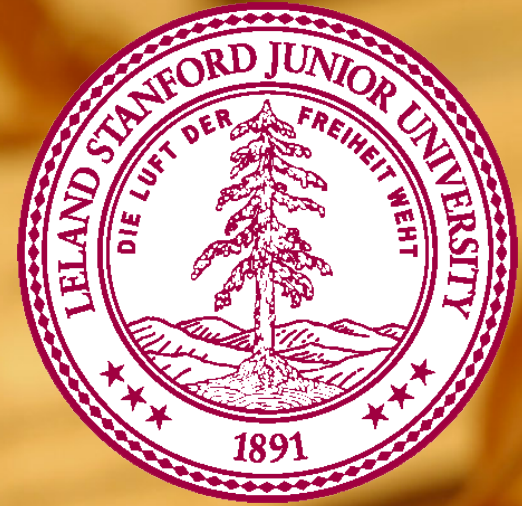




Decomposition

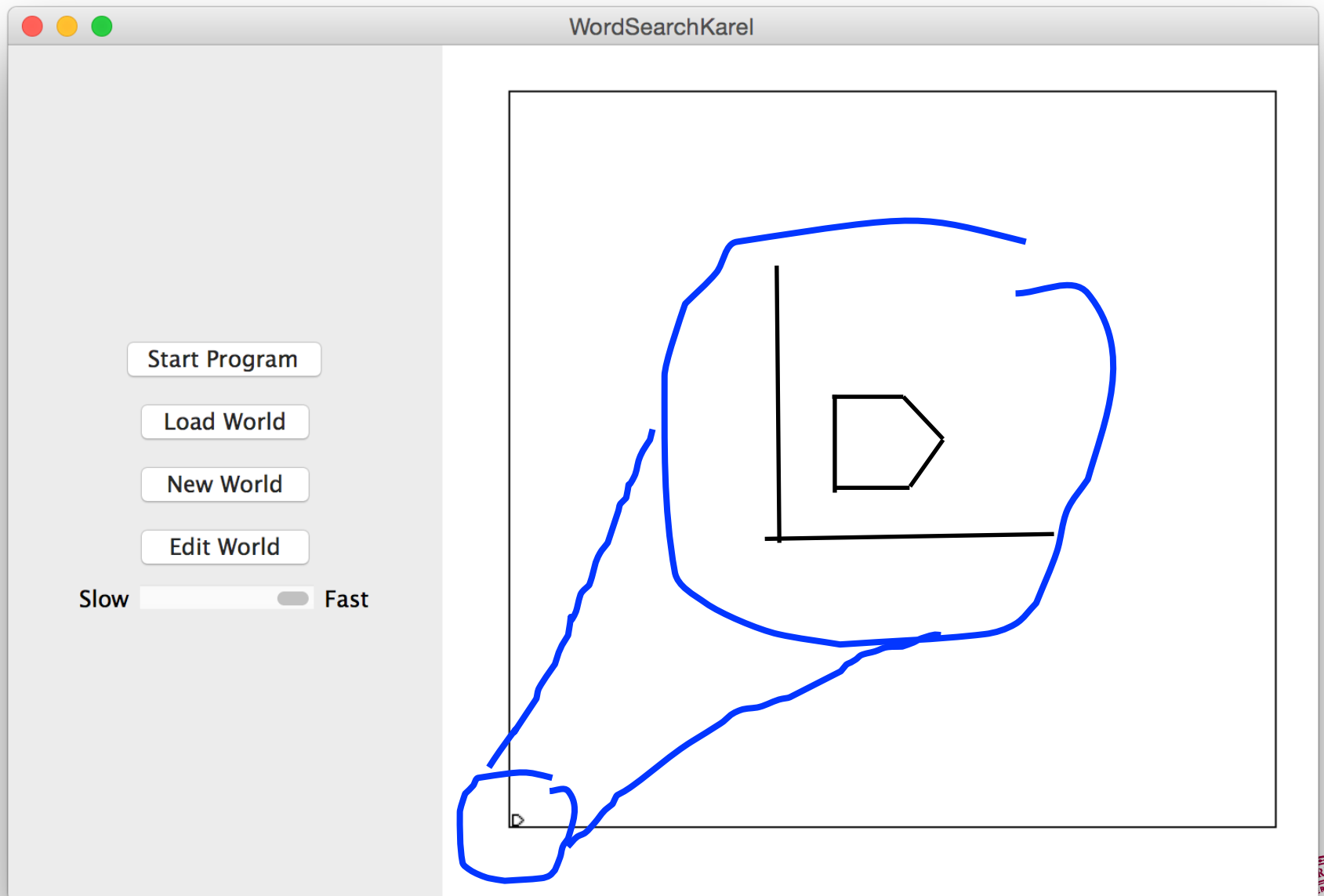
Chris Piech
CS106A, Stanford University

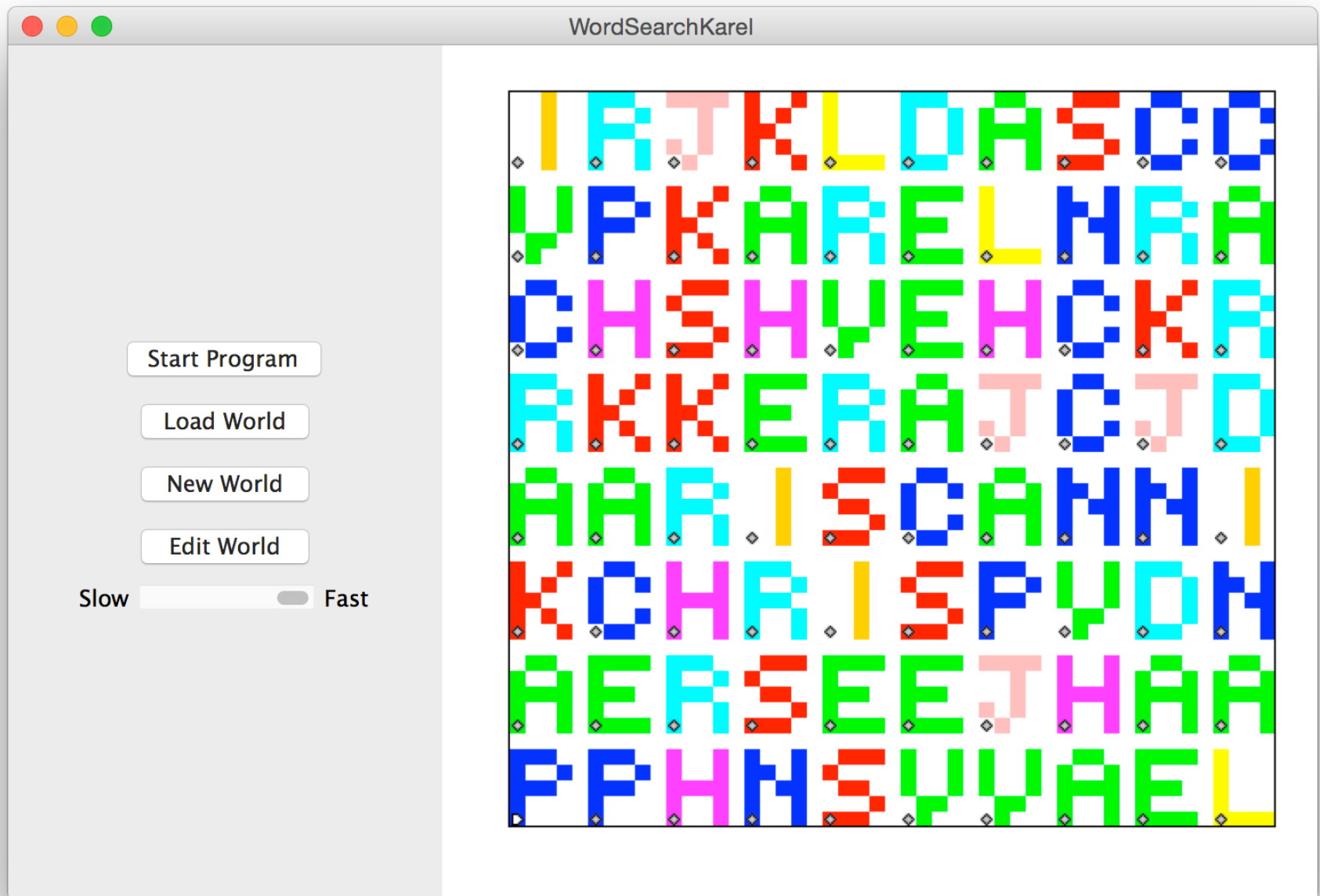
Today's Goal

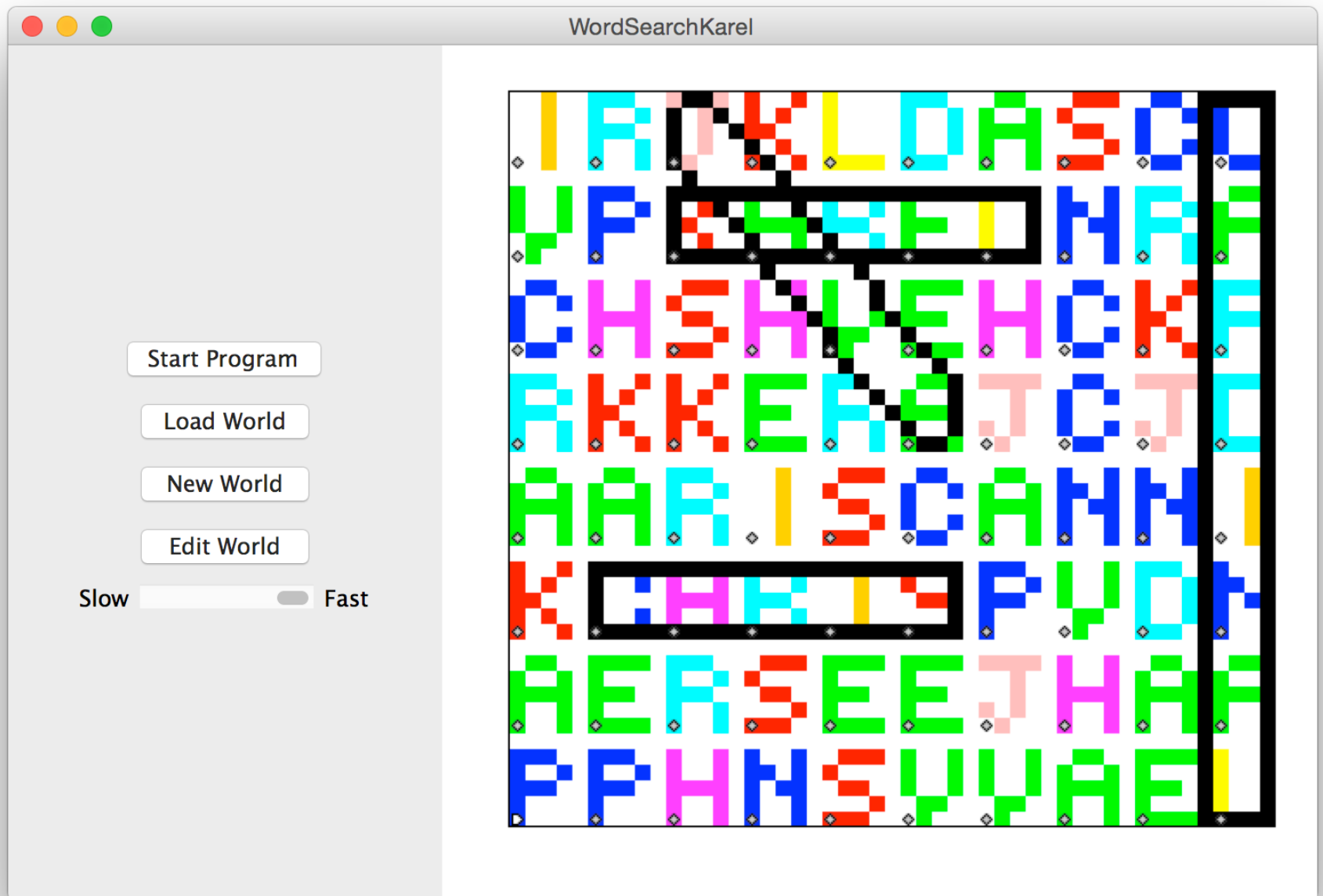
1. Be able to approach a problem “top down” by using decomposition (aka top town refinement)



First, a cool program

