Clement SANCHEZ | Nicolas SCHMITT

Compte rendu JAVA 1

Rapport sur le projet de tennis

Table des matières

[Introduction 2](#_Toc25542942)

[Diagramme de classe 3](#_Toc25542943)

[Description 4](#_Toc25542944)

[Difficultés rencontrées 10](#_Toc25542945)

# Introduction

L’objectif de ce projet était de créer une simulation de tournoi de tennis en utilisant les spécificités du langage Java vu en cours. Le programme permet de gérer l’organisation d’un « Grand Chelem » qui comporte 128 joueurs dans chaque catégorie, ici seront considérés les catégories simples. Il permet entre autres de créer des matchs où s’affronteront les joueurs, arbitrés par des arbitres et assistés par des spectateurs.

# Diagramme de classe

Une image contenant texte, carte

Description générée automatiquement

Les flèches en pointillé représentent les compositions.

Celle avec un bout blanc, les héritages.

# Description

|  |
| --- |
| **Arbitre** |
|  |
| void annoncerScoreMatch()  void annoncerScoreSet()  void annoncerScoreJeu()  void annoncerVainqueurMatch()  Arbitre generer() |

Cette classe permet de créer un objet de type Arbitre. Cette classe hérite de la classe abstraite Personne. Les arbitres ont pour utilités d’annoncer les scores lors des matchs.

La méthode generer permet de créer un objet Arbitre avec des attributs aléatoire.

|  |
| --- |
| **Billet** |
| int prix  Tribune tribune  int place |
| int getPrix()  Tribune getTribune()  int getPlace()  String toString() |

|  |
| --- |
| **Date** |
| int jour  int mois  int annee |
| int getJour()  int getMois()  int getAnnee()  String toString()  Date genererAleatoire()  Date createInterface() |

Cette classe permet de créer un objet de type Date. Les dates caractérisent les dates de naissance et de décès des protagonistes. Les variables d’instance dateNaissance et dateDeces sont implémentés dans la classe Personne.

La méthode generer permet de créer un objet Date avec des attributs aléatoire.

La méthode createInterface créer une interface H/M demandant à l’utilisateur de renseigner les variables d’instances d’une date.

Cette classe permet de créer un objet de type Billet. Les billets sont achetés par les spectateurs afin d’assister aux matchs.

La méthode toString renvoie une chaine de caractère comporte les information d’une date écrit sous la forme jj/mm/aaaa.

Cette classe permet de créer un objet de type Echange. Un objet de type Echange est créer à partir de la classe Jeu. Les échanges sont le plus bas niveau de jeu lors de match entre deux joueurs, c’est-à-dire qu’il commence lorsqu’un joueur d’une équipe sert et se fini lorsque la balle se retrouve soit en faute, soit dans le filet.

La méthode play ici renvoie l’équipe qui remporte l’échange.

|  |
| --- |
| **Echange** |
| Arbitre [] arbitres  Joueur [] equipe1  Joueur [] equipe2  Spectateur [] spectateurs  int service |
| Joueur [] play() |

|  |
| --- |
| **Jeu** |
| Arbitre [] arbitres  Joueur [] equipe1  Joueur [] equipe2  Spectateur [] spectateurs  int service |
| Joueur [] play() |

Cette classe permet de créer un objet de type Jeu. Un objet de type Jeu est créer à partir de la classe Set et permet de créer un objet de type Echange. Un jeu se termine lorsque qu’une équipe remporte 40 points ont remporté un point après avoir eu un avantage (AV).

La méthode play ici renvoie l’équipe qui remporte le jeu.

Cette classe permet de créer un objet de type Joueur. Cette classe hérite de la classe abstraite Personne et implémente l’interface ActionSpectateur. Les joueurs sont les principaux protagonistes où ils s’affrontent lors des tournois, ils pourront aussi assister aux autres matchs lorsqu’ils seront dans les tribunes.

La méthode toString renvoie une chaine de caractère décrivant les attributs d’un joueur.

La méthode servir représente le service d’un joueur où il a une certaine probabilité décrite dans ces statistiques de réussir son service.

La méthode renvoyer, de la même manière que la méthode servir permet au joueur de renvoyer la balle avec une certaine probabilité de réussir son renvoi, de la renvoyer en faute ou de la renvoyer dans le filet.

La méthode play permet d’une part de savoir si l’échange est terminé et d’autre part de sélectionner le prochain joueur qui fera le service.

