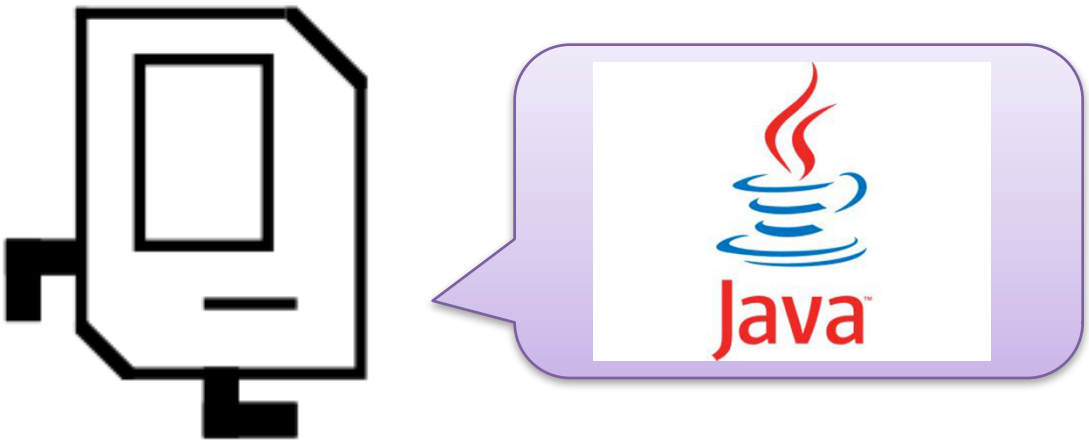
Decomposition

Chris Piech
CS106A, Stanford University

Karel the Robot



* While Karel is in Java, when you program your Karel assignment we ask that you stick to the concepts in the course reader

Method Definition

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

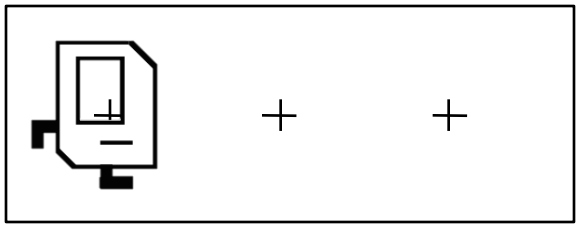
This adds a new command to Karels vocabulary

For Loops

```
public class Place99Beepers extends SuperKarel {
   public void run() {
      move();

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

      move();
   }
}
```

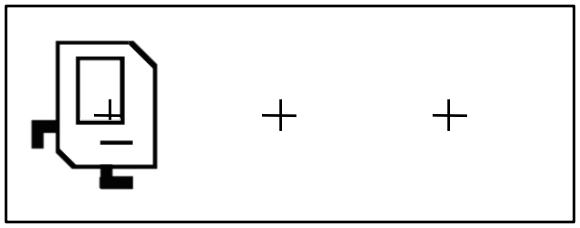

While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // If the condition passes, run "body".
        // Repeat.
        while(frontIsClear()) {
            // the "body"
            move();
            putBeeper();
        }
    }

}
```



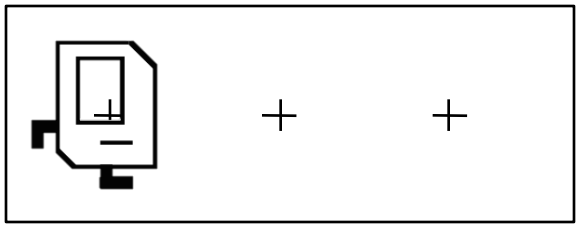
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



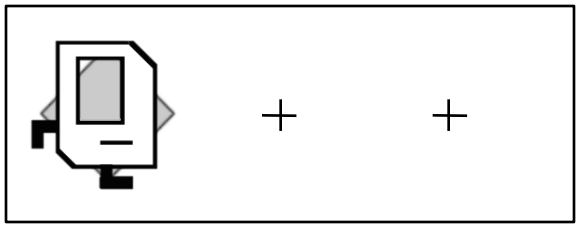
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



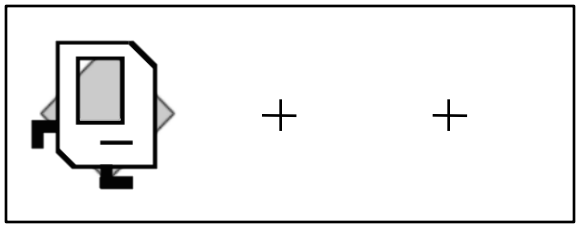
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



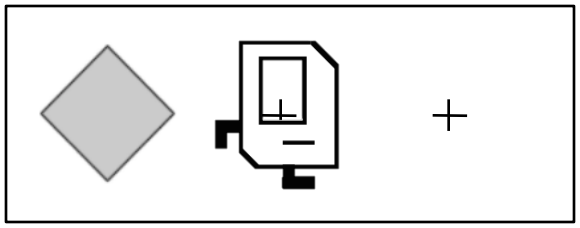
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



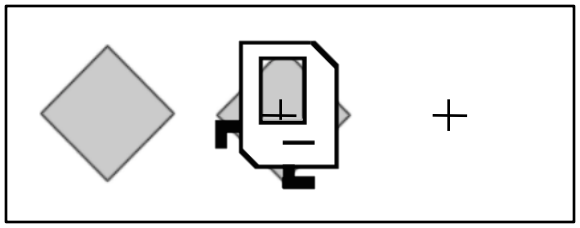
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



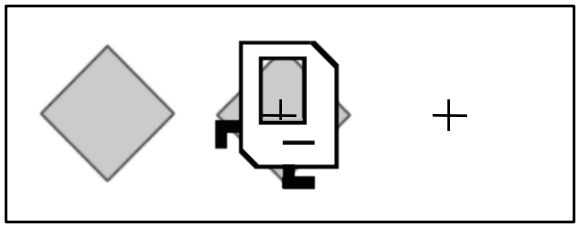
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



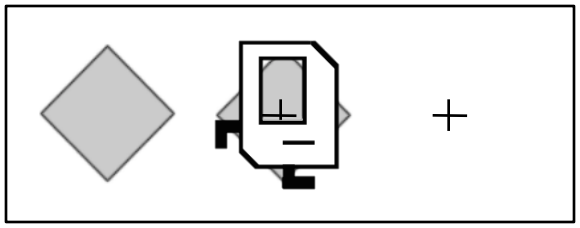
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



While Loops

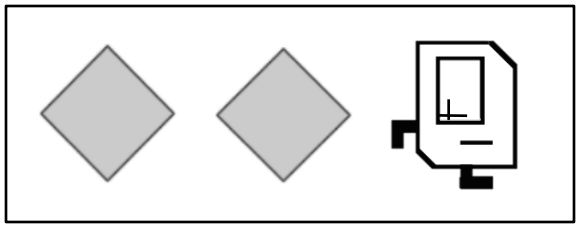
```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