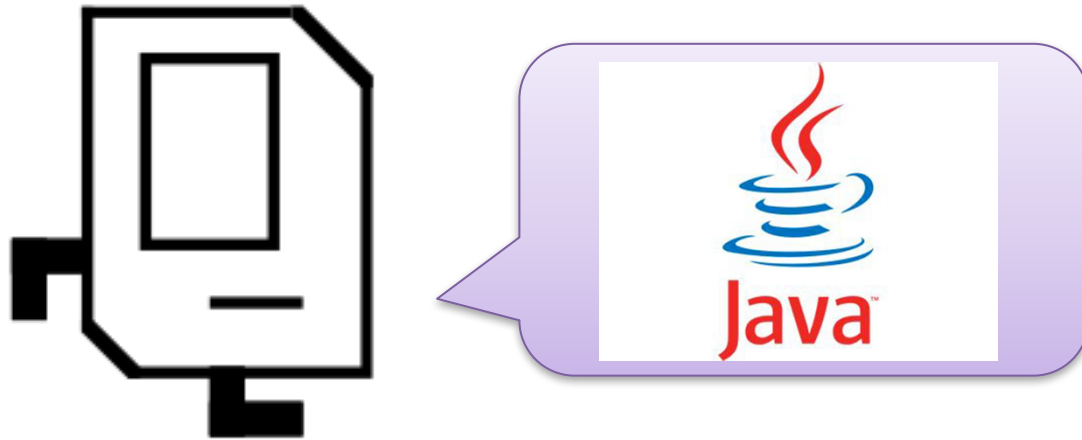






Quick review

Karel the Robot



Methods

```
import stanford.karel.*;

public class MethodExample extends SuperKarel {

    public void run() {
        goToMoon();
    }

    private void goToMoon() {
        buildSpaceship();
        // a few more steps
    }

    private void buildSpaceship() {
        // todo
    }
}
```



