Compte rendu SAE2-01 - POO

DAGNEAUX Nicolas - DEGRAEVE Paul - MARTEL Alexandre - Groupe D1

1 Introduction

Dans le cadre de notre BUT Informatique, nous avons comme projet la création d'une application d'aide à la prise de décision lors d'échanges scolaires dans différents pays. Ce rapport se concentrera sur la partie *Programmation Orientée Objet*.

Notre projet est disponible sur notre page Gitlab.

Référence du dernier commit :

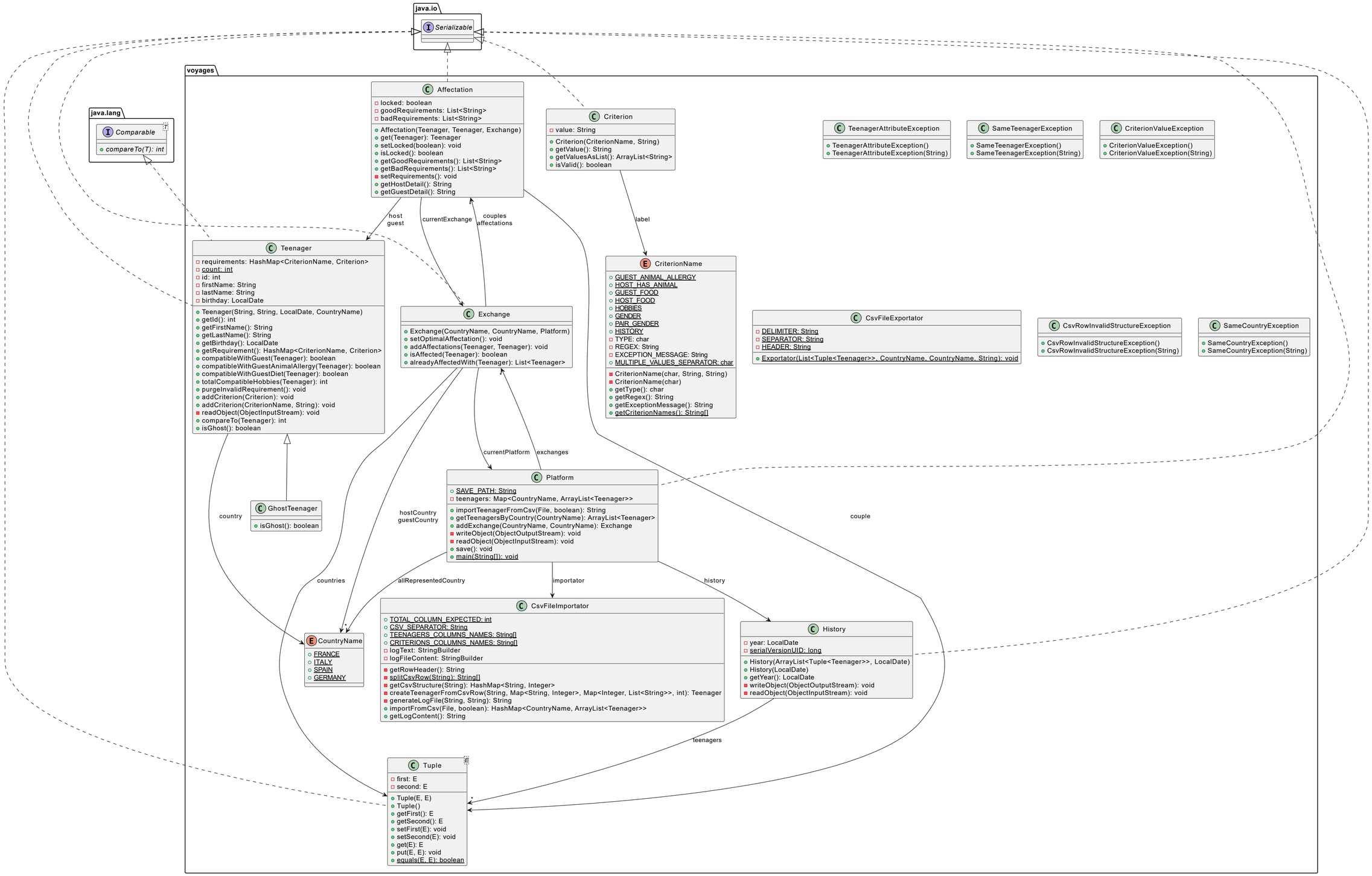
2 Présentation de l'application

2.1 Notice d'utilisation

Pour lancer l'application,

2.2 UML

2.2.1 Diagramme



Voici l'UML de notre package voyages, qui regroupe toutes les classes utilisées pour la partie POO.

La classe principale de ce package est la classe `Platform`. Cette classe contient un dictionnaire d'adolescents associés à leur pays ainsi qu'une chaîne de caractère contenant le chemin de sauvegarde de cette platforme. Cette classe implémente donc `java.io.Serializable`, pour pouvoir être

sauvegardée. Les pays d'origines des adolescents sont implémentés sous la forme d'une énumération des noms de pays, `CountryName`. Les adolescents sont eux représentés par la classe `Teenager`. C'est la 2eme classe la plus importante car elle contient toutes les informations concernant un adolescents (nom, prénom, critère...), et elle contient aussi toutes les méthodes permettant de gérer les adolescents (la compatibilité). Cette classe implémente donc `java.io.Serializable`, pour pouvoir être sauvegardée, mais aussi `java.lang.Comparable`. En effet, il est essentiel de pouvoir comparer, et donc trié, les adolescents. La classe `GhostTeenager` hérite de Teenager est sert uniquement à créer des adolescents fantômes pour équilibrer les listes d'adolescents lors de l'affectation à l'aide de la création d'un graphe. Elle contient une unique méthode `isGhost()` qui renvoie toujours vrai (`Teenager` possède aussi cette méthode qui elle renvoie toujours faux). Les critères des adolescents sont stockés dans un dictionnaire de noms de critères associés à ce critère. Les noms de critère sont stockés dans une énumération `CriterionName`. Quand aux critères en eux-même, ils sont représentés par la classe `Criterion`. Cette classe contient un seul attribut, la valeur du critère que l'adolescent rentre dans le formulaire. Cette classe implémente aussi `java.io.Serializable`.

2.3 Analyse technique

2.3.1 Validité des critères

2.3.2 Adolescents à retirer

2.3.3 Compatibilité avec les français

2.3.4 Imports/exports

2.3.5 Historique

2.3.6 Prise en compte de l'historique

2.3.7 Traitement des données

2.4 Analyse quantitative/qualitative