Building Java Programs

Chapter 2
Variables and For Loops

reading: 2.2 - 2.3

Good Source of FIBER & Made with WHOLE GRAIN
NEW LOOK Same GREAT Taste
Kellogg's
FROOT LOOPS
Toucan SAM
CEREAL

Variables

reading: 2.2

Receipt example

What's bad about the following code?

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);

        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);
        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);
        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .08 +
                           (38 + 40 + 30) * .15);
    }
}
```

- The subtotal expression `(38 + 40 + 30)` is repeated
- So many `println` statements

Variables

- **variable**: A piece of the computer's memory that is given a name and type, and can store a value.
  - Like preset stations on a car stereo, or cell phone speed dial:

  - Steps for using a variable:
    - *Declare* it - state its name and type
    - *Initialize* it - store a value into it
    - *Use* it - print it or use it as part of an expression

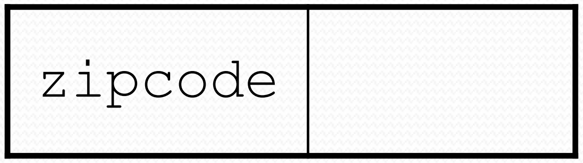
Declaration

- **variable declaration**: Sets aside memory for storing a value.
  - Variables must be declared before they can be used.

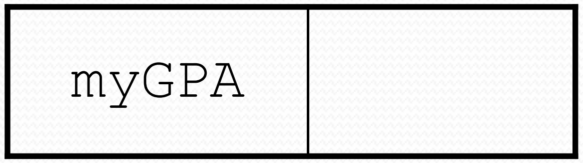
- Syntax:

  type name;

  - `int zipcode;`



  - `double myGPA;`



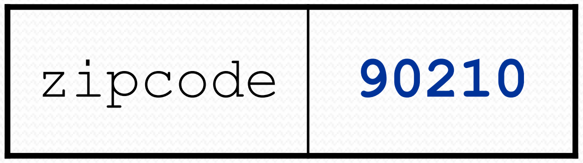
Assignment

- **assignment**: Stores a value into a variable.
  - The value can be an expression; the variable stores its result.

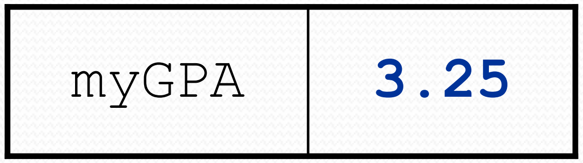
- Syntax:

  name = **expression**;

  - `int zipcode;`
    `zipcode = 90210;`



  - `double myGPA;`
    `myGPA = 1.0 + 2.25;`



Using variables

- Once given a value, a variable can be used in expressions:

```
int x;
x = 3;
System.out.println("x is " + x);        // x is 3
System.out.println(5 * x - 1);          // 14
```

- You can assign a value more than once:

x	11

```
int x;
x = 3;
System.out.println(x + " here");       // 3 here

x = 4 + 7;
System.out.println("now x is " + x);  // now x is 11
```

Declaration/initialization

- A variable can be declared/initialized in one statement.

- Syntax:

  type name = **expression**;

  - `int x = (11 % 3) + 12;`

x	14

  - `double myGPA = 3.95;`

myGPA	3.95

Assignment vs. algebra

- Assignment uses `=` , but it is not an algebraic equation.
  - `=` means, *"store the value at right in variable at left"*
  - `x = 3;` means, *"`x` becomes `3`"* or *"`x` should now store `3`"*

- **ERROR**: `3 = 1 + 2;` is an illegal statement, because `3` is not a variable.

- What happens here?

```
int x = 3;
x = x + 2;    // ???
```

x	**5**

Assignment exercise

- What is the output of the following Java code?

```
int x;
x = 3;
int y = x;
x = 5;
y = y + x;
System.out.println(x);
System.out.println(y);
```

Assignment and types

- A variable can only store a value of its own type.
  - `int x = 2.5;      // ERROR: incompatible types`

- An `int` value can be stored in a `double` variable.
  - The value is converted into the equivalent real number.

  - `double myGPA = 4;`

`myGPA`	`4.0`

  - `double avg = **11 / 2**;`

`avg`	**`5.0`**

    - Why does `avg` store `5.0` and not `5.5` ?

Compiler errors

- A variable can't be used until it is assigned a value.