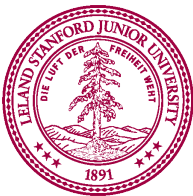
For Loops

```
import stanford.karel.*;

public class WhileExample extends SuperKarel {

    public void run() {

        // repeats the body 99 times
        for(int i = 0; i < 99; i++) {
            // the "body"
            putBeeper();
        }
    }
}
```



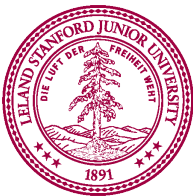
While Loops

```
import stanford.karel.*;

public class WhileExample extends SuperKarel {

    public void run() {

        // while condition holds runs body
        // checks condition after body completes
        while(frontIsClear()) {
            move();
        }
    }
}
```



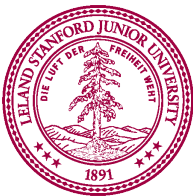
If Statement

```
import stanford.karel.*;

public class IfExample extends SuperKarel{

    public void run() {

        // If the condition holds, runs body
        if(frontIsClear()) {
            move();
        }
    }
}
```



If / Else Statement

```
import stanford.karel.*;

public class IfExample extends SuperKarel{

    public void run() {

        // If the condition holds,
        if(beepersPresent()) {
            // do this
            pickBeeper();
        } else {
            // otherwise, do this
            putBeeper();
        }
    }
}
```



The Full Karel

Built-in Karel commands:

```
move();
turnLeft();
putBeeper();
pickBeeper();
```

Karel program structure:

```
/*
 * Comments may be included anywhere in
 * the program between a slash-star and
 * the corresponding star-slash characters.
 */

import stanford.karel.*;

/* Definition of the new class */

public class name extends Karel {
    public void run() {
        statements in the body of the method
    }

    definitions of private methods
}
```

Karel condition names:

```
frontIsClear()    frontIsBlocked()
leftIsClear()     leftIsBlocked()
rightIsClear()    rightIsBlocked()
beepersPresent() noBeepersPresent()
beepersInBag()   noBeepersInBag()
facingNorth()    notFacingNorth()
facingEast()     notFacingEast()
facingSouth()    notFacingSouth()
facingWest()     notFacingWest()
```

Conditional statements:

```
if (condition) {
    statements executed if condition is true
}
```

```
if (condition) {
    statements executed if condition is true
} else {
    statements executed if condition is false
}
```

Iterative statements:

```
for (int i = 0; i < count; i++) {
    statements to be repeated
}
```

```
while (condition) {
    statements to be repeated
}
```

Method definition:

```
private void name () {
    statements in the method body
}
```

New commands in the SuperKarel class:

```
turnRight();
turnAround();
paintCorner(color);
```

New conditions in the SuperKarel class:

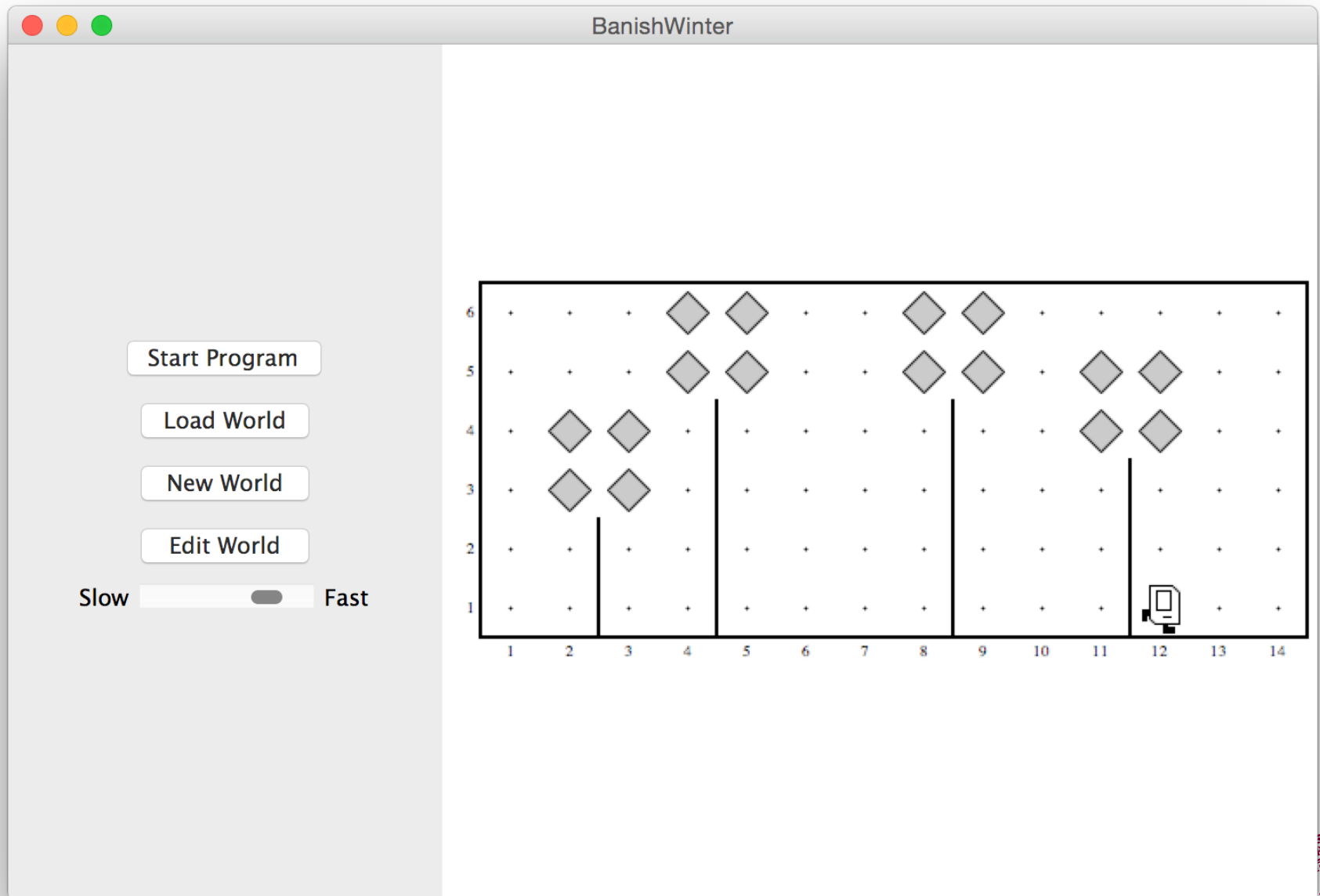
```
random()
random(p)
cornerColorIs(color)
```

Today we will use:

```
frontIsClear()
rightIsBlocked()
beepersPresent()
```



Banish Winter



End review

```
private void friday(){  
    banishWinter();  
    decomposition();  
    doubleBeepers();  
    rhombaKarel();  
    if(extraTime()){  
        wordSearchKarel();  
    }  
}
```

```
private void friday(){  
    banishWinter();  
    decomposition();  
    doubleBeepers();  
    rhombaKarel();  
    if(extraTime()){  
        wordSearchKarel();  
    }  
}
```

