1. Object-oriented part (Java) [8% of your final mark]

Since this solution must follow the object-oriented paradigm, your program must be composed of a set of classes. Specifically, it must include, among others, the classes listed below.

In addition to the source code of your solution, you must also submit a document that includes a UML diagram of all your classes (showing attributes, associations and methods). Do not use static methods, except for the `main` *function. This document must also cite all references used to build your solution.*

- class `Recommendation` and its attributes:
  - `user`: the recommendation is for this user
  - `movie`: recommended movie
  - `score`: probability that the user will like this movie
  - `nUsers`: number of users who likes this movie
- class `Movie`
- class `User`
  - with its `userID`
  - list of liked movies
  - list of not liked movies
- class `RecommendationEngine`
  - Containing the `main` method
    - You specify the ID of the user for whom you want to produce recommendations, filename of the movies and filename of the ratings
      - `java RecommendationEngine 44 movies.csv ratings.csv`
    - The program displays the list of the `N` movies recommendations with their probability.
- plus any other classes, methods or attributes you judge necessary.

1. Partie Orientée-objet (Java) [8% de votre note finale]

Puisque votre solution doit se conformer au paradigme de programmation orienté objet, votre programme doit être composé d'un ensemble de classes. Plus spécifiquement, il doit inclure, parmi d'autres, les classes ci-dessous.

En plus de votre code source, vous devez aussi soumettre un diagramme UML de vos classes, montrant bien les associations, les attributs et les méthodes. Vous ne devez pas utiliser de méthodes statiques (sauf pour la fonction `main`). Ce document doit aussi inclure des références à toute source d'information utilisée pour réaliser votre solution.

- La classe `Recommendation` et ses attributs:
  - `user`: la recommandation est pour cet utilisateur
  - `movie`: le film recommandé
  - `score`: la probabilité que l'utilisateur aimer ace film
  - `nUsers`: nombre d'utilisateur ayant aimé ce film
- La classe `Movie`
- La classe `User`
  - avec son identificateur `userID`
  - sa liste de films aimés
  - sa liste de films non-aimés
- La classe `RecommendationEngine`
  - Contenant la méthode `main`
    - vous devez spécifier le ID de l'utilisateur pour lequel on veut obtenir les recommendation, le nom du fichier de films et le nom du fichier d'évaluations
      - `java RecommendationEngine 44 movies.csv ratings.csv`
    - En sortie, le programme imprime le nom des `N` films recommandés ainsi que leur probabilité.
- Ainsi que tout autre classe, méthodes ou attributs que vous jugez utiles.