  -
    ```
    int x;
    System.out.println(x);    // ERROR: x has no value
    ```

- You may not declare the same variable twice.

  -
    ```
    int x;
    int x;                    // ERROR: x already exists
    ```

  -
    ```
    int x = 3;
    int x = 5;                // ERROR: x already exists
    ```
    - How can this code be fixed?

Printing a variable's value

- Use + to print a string and a variable's value on one line.

  -
    ```
    double grade = (95.1 + 71.9 + 82.6) / 3.0;
    System.out.println("Your grade was " + grade);

    int students = 11 + 17 + 4 + 19 + 14;
    System.out.println("There are " + students +
                       " students in the course.");
    ```
  - Output:

    ```
    Your grade was 83.2
    There are 65 students in the course.
    ```

Receipt question

Improve the receipt program using variables.

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);

        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);

        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);

        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .15 +
                           (38 + 40 + 30) * .08);
    }
}
```

Receipt answer

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        double subtotal = 38 + 40 + 30;
        double tax = subtotal * .08;
        double tip = subtotal * .15;
        double total = subtotal + tax + tip;

        System.out.println("Subtotal:  " + subtotal);
        System.out.println("Tax:  " + tax);
        System.out.println("Tip:  " + tip);
        System.out.println("Total:  " + total);
    }
}
```

Repetition with `for` loops

- So far, repeating an action results in redundant code:

```
drawDiamonds();
drawDiamonds();
drawDiamonds();
drawDiamonds();
drawDiamonds();
drawX();
```

- Java's **`for` loop** statement performs a task many times.

```
for (int i = 1; i <= 5; i++) {    // repeat 5 times
    drawDiamonds();
}
drawX();
```

for loop syntax

```
for (initialization; test; update) {      header
    statement;
    statement;
    ...                                   body
    statement;
}
```

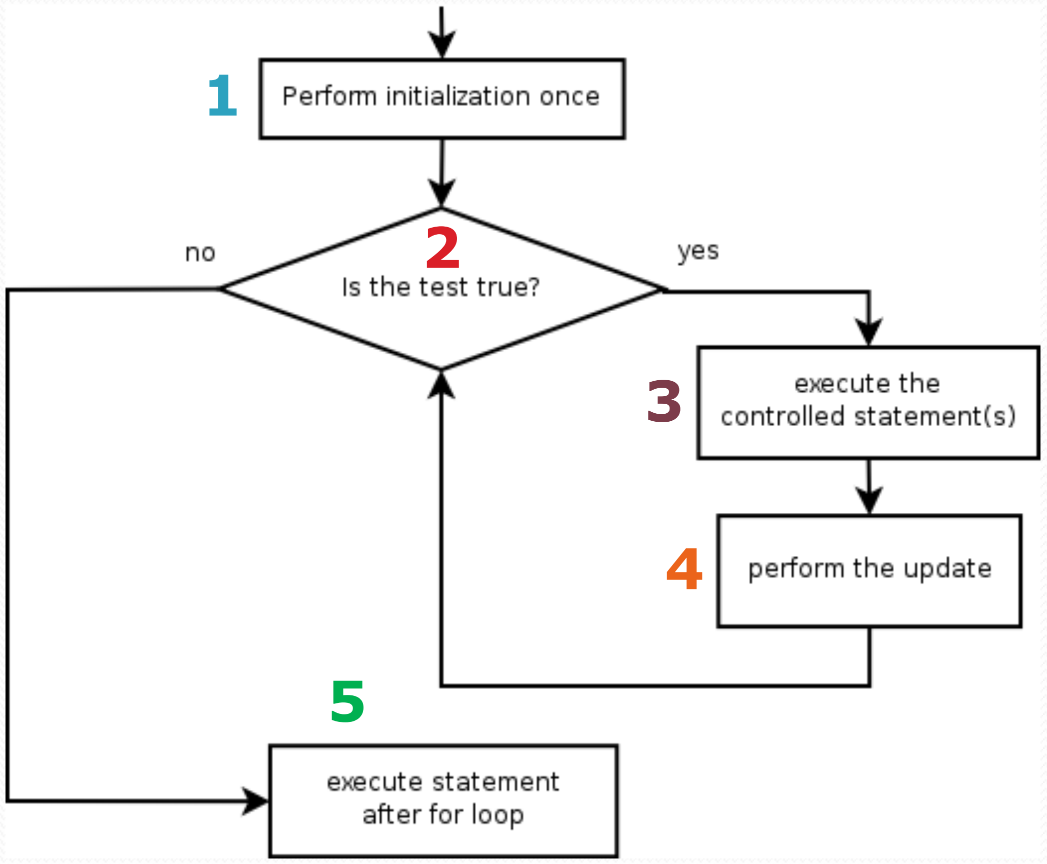
- Perform **initialization** once.
- Repeat the following:
  - Check if the **test** is true. If not, stop.
  - Execute the **statement**s.
  - Perform the **update**.

Loop walkthrough