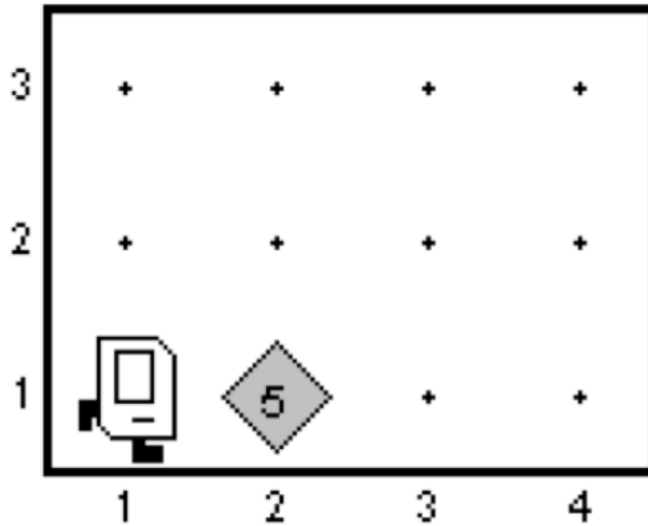


```
private void friday(){
    banishWinter();
    decomposition();
    doubleBeepers();
    rhombaKarel();
    if(extraTime()){
        wordSearchKarel();
    }
}
```

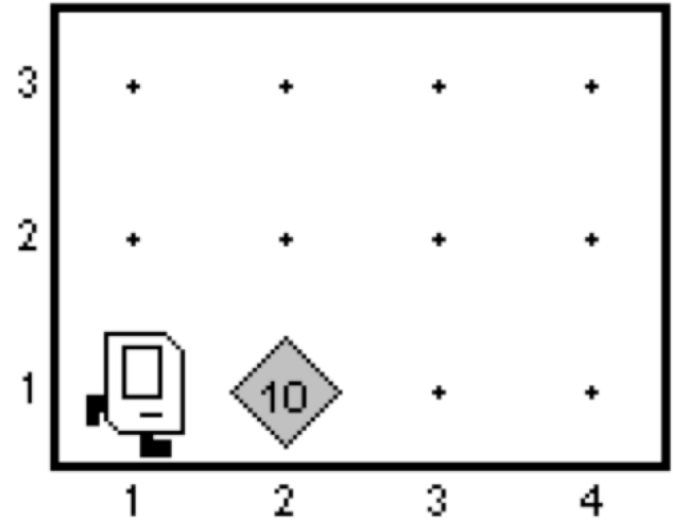
```
private void friday(){
    banishWinter();
    decomposition();
    doubleBeepers();
    rhoombaKarel();
    if(extraTime()){
        wordSearchKarel();
    }
}
```

Double Beeper

Before



After



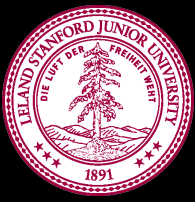
Muhammed ibn
Musa Al Kwarizmi

Pi

:y



DO
YOUR
THING



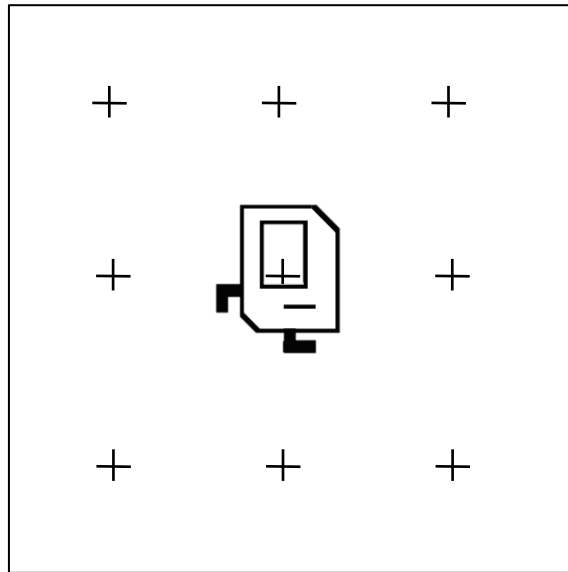
Aside: Common Errors

Lather,
Rinse,
Repeat



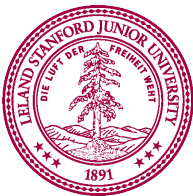
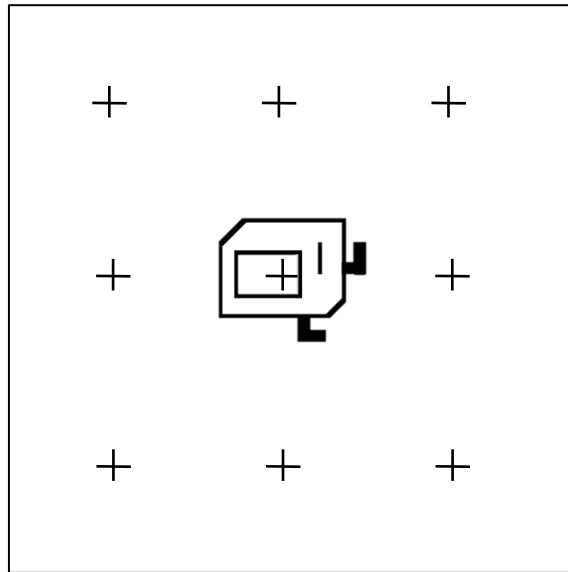
Infinite Loop

```
private void turnToWall() {  
    while(leftIsClear()) {  
        turnLeft();  
    }  
}
```



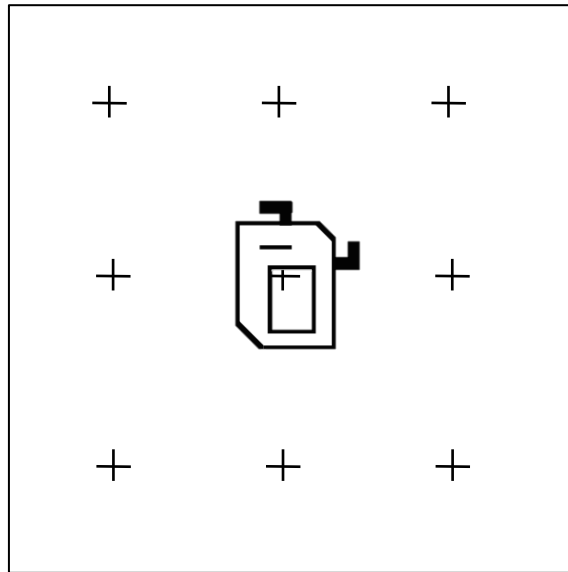
Infinite Loop

```
private void turnToWall() {  
    while(leftIsClear()) {  
        turnLeft();  
    }  
}
```