While Loops

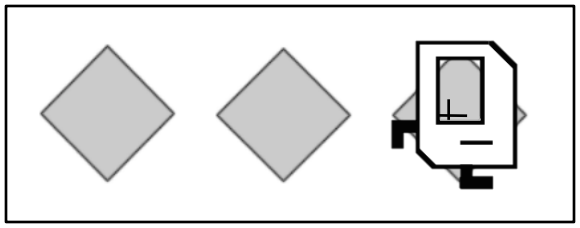
```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```

This is incredibly important!



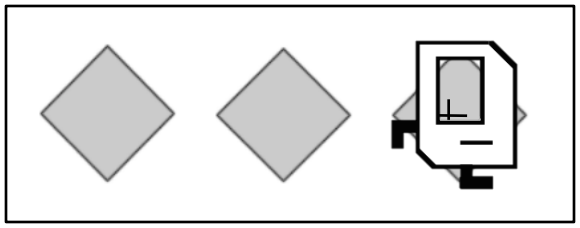
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



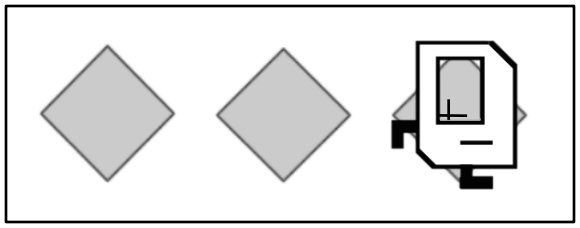
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



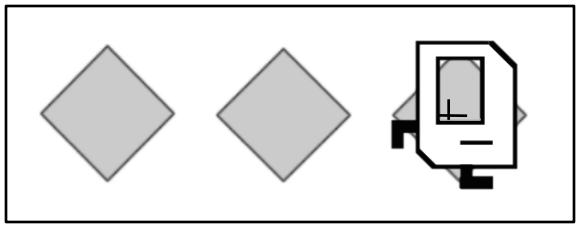
While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```

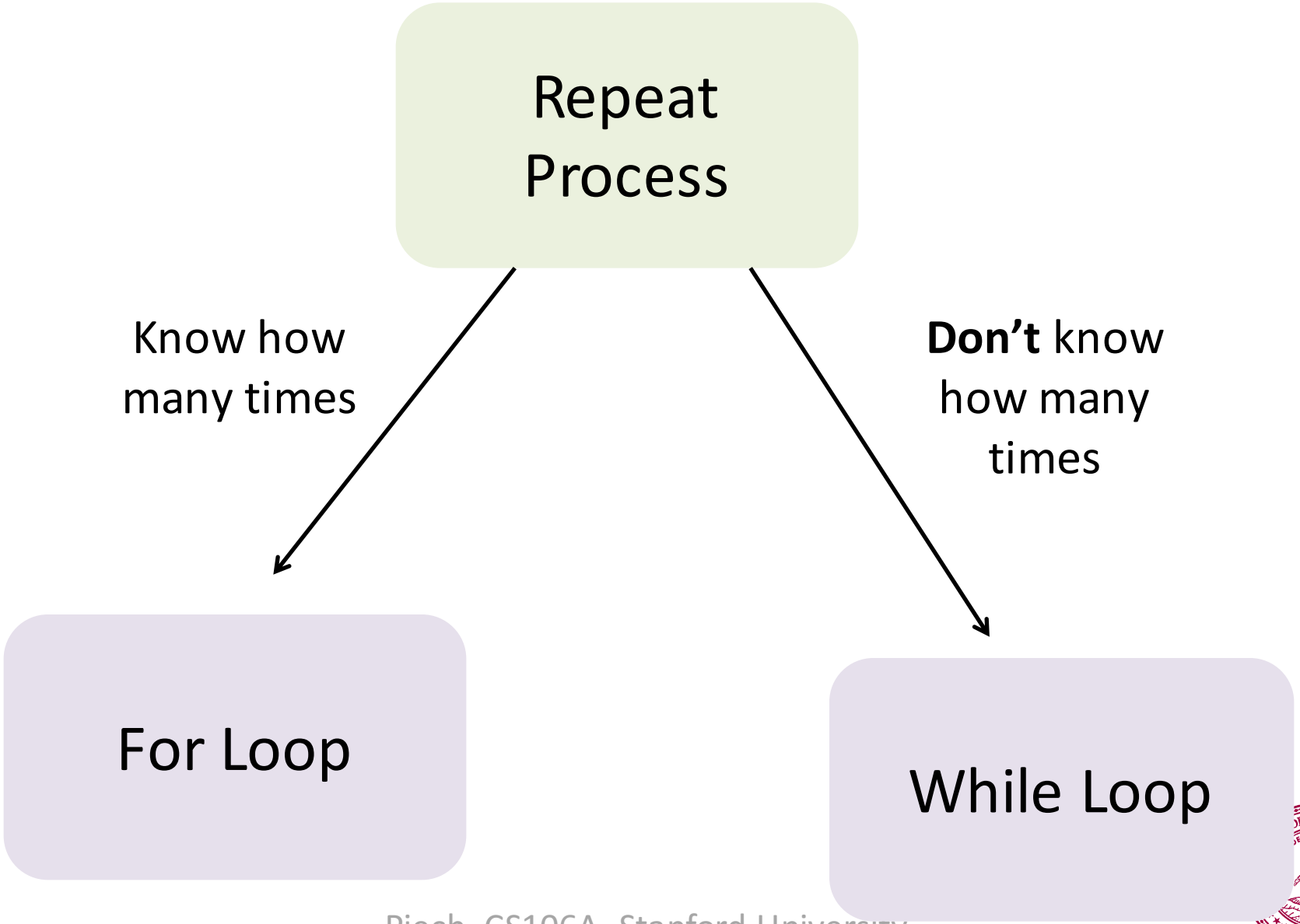


While Loops

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

        // example while loop
        while(frontIsClear()) {
            move();
            putBeeper();
        }
    }

}
```



