amU Faculté des sciences Aix Marseille Université

Année universitaire 2024-2025

Site : ☒ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS

Sujet de : ☒ 1er semestre ☐ 2ème semestre ☐ Session 2 Durée de l'épreuve : 2h

Examen de : L3 Nom du diplôme : Licence informatique : parcours math-info

Code du module : SIN5U34 Libellé du module : Initiation au Génie logiciel

Calculatrices autorisées : NON Documents autorisés : OUI, notes de Cours, supports de cours

---

1 Gestion de groupes de coupe du monde

Question 1 : Quelles parties du code doivent être modifiées pour ajouter une nouvelle méthode de calcul des points qui donne 3 points pour une victoire (et toujours 1 point pour un nul et 0 pour une défaite) ?

On doit modifier le code interne de la méthode `pointsFor`. On est donc en violation d'OCP car il n'est pas possible de facilement ajouter une nouvelle méthode de calcul sans modifier le code interne de la classe `Match`.

Question 2 : Écrivez le code des classes `PointsFunctionOldRuleFactory` et `PointsFunctionNewRuleFactory` ainsi que le code de la ou les classe(s) des objets créés par les deux versions de la méthode `createPointsFunction`.

```
public class PointsFunctionNewRuleFactory implements PointsFunctionFactory {
    public ScoreFunction createPointsFunction() {
        return new PointsFunction(3,1);
    }
}
```

```
public class PointsFunctionOldRuleFactory implements PointsFunctionFactory{
    public ScoreFunction createPointsFunction() {
        return new PointsFunction(2,1);
    }
}
```

```
public class PointsFunction implements ScoreFunction{
    private final int pointsForWin;
    private final int pointsForDraw;

    public PointsFunction(int pointsForWin, int pointsForDraw) {
        this.pointsForWin = pointsForWin;
        this.pointsForDraw = pointsForDraw;
    }
    public int score(Team team, Match match) {
        if (match.isWonBy(team)) {
            return pointsForWin;
        }
        if (match.isADrawWith(team)) {
            return pointsForDraw;
        }
        return 0;
    }
}
```

Question 3 : Écrivez le code de la classe `ScoreComparator`.

```java
public class ScoreComparator implements Comparator<Team> {
    private final Match[] matches;
    private final ScoreFunction scoreFunction;

    public ScoreComparator(Match[] matches, ScoreFunction scoreFunction) {
        this.matches = matches;
        this.scoreFunction = scoreFunction;
    }

    public int compare(Team t1, Team t2) {
        int result = 0;
        for (Match match : matches) {
            result += scoreFunction.score(t2, match) - scoreFunction.score(t1,
                match);
        }
        return result;
    }
}
```

Question 4 : Écrivez le code de la classe `CompositeComparator<T>`.

```java
public class CompositeComparator<T> implements Comparator<T> {
    private final Comparator<T>[] comparator;

    public CompositeComparator(Comparator<T>[] comparator) {
        this.comparator = comparator;
    }

    @Override
    public int compare(T o1, T o2) {
        for (Comparator<T> comparator : comparator) {
            int comparisonResult = comparator.compare(o1, o2);
            if (comparisonResult != 0) {
                return comparisonResult;
            }
        }
        return 0;
    }
}
```

Question 5 : Écrivez le code de la méthode `sort` de la classe `Group`.

```java
public void sort() {
        ScoreFunction pointsFunctionNewRule =
            new PointsFunctionNewRuleFactory().createPointsFunction();
        Comparator<Team> pointComparator =
            new ScoreComparator(matches, pointsFunctionNewRule);
        Comparator<Team> goalDifferenceComparator =
                new ScoreComparator(matches, (t,m) -> m.goalDifferenceFor(t));
        Comparator<Team> goalCountComparator =
                new ScoreComparator(matches, (t,m) -> m.goalCountFor(t));
```

```
        Comparator<Team> comparator = new CompositeComparator<>(List.of(
            pointComparator,
                goalDifferenceComparator, goalCountComparator));
        Arrays.sort(teams, comparator);
    }
```

Question 6 : Décrivez une réorganisation du code permettant de changer la règle de calcul des points en modifiant uniquement une seule ligne de la méthode `main`.

Une façon de procéder est d'ajouter un attribut de type `ScoreFunction` (avec le paramètre correspondant dans le constructeur) qui correspondra à la fonction de calcul des points choisie pour le classement. Cet attribut sera ensuite utilisé dans `sort` pour construire le premier comparateur.

```
public class Group {
    private final Team[] teams;
    private final Match[] matches;
    private final ScoreFunction pointsFunction;

    public Group(Team[] teams, Match[] matches, ScoreFunction pointsFunction)
        {
        this.teams = teams;
        this.matches = matches;
        this.pointsFunction = pointsFunction;

    }

    public void sort() {
        Comparator<Team> pointComparator = new ScoreComparator(matches,
            pointsFunction);
        Comparator<Team> goalDifferenceComparator =
                new ScoreComparator(matches, (t,m) -> m.goalDifferenceFor(t));
        Comparator<Team> goalCountComparator =
                new ScoreComparator(matches, (t,m) -> m.goalCountFor(t));
        Comparator<Team> comparator = new CompositeComparator<>(List.of(
            pointComparator,
                goalDifferenceComparator, goalCountComparator));
        Arrays.sort(teams, comparator);
    }
```

Dans le `main`, on utilisera une instance de `PointsFunctionFactory` pour créer l'instance à donner au constructeur du groupe. Il suffira de changer le constructeur utilisé pour créer la `PointsFunctionFactory` pour changer la méthode de calcul des points.

```
     public static void main(String[] args) {
        Team chili = new Team("Chili");
        Team italy = new Team("Italie");
        Team france = new Team("France");
        Team usa = new Team("USA");
        Team[] teams = { chili, italy, france, usa };
        Match[] matches = {
                new Match(usa, italy, 2, 2),
                new Match(usa, chili, 2, 1),
                new Match(usa, france, 3, 0),
                new Match(italy, chili, 1, 0),
                new Match(italy, france, 3, 0),
                new Match(chili, france, 1, 1)
        };
        PointsFunctionFactory pointsFunctionFactory = new
            PointsFunctionNewRuleFactory();
        ScoreFunction pointsFunction = pointsFunctionFactory.
            createPointsFunction();
        Group group = new Group(teams, matches, pointsFunction);
```

```
18          group.sort();
19          for (Team team : group.teams())
20              System.out.println(team.name());
```