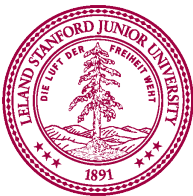
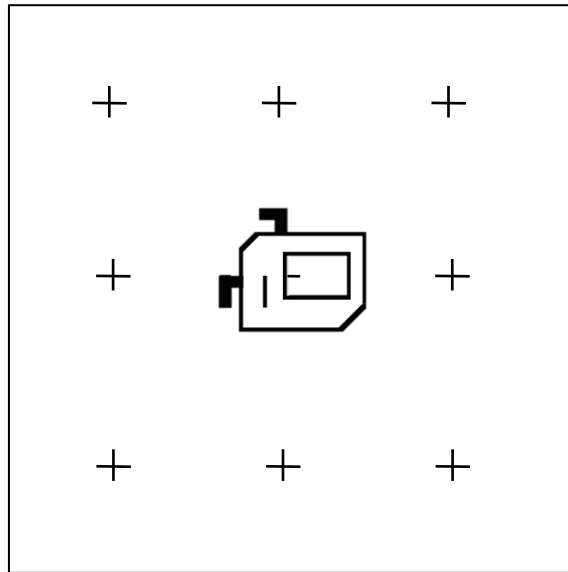
Infinite Loop

```
private void turnToWall() {  
    while(leftIsClear()) {  
        turnLeft();  
    }  
}
```



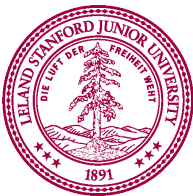
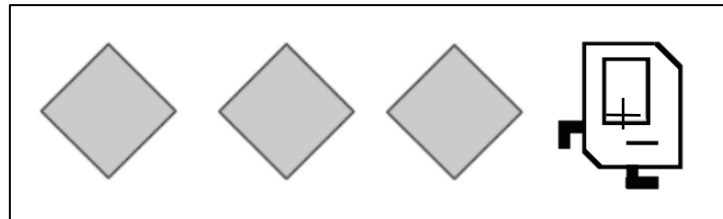
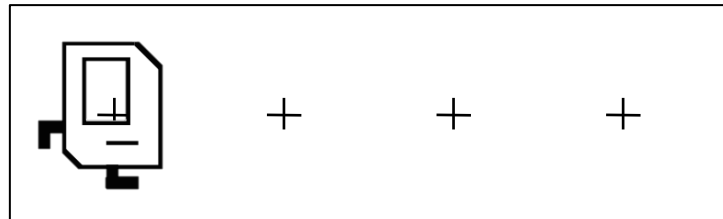
Infinite Loop

```
private void turnToWall() {  
    while(leftIsClear()) {  
        turnLeft();  
    }  
}
```



Off By One

```
private void fillRow() {  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
}
```

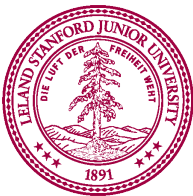


Pre/Post that Don't Match

```
private void addLeavesToTrees() {  
    turnLeft();  
    climbTree();  
    addLeaves();  
    descendToGround();  
    turnLeft();  
}
```

Post: facing East

Pre: facing South



End Aside

Lather,
Rinse,
Repeat

Now that your
longer,
ed
Balsam.

conditions
ing healthy
ch easier to
oo. You just
hower after
sure you get
only Wella makes
the original Balsam, and it's
great stuff. Wella Balsam.

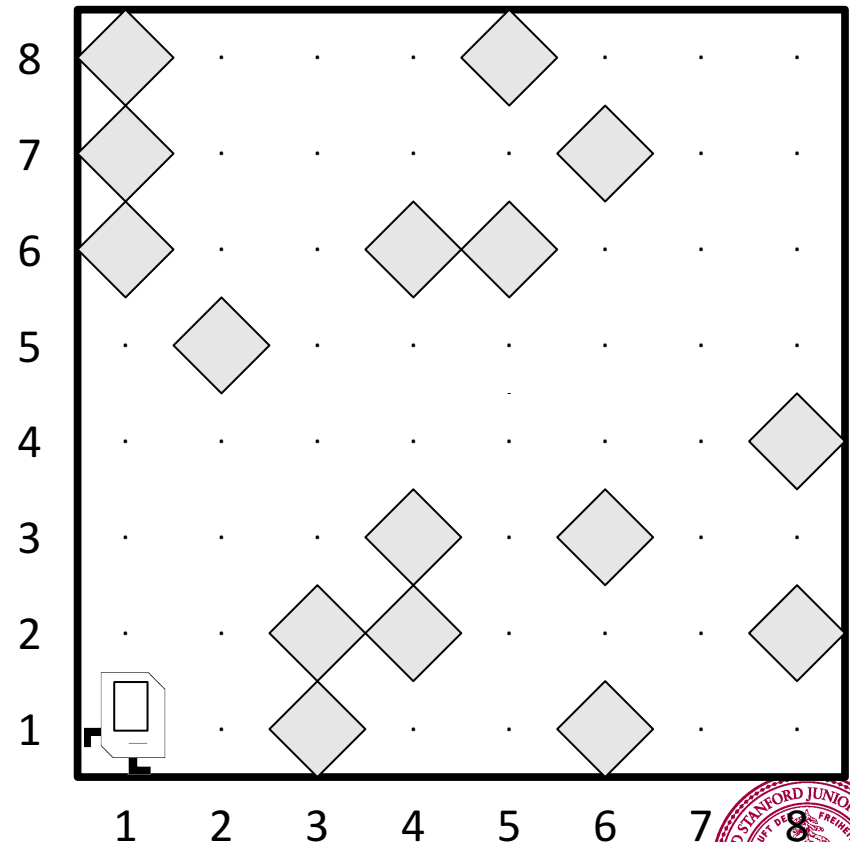


```
private void friday(){
    banishWinter();
    decomposition();
    doubleBeepers();
    rhoombaKarel();
    if(extraTime()){
        wordSearchKarel();
    }
}
```

