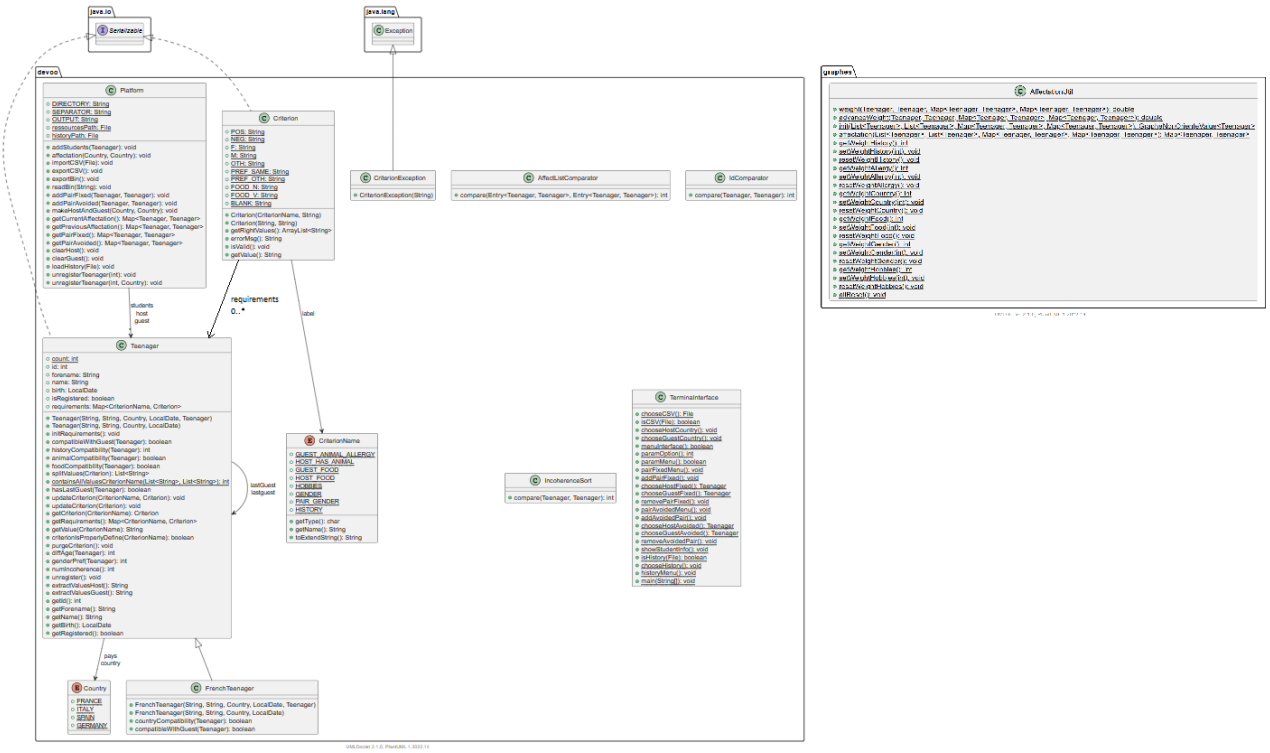
Rapport DEV

Exécution du programme :

Pour exécuter le programme vous avez besoin de cloner entièrement notre dépôt.
Les fichiers csv seront à mettre dans ./rsc/
Les fichiers bin d'historique seront à mettre dans ./hist/

Pour lancer la version en terminal : java -jar D2.jar
Pour lancer la version avec interface :
java --module-path ./lib/javafx/ --add-modules javafx.controls,javafx.fxml -jar ./rendu ihm/D2.jar



La classe "TerminalInterface" offres une interface utilisateur en ligne de commande pour lancer les programmes. Elle est composé uniquement de méthodes statique. Celle-ci permet aux utilisateurs d'interagir avec le système et d'exécuter les fonctionnalités disponibles.

La classe "Platform" est aussi un élément central dans notre programme, mettant en lien les données et les calculs. Elle est responsable de l'importation des fichier csv contenant des étudiants (Teenager) et de l'exportation des données (Sous forme d'historique binaire ou de csv), ainsi que de la visualisation des affectations calculer grâce à la classe "AffectationUtil".

La classe "Teenager" permet la création d'étudiants ayant un nom, un prénom, une date d'anniversaire, un pays et plusieurs critères spécifiques. La date d'anniversaire est de type

LocalDate, le pays est une encapsulation de l’enum "Country", le nom et le prénom sont stocké sous forme de String. Les critères sont quand à eux rangé dans une "HashMap" ayant comme clé un "CriterionName" et comme valeur un "Criterion".

La classe "FrenchTeenager" est une classe fille de la classe "Teenager" spécifique pour les étudiants français, nous avons fait cette classe car les étudiants français possède une règles de compatibilité spécifique (Si l’hôte et le visiteur ne partage pas au moins un hobbies commun il ne sont pas compatibles).

La classe "AffectationUtil" créé un graphe à partir de deux liste d’étudiants, une pour les hôtes et une pour les visiteurs. Grâce à des foreach qui parcours chacune des listes elle créé un certains nombres d’arêtes (Nombre d’hôte fois nombre de visiteur). Elle utilise aussi une méthodes pour calculer le poids de chacune des arêtes, poids que la classe va utiliser pour déterminer les connexions et construire le graphe correspondant.

La classe "Criterion" permet de créé des critères, ces critères sont par exemple, si l’hôte possède un chien ou par exemple les régimes, cette classe permet aussi de savoir si ce critères est valide à l’aide d’un système de levé d’exception si la valeur du Criterion n’est une des valeurs possibles qui sont définis de version statique dans notre code.

Pour les classes de tests :

- La classe "TestCriterion" test uniquement la fonction isValid(), pour cela nous avons créé tout une liste de critères avec des valeurs qui ne marche pas. Puis nous faisons un forEach sur cette liste et testons tout les critères.

- La classe "TestCriterionName" vérifie si le type de retour du CriterionName est bien celui mis à la base.

- Les tests réalisés dans la classe "TestPlatform" examinent si, pour un ensemble donné d'étudiants généré à partir d’un csv, l'affectation des binômes reste la même lors des exécutions répétées de l'algorithme d'affectation.

- Dans "TestTeenager" nous créons un certains nombre de teenager avec des critères plus ou moins valide, ensuite nous testons toute les différentes fonctions de compatibilité pour vérifié si tel ou tel étudiants sont bels et bien compatible les uns entre les autres. Nous testons aussi la fonction qui permet de purgé les critères non valide.