Statement Blocks

- A Compound statement (or block) is a set of statements enclosed in braces

```
{
   int x = 5;
   double y = readDouble("y: ");
   println("y = " + y);
}
```

- Variable's ***scope*** is block in which it is declared
- **Scope**: lifetime of variable
  - When and where the variable is available to be used

`if` statement

- **General form:** `if (`*condition*`) {`
  statements
  `}`

Any boolean condition/variable

```
if ((num % 2) == 0)         // can omit braces
  println("num is even"); // if one statement

if ((num % 2) == 0) {
  println("num is even");
  println("Oh, and Beat Cal!");
}
```

- Use braces with **`if`** with more than one statement
- Good idea to use braces (block) even if there is only one statement in the **`if`**

`if-else` statement

- **General form:** `if (`*condition*`) {`
  statements
  `} else {`
  statements
  `}`

```
if ((num % 2) == 0) {
  println("num is even");
} else {
  println("num is odd");
  println("and so are you");
}
```

Cascading `if`

```
if (score >= 90) {
  println("A");
} else if (score >= 80) {
  println("B");
} else if (score >= 70) {
  println("C");
} else {
  println("Bad Times");
}
```

switch statement

```
int day = readInt("Day of week as int: ");
switch (day) {
    case 0:
         println("Sunday");
         break;
    case 6:
         println("Saturday");
         break;
    default:
         println("Weekday");
         break;
}
```

`for` loop

- **General form:**

```
for (init ; condition ; step) {
        statements
}
```

- ***init*** done once at start of loop
- ***condition*** checked before every iteration through loop
  - we execute ***statements*** if condition is true
- ***step*** every time through loop after ***statements***

`for` loop

- **Example:**

```
for (int i = 0; i < 5; i++) {
    println(i);
}
```

```
0
1
2
3
4
```

- As computer scientists, we count starting at 0

`for` loop

- **Another Example:**

```
for (int i = 6; i > 0; i -= 2) {
    println(i);
}
```

```
6
4
2
```

- Note that 0 is not displayed!

`while` loop

- **General form:**

```
while ( condition ) {
        statements
}
```

- ***condition*** checked before every iteration through loop
  - we execute ***statements*** if condition is true

`while` loop

- **Example:**

```
int x = 15;
while (x > 1) {
    x /= 2;
    println(x);
}
```

```
7
3
1
```

Loop and a half ?!

```
public class Add extends ConsoleProgram {

  // Constant value for SENTINEL
  private static final int SENTINEL = 0;

  public void run() {
     int total = 0;

     int val = readInt("Enter val:");
     while (val != SENTINEL) {
          total += val;
          val = readInt("Enter val:");
     }
     println("Total = " + total);
  }
}
```

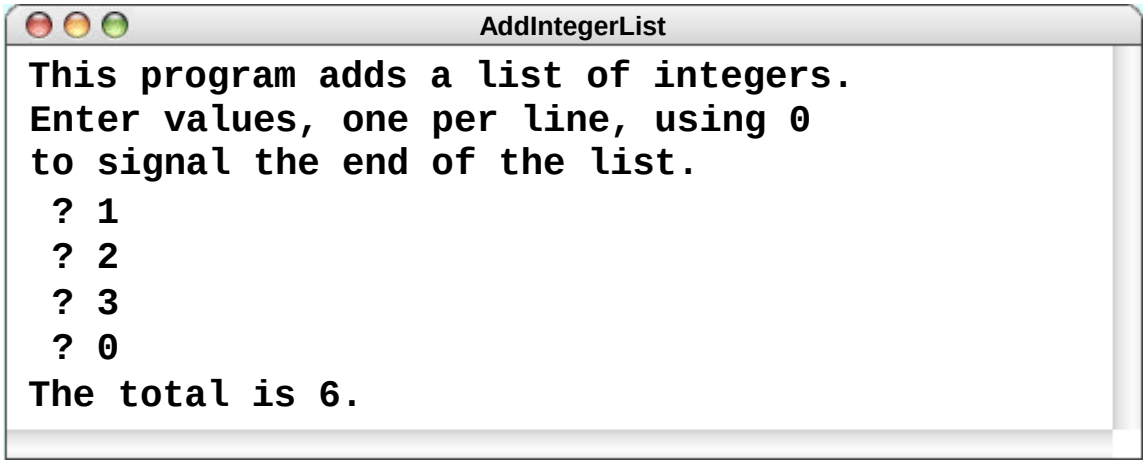
Loop and a half ?!

```
public class Add extends ConsoleProgram {

  // Constant value for SENTINEL
  private static final int SENTINEL = 0;

  public void run() {
     int total = 0;

     while (true) {
          int val = readInt("Enter val:");
          if (val == SENTINEL) break;
          total += val;
     }
     println("Total = " + total);
  }
}
```

The `AddIntegerList` Program

```
public void run() {
   println("This program adds a list of integers.");
   println("Enter values, one per line, using " + SENTINEL);
   println("to signal the end of the list.");
   int total = 0;
   while (true) {
      int value = readInt(" ? ");
      if (value == SENTINEL) break;
      total += value;
   }
   println("The total is " + total + ".");
}
```

value	total
0	6



skip simulation

`for` versus `while`

```
for (init ; test ; step) {
    statements
}
```

- **`for`** loop used for *definite* iteration
- Generally, we know how many times we want to iterate

```
init
while ( test ) {
    statements
    step
}
```

- **`while`** loop used for *indefinite* iteration
- Generally, don’t know how many times to iterate beforehand