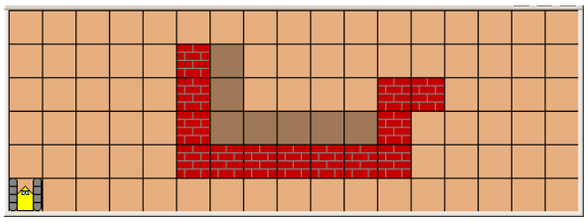
Laboration 2.

I denna laboration skall ni programmera en robot som modelleras av den givna klassen Robot. En robot vistas i en enkel värld, som modelleras av klassen RobotWorld. Världen består av ett rutmönster av celler som är objekt av klassen Cell. Det finns tre olika typer av celler, mörka celler, ljusa celler och murar.



En robot befinner sig på en viss plats i världen (som anges av ett objekt av klassen Location) och har en riktning (som anges med statiska heltalskonstanterna Robot.NORTH, Robot.EAST, Robot.SOUTH eller Robot.WEST). Roboten kan förflytta sig i världen i horisontell respektive vertikal riktning och utföra enkla instruktioner. Om roboten försöker förflytta sig utanför världen eller ta sig igenom en mur uppstår ett exekveringsfel.

De instruktioner som en robot kan utföra är:

void move()	förflyttar sig ett steg framåt. Om roboten hamnar utanför världen eller i en mur fås ett exekveringsfel.
void turnLeft()	vrider sig 90° åt vänster.
void makeDark()	färgar rutan den står på till mörk. Om rutan redan är mörk fås ett exekveringsfel.
void makeLight()	färgar rutan den står på till ljus. Om rutan redan är ljus fås ett exekveringsfel.
boolean onDark()	returnerar **true** om roboten står på en mörk ruta, annars returneras **false**.
boolean frontIsClear()	returnerar **true** om det är möjligt för roboten att göra move() utan att ett exekveringsfel erhålls, annars returnera **false**.
boolean atEndOfWorld()	returnerar **true** om det är omöjligt för roboten att göra move() utan att hamna utanför världen, annars returneras **false**.
Location getLocation()	returnerar robotens position i världen som ett objekt av typen Location.
int getDirection()	returnerar robotens riktning.
void setDelay(**int** t)	sätter hastigheten med vilken roboten rör sig. Parametern t anger antalet millisekunder mellan varje rörelse.

Syfte

Syftet med dessa uppgifter är att få övning i att använda en färdig klass samt att bryta ner ett programmeringsproblem i små meningsfulla delproblem, dvs tänka i olika abstraktionsnivåer.

Ni skall försöka och minimera användningen av upprepade instruktionssekvenser, och i stället deklarera lämpliga metoder (abstraktioner). Som en riktlinje bör de metoder ni definierar inte innehålla mer än 10 satser.

Metoderna ni utvecklar skall ha beskrivande namn och en kommentar som anger för- och eftervillkor:

```
//before:  none
//after:   robot is facing opposite direction
public static void turnAround() {
    robot.turnLeft();
    robot.turnLeft();
}//turnAround
```

Redovisning

Sista inlämningsdag i Fire är torsdag 7/10. Lämna in samtliga källkodsfiler som en komprimerad zip-fil.

Uppstart

Börja med att skapa en nytt projekt i IntelliJ och kopiera samtliga filer från zip-filen som finns på kursens hemsida under fliken Laborationer med namn FilesLab2.zip i projektets src catalogen.

Uppgift 1

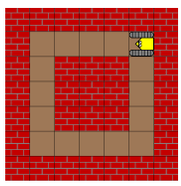
I filen Cleaner.java finns ett program i vilket en värld och en robot skapas. Koden har följande utseende:

```
public class Cleaner {
    private Robot robot;
    public static void main(String[] args) {
        Cleaner cleaner = new Cleaner();
        cleaner.createEnviroment();
        cleaner.cleanCorridors();
    } //main

    private void createEnviroment() {
        RobotWorld world = RobotWorld.load("src/square.txt");
        robot = new Robot(1, world.getNumCols() - 2, Robot.WEST, world);
        robot.setDelay(250);
     }//createEnviroment

    //The robot cleans the corridor by making the colour of each cell light
    //before: The room has four corridors, forming a square
    //          The robot is located in beginning of one of the corridors, facing the corridor
    //           in counter-clockwise direction.
    //          Each corridor has a length of five cells.
    //          All cells in the corridors are dark.
    //after:   The robot has the same location and facing the same direction
    private void cleanCorridors() {
        //This is your work to do!!
    }//cleanCorridors
}//Cleaner
```

Om du kompilerar och kör klassen Cleaner visas ett fönster enligt figuren bredvid. Inget mer händer eftersom metoden CleanCorridors ännu inte gör någonting. Det är din uppgift att fullborda denna metod, enligt den specifikation av metoden som anges av kommentarerna i koden ovan och som vidareutvecklas i texten nedan.