Which Loop


Repeat Process
Know how many times
Don't know how many times
For Loop
While Loop

If Statement

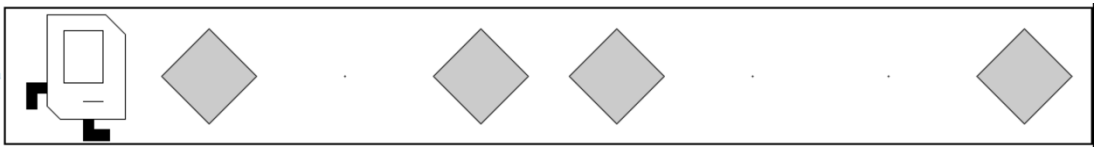
```
import stanford.karel.*;

public class IfExample extends SuperKarel{

    public void run() {
        safeMove();
    }

    private void safeMove() {
        if(frontIsClear()) {
            move();
        }
    }

}
```


If / Else Statement

```
import stanford.karel.*;

public class IfExample extends SuperKarel{

    public void run() {
        invertBeeper();
    }

    private void invertBeeper() {
        if(beepersPresent()) {
            pickBeeper();
        } else {
            putBeeper();
        }
    }

}
```

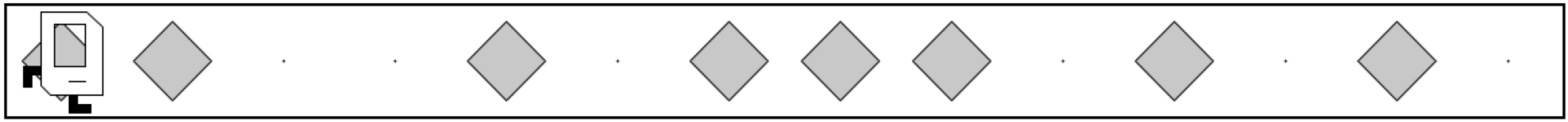

Put it All Together

Before:



After:

Before:



After:

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:

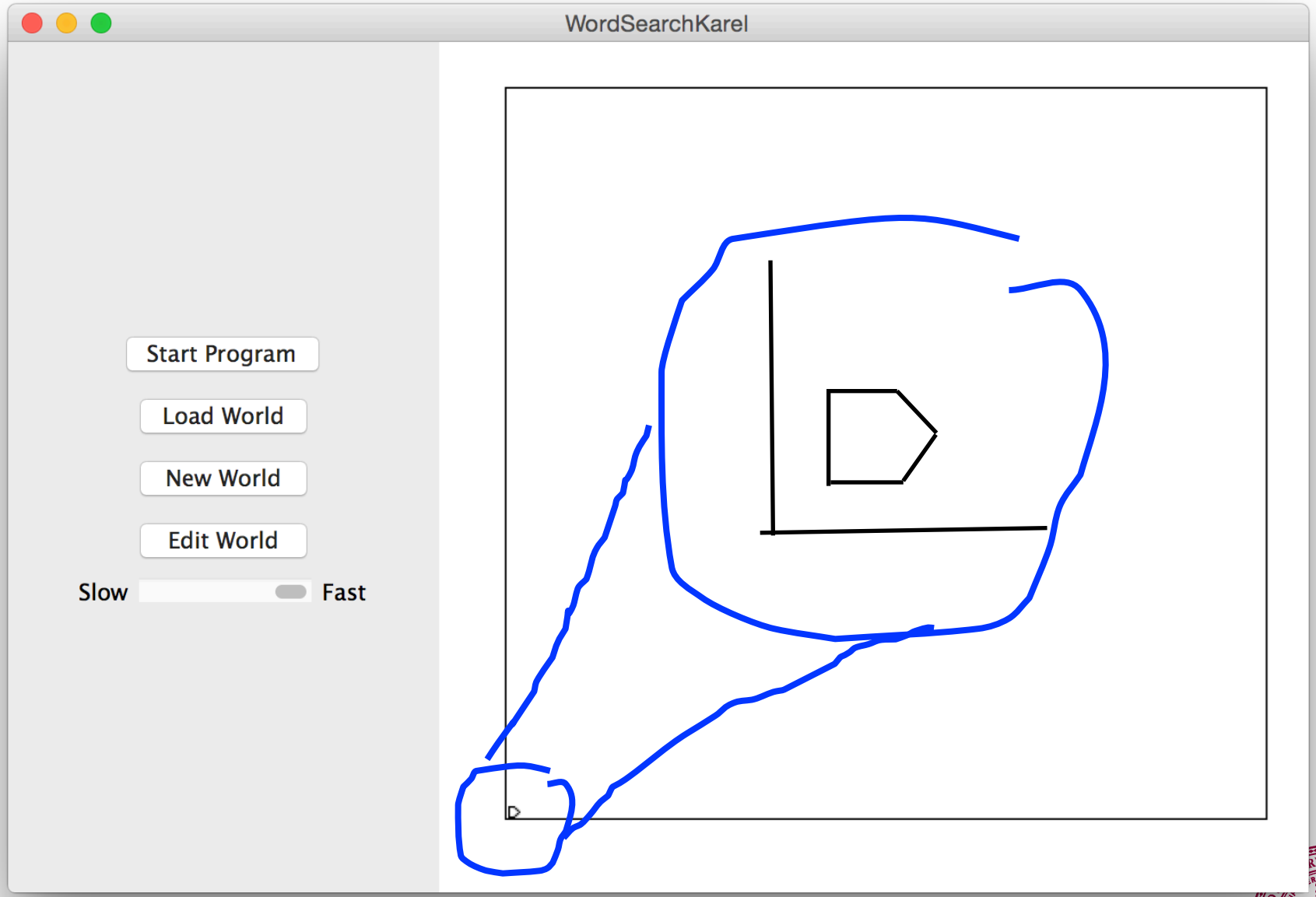
```
random()
random(p)
cornerColorIs(color)
```

