Lesson 30: Variables and Data Types #1 (W09D2)

Balboa High School

Michael Ferraro

October 14, 2015

Do Now

Part I: Create a new Eclipse workspace

1. After mounting your locker directory, create a new workspace folder alongside `workspace0` called `workspace1`.
2. Start Eclipse, pointing it to `workspace1`.
3. Create a new Java project named `Lesson30`.

proceed to the next slide →

Do Now

Write a program utilizing `ArrayList`:

1. Add class `DivBy10` to the new project and include a `main()` method.
2. Inside `main()`:
    1. create an `ArrayList<Integer>` and add these elements: $\{30, 52, 739, 1024\}$
    2. using a `for-each()` loop, print out each element
    3. once you're able to print each element, modify the print statement to print the element divided by 10
3. Summarize the effect of taking an `int` and dividing it by 10.

Aim

Students will formally learn about variables, variable assignment, and Java's primitive data types.

PS04b Bonus

Let's go over the bonus — `printTriangle()` using recursion.

What's a variable?

A *variable*, in programming, is a "container" that holds a value. The *kind* of value is the variable's data type.

Data Types

Java has two types of data:

Data Types

Java has two types of data:

- **Primitives** — simple data, like numbers and individual characters (e.g., `a-z`, `A-Z`, `0-9`, `!#&%...`)

Data Types

Java has two types of data:

- **Primitives** — simple data, like numbers and individual characters (e.g., `a-z`, `A-Z`, `0-9`, `!#&%...`)
- **Objects** — more complex data: structures that involve data (primitives and/or other objects), methods, etc.

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**

[1] ≈ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] ≈ twice the precision of a `float`

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**
`boolean`	1	`true` or `false`	`TRUE`

[1] ≈ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] ≈ twice the precision of a `float`

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**
`boolean`	1	`true` or `false`	`TRUE`
`char`	2	any character that is part of the Unicode character set	`Q`

[1] ≈ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] ≈ twice the precision of a `float`

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**
`boolean`	1	`true` or `false`	`TRUE`
`char`	2	any character that is part of the Unicode character set	`Q`
`int`	4	-2^{31} to $2^{31}-1$ ($\approx \pm 2 billion$)	$-29,386$

[1] $\approx$ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] $\approx$ twice the precision of a `float`

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**
`boolean`	1	`true` or `false`	`TRUE`
`char`	2	any character that is part of the Unicode character set	`Q`
`int`	4	-2^{31} to $2^{31}-1$ ($\approx \pm 2 billion$)	$-29,386$
`long`	8	-2^{63} to $2^{63}-1$ ($\approx \pm 10^{19}$)	$4,294,967,296$

[1] $\approx$ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] $\approx$ twice the precision of a `float`

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**
`boolean`	1	`true` or `false`	`TRUE`
`char`	2	any character that is part of the Unicode character set	`Q`
`int`	4	-2^{31} to $2^{31}-1$ ($\approx \pm 2 billion$)	$-29,386$
`long`	8	-2^{63} to $2^{63}-1$ ($\approx \pm 10^{19}$)	$4,294,967,296$
`float`	4	$\approx -3.4 \times 10^{38}$ to 3.4×10^{38}	72.23564[1]

[1] $\approx$ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] $\approx$ twice the precision of a `float`

Java Primitive Data Types

Refer also to the table in Litvin §5.3.

Type	**Size** (bytes)	**Range**	**Example**
`boolean`	1	`true` or `false`	`TRUE`
`char`	2	any character that is part of the Unicode character set	`Q`
`int`	4	-2^{31} to $2^{31}-1$ ($\approx \pm 2 billion$)	$-29,386$
`long`	8	-2^{63} to $2^{63}-1$ ($\approx \pm 10^{19}$)	$4,294,967,296$
`float`	4	$\approx -3.4 \times 10^{38}$ to 3.4×10^{38}	72.23564[1]
`double`	8	$\approx -1.8 \times 10^{308}$ to 1.8×10^{308}	1,134,876,320,238.94[2]

[1] $\approx$ 7 significant figures (i.e., a 7-digit integer times a positive or negative power of ten); 72.23564 can be expressed as 7223564×10^{-5}

[2] $\approx$ twice the precision of a `float`

For this course...