```
               1              2          4
for (int count = 1; count <= 4; count = count + 1) {
  3 System.out.println("Hello World!");
}
System.out.println("Whoo!");
5
```

Output:

```
Hello World!
Hello World!
Hello World!
Hello World!
Whoo!
```


1
Perform initialization once
2
Is the test true?
no
yes
3
execute the controlled statement(s)
4
perform the update
5
execute statement after for loop

Control structures

- **Control structure**: a programming construct that affects the flow of a program's execution

- Controlled code may include one or more statements

- The for loop is an example of a looping control structure

Initialization

```
for (int i = 1; i <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tells Java what variable to use in the loop
  - The variable is called a *loop counter*
    - can use any name, not just `i`
    - can start at any value, not just `1`
    - only valid in the loop
  - Performed once as the loop begins

Test

```
for (int i = 1; i <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tests the loop counter variable against a limit
  - Uses comparison operators:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Increment and decrement

shortcuts to increase or decrease a variable's value by 1

Shorthand	Equivalent longer version
variable`++;`	**variable** `=` **variable** `+ 1;`
variable`--;`	**variable** `=` **variable** `- 1;`

```
int x = 2;
x++;                    // x = x + 1;
                        // x now stores 3

double gpa = 2.5;
gpa--;                  // gpa = gpa - 1;
                        // gpa now stores 1.5
```

Modify-and-assign operators

shortcuts to modify a variable's value

Shorthand	Equivalent longer version
variable `+=` **value**;	**variable** `=` **variable** `+` **value**;
variable `-=` **value**;	**variable** `=` **variable** `-` **value**;
variable `*=` **value**;	**variable** `=` **variable** `*` **value**;
variable `/=` **value**;	**variable** `=` **variable** `/` **value**;
variable `%=` **value**;	**variable** `=` **variable** `%` **value**;

```
x += 3;                    // x = x + 3;

gpa -= 0.5;                // gpa = gpa - 0.5;

number *= 2;               // number = number * 2;
```

Repetition over a range

```
System.out.println("1 squared = " + 1 * 1);
System.out.println("2 squared = " + 2 * 2);
System.out.println("3 squared = " + 3 * 3);
System.out.println("4 squared = " + 4 * 4);
System.out.println("5 squared = " + 5 * 5);
System.out.println("6 squared = " + 6 * 6);
```

- Intuition: "I want to print a line for each number from 1 to 6"

- The `for` loop does exactly that!