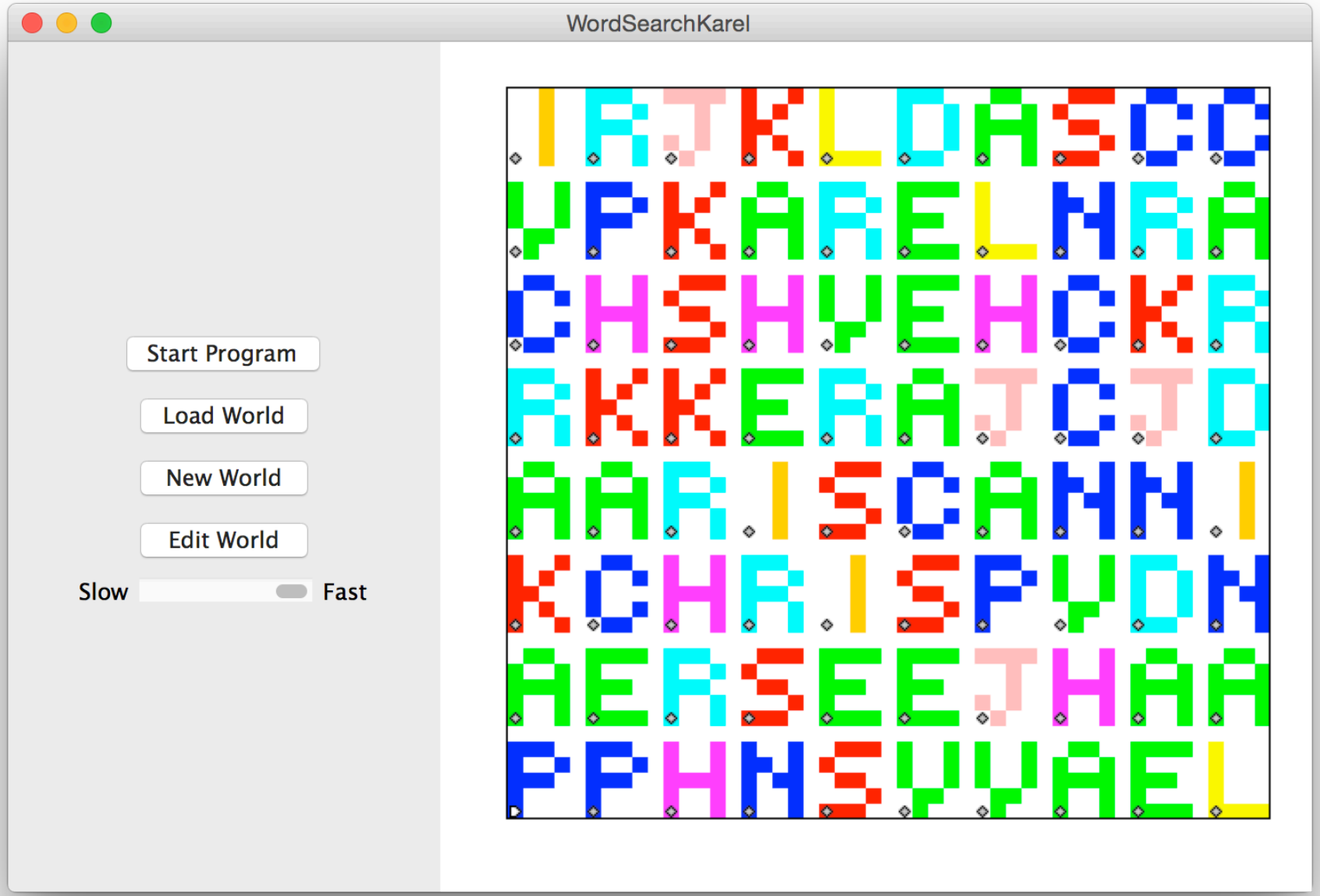
Today we will use:

