The General Form of an \textbf{if} Statement:
\begin{verbatim}
if ("test")
{ "list of statements"
}
\end{verbatim}
Examples:
\begin{verbatim}
if (karel.canPickThing())
{ karel.pickThing();
  karel.turnLeft();
}

if (this.frontIsClear())
{ this.move();
}
\end{verbatim}

The General Form of a \textbf{while} Statement:
\begin{verbatim}
while ("test")
{ "list of statements"
}
\end{verbatim}
Examples:
\begin{verbatim}
while (karel.canPickThing())
{ karel.pickThing();
  karel.turnLeft();
}

while (this.frontIsClear())
{ this.move();
}
\end{verbatim}
While Loops Refresh

As we shall soon see, **while** loops are commonly used for **counting** and implementing **counters** inside of programs.

```plaintext
While(<<BooleanExpression>>)  
{ <<statement>>  }

While(<<BooleanExpression>>)  
{ <<statement>>  
  <<statement>>  
  <<statement>>  }
```

The General Form of a **while** Statement:

```
while («test»)
{ «list of statements» }
```

Examples:

```
while (karel.canPickThing())
{ karel.pickThing();
  karel.turnLeft();
}

while (this.frontIsClear())
{ this.move();
}
```
Chapter 5.2 – Temporary Variables (Local Variables)

Data Types

A data type is nothing but the type of the data that will be stored in memory.

Different data types require smaller or larger amounts of storage capacity, so when you declare a data type you are telling the program upfront the size of the storage container you want to set aside in memory. The datatype also dictates how the values in that container are interpreted.

Example by Analogy:
If you want to store a quart of water, then you need a container that will hold a quart. If you want to store a gallon of water, then you need a container that will hold a gallon. In both cases, you need the container before you can put the water in it. Declaring a data type is getting the container ready before you put anything in it—first the container, then what goes into it.
Temporary Memory, Variables

```java
int num = 0;  // int - gets the container ready
// num - tells which container to use
// 0 - integer that is put into the container

int numThings = 0, numStuff = 1;
```

**Data Types**

A variable in Java MUST have certain type associated with it which tells us what kind of data a variable can store, and whether that storage requires a small “container” in memory or a larger “container” (i.e., number of bits making up the storage space).

Data Types in the *Java Programming Language* are classified into two main groups:

- **Primitive Data Types**
  - We will discuss Primitive Data Types briefly now
- **Reference Data Types**
  - We will discuss Reference Data Types later in the Quarter
Classification of Data Types in Java Programming Language

- Primitive Types
  - boolean
  - returnAddress
- Numeric Types
- Integral Types
  - byte
  - short
  - int
  - long
  - char
- Floating-Point Types
  - float
  - double
- Reference Types
  - reference
- Class Types
- Interface Types
- Array Types
## Binary (A Quick Look)

### Byte (8 Bits)

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Signed vs. Unsigned

"Two's Complement"

## Binary Representation

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

## BYTE (Signed)

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

## BYTE (Unsigned)

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

0

-128

127

255
Primitive Data Types

- Primitive data types are built into the Java language and are not derived from classes.
- There are 8 Java primitive data types:

  - **byte**  -128 to 127
  - **short** -32,768 to 32,767
  - **int**  -2,147,483,648 to 2,147,483,647 (billion)
  - **long** -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (quintillion)
  - **float** 127 digits after the decimal point
  - **double** 1023 digits after the decimal point
  - **boolean**
    - True or False
  - **char**
    - Alphabetical characters stored as an ASCII numerical value (e.g., ‘A’ is 65, ‘B’ is 66)
Integer Data Types

- `byte`, `short`, `int`, and `long` are all integer data types.

- They can hold whole numbers such as 5, 10, 23, 89, etc.

- Integer data types cannot hold numbers that have a decimal point in them. (You need a `float` or `double` for that.)

- Integers embedded into Java source code are called `integer literals`. (You can also have `character literals`, such as 'a'.)
Variable Declarations

Variable Declarations take the following form:

\[ <\text{DataType}> \text{VariableName}; \]

- `byte` inches;
- `short` month;
- `int` speed;
- `long` timeStamp;
- `float` salesCommission;
- `double` distance;
Declaring an Integer Variable

Declaring an integer variable:

```java
int counter = 0;
```

Using a variable’s value:

```java
if (counter == 0) while (counter > 0) this.myMethod(counter);
{ ... } { ... }
```

Changing a variable’s value:

```
counter = counter + 1;
```

1. Get the variable’s current value
2. Add 1 to it.
3. Put the result back into the variable
To Summarize:
Temporary Variables (Local Variables)

Temporary variables
- are also called “local variables.”
- occur inside a method.
- store a value until the end of the method.
- have a type such as `Robot` or `int` that determines what kind of values may be stored in the variable.
- must be given an initial value.

Two examples:

```
int counter = 0;
Robot karel = new Robot(sanDiego, 1, 2, Direction.EAST);
```
A Quick Word About **Scope**

*Scope* is the region of the program in which a variable may be used.

Examples:

```java
public void method() {
    int tempVar = 0;
    // statements
}
```

```java
public void method() {
    int tempVar = 0;
    // statements
}
```

```java
public void method() {
    if («booleanExpression») {
        // statements
        int tempVar = 0;
        // statements
    }
    // statements
}
```

```java
public void method() {
    // statements
    int tempVar = 0;
    while («booleanExpression») {
        // statements
    }
    // statements
}
```

Rule: Use a variable from where it is declared to the end of the smallest enclosing block (set of braces).
FYI (Just for fun)

If you want to give the Robot a label when the program runs, then you will have to do something like this:

```java
mary.setLabel("Mary");
```

If you want to change the Robot's color, then you will have to include this at the top of your file

```java
import java.awt.Color;
```

And then do something like this:

```java
mary.setColor(Color.ORANGE);
```

Also, if you want to change the color of a wall, first give the wall a name, then color it according to that name:

```java
Wall w1 = new Wall(bothell,2,1,Direction.NORTH);
w1.setColor(Color.BLUE);
Wall w2 = new Wall(bothell,2,2,Direction.NORTH);
w2.setColor(Color.GREEN);
```