```
for (int i = 1; i <= 6; i++) {
    System.out.println(i + " squared = " + (i * i));
}
```

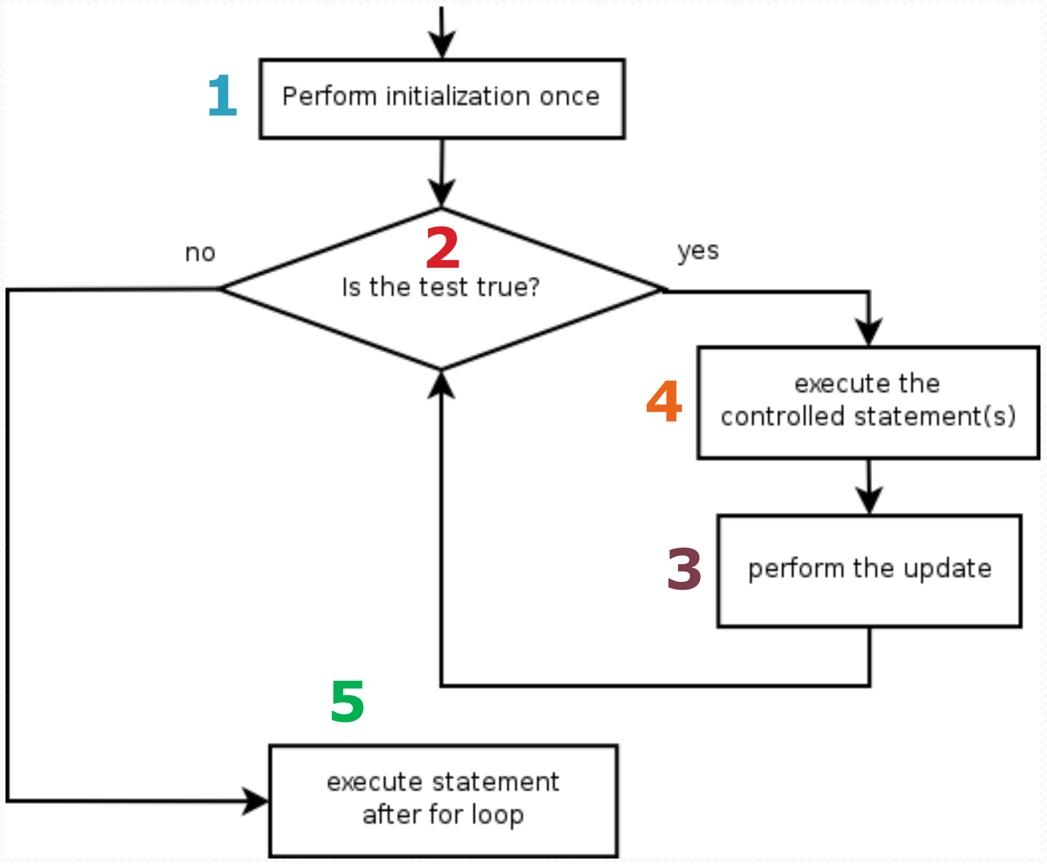
- "For each integer **i** from 1 through 6, print ..."

Loop walkthrough

```
               1       2     3
for (int i = 1; i <= 4; i++) {
  4 System.out.println(i + " squared = " + (i * i));
}
System.out.println("Whoo!");
5
```

Output:

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
Whoo!
```


1
Perform initialization once
2
Is the test true?
no
yes
4
execute the controlled statement(s)
3
perform the update
5
execute statement after for loop

Multi-line loop body

```
System.out.println("+----+");
for (int i = 1; i <= 3; i++) {
    System.out.println("\\    /");
    System.out.println("/    \\");
}
System.out.println("+----+");
```

- Output:

```
+----+
\    /
/    \
\    /
/    \
\    /
/    \
+----+
```

Expressions for counter

```
int highTemp = 5;
for (int i = -3; i <= highTemp / 2; i++) {
    System.out.println(i * 1.8 + 32);
}
```

- Output:

```
26.6
28.4
30.2
32.0
33.8
35.6
```

System.out.print

- Prints without moving to a new line
  - allows you to print partial messages on the same line

```
int highestTemp = 5;
for (int i = -3; i <= highestTemp / 2; i++) {
    System.out.print((i * 1.8 + 32) + "  ");
}
```

  - Output:

```
26.6  28.4  30.2  32.0  33.8  35.6
```

    - Concatenate " " to separate the numbers

Counting down

- The **update** can use `--` to make the loop count down.
  - The **test** must say `>` instead of `<`

```
System.out.print("T-minus ");
for (int i = 10; i >= 1; i--) {
     System.out.print(i + ", ");
}
System.out.println("blastoff!");
System.out.println("The end.");
```

  - Output:

```
T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, blastoff!
The end.
```