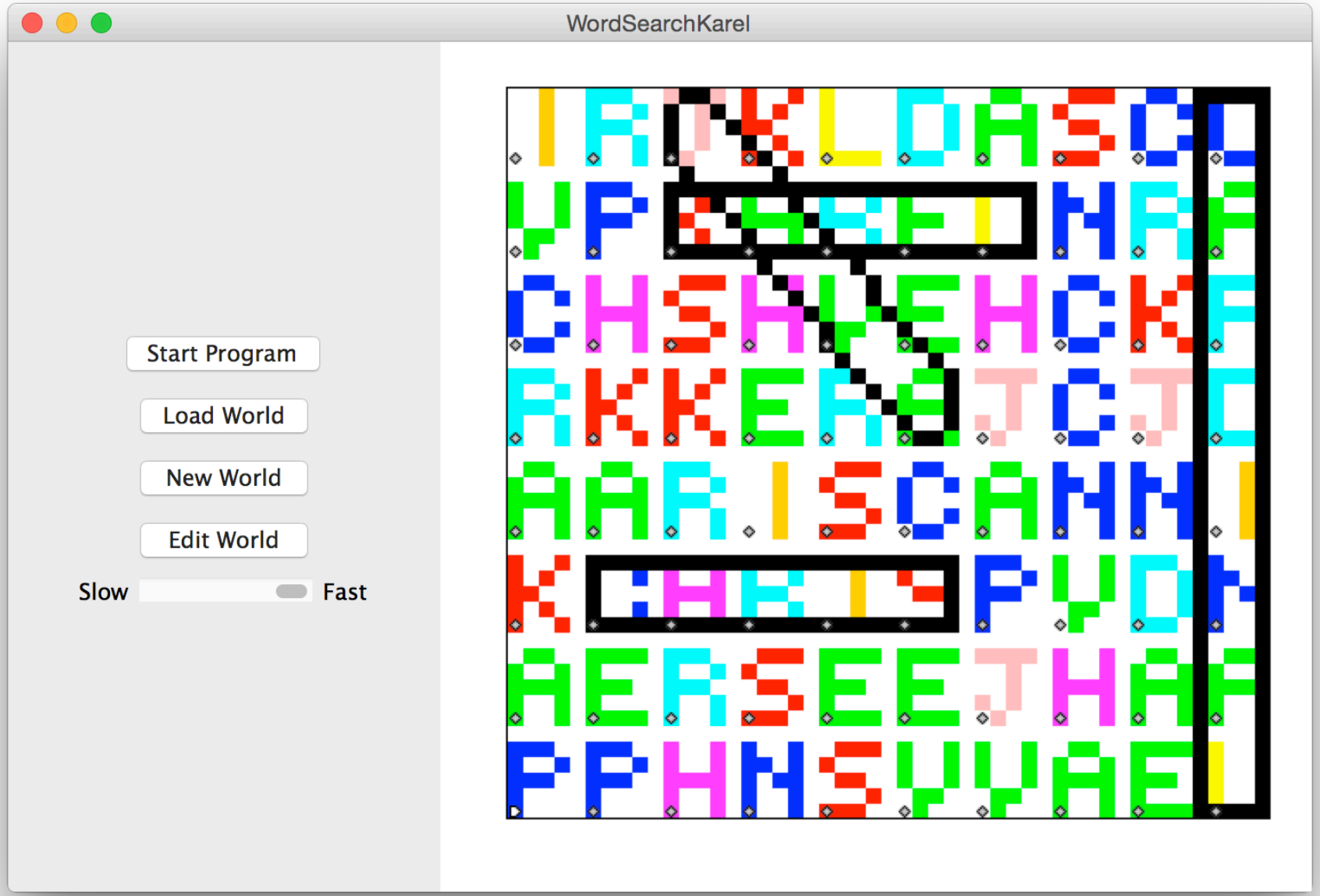
```
frontIsClear()
rightIsBlocked()
beepersPresent()
```

