI);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**(2, animauxSurPodiumBleu.**size**());

**assertEquals**("OURS", animauxSurPodiumBleu.**get**(0).**recupereNom**());

**assertEquals**("ELEPHANT", animauxSurPodiumBleu.**get**(1).**recupereNom**());

**assertEquals**(1, animauxSurPodiumRouge.**size**());

**assertEquals**("LION", animauxSurPodiumRouge.**get**(0).**recupereNom**());

    }

    @Test

    void **elephantLionOursSurPodiumRouge**() {

        ArrayList<Ordres> ordres = new ArrayList<Ordres>();

        ordres.**add**(Ordres.SO);

        ordres.**add**(Ordres.NI);

        ordres.**add**(Ordres.KI);

        ordres.**add**(Ordres.MA);

        ordres.**add**(Ordres.KI);

        situation.**executerOrdres**(ordres);

        ArrayList<Animal> animauxSurPodiumRouge = podiumRouge.**recupererAnimaux**();

        ArrayList<Animal> animauxSurPodiumBleu = podiumBleu.**recupererAnimaux**();

**assertEquals**(0, animauxSurPodiumBleu.**size**());

**assertEquals**(3, animauxSurPodiumRouge.**size**());

**assertEquals**("ELEPHANT", animauxSurPodiumRouge.**get**(0).**recupereNom**());

**assertEquals**("LION", animauxSurPodiumRouge.**get**(1).**recupereNom**());

**assertEquals**("OURS", animauxSurPodiumRouge.**get**(2).**recupereNom**());

    }

}

package CrazyCircus.tests;

import java.util.ArrayList;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import CrazyCircus.Sequence;

import CrazyCircus.Ordres;

class **SequenceTest** {

    @Test

    void **recupereOrdre**() {

        Sequence sequence = new **Sequence**("KIMALONI");

        ArrayList<Ordres> ordresExpecte = new ArrayList<>();

        ordresExpecte.**add**(Ordres.KI);

        ordresExpecte.**add**(Ordres.MA);

        ordresExpecte.**add**(Ordres.LO);

        ordresExpecte.**add**(Ordres.NI);

        ArrayList<Ordres> ordresActuel = sequence.**recupereOrdre**();

**assertEquals**(ordresExpecte, ordresActuel);

    }

}

package CrazyCircus.tests;

import CrazyCircus.Joueur;

import CrazyCircus.Partie;

import org.junit.jupiter.api.Test;

import java.util.ArrayList;

import static org.junit.jupiter.api.Assertions.\*;

class **PartieTest**

{

    @Test

    void **recupererJoueurs**() throws Exception {

        Partie partie = new **Partie**("PC TC");

        partie.**lancerPartie**();

        ArrayList<Joueur> listDesJoueurs = partie.**recupererJoueurs**();

**assertEquals**(2, listDesJoueurs.**size**());

**assertEquals**("PC", listDesJoueurs.**get**(0).**getIdentifiant**());

**assertEquals**("TC", listDesJoueurs.**get**(1).**getIdentifiant**());

    }

    @Test

    void **testRecupererJoueurs**() {

    }

}

package CrazyCircus.tests;

import CrazyCircus.Animal.Elephant;

import CrazyCircus.Animal.Lion;

import CrazyCircus.Animal.Ours;

import CrazyCircus.ComparateurSituations;

import CrazyCircus.Podium.Bleu;

import CrazyCircus.Podium.Rouge;

import CrazyCircus.Situation;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class **ComparateurSituationsTest** {

    Bleu podiumBleu;

    Rouge podiumRouge;

    Bleu podiumBleuSouhaite;

    Rouge podiumRougeSouhaite;

    @BeforeEach

    void **setUp**() {

        podiumBleu = new **Bleu**("BLEU");

        podiumRouge = new **Rouge**("ROUGE");

        podiumBleuSouhaite = new **Bleu**("BLEU");

        podiumRougeSouhaite = new **Rouge**("ROUGE");