Klassen innehåller en instansvariabel robot av klassen Robot, de båda instansmetoderna createEnviroment() och cleanCorridors(), samt en main-metod.

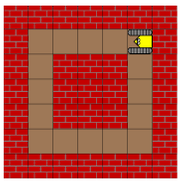
Metoden createEnviroment skapar först en värld. Världen är ett objekt av klassen RobotWorld som refereras via variabeln world. Världen skapas genom att anropa metoden RobotWorld.load, som har en textfil square.txt som parameter. Textfilen square.txt definierar hur världen kommer att se ut. Sedan skapar metoden createEnviroment roboten som refereras via instansvariabeln robot. Detta sker i satsen

```
robot = new Robot(1, world.getNumCols() - 2, Robot.WEST, world);
```

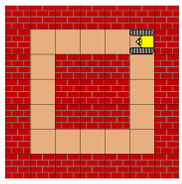
De två första parametrarna till konstruktorn (1 och world.getNumCols() - 2) anger i vilken position i världen roboten skall placeras. Den tredje parametern (Robot.WEST) anger robotens riktning och den sista parametern (world) anger vilken värld roboten skall bebo.

I världen har cellen längst upp till vänster positionen (0,0). Det är möjligt att fråga världen hur stor den är. Genom att anropa metoden getNumCol() tar vi reda på hur många celler världen har i horisontell led (= antalet kolumner). Om vi vill ta reda på antalet celler världen har i vertikal led (=antal rader) finns metoden getNumRow(). Slutligen gör metoden createEnviroment ett anrop av metoden setDelay för att sätta med vilken hastighet roboten skall röra sig. Parametern till setDelay anger hur många millisekunder det är mellan varje rörelse.

a) Din uppgift är att komplettera metoden cleanCorridors(), på så sätt att roboten genomlöper korridorerna *moturs* och ”städar” dessa genom att sätta de mörka cellerna till ljusa (enligt figurerna nedan).



Före



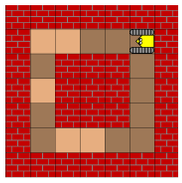
Efter

Du får anta att samtliga korridorer är 5 celler långa och att alla celler är mörka. Vidare får du anta att roboten står i ett hörn och är riktad på så sätt att den har ytterväggen på sin högra sida (vilket innebär att en förflyttning av roboten sker i moturs riktning i korridoren). Samtliga dessa antaganden är förvillkor till metoden cleanCorridors(). När roboten utfört sin uppgift skall den befinna sig i samma tillstånd (dvs ha samma position och riktning) som den hade innan uppgiften utfördes. Din lösning skall endast använda instruktionerna move(), turnLeft() och makeLight().

Tips: Du skall vara observant på det så kallade *staket-stolp-problemet*, som resulterar i ett *off-by-one-error*. Problemet har fått sitt namn pga det faktum att det åtgår n+ 1 stolpar för att sätta upp n sektioner staket, vilket ofta leder till att en stolpe saknas när sista sektionen skall sättas upp.

Analogin här är att om roboten befinner sig i början av en korridor som är 5 celler lång, behöver roboten endast göra 4 move-operationer för att komma till slutet av korridoren. När roboten står i början av en korridor är det därför lämpligt att den inte ”städar” cellen den står i utan endast de celler som den har *framför sig* i korridoren. Inför en sådan metod (abstraktion) och ge den namnet clearCorridorInFront. Specificera även för- och eftervillkor (skall ges som kommentarer i koden).

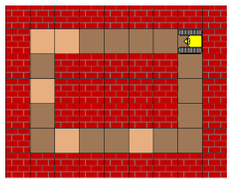
b) Förändra ditt program från deluppgift a) på så sätt att programmet klarar av att handha korridorer som redan delvis är ”städade”, dvs där någon cell initialt är ljus (se figuren bredvid).