End review

First, a cool program

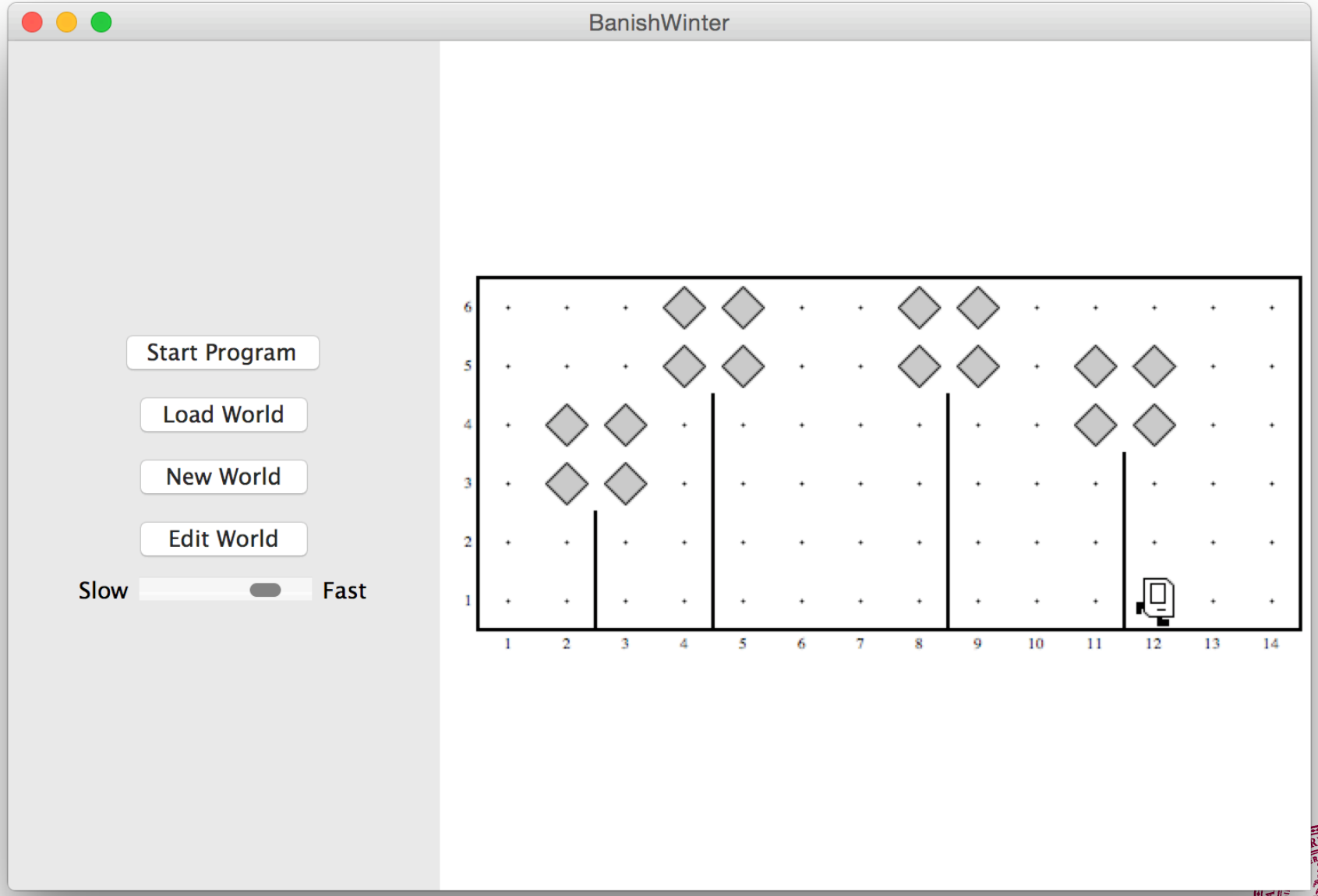

WordSearchKarel
Start Program
Load World
New World
Edit World
Slow
Fast


WordSearchKarel
Start Program
Load World
New World
Edit World
Slow
Fast


WordSearchKarel
Start Program
Load World
New World
Edit World
Slow
Fast

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

Banish Winter



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

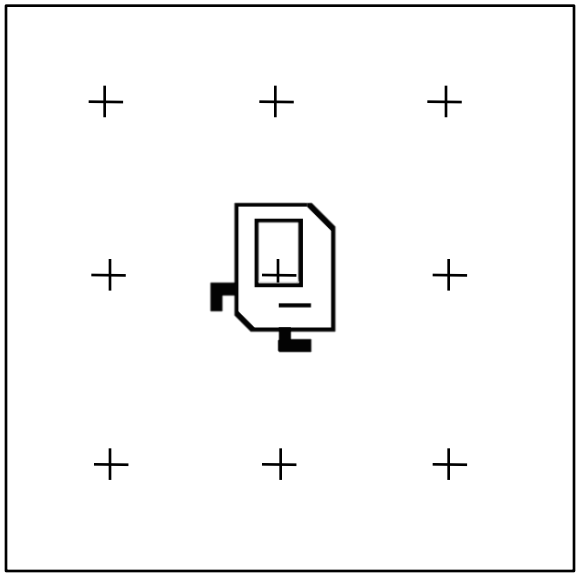
```
private void friday(){
    banishWinter();
    commonErrors();
    decomposition();
    doubleBeepers();
    if(extraTime()){
        wordSearchKarel();
    }
}
```

