Intro to Java I: Variables, Arrays, Loops

CS 1025 Computer Science Fundamentals I

Stephen M. Watt
University of Western Ontario

A First Java Example

```
// This is a first example in Java for those who have
// written programs in other languages.

class Hello {
      public static void main(String[] args) {
            System.out.println("Hello world!");
      }
}
```

- The blue text is a comment.
- The green text is boiler plate that we will go over later.
- The red text is where you can put something to output.

Constants

- Java provides syntax for constants of various types
  - Integers: *e.g.* 1, 2, 34567
  - Floating point numbers: *e.g.* 3.2, 6.022e23 *(6.022×10^23)*
  - Text strings: *e.g.* “Hello World”
  - Logical values: *i.e.* true, false

Variables

- Computed values may be stored in *variables.*

- A variable has
  - a *name*,

    e.g. `myCounter` (made up of letters and numbers)
  - a *type*,

    e.g. `int`, `String`, `double` (floating point), `boolean`
  - a *scope*,

    where the variable may be used.
  - a *value*,

    may change over time.

Uses of variables

- To give a *name* to some value:

```
nine = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1;
```

- To keep track of something that changes over time:

```
howMany = 0;
...
howMany = howMany+1;
```

Declarations

- A variable must be *declared* before it is *used*.
- A declaration is given by specifying a *type* and naming variables that will have that type.
- Variables may optionally be given initial values.
- Several declarations may be given.

Output

- You can display values of many types using

```
System.out.print("Hello");          // Display text.
System.out.print(9);                // Display a number.
System.out.print("Value is" + 9); // Text then number.

System.out.println("Hello");      // Same, but with
System.out.println(9);             // new line afterwards.
System.out.println("Value is" + 9);
```

- Why this works will come later. For now just use it.
- The "+" is *not* addition.

Expressions

- Values may be combined using various operations in *expressions*.

```
(a+b)*(c+d); // Addition, multiplication

a == b         // Equal?  Returns true or false.
a != b         // Not equal?

a < b    a <= b   a > b   a >= b
```

Example 2

```
// This is a first example in Java for those who have
// written programs in other languages.

class Hello {
     public static void main(String[] args) {
          int n = 2;
          n = n * n;
          n = n * n;
          n = n * n;
          n = n * n;
          System.out.println("The value of n is " + n);
     }
}
```

Arrays

- Several values of the same type may be kept together in an *array.*

- A variables whose values are arrays of elements of type *T* is declared, *e.g.* as

```
T[] myVariable, yourVariable;
```

Here *T* would be replaced by **`int`**, **`String`**, ...

Arrays II

- The places in an array are specified by giving their position, which is an integer, starting from 0:

```
A[0] = 12;   A[1] = A[1] + A[1];
```

- The number of elements in an array A is given by A.length.

```
n = A.length;
```

Arrays III

- Arrays are created using the *"new"* operator.

```
int[] a;
...
a = new int[4];

a[0] = 0;
a[1] = 1;
a[2] = 4;
a[3] = 9;
```

Control Flow I

- A *compound statement* , using *braces* (“{”, “}”) executes any number of statements, one after another:

```
{
  a = 2;
  System.out.println(a);
  a = a + a;
  System.out.prinln(a);
}
```

Displays

```
2
4
```

Control Flow II

- A *conditional statement* , using “if”, executes a statement (or doesn’t) depending on whether a value is true (or false):

```
if (a < 4) {
   a = b + 2;
   System.out.println(a);
 }
```

Control Flow III

- A *loop*, or “iteration statement” is given using “for”

```
for (i = 0; i < a.length; i = i + 1)
   a[i] = 0;
```

- The general form is

```
for (start; check; step) body
```

- Often *body* is a compound statement. { S1; S2; ... Sn;}

Example 3: Prime Number Sieve

- We can put this all together in an example.
- Remember that a positive integer is *prime* if it is not a multiple of another.
- We can find prime numbers by crossing out multiples.

Example (contd)

- Start with a row of markers:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
 x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x  x
```

- Start with 2. Keep its marker, then remove the markers for all its multiples.

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
 x  x  x  x     x     x     x     x     x     x     x
```

- Now do the same thing for 3. Keep it, but remove its multiples.

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
 x  x  x  x     x     x           x     x           x
```

Example (contd)

- Now look at the slot for 4. There is no marker there because it was removed when we were removing multiples of 2. So its multiples are already taken care of.

- Represent the row of markers as an array of booleans, and initialize them all to be `true`. We'll look up to 1000.

```
boolean[] marks = new boolean[1000];

for (int i = 0; i < marks.length; i = i+1)
    marks[i] = true;
```

- We can remove the markers (set the spots in the array to false) for the multiples of k (but not k itself) using

```
for (int j = 2; j*k < marks.length; j = j + 1)
      marks[j*k] = false;
```

- But we don't want to do this unless `marker[k]` is true.

```
if (marks[k])
    for (int j = 2; j*k < marks.length; j = j + 1)
        marks[j*k] = false;
```

Example (contd)

- We'll do this for all k from 2 up to the array size.

```
// For each number k >= 2
for (int k = 2; k < marks.length; k = k+1) {
    // If it is prime, cross out its muliples.
    if (marks[k])
        for (int j=2; j*k < marks.length; j =j+1)
            marks[j*k] = false;
}
```

Example (conclusion)

```
class Primes {
     public static void main(String[] args) {
          // Create array and initialize all entries.
          boolean[] marks = new boolean[1000];
          for (int i = 0; i < marks.length; i = i+1)
               marks[i] = true;
          // For each number k >= 2
          for (int k = 2; k < marks.length; k = k+1)
               // If it is prime, cross out its multiples.
               if (marks[k])
                    for (int j=2; j*k< marks.length; j =j+1)
                         marks[j*k] = false;
          // Display results
          for (int i = 0; i < marks.length; i = i+1)
               if (marks[i])
                    System.out.println("A prime is " + i);
     }
}
```