Förvillkoret *”all cells in the corridors are dark”* till metoden cleanCorridors() skall alltså tas bort.

Gör inte förändringarna direkt i klassen som du skapade i deluppgift a) utan kopiera koden till en ny klass Cleaner2 och gör ändringarna i denna klass. Glöm inte att uppdatera förvillkoren, samt att byta alla förekomster av Cleaner till Cleaner2 (fyra ställen med en kommentar inräknat). Programmet skall testas med världen som finns beskriven i filen square2.txt. Du behöver därför också byta ut filnamnet square.txt mot square2.txt i metoden createEnviroment().

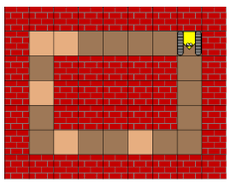
c) Nu skall du förändra ditt program från deluppgift b) på så sätt att roboten klarar att ”städa” byggnader där de horisontella och vertikala korridorerna är olika långa (se figuren nedan).



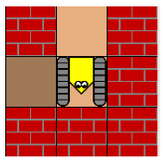
Förvillkoret *”each corridor has a length of five cells”* till metoden cleanCorridors() tas alltså bort.

Gör inte förändringarna direkt i klassen som du skapade i deluppgift b) utan kopiera koden till en ny klass Cleaner3 och gör ändringarna i denna klass. Du skall testa ditt program med världen som beskrivs av filen square3.txt.

d) Nu skall du förändra ditt program från deluppgift c) på så sätt att förvillkoret *"facing the corridor in counter-clockwise direction"* till metoden CleanCorridors() tas bort. Roboten skall alltså klara av att "städa" korridorerna även *medurs* (se figuren bredvid).

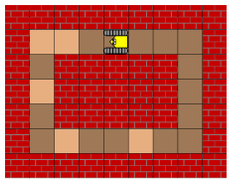


Om roboten står i slutet av en korridor och är riktad mot väggen, är det enkelt att ta reda på om den förflyttar sig medurs eller moturs. Har roboten en vägg på sin vänstra sida är riktningen medurs och roboten skall svänga runt hörnet genom att vrida sig 90° åt höger, annars är riktningen moturs och roboten skall svänga runt hörnet genom att vrida sig 90° åt vänster.



Gör inte förändringarna direkt i klassen som du skapade i deluppgift c) utan kopiera koden till en ny klass Cleaner4 och gör ändringarna i denna klass. Du skall testa ditt program med världen som beskrivs av filen square3.txt. Testa ditt program för såväl då robotens rörelseriktning är moturs som medurs. För att få roboten att gå medurs ändras robotens riktning i metoden CleanCorridors() till Robot.SOUTH.

e) Nu skall du förändra ditt program från deluppgift d) på så sätt att det klarar av att roboten placeras på en godtycklig position i någon korridors längdriktning (se figuren bredvid).



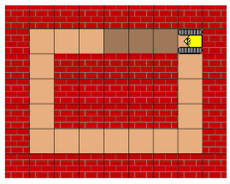
Förvillkoret *"the robot is located in beginning of one of the corridor"* till metoden cleanCorridors() tas alltså bort.

Tips: För att uppfylla eftervillkoret *"the robot has the same location"* till metoden cleanCorridors() är det nödvändigt att spara undan robotens startposition. Detta görs med hjälp av operationen getLocation(), som returnerar positionen som ett objekt av klassen Location.

För att kontrollera om två objekt startPosition och currentPosition av klassen Location är lika används instansmetoden equals enligt

```
startPosition.equals(currentPosition)
```

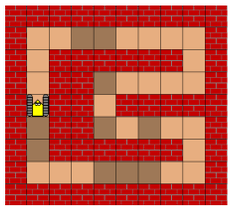
Om ni kör programmet från deluppgift d) med världen som ges i bilden ovan kommer resultatet att bli enligt bilden bredvid. Den första delen av korridoren i vilken roboten påbörjade städningen är alltjämt ostädad. Ni måste alltså införa en metod som städar en korridor fram till och med en viss given position (i detta fall robotens ursprungliga startposition). Positionen (ett objekt av klassen Location) ges som parameter till metoden. Kalla metoden cleanCorridorUpToPosition.



