Indledende Programmering - Hjemmeopgave 2

Asbjørn Kjær Olling S163615
Oliver Sander Poulsen S174122

Problem 1: `TextAnalysis`

I `TextAnalysis.java` bruges BufferedReader til at læse linjer fra en given fil. Hvis filen ikke kan findes, kastes der en `FileNotFound` undtagelse.

Der bruges en Regular Expression som parameter til metoden `String.split()`, for at adskille hverlinje til ord i et array.

Ordene løbes igennem vha. et `for`-loop. Der forekommer et sanity-check, for at bestemme om ordet overhovedet er validt. Ordet sammenlignes med en `ArrayList` af unikke ord, for at bestemme om det er unikt. Det sammenlignes derefter med `previousWord` for at bestemme om det er en umiddelbar gentagelse i teksten.

TextAnalysen foregår udelukkende i konstruktøren, mens værdien af objektets felter kan hentes vha. specifike metoder.

Problem 2: `TrianglePattern`

`TrianglePattern` drejer hovedsagligt om at udfylde et array af arrays af integers, `int[][] grid`, som repræsenterer koordinater for hvert felt i mønstret. Grid har dimensionerne $n \times h$, som beskrevet i opgaven.

Udfylding af mønstret i `grid` foregår i klassens konstruktør. Resten af metoderne har formålet at returnere information om objektet.

Første række udfyldes ved at løbe igennem et givet array over indices, som bør være udfyldt. `1`-taller sættes i de passende indices i `grid[0]`;

Derefter bruges et `for`-loop til at løbe gennem resten af felterne. De forskellige cases i opgaven er i forvejen sorteret, så vi kan ud fra rækkefølgen i PDF'en kigge på f.eks. kassen øverst til venstre, for et udelukke halvdelen af cases'ne. Vi har til sidst kunnet snævre case-detection ned til kun to if-statements, med tre boolske udtryk i hver.

Felterne i de yderste søjler checkes ikke, da de ikke gyldigt kan udfyldes p.g.a. manglende felter (hhv. til højre og venstre for).

Problem 3: `MovingPoint`

`MovingPoint`-klassen er en subklasse af `Point2D.Double`. Fra superklassen bruges `setLocation()` metoden, samt felterne `double x, y`.

De to konstruktører er uhyrre simple. Værdier overføres til klassens felter. Hvis der ikke gives nogen parametre til konstruktøren, bruges der default-værdier som angivet i opgaven.

Metoden `move()` konverterer `direction` fra grader til radianer, og gør brug af trigonometriske funktioner fra `Math` til at udregne punktets nye position.

Metoden `turnBy()` lægger en vinkel til `direction` feltet, og bruger `while`-loops til at flytte værdien ind i intervallet $[0; 360]$. Alternativt kunne modulo bruges, for at finde hovedargumentet.

Metoden `accelerateBy()` lægger en hastighedsændring til `speed` feltet, og bruger et `if`-statement til at korrigere negative tal. Oprindeligt antog vi at negative hastiheds-værdier betød at punktet "bakker", altså at vinklen ændres med 180°, og hastighedn blev den absolutte værdi, af hvad den var. Den alternative løsning står stadig (udkommenteret) i `MovingPoint.java` på linjer 57-62.

Arbejdsfordeling:

- **Problem 1:** Oliver + Asbjørn
- **Problem 2:** Asbjørn
- **Problem 3:** Oliver + Asbjørn