- When you need to represent an integer[3] whose absolute value is guaranteed to stay under 2×10^9, use `int`; otherwise, use `long`.

[3] a $\pm$ whole #, including 0

[4] B is byte(s), b is bit(s)

For this course...

- When you need to represent an integer[3] whose absolute value is guaranteed to stay under 2×10^9, use `int`; otherwise, use `long`.

- When representing a mixed number or fraction, in decimal form, use `double` rather than `float`. This way, calculations can have more than 7 significant digits.

[3] a $\pm$ whole #, including 0

[4] `B` is byte(s), `b` is bit(s)

For this course...

- When you need to represent an integer[3] whose absolute value is guaranteed to stay under 2×10^9, use `int`; otherwise, use `long`.

- When representing a mixed number or fraction, in decimal form, use `double` rather than `float`. This way, calculations can have more than 7 significant digits.

  $\rightarrow$ downside: when we don't *really* need that kind of precision or a number that large (or negative), we waste 4B [4] of memory.

[3] a $\pm$ whole #, including 0

[4] B is byte(s), b is bit(s)

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- 1.3

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- 1.3 $\rightarrow$ `float`

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456$

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- 73

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`
- $772,836.23856$

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`
- $772,836.23856 \rightarrow$ `double`

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`
- $772,836.23856 \rightarrow$ `double`
- $-516,038.285$

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`
- $772,836.23856 \rightarrow$ `double`
- $-516,038.285 \rightarrow$ `double`

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`
- $772,836.23856 \rightarrow$ `double`
- $-516,038.285 \rightarrow$ `double`
- 12.38601

Choose the Right Type

For each of the following, chose the *most* appropriate primitive data type to hold the given value.

- $1.3 \rightarrow$ `float`
- $-93,456 \rightarrow$ `int`
- $73 \rightarrow$ `int`
- $772,836.23856 \rightarrow$ `double`
- $-516,038.285 \rightarrow$ `double`
- $12.38601 \rightarrow$ `float`

Variable Value Assignment

Why are the following statements different?

```
a = 7;
```

```
7 = a;
```

Variable Value Assignment

Why are the following statements different?

```
a = 7;
```

Variable `a` is being assigned the value 7.

```
7 = a;
```

Variable Value Assignment

Why are the following statements different?

`a = 7;`	`7 = a;`
Variable `a` is being assigned the value 7.	LHS [lefthand side] needs to be a variable! This is not valid Java syntax.

Variable Value Assignment

Assignment in Java (and *most* other languages) works this way:

- evaluate the RHS
- store the RHS' value in the memory location for the variable

Variable Value Assignment

Example

```
int x = -6;
int y = 5 + 9 - x;
```

Variable Value Assignment

Example

```
int x = -6;
int y = 14 - x;
```

Variable Value Assignment

Example

```
int x = -6;
int y = 14 - (-6);
```

Variable Value Assignment

Example

```
int x = -6;
int y = 20;
```

Variable Value Assignment

Example

```
int x = -6;
int y = 20;
```

$\rightarrow$ The value 20 is now stored at `y`'s memory location

Variable Value Assignment

Example

```
int x = -6;
int y = 20;
```

$\rightarrow$ The value 20 is now stored at `y`'s memory location

Coming soon: Java's Arithmetic Evaluation (Java's order of operations with respect to operators)

Default Values

Try running this:

```
public class DefaultValues {
    static int instanceVar;

    public static void testFcn() {
        System.out.println("value of instanceVar is"
            + instanceVar);
    }

    public static void main(String[] args) {
        int localVar;
        System.out.println("value of localVar is"
            + localVar);
        testFcn();
    }
}
```

Default Values

By default, an `int` is set to 0 if you don't specify a value

Default Values

By default, an `int` is set to 0 if you don't specify a value. . . except when it's a *local variable*, in which case the compiler complains and won't build a class file!

Items Not Discussed Here

Items not included in these slides, but that are in the reading:

- Default value for an object
- How `new` works
- What an object variable really is: A reference to a memory location

HW

PS #5, §§1-2.2, inclusive

(required reading here)