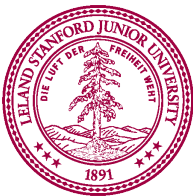
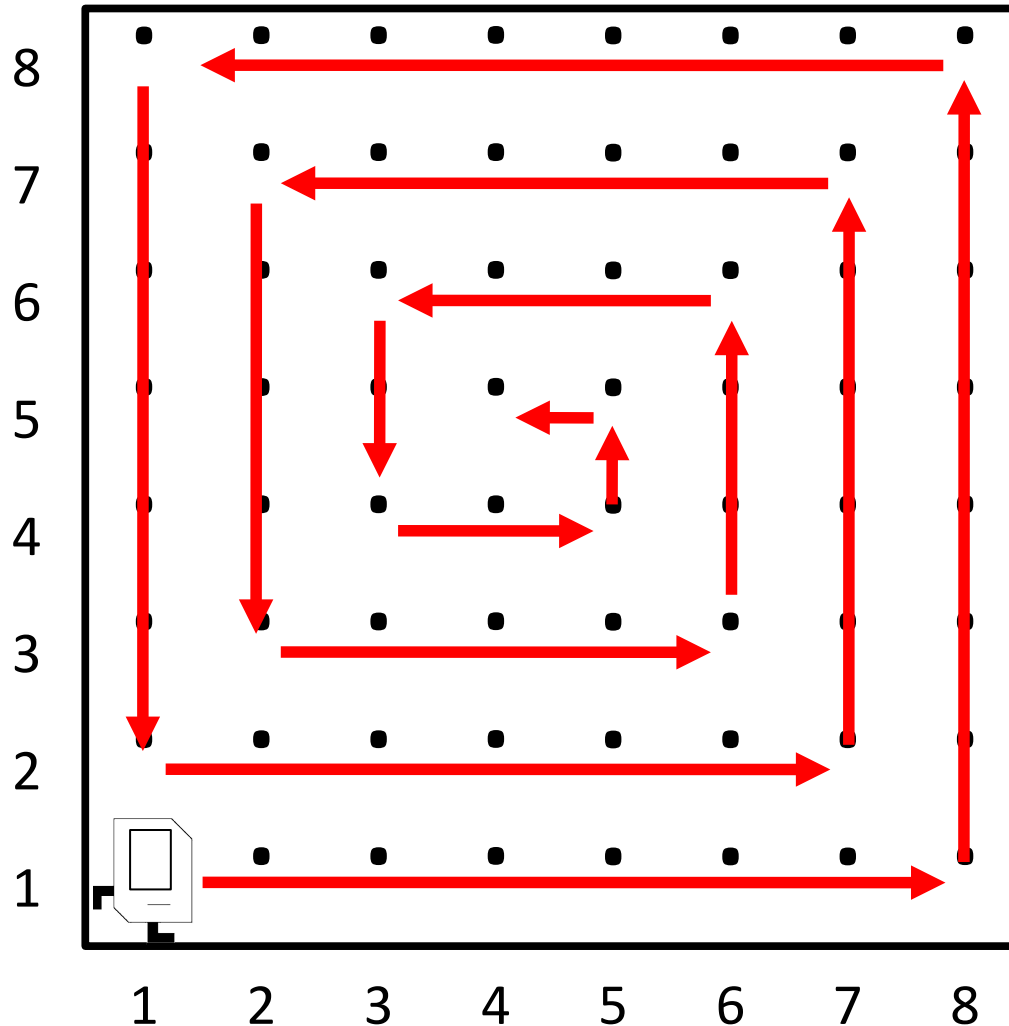
```
private void friday(){  
    banishWinter();  
    decomposition();  
    doubleBeepers();  
    rhombaKarel();  
    if(extraTime()){  
        wordSearchKarel();  
    }  
}
```

Rhoomba Karel

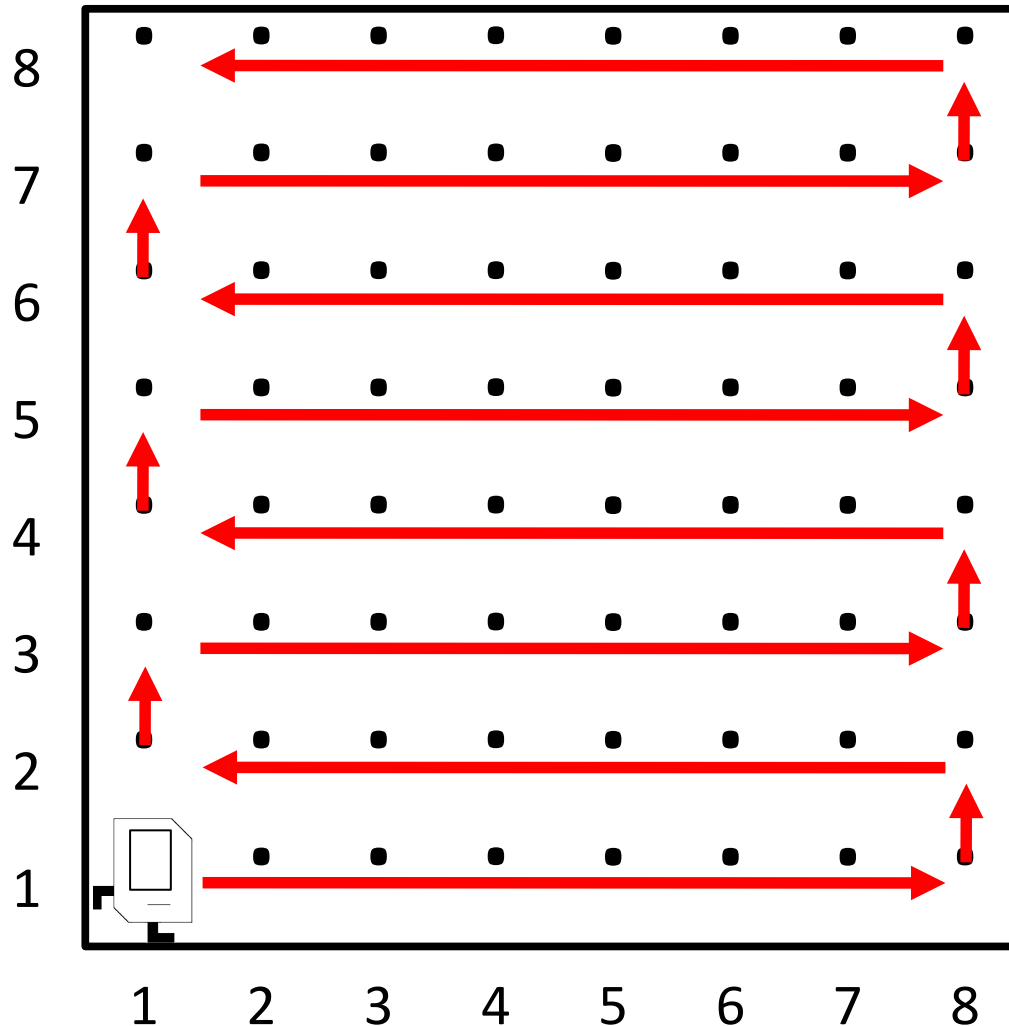
- Write a **Roomba** Karel that sweeps the entire world of all beepers.
 - Karel starts at (1,1) facing East.
 - The world is rectangular, and some squares contain beepers.
 - There are no interior walls.
 - When the program is done, the world should contain 0 beepers.
 - Karel's ending location does not matter.
- How should we approach this tricky problem?



