When I choose algorithm I wanted something simple and had as few turns as possible. I found out that going in serpentine, going in an inwards spiral and going in a more complicated pattern that is a double spiral(first inwards leaving space for the later outwards spiral) to be good algorithms. All of them have very few turns and each of them have there own advantages and disadvantages. The serpentine had the biggest amount of turns, and you always ended up very far away from home. The double spiral had the advantage of never visit a square twice, but the algorithm for it is more complicated than the others and cant cover all squares if both sides are an odd number. Thus I choose to go in an inwards circle. It was the simplest and had very few turns.

The way to decide how to move was a bit troublesome to use. I didnt like that I couldnt stack commands. For that I created a actionQueue. At every call to the execute command I check if there already is a command in the actionQueue. If there is I will return the first command in the actionQueue.

There is no reason not to suck when there is dirt on the floor tile pacman is standing on.

By using the MyAgentState class I stored different data about the room. I didnt bother to save if there was dirt anywhere as I made the agent to always clean up dirt if there was any. When the agent hits a wall it stores the data for that square too.

turn on bump The basic algorithm is very simple, if pacman bumps into something turnRight, else go formward.When the next square has been visited before pacman also turns to the right. To calculate if the next square has been visited I created a function beenVisited(), I also created two functions, nextX() and nextY(), that would calculate the next square pacman would go to, these made it easy to define the condition that checks when to turn.

How do pacman know when he have been in the whole room? By using the beenVisited() function I defined the function surroundedByVisited() that checks if pacman has been on all surrounding squares. When there are no obstacles in the room pacman dont need to check if all unknown squares are out of reach. Pacman cant be surrounded by visited squares if he isnt done.

The agent also have two variables storing the x and y of the home square. By using this information I made a very simple pathfinding algorithm.

```
while my x is different from the target x:
    move in x towards target.
while my y is different from the target y:
    move in y towards target.
```

Here I had much use of my abstraction. Being able to stacking commands made this very simple, I also made the functions goWest(), goNorth(), goSouth(), goEast(), that made some abstraction for turning the agent in the right direction.