Proeftentamen Datastructures Make IT Work

Naam	
Studentnr.	
Inlevertijd (door surveillant in te vullen	

Opdracht 1

Opdracht

Tekstbestand 'grades.txt' bevat meerdere regels met elk de resultaten van studenten. Per regel staat een *student-id* (int) en een *grade* (int, score van 50-100).

1. Schrijf een klasse `Grades` dat:
   - Deze file inleest (met een `Scanner`) en de waardes opslaat. Kies een geschikte datastructuur hiervoor.

2. Schrijf een main() method dat:
   - Een lijst print met per regel:
     Student-id aantal_grades gemiddelde_score

     Bv. (fictieve nummers):
     1000 9 67

Opdracht 2

Inleiding

Het start-project HashMapProblem.zip bevat een project met daarin twee klassen `Student` en `StudentMap`, en er is een UnitTest voor StudentMap.
Bestudeer alle 3 de klassen en run de UnitTest. Zie dat de UnitTest faalt.

De UnitTest test de StudentMap door een aantal studenten en grades toe te voegen. Daarna checkt de test of een bepaalde student in de studentmap zit en probeert de bijbehorende grade te vinden.

Opdracht

Achterhaal waarom de UnitTest faalt.

1. Geef een verklaring hiervoor.

2. Verbeter de code (Let op, de UnitTest mag je niet aanpassen!) Na afloop moet de UnitTest slagen.

Antwoord punt 1): ....

Opdracht 3

Inleiding

Startproject "QuickSort.zip" bevat code voor zowel QuickSort als ook InsertionSort. QuickSort is al een heel snel sorteer-algoritme, maar voor kleine lijsten (bv lijsten met maximaal 10 elementen) is InsertionSort efficienter.

Opdracht

Optimaliseer de code van QuickSort zodat het algoritme gebruik maakt van InsertionSort zodra de lijst maximaal 10 elementen bevat.