    }

    @Test

    void **situationsNeSontPasIdentiques**() {

        podiumBleu.**ajouteAnimal**(new **Lion**());

        podiumBleu.**ajouteAnimal**(new **Ours**());

        podiumRouge.**ajouteAnimal**(new **Elephant**());

        Situation actuelle = new **Situation**(podiumBleu, podiumRouge);

        podiumBleuSouhaite.**ajouteAnimal**(new **Elephant**());

        podiumRougeSouhaite.**ajouteAnimal**(new **Ours**());

        podiumRougeSouhaite.**ajouteAnimal**(new **Lion**());

        Situation souhaitee = new **Situation**(podiumBleuSouhaite, podiumRougeSouhaite);

        ComparateurSituations comparateur = new **ComparateurSituations**(actuelle, souhaitee);

        boolean situationsSontIdentiques = comparateur.**situationsSontIdentiques**();

**assertFalse**(situationsSontIdentiques);

    }

    @Test

    void **ordreNEstPasIdentique** () {

        podiumBleu.**ajouteAnimal**(new **Elephant**());

        podiumBleu.**ajouteAnimal**(new **Lion**());

        podiumRouge.**ajouteAnimal**(new **Ours**());

        Situation actuelle = new **Situation**(podiumBleu, podiumRouge);

        podiumBleuSouhaite.**ajouteAnimal**(new **Lion**());

        podiumBleuSouhaite.**ajouteAnimal**(new **Elephant**());

        podiumRougeSouhaite.**ajouteAnimal**(new **Ours**());

        Situation souhaitee = new **Situation**(podiumBleuSouhaite, podiumRougeSouhaite);

        ComparateurSituations comparateur = new **ComparateurSituations**(actuelle, souhaitee);

        boolean situationsSontIdentiques = comparateur.**situationsSontIdentiques**();

**assertFalse**(situationsSontIdentiques);

    }

    @Test

    void **situationsSontIdentiques**() {

        podiumBleu.**ajouteAnimal**(new **Elephant**());

        podiumRouge.**ajouteAnimal**(new **Ours**());

        podiumRouge.**ajouteAnimal**(new **Lion**());

        Situation actuelle = new **Situation**(podiumBleu, podiumRouge);

        podiumBleuSouhaite.**ajouteAnimal**(new **Elephant**());

        podiumRougeSouhaite.**ajouteAnimal**(new **Ours**());

        podiumRougeSouhaite.**ajouteAnimal**(new **Lion**());

        Situation souhaitee = new **Situation**(podiumBleuSouhaite, podiumRougeSouhaite);

        ComparateurSituations comparateur = new **ComparateurSituations**(actuelle, souhaitee);

        boolean situationsSontIdentiques = comparateur.**situationsSontIdentiques**();

**assertTrue**(situationsSontIdentiques);

    }

}

package CrazyCircus.tests;

import CrazyCircus.Carte;

import CrazyCircus.Situation;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class **CarteTest** {

    @Test

    void **recupererSituation**() throws Exception {

        Carte carte = new **Carte**();

        Situation situation = carte.**recupererSituation**(0);

        String afficher = situation.**recupererPodiumBleu**().**recupererNom**();

        System.out.**println**(afficher);

    }

}

package CrazyCircus.tests.Podium;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import CrazyCircus.Podium.Bleu;

class **BleuTest** {

    @Test

    void **ajouteAnimal**() {

        Bleu podiumBleu = new **Bleu**("BLEU");

    }

    @Test

    void **executerOrdre**() {

    }

}

package CrazyCircus.tests.Animal;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import CrazyCircus.Animal.Elephant;

import static org.junit.jupiter.api.Assertions.\*;

class **ElephantTest** {

    @Test

    void **creeElephant**(){

        Elephant elephant = new **Elephant**();

**assertEquals**("ELEPHANT", elephant.**recupereNom**());

    }

}