Ambos integrantes do grupo contribuíram com 50% da realização do projeto, tanto no auxílio da formulação de algoritmos quanto na estruturação do código. A maior parte do projeto foi realizada por meio de chamada zoom em que era implementado ambas abordagens algorítmicas dos integrantes do grupo.

Ordenação da implementação das classes:

Foi definido uma classe para os skippers e uma para os Requests, a partir dessas classes foi definido duas coleções, a SkipperCollection e a RequestCollection onde as classes skippers e Request foram implementadas por delegação.

Após esse passo implementamos a coleção Schedule que executa, novamente por delegação as coleções RequestCollection e SkipperCollection.

Utilizamos o ficheiro Update que funciona como o ficheiro principal para a execução do software, o ficheiro em questão cria coleções de Skippers, Requests e Schedule a partir de três ficheiros de texto tipo Skipper, Request e Schedule. Em seguida, faz o pareamento das coleções skippers e Request na coleção Schedule por meio dos métodos: FindMatchingSkipper, MatchCol e TieBreaker e a partir da coleção Schedule produz duas versões atualizadas, uma do ficheiro Skipper e outra do ficheiro Schedule.

Classe Skipper:

Recebe os atributos nome, língua, categoria, preço, horas máximas de trabalho, horas acumuladas e a data.

Além disso possui três atributos responsáveis pela atualização da hora dos skippers e da hora e data do Schedule necessário para a escrita dos ficheiros Schedule e Skippers

Sua função consiste em criar objetos Skippers necessários para guardar as informações individuais de cada skipper e representá-los no código.

Classe Request:

Recebe os atributos nome, língua, categoria, tipo e número de horas de viagem.

Sua função consiste em criar objetos Request necessários para guardar as informações individuais de cada pedido e representá-los no código.

Classe SkipperCollection:

Gera uma coleção vazia e a partir do método createCollection() adiciona à coleção objetos da classe skipper, esse método implementa a classe skipper por delegação para gerar objetos skippers.

Classe RequestCollection:

Gera uma coleção vazia e a partir do método createCollection() adiciona à coleção objetos da classe Request, esse método implementa a classe Request por delegação para gerar objetos Request.

Classe Schedule:

Gera uma coleção vazia chamada match e a partir do método matchCol() adiciona à coleção tuplos de objetos da classe Request e objetos da classe Skipper, esse método utiliza outros dois métodos internos da classe Schedule: O FindMatchingSkipper() e o método TieBreaker().

Método FindMatchSkipper():

Este método foi usado na classe Schedule de modo a conseguirmos criar uma lista de possível match entre um objeto Request e vários objetos Skipper. Para a obtenção desta lista foi preciso relacionar as características do Request com as características dos Skippers e descobrir quais os Skippers que estavam aptos. Depois de criada a lista foi chamado o método TieBreaker(). Por fim este método devolve apenas o Skipper mais apto ao Request dado.

Método TieBreaker():

Este método recebe uma lista provinda do método FindMatchSkipper(), de todos os Skippers aptos a um dado Request. A função percorre a lista dada e realiza o critério de desempate, devolvendo por último o Skipper mais apto a dado Request.

Outros métodos da classe Schedule:

dateUpdate(): Atualiza a data e hora do Skipper conforme o Request que lhe foi fornecido.

dateUpdateSCH(): Atualiza a data e hora do começo do Request para determinado Skipper.

Implementações por fazer:

O programa corre tudo como deve de ser e cria os ficheiros de forma correta, apenas em casos específicos há erros na incrementação das horas, entretanto esses erros não comprometem o funcionamento do programa. Apesar deste erro do programa, o uso das classes foi conseguido para a realização dos outros testSets.