Definite Loops

and

Interactive Input

But first some really cool assignment expressions!!

Definitive Loops

The `for` loop

Repetition with `for` loops

- So far, repeating a statement is redundant:

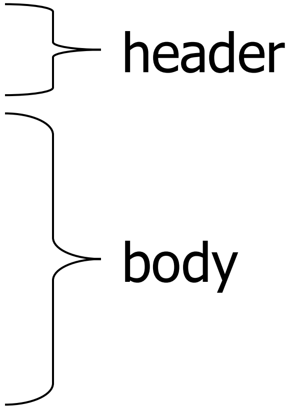
```
System.out.println("Homer says:");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

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

```
System.out.println("Homer says:");
for (int i = 1; i <= 4; i++) {   // repeat 4 times
    System.out.println("I am so smart");
}
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

`for` loop syntax

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



header

body

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

Initialization

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

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

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

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));
}
```

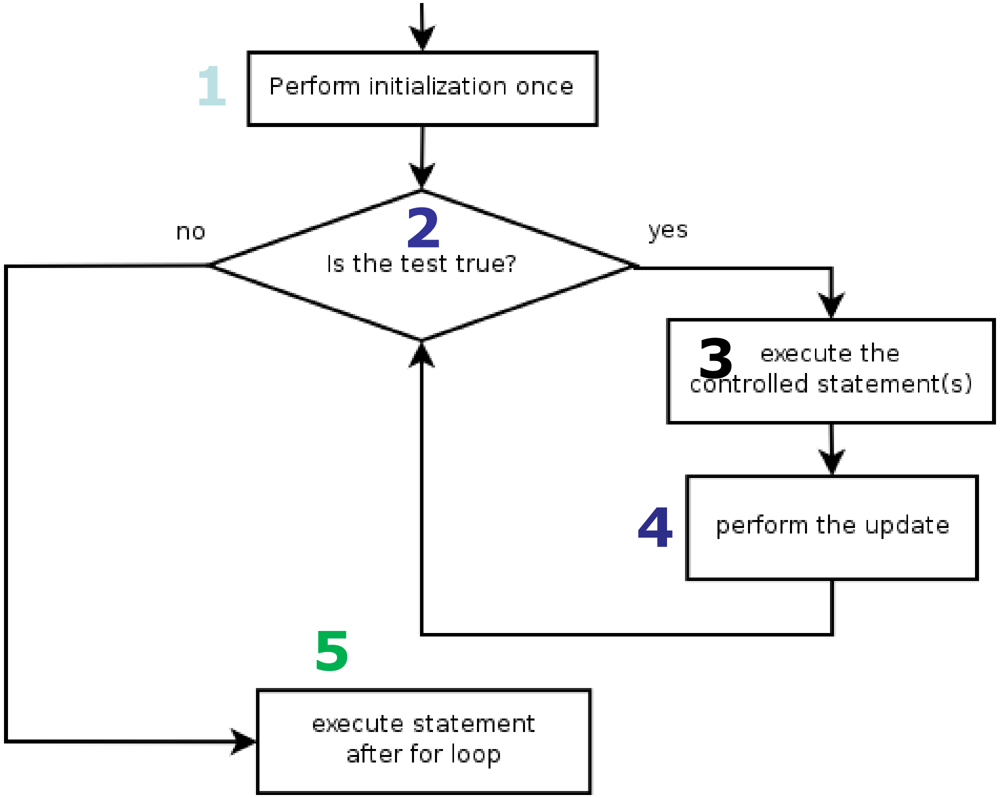
  - "For each integer **i** from 1 through 6, print ..."

Loop walkthrough

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

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

3 execute the controlled statement(s)

4 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:

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

Counter Expressions

```
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.
```

```
// Program : Assignment Average Calculator
// Author: LillAnne Jackson
// Date: May 25, 2009
// Input: None
// Output: Weighted Average of 5 Assignments
public class AssignmentAverager {
  // method main(): application entry point
  public static void main(String[] args) {
    // Assignment scores for J. Doe
    double assignment1John = 75.5;
    double assignment2John = 83;
    double assignment3John = 86;
    double assignment4John = 88.5;
    double assignment5John = 90;

    // weighted Assignment scores for J. Doe
    double weightedAssignment1John = 6 * assignment1John;
    double weightedAssignment2John = 6 * assignment2John;
    double weightedAssignment3John = 6 * assignment3John;
    double weightedAssignment4John = 6 * assignment4John;
    double weightedAssignment5John = 6 * assignment5John;

