Verslag opdracht 2

Victor van de Riet & Rob logtenberg

2016

# Analyse

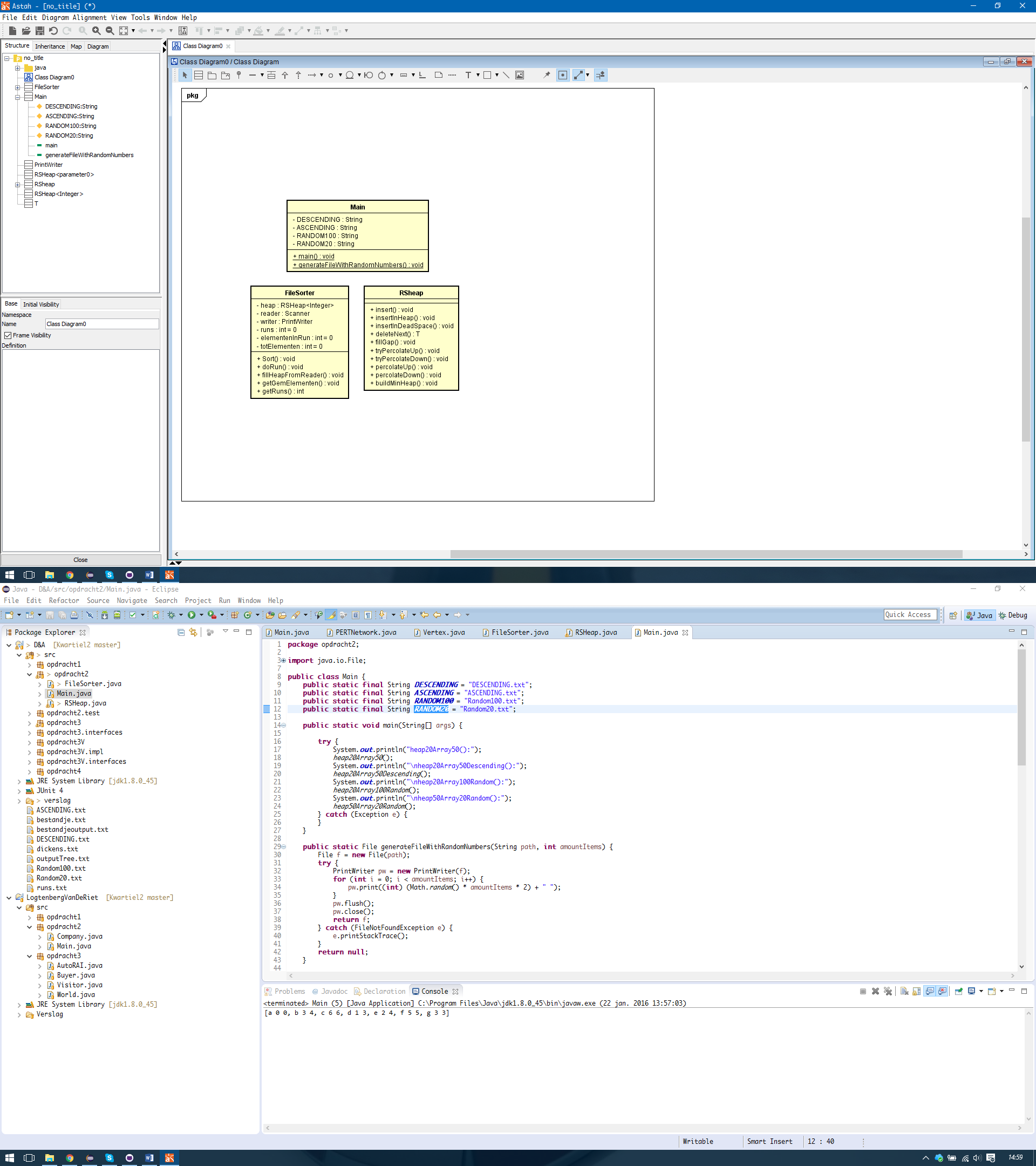
Onze verwachting is dat de snelheid van het invoegen van de data in de heap, O(log(N)) zal zijn. Dit komt omdat in de worst-case door de log van het aantal elementen van de boom gelopen moeten worden. Daarnaast worden ook veel elementen geinsert in de huildheap methode. Hierin is de snelheid O(N). De grootste delen van de inserts zullen ook de snelheid van O(N) hebben, omdat we eerst alles toevoegen en daarna in een heap “formaat” zetten.

# Ontwerp

## Gemaakte keuzes:

We hebben we besloten om een aparte .txt bestanden te gebruiken zodat we de invoer en uitvoer beter konden vergelijken. En niet een Array waar we data instoppen en dat uitprinten in de console.

## Class Diagram:

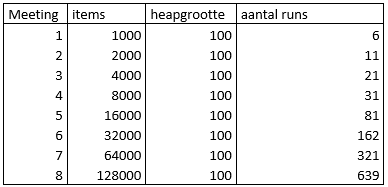


# Implementatie

De implementatie is de code zelf en hierin zit ook Javadoc. Ook zijn er 3 JUnit-tests meegeleverd.

# Metingen

Hieronder een tabel waarin we het heapgrootte gelijk houden en het aantal items verdubbelen.



Uit deze metingen kan je de conclusie trekken dat de volgende stelling juist is: “Dankzij dit algoritme wordt het aantal runs ongeveer half zoveel. Daarnaast wordt de runlengte 2 keer zo lang”. Dus dit maakt het algoritme zeer geschikt meedere files te mergen.