Toets Algoritmen en Datastructuren – 28-6-2022 V2

Algemeen

Download het zip-bestand sources.zip. Werk in je favoriete ontwikkelomgeving. Lees de opdrachten eerst goed door. Bij onduidelijkheden mag je aannames maken, maar vermeld die in je commentaar.

Je mag alle mogelijke hulpbronnen gebruiken, alleen niet samenwerken.

Zet duidelijk in commentaar bij alle te maken of aan te passen methodes en gekozen datastructuren:

- Antwoord op de gestelde vragen en uitleg van je antwoord
- Onderbouwing van gemaakte keuzes

Als je de main-methode van een project uitvoert, worden de gewenste test automatisch gestart.

Inleveren

Als je klaar bent, pak je de aangepaste Java-bestanden weer in één zipfile met twee sub-mappen: opgave1 en opgave 2. We willen alleen Java-sources zien, dus geen targets, class-files of projectbestanden.

Opgave 1

Maak een nieuw project "opgave1". Zet daar de sources uit opgave1 uit het zip-bestand in:

Main.java	Niets aan wijzigen!
Test.java	Niets aan wijzigen
Pirate.java	Niets aan wijzigen!
LinkedList.java	Aanpassen om de vijf opgaven uit te werken

De vijf deelopgaven voor deel 1 staan in de source van LinkedList.java

Opgave 2

Maak een nieuw project "opgave2". Zet daar de sources uit opgave2 uit het zip-bestand in:

Main.java	Niets aan wijzigen!
Test.java	Niets aan wijzigen!
Admirality.java	Wijzigen voor requirement 1,2,3 en 4
Pirate.java	Wijzigen voor requirement 5 en anders als je dat nodig vindt

Vergeet de zelfgebouwde LinkedList uit opgave1 en gebruik het collection framework voor het voldoen aan de volgende eisen.

Requirements

Admiraal Dreispitz is de eindbaas van een groep piraten. Hij wil de meldingen van hun veroveringen op een efficiënte manier bijhouden.

Algemene niet-functionele eisen:

Omdat er heel veel veroveringen worden verwacht, streeft Dreispitz naar tijdsefficiëntie van Big O(1) of maximaal Big O(log(n)) voor de methoden genoemd in de requirements. Bahalve voor de lijstjes, die blijven natuurlijk O(N). Help hem daarbij door de juiste containerklasse(n) te kiezen.

Voor de onderhoudbaarheid wil Dreispitz in de sources duidelijke onderbouwing van gemaakte keuzes, en verklaring van de **tijdsefficiëntie (Big O) in commentaar** bij alle te maken of aan te passen methodes en gekozen datastructuren.

Functionele requirement 1.

Piraten moeten bij hem melden welke schepen ze hebben buitgemaakt. Dreizpitz gebruikt hiervoor de methode: void addConquest(Pirate pirate, String Ship). Als een schip voor de tweede keer door dezelfde piraat wordt gemeld, wordt dat niet nog eens opgeslagen.

Functionele requirement 2.

Hij wil een alfabetische lijst van piraten die iets veroverd hebben plus per piraat hun veroveringen: void showPirates()

Functionele requirement 3.

Hij wil een alfabetische lijst van buitgemaakte schepen, met per schip de naam van de eerste piraat die het gemeld heeft: void showShips()

Functionele requirement 4.

Voor onderzoek naar bedrog wil hij een lijst van schepen die meer dan één keer zijn aangemeld, met daarbij de namen van de piraten: void showCheats()

Requirement 5 (niet-functioneel).

De methode Pirate.getByName() heeft nu lineaire tijdsefficiëntie. Maak die efficiënter door het array allPirates te vervangen door een andere collectie zodat er geen loop meer nodig is in getByName(). Wat verandert er hierdoor in de Big O?

Einde