    // calculate the weighted average
    double sumJohn = weightedAssignment1John + weightedAssignment2John +
        weightedAssignment3John + weightedAssignment4John +
        weightedAssignment5John + weightedAssignment6John +
        weightedAssignment7John;
    double averageJohn = sumJohn / 30;

    System.out.println ("J. Doe weighted average = " + averageJohn);
  }
}
```

Example: For Interactive Input topic

A VERY Redundant Program

Interactive programs

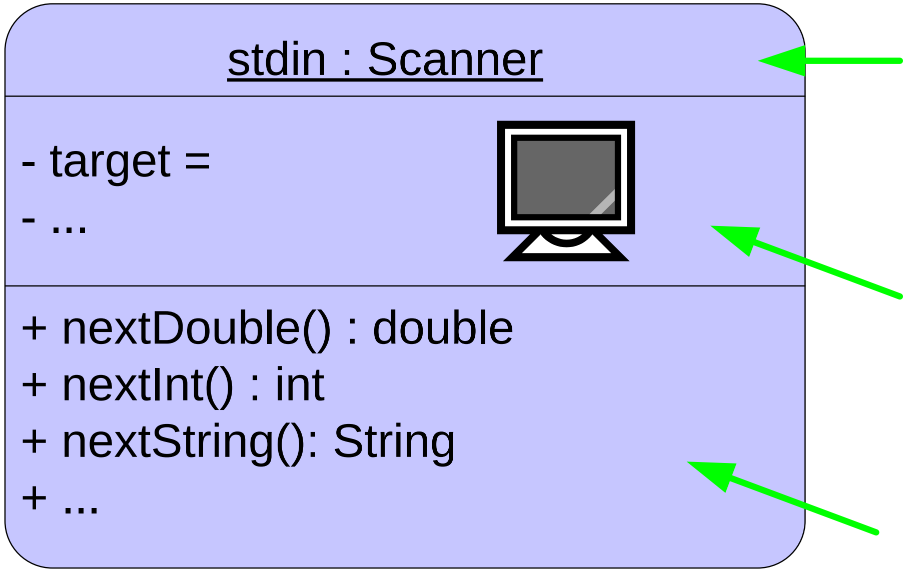
- Programs that interact with their users through statements performing input and output

- **`Assignment 1 Programs:`**
  - Not interactive – The values used in the programs are fixed.

Interactive programs: Console

- Variable System.in
  - Uses the standard input stream – the keyboard
- Class Scanner
  - Extracts numbers, characters, and strings

```
Scanner stdin = new Scanner(System.in);
```



stdin : Scanner

- target =

- ...

+ nextDouble() : double

+ nextInt() : int

+ nextString(): String

+ ...

Variable stdin gives access to an input stream that supports the extraction (reading) of input as values

The input destination attribute for this Scanner object is a stream associated with standard input — System.in

The behaviors of a Scanner object support a high-level view of extracting values

Interactive Assignment Averager

```
// Program : Assignment Average Calculator
// Author: LillAnne Jackson
// Date: May 25, 2009
// Input: 5 assignment scores (each a double)
// Output: Weighted Average of 5 Assignments

import java.util.*;

public class AssignmentAverager {
  // method main(): application entry point
  public static void main(String[] args) {
    Scanner stdin = new Scanner(System.in);

    // Assignment scores for J. Doe
    double assignment1John = stdin.nextDouble();
    double assignment2John = stdin.nextDouble();
    double assignment3John = stdin.nextDouble();
    double assignment4John = stdin.nextDouble();
    double assignment5John = stdin.nextDouble();

    // the rest is the same !!
```

Also: `nextInt()`

`next()`

Would a static method be useful here?