Infinite Loop

Lather,
Rinse,
Repeat

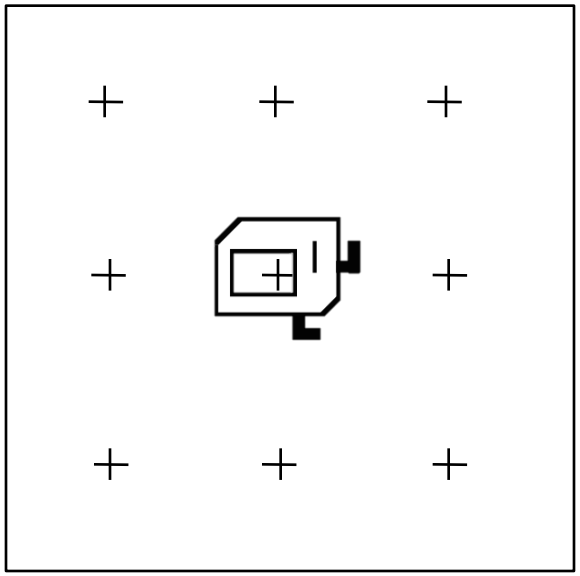
Infinite Loop

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



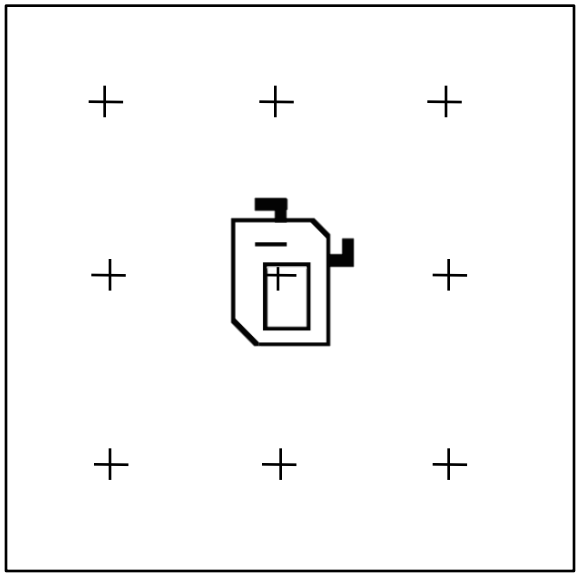
Infinite Loop

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



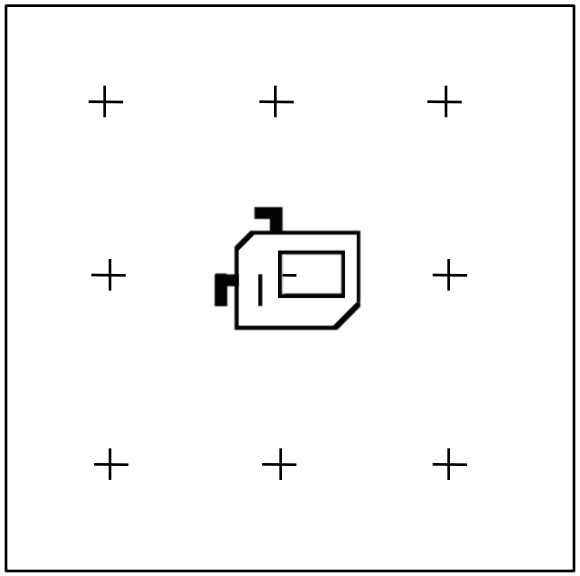
Infinite Loop

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



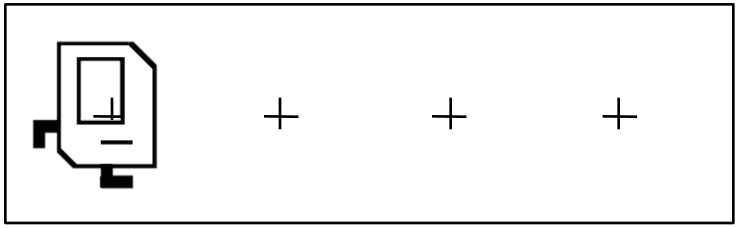
Infinite Loop

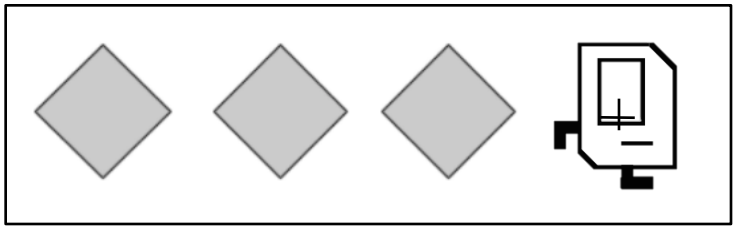
```
private void turnToWall() {
    while(leftIsClear()) {
        turnLeft();
    }
}
```



Off By One

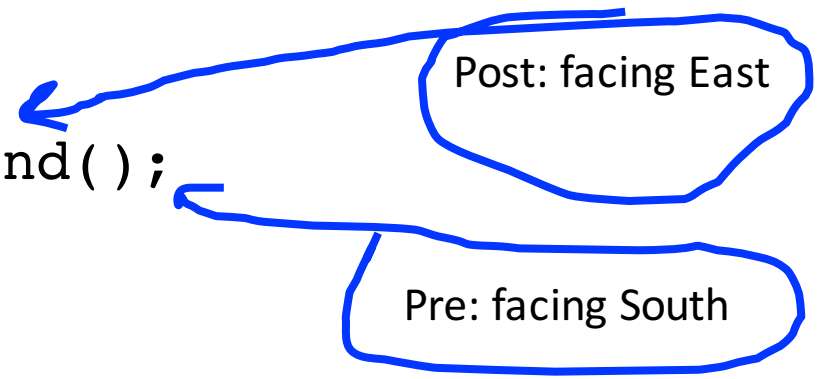
```
private void fillRow() {
    while(frontIsClear()) {
        putBeeper();
        move();
    }
}
```





