**Dossier de développement logiciel**

Interpréteur de commandes

---

**Une image contenant patinage, personne, skiant, neige

Description générée automatiquement**Poursuite par équipes en patinage de vitesse

*Crédit : Damien MEYER, AFP*

Jules Doumèche & Gwénolé Martin

2019 - Groupe 111

Sommaire

[I - Présentation du projet 3](#_Toc23336142)

[1 – Introduction 3](#_Toc23336143)

[2 – Entrées/Sorties 3](#_Toc23336144)

[II – Bilan de validation 3](#_Toc23336145)

[III – Bilan de projet 3](#_Toc23336146)

[IV – Annexes 3](#_Toc23336147)

[1 – Code source du sprint 5 3](#_Toc23336148)

[2 – Trace d’exécution du sprint 5 3](#_Toc23336149)

*Contexte du projet :*

Ce projet a été réalisé en période A (Semestre 1). Nous avons été encadrés par Mme Caraty, M. Alles-Bianchetti et M. Poitreneau.

# 

# I - Présentation du projet

## 1 – Objectif du projet

Dans le cadre du projet d’IAP de la période A (2019-2020), nous devions coder un interpréteur de commandes capable de gérer des poursuites par équipes en patinage de vitesse. Cette épreuve olympique est définie de la sorte : 2 équipes de 3 patineurs partent chacune à l’opposée d’une piste circulaire, le but étant pour chaque équipe de faire un nombre de tour donné le plus rapidement possible, l’équipe faisant le meilleur temps gagne. Le temps retenu est celui du dernier patineur de l’équipe.

## 2 – Entrées/Sorties

L’interpréteur de commandes doit être capable de lire 8 commandes, envoyées directement via la console ou avec une redirection d’un fichier texte (.txt). Les commandes que nous avons codées sont les suivantes : exit, definir\_parcours, definir\_nombre\_epreuves, inscrire\_equipe, afficher\_equipes, enregistrer\_temps, afficher\_temps et afficher\_temps\_equipes.

Voici un tableau récapitulatif des entrées/sorties pour chaque commande :

|  |  |  |
| --- | --- | --- |
| Nom de la commande | Entrée(s) | Sortie(s) |
| definir\_parcours | 1 entier positif | Définie le nombre de tours par épreuve |
| definir\_nombre\_epreuves | 1 entier positif | Définie le nombre d’épreuves |
| inscrire\_equipe | 4 chaines de caractères :  Pays + Nom de chaque patineur | Affiche le numéro de dossard de chaque patineur inscrit : « inscription dossard + num » |
| afficher\_equipes | Pas d’entrée | Affiche les deux équipes actuellement en compétition |
| enregistrer\_temps | 2 entiers et 1 float :  numéro de dossard + numéro de tour + temps | Enregistre le temps d’un patineur pour un tour donné |
| afficher\_temps | 1 entier positif :  le numéro de dossard | Affiche le temps d’un patineur (pour chaque tour) |
| afficher\_temps\_equipes | 1 entier positif :  le numéro du tour | Affiche les temps des équipes pour un tour donné (par ordre croissant) |

# II – Bilan du développement

## 1 – Organisation des tests

Pour développer cet interpréteur, nous avons utilisé une méthode de développement par sprint, avec 5 niveaux de sprints différents, chaque sprint permettant d’implémenter une fonctionnalité supplémentaire dans le programme. Nous avons utilisé les jeux de données de test fournis pour tester chaque sprint, plus quelques jeux de données personnalisés, pour être sûr que les sprints ne comportaient pas de bug.

## 2 – Bilan de validation

Le sprint de plus haut niveau validé est le sprint 5. Le fichier *out* est identique au *out* de référence. Ci-dessous une capture d’écran sur DiffChecker pour le Jeu de Données de Test (JDT) du sprint 5 :

Une image contenant capture d’écran

Description générée automatiquement

# III – Bilan de projet

Comme nous avions commencé une semaine avant le cours de projet, nous avons refais plusieurs fois notre sprint 1. Nous avions au tout début une version sans structure, que nous avons entièrement refaite, puis optimisée pour partir sur une base « fiable » pour le reste du développement.

Pour le sprint 2, nous avons eu quelques difficultés le temps de bien comprendre comment attribuer les valeurs aux bons endroits grâce aux pointeurs, comme nous avions une idée claire de ce que nous devions faire, le reste n’a pas posé de problème.

Le sprint 3 ne nous a pas posé de problème, nous l’avons codé rapidement.

Nous avons codé les sprints 4 et 5 en même temps, il était plus simple pour nous d’implémenter les deux fonctionnalités en même temps, une fois que detection\_fin\_poursuite fonctionnait, il était simple d’implémenter detection\_fin\_competition.

Le développement n’a pas toujours été simple notamment au niveau de l’algorithme de tri pour le sprint 5, qui a nécessité de nombreux tests mais en prenant le temps de poser les idées (en écrivant sur papier le déroulement de la fonction par exemple), nous avons pu surmonter toutes les difficultés.

# IV – Annexes

## 1 – Code source du sprint 5

## 2 – Trace d’exécution du sprint 5