```
// Author: L. Jackson
// Purpose: To demonstrate repetitive code!
public class VeryRepetitive {
  // method main(): application entry point
  public static void main(String[] args) {
    Scanner stdin = new Scanner(System.in);

    // Exam scores for John
    double midterm1John = stdin.nextDouble();
    double midterm2John = stdin.nextDouble();
    double finalJohn = stdin.nextDouble();
    double averageJohn = (midterm1John + midterm2John + finalJohn) / 3;

   // Exam scores for Jane
    double midterm1Jane = stdin.nextDouble();
    double midterm2Jane = stdin.nextDouble();
    double finalJane = stdin.nextDouble();
    double averageJane = (midterm1Jane + midterm2Jane + finalJane) / 3;

   // Exam scores for Jim
    double midterm1Jim = stdin.nextDouble();
    double midterm2Jim = stdin.nextDouble();
    double finalJim = stdin.nextDouble();
    double averageJim = (midterm1Jim + midterm2Jim + finalJim) / 3;
  }
}
```

Adjust Program

- Interactive in-class exercise.

Determine the Ouputs

```
int limeTray = 17;
limeTray = limeTray + 1;
System.out.print("Lime: " + limeTray + " blocks.");
  .
  .
  .
limeTray = limeTray - 7;
System.out.print("Lime: " + limeTray + " blocks.");
```

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

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;
```

Using the Shortcuts

```
int limeTray = 17;
limeTray = limeTray + 1;
limeTray++;
System.out.print("Lime: " + limeTray + " blocks.");
   .
   .
   .
limeTray = limeTray - 7;
limeTray -= 7;
System.out.print("Lime: " + limeTray + " blocks.");
```

Data types

- **type**: A category or set of data values.
  - Constrains the operations that can be performed on data
  - Many languages ask the programmer to specify types

  - Examples: integer, real number, string

- Internally, computers store everything as 1s and 0s

```
104   → 01101000
"hi"  → 0110100001101001
```

Java's primitive types

- **primitive types**: 8 simple types for numbers, text, etc.
  - Java also has **object types**, which we'll talk about later

Name	Description		Examples
`int`	integers	(up to 2^{31} - 1)	`42, -3, 0, 926394`
`double`	real numbers	(up to 10^{308})	`3.1, -0.25, 9.4e3`
`char`	single text characters		`'a', 'X', '?', '\n'`
`boolean`	logical values		`true, false`

- Why does Java distinguish integers vs. real numbers?

Precedence questions

- What values result from the following expressions?

9 / 5	1
695 % 20	15
7 + 6 * 5	37
7 * 6 + 5	47
248 % 100 / 5	9
6 * 3 - 9 / 4	16
(5 - 7) * 4	-8
6 + (18 % (17 - 12))	9

Real numbers (type `double`)

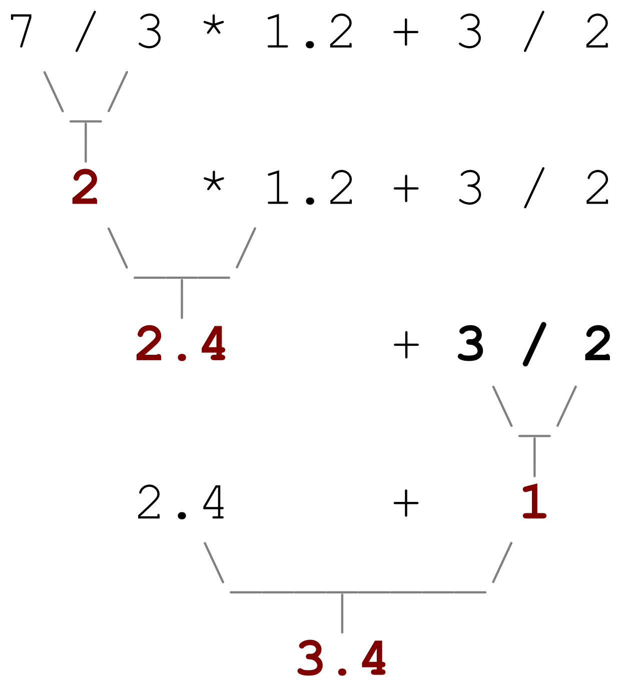
- Examples: `6.022` , `-42.0` , `2.143e17`
  - Placing `.0` or `.` after an integer makes it a `double`.

