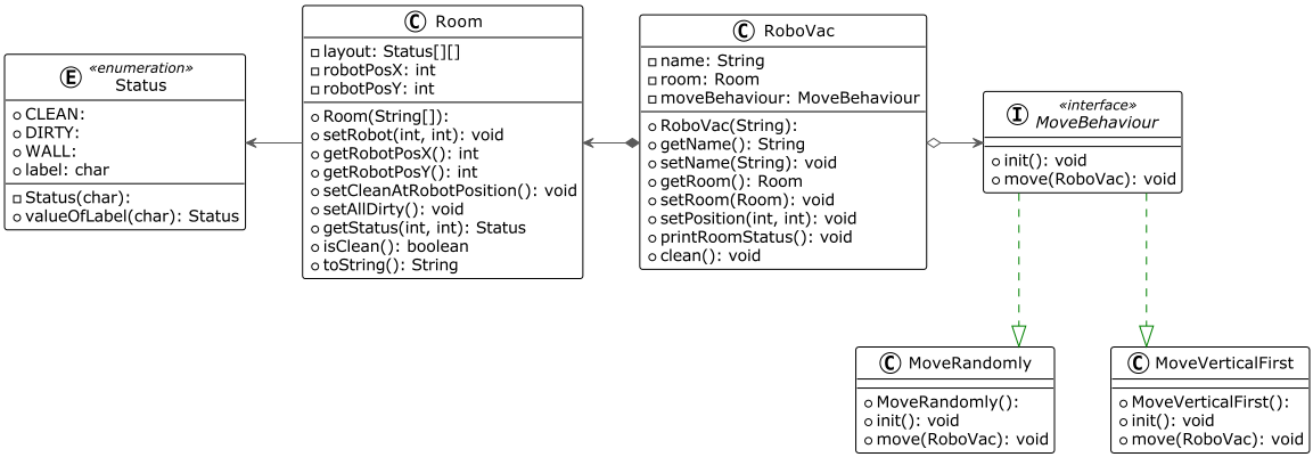
RoboVac 1.0

Implementieren Sie eine Simulation des Staubsaugerroboters RoboVac 1.0. Verwenden Sie als Anhaltspunkt folgendes Klassendiagramm sowie die unten angeführten Anmerkungen:

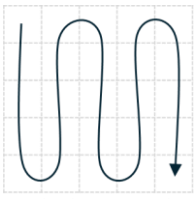


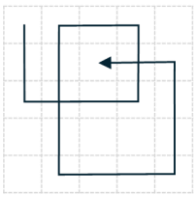
Anmerkungen:

- Der Staubsaugerroboter verfügt über einen Namen, ein Verweis auf einen Raum sowie sowie eine Bewegungsstrategie. Über Methoden können u. a.
  - der Raum die Position im Raum gesetzt werden (*setPosition*).
  - Die Reinigung des Raums angestoßen werden (*clean*).
- Der Raum besteht stets aus einer rechteckigen Matrix. Jede Position in der Matrix verfügt über einen Status: *WALL, CLEAN, DIRTY*. Darüber hinaus gibt es die aktuelle Position des Robots im Raum.
  Der Konstruktor erwartet ein 1-dimensionales String-Array, welches in das Raum-Layout (d. h. ein 2-dimensionales Status-Array) übersetzt werden muss.
  Beispiel:

```
{ "####",          { { Status.WALL, Status.WALL, Status.WALL, Status.WALL },
  "#  #",            { Status.WALL, Status. CLEAN, Status. CLEAN, Status.WALL },
  "#  #",      ➔     { Status.WALL, Status. CLEAN, Status. CLEAN, Status.WALL },
  "#  #",            { Status.WALL, Status. CLEAN, Status. CLEAN, Status.WALL },
  "####" }           { Status.WALL, Status.WALL, Status.WALL, Status.WALL } }
```

  Über Methoden können weiters u. a.
  - der Roboter positioniert werden (*setRobot*)
  - die aktuelle Position des Roboters als sauber markiert werden(*setCleanAtRobotPosition*)
  - der gesamte Raum als schmutzig gesetzt werden (*setAllDirty*)
  - der Status einer bestimmten Position abgefragt werden (*getStatus*),
  - abgefragt werden, ob der gesamte Raum sauber ist (*isClean*)
  - eine String-Repräsentation des Raum-Zustands abgefragt werden (*toString*)
- Der Roboter kann mit unterschiedlichen Bewegungsstrategien betrieben werden.
  - VerticalFirstMove: Der Roboter bewegt sich grundsätzlich vertikal im Raum. Erst wenn er an eine Wand stößt, führt er eine kurze Horizontalbewegung durch, um anschließend wieder vertikal weiterzuarbeiten.
  - RandomMove: Der Roboter ändert ein unregelmäßigen Zeitabständen zufällig seine Bewegungsrichtung.

  
  VerticalFirstMove-Strategie

  
  RandomMove-Strategie

  Jede Bewegungsstrategie verfügt über
  - eine *init*-Methode zum Initialisieren der Strategie.
  - eine *move*-Methode, die genau einen Bewegungsschritt entsprechend der Strategie ausführt.
- Wenn der Roboter mit der Reinigung beginnt (*clean()*), wird zunächst der ganze Raum als schmutzig markiert, dann beginnt er sich gemäß seiner Strategie zu bewegen (wiederholtes aufrufen von *move*), bis der gesamte Raum gereinigt ist.
  Der Reinigungsvorgang soll in der Konsole protokolliert werden, indem nach jedem Bewegungsschritt das aktuelle Raumlayout gedruckt wird.

Beispiel:

RoboVac EG starts cleaning... Cleaning... ###### #R...# #....# #....# ######	Cleaning... ###### # ...# #R...# #....# ######	Cleaning... ###### # ...# # ...# #R...# ######	Cleaning... ###### # ...# # ...# # R..# ######
Cleaning... ###### # ...# # R..# # ..# ######	Cleaning... ###### # R..# # ..# # ..# ######	Cleaning... ###### # R.# # ..# # ..# ######	Cleaning... ###### # .# # R.# # ..# ######
Cleaning... ###### # .# # .# # R.# ######	Cleaning... ###### # .# # .# # R# ######	Cleaning... ###### # .# # R# # # ######	Cleaning completed in 11 moves ###### # R# # # # # ######

Einfaches Testskript:

```
public class roboVacApp {
    public static void main(String[] args) {
        RoboVac myRoboVac = new RoboVac("RoboVac EG");
        myRoboVac.setRoomLayout(new RoomLayout(new String[] {
            "######",
            "#    #",
            "#    #",
            "#    #",
            "######"
        }));
        myRoboVac.setPosition(1,1);
        myRoboVac.clean();
        myRoboVac.clean();

    }
}
```

Hinweise:

- Setzen Sie zunächst nur die *MoveVerticalFirst*-Strategie um.
- Prüfen Sie das Reinigen mit unterschiedlichen Raum-Layouts!