TP3 – Premier projet JAVA

Dans ce projet en JAVA, l'objectif était de s'occuper de la gestion d'une course cycliste.
Nous avons donc créé deux classes : Cycliste et Principal. Dans la classe Cycliste nous définissons l'objet Cycliste et définissons les attributs nécessaires (définir le nom, le numéro, le rang, le chrono, les attributs booléens pour savoir si le cycliste a fini, abandonné, voire disqualifié), ainsi que les getters et setters. Dans la classe Principal, nous définissons les différentes méthodes.

```
public class Cycliste {
    private String name;
    private int number;
    private int rank;
    private boolean finished;
    private boolean abandon;
    private boolean disqualified;
    private int chrono;
```

Pour générer les cyclistes, nous avons réalisé le tableau allRunners issu de Cycliste, et en sachant combien il y a de cyclistes dans la course en ayant demandé à l'utilisateur, nous ajoutons avec une boucle for le nouveau cycliste en enregistrant automatiquement son rang dans l'ordre (i+1).

```
for (int i=0 ; i < a ; i++) {
    String str;
    // get name of the runner
    Scanner s1 = new Scanner(System.in);
    System.out.print("Donnez le nom du coureur suivant :");
    str = s1.nextLine();
    allRunners[i] = new Cycliste(str,i,(i+1), finished: false, abandon: false, disqualified: false, chrono: 0);
}
```

Pour afficher la liste des cyclistes, nous parcourons le tableau allRunners en affichant pour chaque attribut le getter de la classe Cycliste, puisque les attributs sont privés dans la classe Cycliste et que nous travaillons actuellement dans la classe Principal.

```java
static void displayCoureur(Cycliste[] allRunners, int a){
    // displays cyclistes
    for (int i=0; i < a ; i++) {
        System.out.println("Nom : " + allRunners[i].getName()
                + " / Num@ro : " + allRunners[i].getNumber()
                + " / Classement : " + allRunners[i].getRank()
                + " / A fini : " + allRunners[i].getFinished()
                + " / Abandon : " + allRunners[i].getAbandon()
                + " / Est disqualifi@ : " + allRunners[i].getDisqualified()
                + " / Temps : " + allRunners[i].getChrono()
```

Pour afficher le classement, l'objectif était de parcourir deux fois la liste des cyclistes, et de comparer chaque chrono avec le cycliste suivant. Si le premier cycliste possède un chrono supérieur par rapport au cycliste suivant, l'objectif était d'échanger ces deux cyclistes, en instanciant un nouvel objet Cycliste afin de pouvoir stocker le second cycliste lors du switch. Malheureusement nous rencontrons un bug dans cette méthode.

```java
static void displayRanking(Cycliste[] allRunners, int a){
    for (int i=0; i < a ; i++) {
        for (int j=0; j < a ; j++) {
            if (allRunners[i].getRank()>allRunners[j].getRank()) {
                Cycliste c = new Cycliste( name: " ", number: 0, rank: 0, finished: false, abandon: false, disqualified: false, chrono: 0);
                c = allRunners[j];
                allRunners[j]=allRunners[i];
                allRunners[i]=c;
            }
        }
    }
    displayCoureur(allRunners,a);
```

Pour enregistrer qu'un cycliste est arrivé, nous demandons à l'utilisateur de préciser le numéro du cycliste en question. Nous parcourons la liste des cyclistes, et lorsque nous arrivons au cycliste nommé, nous attribuons à l'attribut finished « true ».

```java
static Cycliste[] registerArrival(Cycliste[] allRunners, int a , int[] rank){
    //make sure the user selects an existing number
    int c=0;
    while (c==0){
        //int nb=-1;
        Scanner s3 = new Scanner(System.in);
        System.out.print("Quel cycliste est arriv@ ? ");
        int nb = s3.nextInt();
        for (int i=0; i < a ; i++) {
            if (nb==allRunners[i].getNumber()) {
                c=1;
                allRunners[nb].setFinished(true);
            }
        }
    }
    return allRunners;
}
```

Pour enregistrer l'abandon, la disqualification, le temps, et afficher le temps, nous avons utilisé le même principe. Parcourir le tableau de cyclistes, et attribuer à l'attribut en question le « true » ou le temps, grâce aux setters de la classe Cycliste.

Pour calculer l'écart de temps entre deux cyclistes, nous demandons à l'utilisateur de premièrement renter le premier coureur puis le second. Pour chaque coureur nous parcourons notre tableau de cyclistes, puis lorsque nous trouvons le bon numéro de cycliste, nous récupérons son chrono grâce au getChrono de la classe Cycliste. Ainsi nous faisons la différence des deux temps, en faisant attention quel chrono est supérieur pour ne pas avoir un nombre négatif.

```
if (allRunners[val1].getChrono() > allRunners[val2].getChrono()) {
    System.out.print("La diff@rence est de : " + (allRunners[val1].getChrono()-allRunners[val2].getChrono()) + " minutes.");
}
else {
    System.out.print("La diff@rence est de : " + (allRunners[val2].getChrono()-allRunners[val1].getChrono()) + " minutes.");
}
```