- The operators `+ - * / % ()` all still work with `double`.
  - `/` produces an exact answer: `15.0 / 2.0` is `7.5`
  - Precedence is the same: `()` before `* / %` before `+ -`

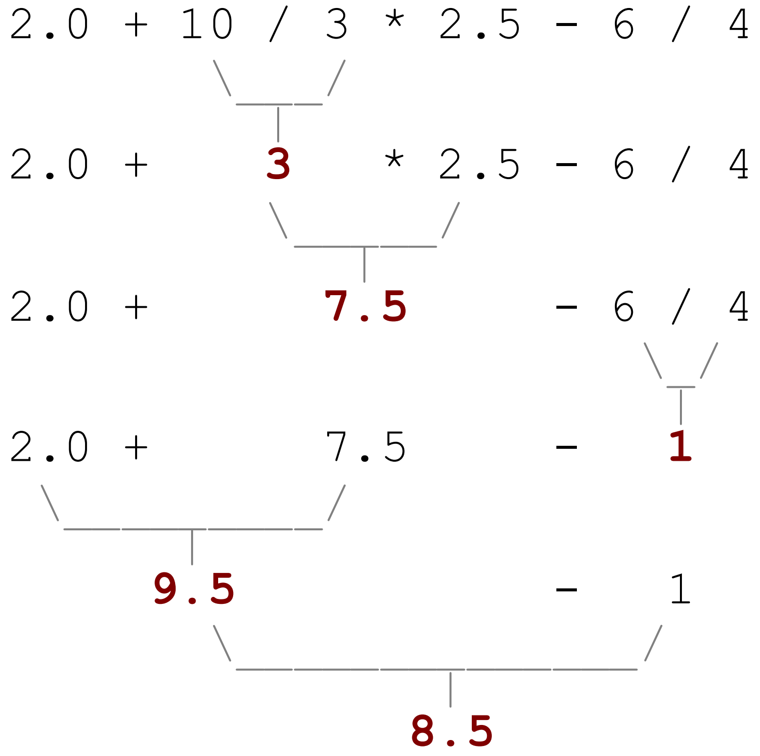
Mixing types

- When `int` and `double` are mixed, the result is a `double`.
  - `4.2 * 3` is `12.6`

- The conversion is per-operator, affecting only its operands.



```
7 / 3 * 1.2 + 3 / 2
2     * 1.2 + 3 / 2
2.4         + 3 / 2
2.4         +   1
3.4
```



```
2.0 + 10 / 3 * 2.5 - 6 / 4
2.0 +    3   * 2.5 - 6 / 4
2.0 +       7.5    - 6 / 4
2.0 +       7.5    -   1
      9.5          -   1
           8.5
```

String concatenation

- **string concatenation**: Using `+` between a string and another value to make a longer string.

`"hello" + 42`	`"hello42"`
`1 + "abc" + 2`	`"1abc2"`
`"abc" + 1 + 2`	`"abc12"`
`1 + 2 + "abc"`	`"3abc"`
`"abc" + 9 * 3`	`"abc27"`
`"1" + 1`	`"11"`
`4 - 1 + "abc"`	`"3abc"`

- Use `+` to print a string and an expression's value together.
  - `System.out.println("Grade: " + (95.1 + 71.9) / 2);`
  - Output: `Grade: 83.5`

Variables

Yes . . .It is a review!

And then a little bit more!

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

Assignment and algebra

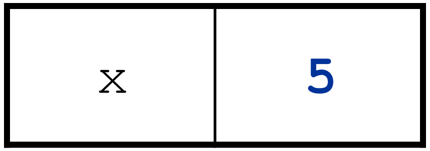
- Assignment uses = , but it is not an algebraic equation.

  = means, *"store the value at right in variable at left"*

  - The right side expression is evaluated first, and then its result is stored in the variable at left.

- What happens here?

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

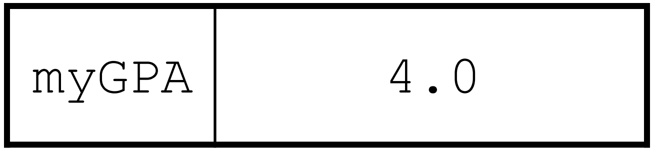


x 5

Assignment and types

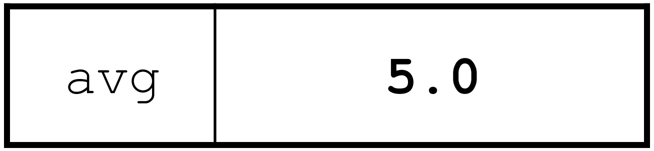
- A variable can only store a value of its own type.
  - `int x = 2.5;       // ERROR: incompatible types`

- 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 = 161 + 147;
System.out.println("There are " + students +
                   " students in the course.");
```

- Output:

```
Your grade was 83.2
There are 308 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
        int 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);
    }
}
```