Repetition

Loops

- We have already seen instances where a robot needs to repeat instructions to perform a task
  - turnRight();
  - moveMile();

- By defining new instructions we have been able to minimize repetitive instructions
  - moveTenMiles();
  - moveHundredMiles();
  - moveThousandMiles();
WHILE loop: General form

while (<test>) {
    <instruction-list>;
}

- Reserved word `while` starts the instruction
- Parentheses enclose the `<test>`
- Braces enclose the `<instruction-list>;` in the usual way
- `<test>`: The same conditions used in the IF instructions

- A robot starts the loop by checking the `<test>` in its current situation
  - If TRUE, the robot executes the `<instruction-list>;` and rechecks the situation
  - If FALSE, the robot executes instructions that follow the WHILE loop
WHILE loop: find the beeper

```java
class BeeperFinder extends Robot {
    // usual constructor

    // assumes no intervening wall segments
    void findBeeper() {
        while (! nextToABeeper()) {
            move();
        }
    }
}
```
A robot is on a corner that may or may not have beepers on it. The robot is to make sure that the corner is clear of all beepers

```c
// fill in the blanks
void clearCorner() {
    while ( _____________________________ ) {
        ________________________________;
    }
}
```
Build a WHILE loop: 4 step process

- When a robot has to do something an unknown number of times (clear a corner of beepers)
  - **Step 1**: Identify the one test that must be true when the robot is finished with the loop
    - `! nextToABeeper()`
  - **Step 2**: Use the opposite form of the test
    - `nextToABeeper()`
  - **Step 3**: Within the WHILE, make progress toward completion of the WHILE
    - `pickBeeper()`
  - **Step 4**: Do whatever you need to do before or after the loop to complete solving the problem
    - For this problem, nothing else needs to be done
Errors to avoid with WHILE loops

- Infinite Execution
  - Look at the following loop

    ```java
    while (facingNorth()) {
        if (nextToABeeper())
            pickBeeper();
        move();
    }
    ```

  - Think of step 3 in building loops (Within the WHILE, make progress toward completion of the WHILE). Is this happening here?

- What will happen to the robot if he is not facing north? If he is facing north?
WHILE test checking

- When is the test of a WHILE loop checked?
  
  ```
  while ( facingNorth()) {
    if (nextToABeeper() )
      pickBeeper();
    move();
  }
  ```

- If there is more than one beeper on a corner, will the robot pick them all before moving on?
  - The robot checks the `<test>` only before it executes the `<instruction-list>`, not after each instruction in the list

- The number of test evaluations is always one more then the number of executions of the body of the WHILE instruction

- The `<instruction-list>` must affect the `<test>`
WHILE and IF instructions

What will happen with this code fragment?

```java
if (facingSouth()) {
    while (! facingSouth()) {
        turnLeft();
    }
}
```

Conflicting tests
What will happen with this code fragment?

```java
while (! frontIsClear() ) {
    if (frontIsClear()) {
        move();
    } else {
        turnLeft();
    }
}
```

Unnecessary test
WHILE and IF instructions

- What will happen with this code fragment?

```java
while (nextToABeeper() ) {
    if (nextToABeeper()) {
        pickBeeper();
    }
}
```

- Redundant test