Uppgift 2: Robotar och Labyrinter

Din uppgift är att skriva klasser för att representera en labyrint, ett antal olika robottyper, samt en klass för att representera positioner i labyrinten. Klasserna **ska** följa designen i UML-diagrammet nedan. Ni ska sedan låta robotarna tävla mot varandra i att ta sig från en startpunkt till ett mål i en labyrint (d.v.s. skriva ett litet program som testar robotarnas prestanda). För klasserna Maze och Position ska också testprogram skrivas som testar dessa klassers funktionalitet.

Robotarna

Skriv en abstrakt klass Robot som håller reda på

- en labyrint
- robotens position

och har följande metoder

- en abstrakt move()-metod vars syfte är att flytta roboten ett steg
- En metod för att kolla robotens nuvarande position
- En metod för att kolla om roboten nått målet

Du ska sedan tillhandahålla följande två varianter på robotar. Dessa kommer eventuellt behöva ha ytterligare attribut förutom de ärvda.

- RightHandRuleRobot - Rör sig runt i labyrinten så att den hela tiden håller höger hand mot en vägg. Observera att RightHandRuleRobot endast klarar labyrinter då målet finns vid en vägg som hänger ihop med en vägg vid startpunkten.
- MemoryRobot - Håller reda på positioner där den varit samt vägen den gått hittills. Hittar den ingen position som den inte redan har besökt (och som den kan gå till) ska den ta ett steg tillbaka till föregående position i en väglista (och radera positionen den stod på ur den listan (Tips: använd en stack)). Roboten ska alltså göra en (iterativ variant på) djupet-först-sökning av labyrinten.

Labyrinten

Skriv en klass Maze som representerar en labyrint. I denna klass måste ni själva bestämma vilka attribut som behövs. Labyrintobjekt ska kunna skapas genom att läsas in från en textfil. Textfilen ska ha följande format:

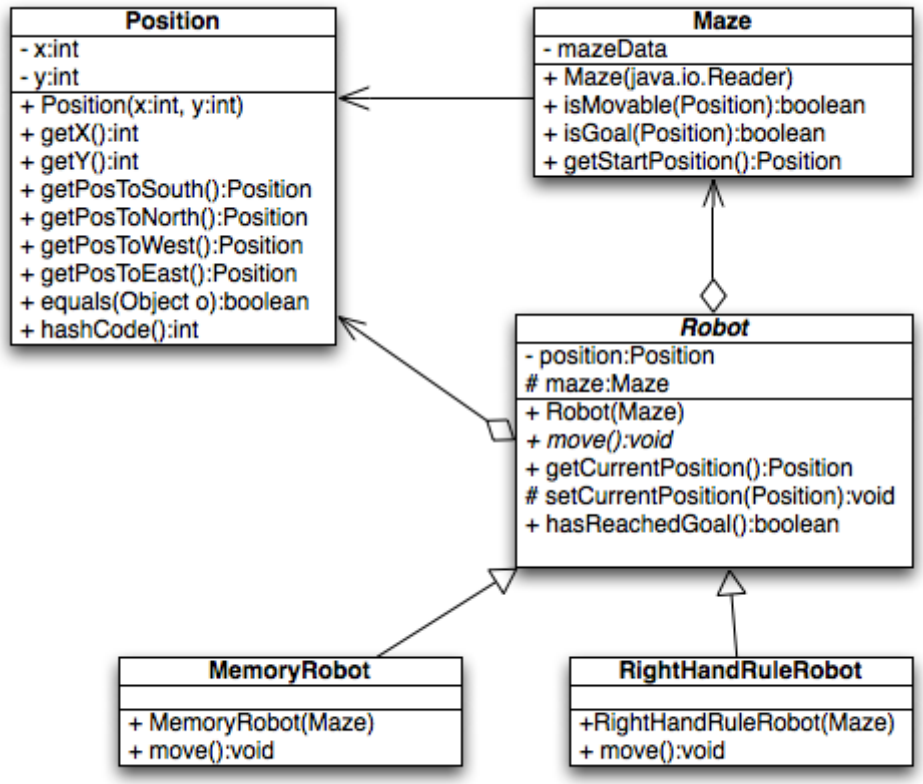
- Väggar markeras med *.
- Platser man kan gå till markeras med mellanslag.
- Varje labyrint ska ha en startposition markerad med S.
- Minst en målposition, markerad(e) med G.

ex:

```
*S**********
*          *
* ******** *
*    *   * *
*** ** * * *
*      * * *
********G***
```

Design

Ni har följande design att utgå ifrån. Observera att ytterligare attribut (och ev hjälpmetoder) kan komma att behövas för att implementera beteendena. Robotarna ska inte för sin förflyttning eller annat grundläggande beteende utnyttja några andra publika metoder än de nedan. För att underlätta testning får ni om ni vill lägga till ytterligare metoder i klasserna (t.ex. `toString`-metoder)



Krav

- Labyrinter ska kunna läsas in från en textfil.
- Inga objekt ska kunna hamna i ett tillstånd som är ogiltigt (t.ex.: inga Maze-objekt ska kunna skapas som inte innehåller alla nödvändiga värden och ingen robot ska kunna skapas så att den befinner sig på en plats där den inte kan vara).

Er lösning måste vara implementerad enligt de principer om objektorientering som vi gått igenom på kursen. I den mån ni anser att specifikationen är oklar är det upp till er att reda ut dessa oklarheter (genom att fråga läraren).

Redovisning

Vid redovisningen ska följande visas upp:

- Kommenterad källkod
- Javadoc
- UML-klassdiagram (inkl. alla egna klasser, deras metoder och attribut)
- Testkörningar, inklusive analys av robotarnas prestanda i olika labyrinter

Extrauppgift: Gör så att man ser hur roboten rör sig i labyrinten.