Gör inte förändringarna direkt i klassen som du skapade i deluppgift d) utan kopiera koden till en ny klass Cleaner5 och gör ändringarna i denna klass.

Testa programmet med samma värld och med samma initiala placering av roboten som i deluppgift d) . Testa också programmet såväl för då roboten förflyttar sig medurs som moturs, samt med att placera roboten mitt på en korridor (det du behöver göra är att ändra parametern world.getNumCols() - 2 i metoden createEnviroment() till förslagsvis värdet 4).

f) Slutligen skall vi ta bort förvillkoret *"the room has four corridors, forming a squarer"* till metoden cleanCorridors() och ersätta detta med förvillkoret *"the corridors form a closed loop"*.



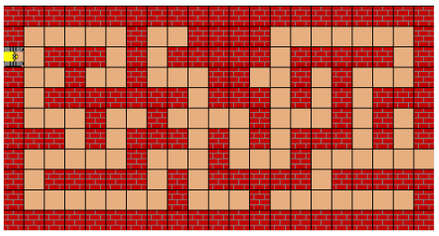
Roboten skall alltså klara av att "städa" korridorer enligt scenariot i figuren bredvid.

Det som behöver göras är att kombinera abstraktionerna cleanCorridorInFront och cleanCorridorUpToPosition till en enda abstraktion – *"städa" korridoren tills en given position nåtts eller om en sådan position inte påträffas i korridoren tills slutet av korridoren*. Abstraktionen måste återrapportera om "städningen" avslutats vid den givna positionen eller inte.

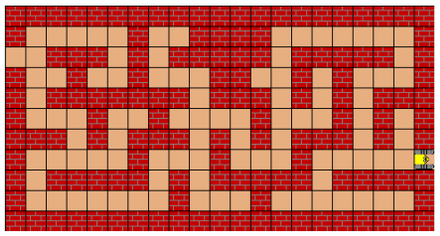
Testa programmet med världen som beskrivs av filen loop.txt, såväl för då roboten förflyttar sig medurs som moturs.

Uppgift 2

I filen MazeFinder.java finns ett program i vilket en värld och en robot har skapats. Roboten befinner sig i ingången till en labyrint och uppdraget roboten skall utföra är att förflytta sig till utgången av labyrinten (enligt figurerna nedan).



Före

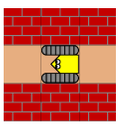


Efter

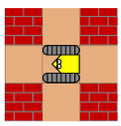
Din uppgift är att komplettera metoden findExit().

Problemet med att ta sig igenom en labyrint, som är enkelt sammanhängande (dvs där vägarna inte bildar några slingor) kan lösas genom att följa ena väggen. Man skall alltså se till att alltid ha en vägg på på sin vänstra sida (eller att alltid ha en vägg på sin högra sida).

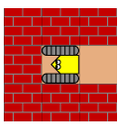
För att kunna följa en vägg (i detta fall på den vänstra sidan) och förflytta sig ett steg har vi fyra olika fall att beakta:



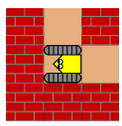
Flytta ett steg framåt



Sväng vänster och flytta ett steg framåt



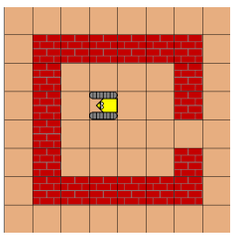
Vänd om och flytta ett steg framåt

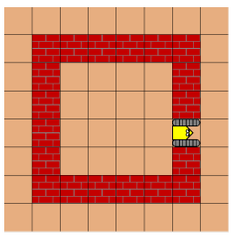


Sväng höger och flytta ett steg framåt

Uppgift 3

I filen Escaper.java finns ett program i vilket en värld och en robot har skapats. I världen finns ett rektangulärt rum med en dörr som är en cell bred. Roboten befinner någonstans *inuti* rummet och har en godtycklig riktning. Det uppdrag roboten skall utföra är att förflytta sig till utgången och stanna i denna riktad mot världen utanför rummet (enligt figurerna nedan).





Din uppgift är att komplettera metoden moveToEntrance(). Testa ditt program genom att placera roboten på olika positioner i rummet och med olika startriktningar på roboten.