Types and Variables

Types and Variables

- Every value has a type
- Variables
  - Store values
  - Can be used in place of the objects they store
  - Essentially, a name for a memory location
- Variable declaration examples:

```
String greeting = "Hello, World!";
PrintStream printer = System.out;
int luckyNumber = 13;
```

Syntax 2.1: Variable Definition

typeName variableName = value;
or
typeName variableName;

Example:

```
String greeting = "Hello, Dave!";
```

Purpose:
To define a new variable of a particular type and optionally supply an initial value

Any idea how you would define a variable to hold your name?

String myName = "Vincent Cicirello";

Identifiers

- Identifier: name of a variable, method, or class
- Rules for identifiers in Java:
  - Can be made up of letters, digits, and the underscore (_) character
  - Cannot start with a digit
  - Cannot use other symbols such as ? or %
  - Spaces are not permitted inside identifiers
  - You cannot use reserved words
  - They are case sensitive

Continued…

Which of the following do you think are **valid** identifiers?

- Greeting yes
- i yes
- 4me no
- testing123 yes
- LotsOf$ no
- filename yes
- Filename yes
- EPSILON yes
- abc123+- no
- go+team no
- go_team yes
- Pi yes
- PI yes
- E yes

Naming Conventions

- By convention, variable names start with a lowercase letter
  - For longer variable names use camelcase
    - 1st letter lowercase
    - Periodically have an uppercase letter
- By convention, class names start with an uppercase letter
  - For longer class names use camelcase
  - 1st letter still uppercase

Which are **good** choices for name of a variable? A class?

- Greeting — class
- i — variable
- testing123 — variable
- filename — variable
- Filename — class
- EPSILON — neither
- go_team — neither
- Pi — class
- PI — neither
- E — Neither… be more descriptive

The Assignment Operator

- Assignment operator: =
- Not used as a statement about equality
- Used to change the value of a variable

```
1 int luckyNumber = 13;
2 luckyNumber = 12;
```

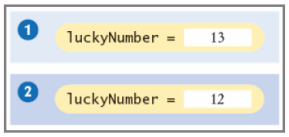

Figure 1:
Assigning a New Value to a Variable

Uninitialized Variables

- Error:

```
int luckyNumber;
System.out.println(luckyNumber);
   // ERROR - uninitialized variable
```

Syntax 2.2: Assignment

***variableName* = *value*;**

Example:
```
luckyNumber = 12;
```

Purpose:
To assign a new value to a previously defined variable.

Quick Question

- You've previously defined a variable greeting with:
  String greeting = "Hello, World!";
- How do you change the value of the greeting variable to `"Hello, Class!"`?
  greeting = "Hello, Class!";
- Note: **String greeting = "Hello, Class!";** is not correct since the greeting variable is already defined.

Number Types

Number Types

- Integers: `short, int, long`
  `13`
- Floating point numbers: `float, double`
  `1.3`
  `0.00013`

Continued…

Number Types

- When a floating-point number is multiplied or divided by 10, only the position of the decimal point changes; it "floats". This representation is related to the "scientific" notation 1.3×10^{-4}.

```
1.3E-4       // 1.3 × 10^-4 written in Java
```

- Numbers are not objects; numbers types are primitive types

Arithmetic Operations

- Operators: + - *

```
10 + n
n - 1
10 * n       // 10 × n
```

As in mathematics, the * operator binds more strongly than the + operator

```
x + y * 2    // means the sum of x and y * 2
(x + y) * 2  // multiplies the sum of x and y with 2
```

Quick Question

- Let's say that we have previously defined variables x and y as follows:
  int x = 12;
  int y = 67;
- Write a Java statement that will:
  - Compute the average of **x** and **y** (use only ***** and **+**)
  - And store the result in a newly defined variable named **average**.

double average = (x + y) * 0.5;