La méthode generer permet de créer un objet Joueur avec des attributs aléatoire.

|  |
| --- |
| **Joueur** |
| Main main  String sponsor  String entraineur  int classement  Vetement vetement  Couleur couleur  Statistique statistique |
| Main getMain()  String getSponsor()  String getEntraineur()  int getClassement()  Vetement getVetement()  Couleur getCouleur()  Statistique getStatistique()  void setMain()  void setSponsor()  void setEntraineur()  void setClassement()  void setVetement()  void setCouleur()  String toString()  Action servir()  Action renvoyer()  Action play()  Joueur generer()  void applaudir()  void crier()  void huer()  void dormir() |

|  |
| --- |
| **Match** |
| Categorie categorie  Phase phase  Arbitre [] arbitres  Joueur [] equipe1  Joueur [] equipe2  Spectateur [] spectateurs |
| Categorie getCategorie()  Phase getPhase()  Arbitre [] getArbitres()  Joueur [] getEquipe1()  Joueur [] getEquipe2()  Spectateur [] getSpectateurs()  Joueur [] play()  String toString() |

Cette classe abstraite est la classe mère des classes filles Arbitre, Joueur et Spectateur. Elle y répertorie les variables d’instances et méthodes communes à ses classes filles. De plus, elle possède des variables statiques sous forme de tableau (ArrayList) :

* prenomMasculin
* prenomFeminin
* nomFamille
* villeNaissance
* pays
* sponsor

Ces tableaux sont initialisés par l’appel de la fonction getData de la classe FileData qui lit les donnes dans les différents fichiers portant le même nom que les variables. Ils serviront pour la génération aléatoire d’arbitres, de joueurs et de spectateurs.

La méthode toString renvoie une chaine de caractère décrivant les attributs communs entre arbitres, joueurs et spectateurs.

Cette classe permet de créer un objet de type Match. Les matchs sont generer à partir de la classe Tournoi au nombre de 127. Ils matchs opposent deux équipes/joueurs, arbitrer par au moins un arbitre et assister par des spectateurs.

La méthode toString renvoie une chaine de caractère décrivant les attributs d’un match.

La méthode play permet de lancer le match en 2 set, plus un set décisif si il y a égalité. Elle renvoie l’équipe/joueur qui remporte la match.

|  |
| --- |
| **Personne** |
| String nomNaissance  String nomCourant  String prenom  String surnom  String lieuNaissance  String nationalite  Date dateNaissance  Date dateDeces  int taille  int poids |
| String getNomNaissance()  String getNomCourant()  String getPrenom()  String getSurnom()  String getLieuNaissance()  String getNationalite()  Date getDateNaissance()  Date getDateDeces()  int getTaille()  int getPoids()  int getAge()  void setNomCourant()  void setSurnom()  void setNationalite()  void setTaille()  void setPoids()  String toString() |

Cette classe permet de créer un objet de type Set. Un objet de type Set est créer à partir de la classe Match et permet de créer un objet de type Jeu. Un Set se termine lorsque qu’une équipe remporte 5 jeux ou deux jeux d’avances dans le cas où le score aurait été de 6-5.

La méthode play ici renvoie l’équipe qui remporte le set.

|  |
| --- |
| **Set** |
| Arbitre [] arbitres  Joueur [] equipe1  Joueur [] equipe2  Spectateur [] spectateurs  int service |
| Joueur [] play() |

|  |
| --- |
| **Spectateur** |
| Vetement vetement  Couleur couleur  Billet Billet |
| Vetement getVetement()  Couleur getCouleur()  Billet getBillet()  void setVetement()  void setCouleur()  void acheterBillet()  void applaudir()  void crier()  void huer()  void dormir()  String toString()  Spectateur generer() |

Cette classe permet de créer un objet de type Spectateur. Cette classe hérite de la classe abstraite Personne et implémente l’interface ActionSpectateur.

La méthode toString renvoie une chaine de caractère décrivant les attributs d’un spectateur.

La méthode generer permet de créer un objet Joueur avec des attributs aléatoire.

Cette classe permet de créer un objet de type StatMach. Elle répertorie les statistiques du joueur pour chacun de ses matchs.

Les méthodes de type addxxx permettent d’incrémenter les variables d’instances.