Pre/Post that Don't Match

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



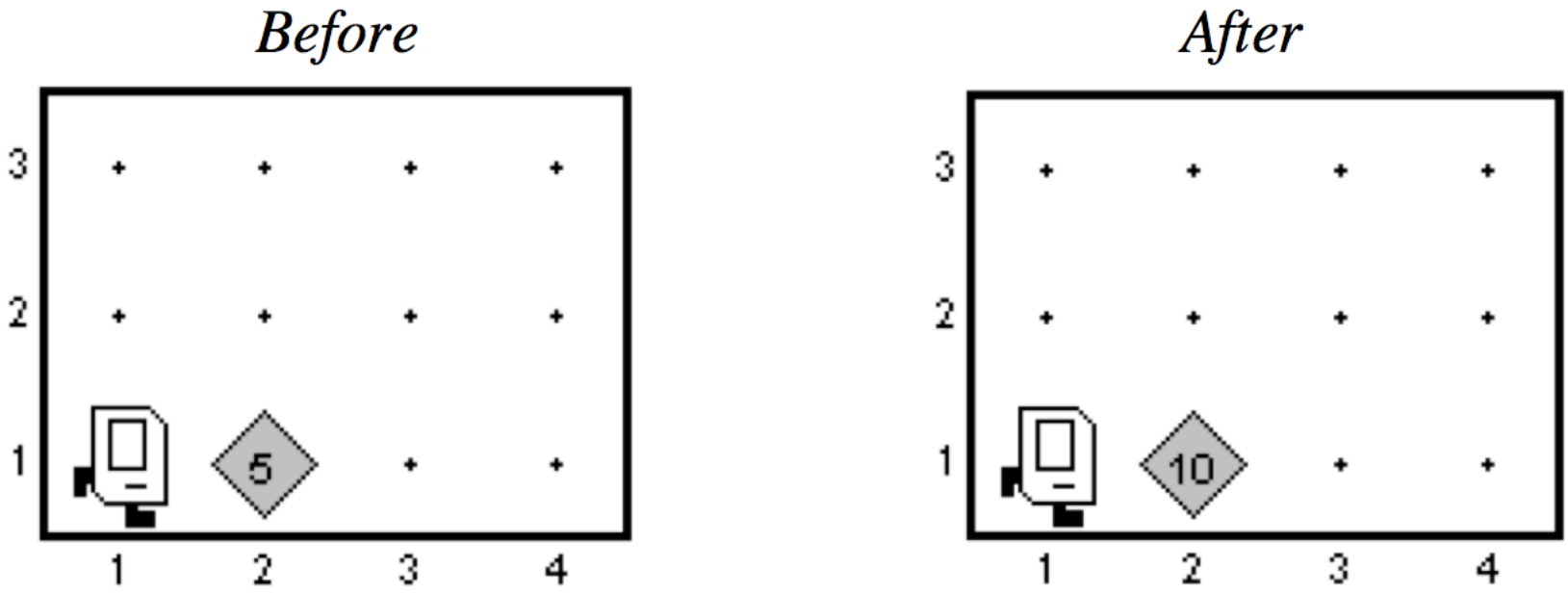
```
private void friday(){
    banishWinter();
    commonErrors();
    decomposition();
    doubleBeepers();
    if(extraTime()){
        wordSearchKarel();
    }
}
```

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

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

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

Double Beepers

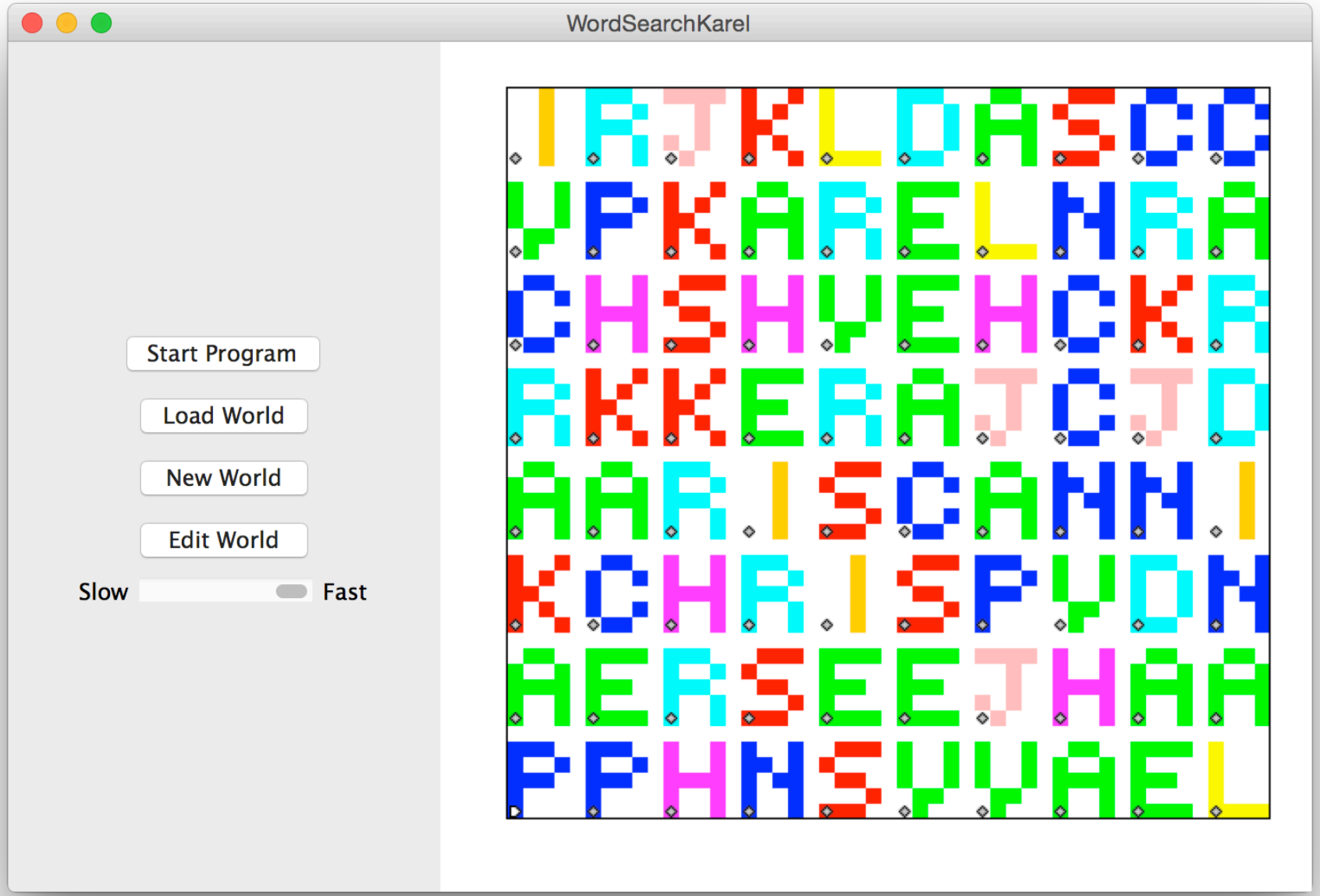



DO
YOUR
THING

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

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

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


WordSearchKarel
Start Program
Load World
New World
Edit World
Slow
Fast

Happy Friday