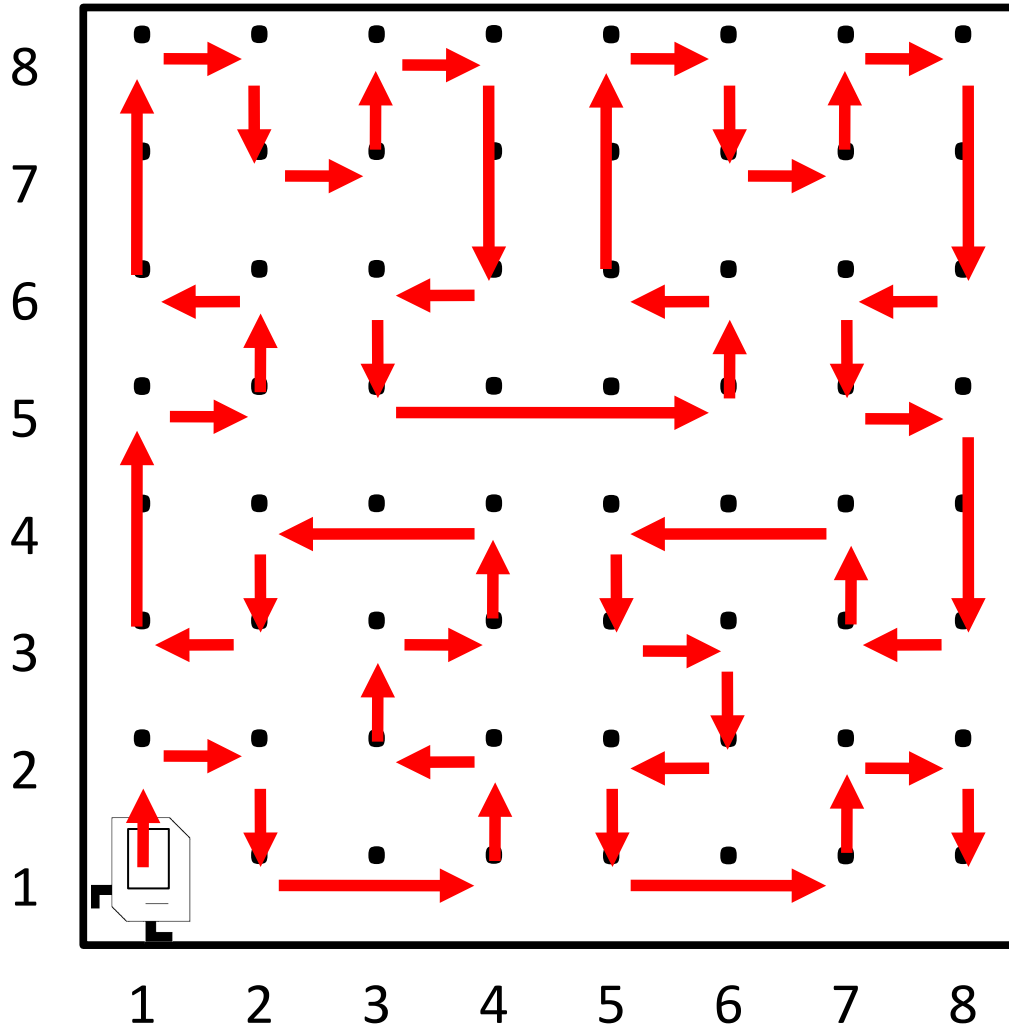
Possible Algorithm 1



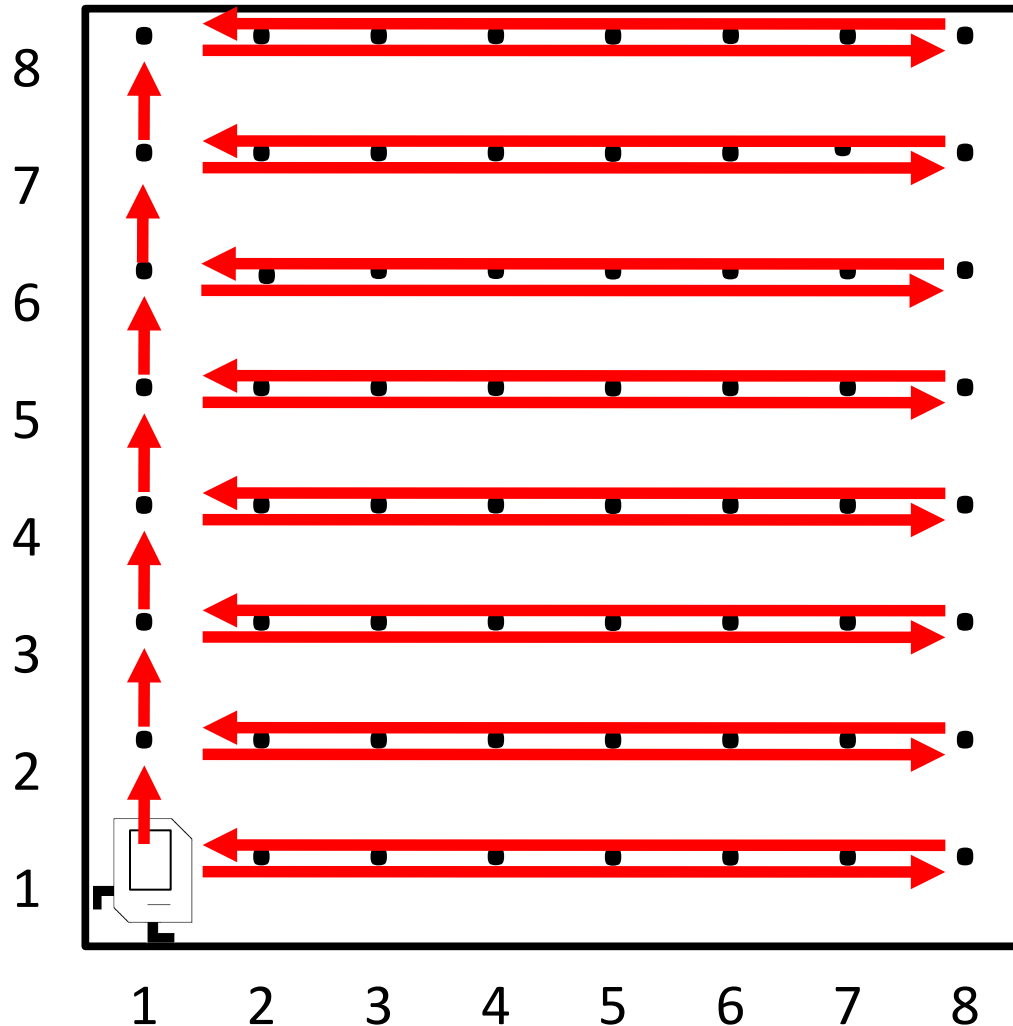
Possible Algorithm 2



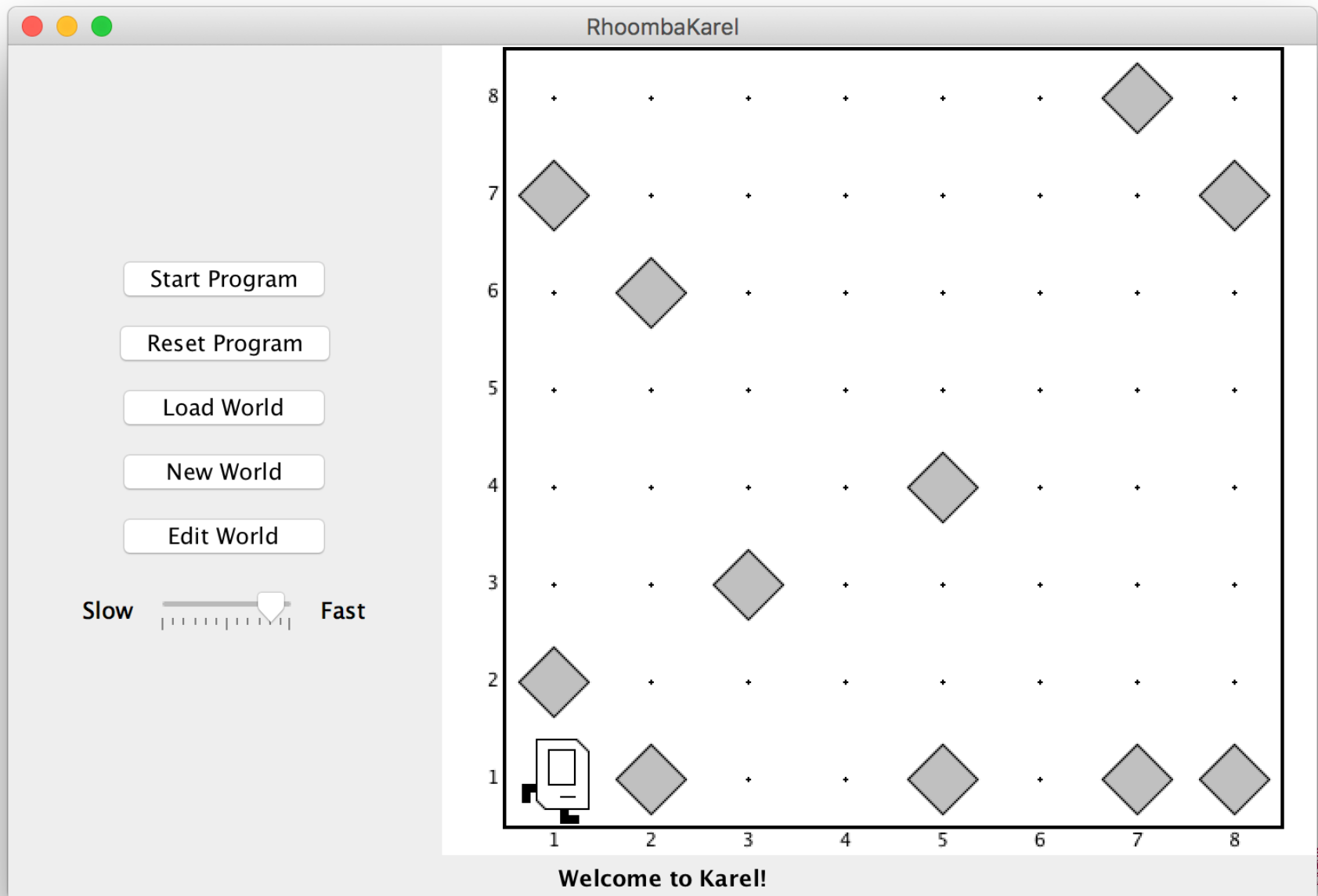
Possible Algorithm 3



Possible Algorithm 4



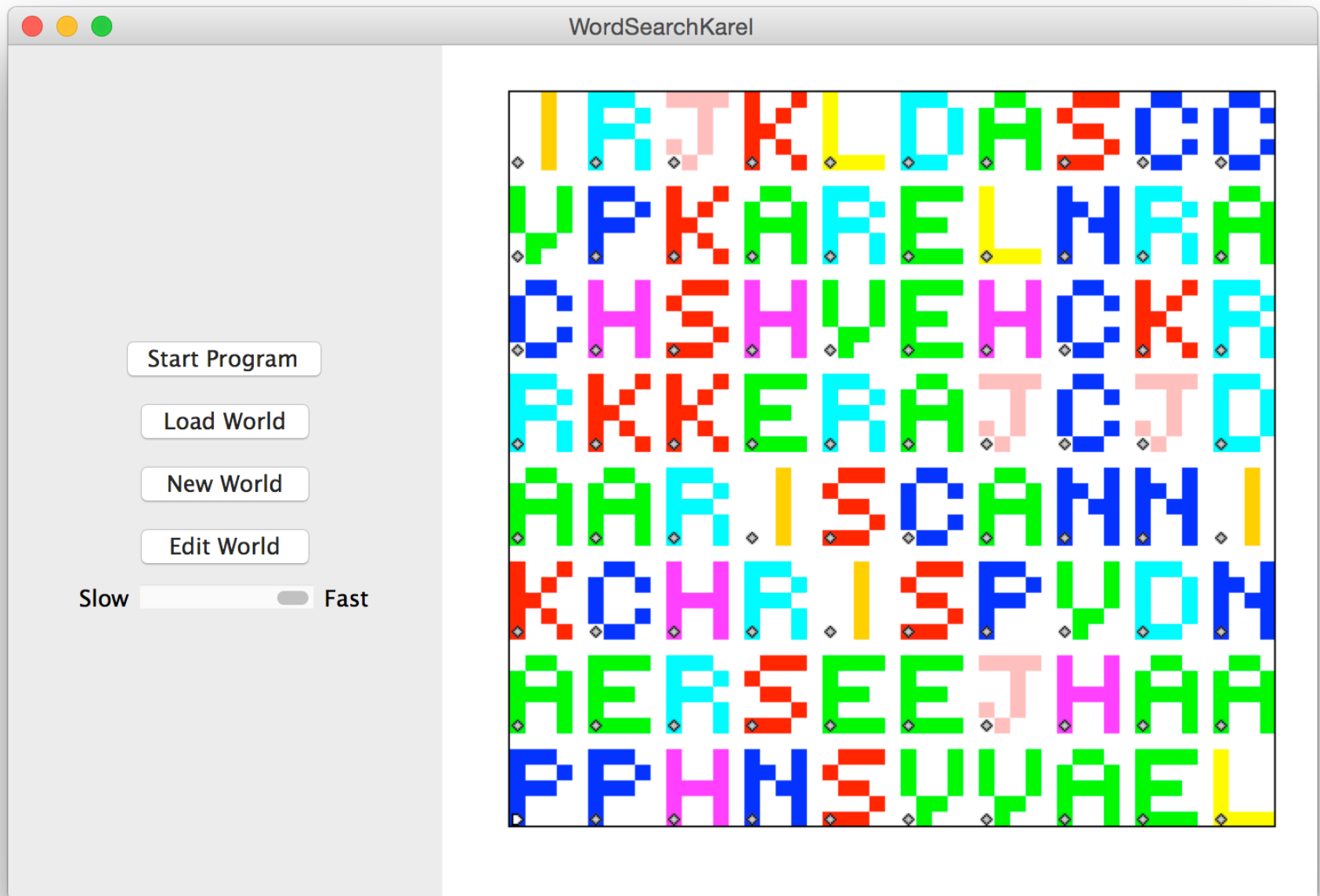
Rhoomba Karel



```
private void friday(){
    banishWinter();
    decomposition();
    doubleBeepers();
    rhombaKarel();
    if(extraTime()){
        wordSearchKarel();
    }
}
```

```
private void friday(){  
    banishWinter();  
    decomposition();  
    doubleBeepers();  
    rhoombaKarel();  
    if(extraTime()){  
        wordSearchKarel();  
    }  
}
```

```
private void friday(){  
    banishWinter();  
    decomposition();  
    doubleBeepers();  
    rhombaKarel();  
    if(extraTime()){  
        wordSearchKarel();  
    }  
}
```



Happy Friday