Les méthodes de type updateVitesse

|  |
| --- |
| **StatMatch** |
| int nbSet  int nbJeu  int nbEchange  int nbPoint  int nbAce  int nbPremierService  int nbSecondService  int nbDoubleFaute  int nbBalleBreak  int nbBalleBreakRemporte  float vitesseBallePremierService  float vitesseBalleSecondService |
| int getNbSet()  int getNbJeu()  int getNbEchange()  int getNbPoint()  int getNbAce()  int getNbPremierService()  int getNbSecondService()  int getNbDoubleFaute()  int getNbBalleBreak()  int getNbBalleBreakRemporte()  float getVitesseBallePremierService()  float getVitesseBalleSecondService()  void addSet()  void addJeu()  void addNbEchange()  void addNbPoint()  void addNbAce()  void addNbPremierService()  void addNbSecondService()  void addNbDoubleFaute()  void addNbBalleBreak()  void addNbBalleBreakRemporte()  void updateVitesseBallePremierService()  void updateVitesseBalleSecondService() |

|  |
| --- |
| **Statistique** |
| int matchJoues  int matchRemporte  int nbTournoi  int gains  double probaServiceCorrect  double probaRenvoieCorrect  double probaRenvoieFaute  LinkedList <StatMatch> statMatch |
| int getMatchJoues()  int getMatchRemporte()  int getNbTournoi()  int getGains()  double getProbaServiceCorrect()  double getProbaRenvoieCorrect()  double getProbaRenvoieFaute()  LinkedList <StatMatch> getStatMatch()  void setProbaServiceCorrect()  void setProbaRenvoieCorrect()  void setProbaRenvoieFaute()  void addMatchJoues()  void addMacthRemporte()  void addNbTournoi()  void addGains()  void addStatMatch() |

Cette classe permet de créer un objet de type Statistique. Elle y répertorie les statistiques globales, par match et les performances du joueur.

|  |
| --- |
| **Tournoi** |
| Ville ville  Surface surface  Categorie categorie  ArrayList <Joueur> joueurs  ArrayList <Match> matchs |
| Ville getVille()  Surface getSurface()  Categorie getCategorie()  ArrayList <Joueur> getJoueurs()  ArrayList <Match> getMatchs()  void genererMatch()  void genererJoueur()  Joueur play() |

Cette classe permet de créer un objet de type Tournoi.

La méthode genererMatch permet de créer les 127 matchs du tournoi.

La méthode genererJoueur permet de remplir la liste des 128 joueurs du tournoi.

La méthode play permet de joueur tous les matchs de manière automatique.

Cette classe abstraite permet de lire les données des fichiers textes et de les retransmettre aux variables statiques se trouvant dans la classe Personne.

|  |
| --- |
| **FileData** |
|  |
| ArrayList <String> getData()  ArrayList <String> getDataFromFile() |

|  |
| --- |
| **Menu** |
| Scanner scanner  String title  ArrayList <MenuItems> |
| void addChoice()  int afficherMenu()  int getChoice() |

Cette classe permet de créer un objet de type Menu. Elle est utilisée pour interagir avec l’utilisateur.

Cette classe permet de créer un objet de type MenuItems.

Custom exception qui permet de dire lorsqu’il y un conflit entre le genre des joueurs (dans un match / tournoi Homme, une femme se présente ou l’inverse )

|  |
| --- |
| **MenuItems** |
| String description  int value |
|  |

|  |
| --- |
| **InvalidGenderException** |
|  |
|  |

# Difficultés rencontrées

Une des difficultés rencontrées fût la lecture des données (utilisées pour générer des personnes de manière aléatoire) de chaque fichiers textes et leurs enregistrements dans les tableaux respectifs (*prenomMasculin*, *prenomFeminin*, …) se trouvant dans la classe *Personne*. La première étape était la création de la classe *FileData*.

La deuxième étape était l’enregistrement des données dans les fichiers tel qu’il y ait une chaine de caractère par ligne. La troisième étape était l’implémentation de la méthode *getDataFromFile(String data)* qui permettait de lire et d’enregistrer chacune des lignes du fichier *data.*txt dans un tableau de chaine de caractère *ArrayList <String>.* Ce tableau est ensuite renvoyer à la méthode *getData(String data)* qui renverra de nouveau par la suite aux variables statiques de la classe *Personne*.

L’intérêt de ne pas appeler directement la méthode *getDataFromFile(String data)* mais d’appeler la méthode *getData(String data)* en intermédiaire se justifie par l’utilisation d’une exception au moyen d’un *try*-*catch* dans cette dernière méthode au lieu de l’utiliser dans la classe *Personne*. De plus, il a fallu changer les types de donner des variables statiques de la classe *Personne* initialement de type *String []* en type *ArrayList* <*String*> impliquant des adaptations des méthodes *generer()* des classes *Arbitre*, *Joueur* et *Spectateur*.

Sur la class Menu et Menuitems, on a créé ces class pour permettre un développement plus générique et plus rapide de l’interface homme machine.

Elles ont l’avantage de normaliser l’affichage et de généraliser les menu, plus simple dans l’implémentation.

# Conclusion