More Declaration Syntax

- We can assign a value to a variable in a variable declaration statement.

Syntax:

```plaintext
dataType variableName = value;
```

- The data type, valid identifier, and initial value are specified.

Initializing Variables

**EX:**

```plaintext
double a;
da = 5.0;
```

- These are equivalent to the single statement:

```plaintext
double a = 5.0;
```

Initializing Variables

**EX:**

```plaintext
double x, y, z;
x = 4.3;
z = 67.5;
```

- These are equivalent to the single statement:

```plaintext
double x = 4.3, y, z = 67.5;
```
Variable Input

- We can write much more useful programs if we can get data values from the user while the program is running.
- The `Input` class provides methods for getting input values from the user.
- To use methods from the `Input` class you must place a copy of the file `Input.class` in the same folder with your source code file.

Variable Input

- The `Input` class provides the `readDouble` method for getting an input real number.
- `readDouble` takes one argument:
  - A text string prompt.
- `readDouble` returns a real number.
- It can be invoked on the right-hand side of an assignment.
- Replaced with the user-input value.

readDouble

- Syntax:
  
  ```java
  variableName = Input.readDouble("prompt");
  ```

  - Assigned user-input value.
  - Method invocation.
  - Tells user what to input.

EX:

```java
double width;
width = Input.readDouble("Enter the width");
```

- Prompts user with “Enter the width”.
- After this statement executes, `width` will have the user-input value.
Example

```java
public class Sum {
    public static void main (String [] args) {
        double x, y;
        double sum;
        x = 32.5;
        y = 9.4;
        sum = x + y;
        Output.showValue("The first number is ", x);
        Output.showValue("The second number is ", y);
        Output.showValue("The sum is ", sum);
    } // method main
} // class Sum
```

Example with Input

```java
public class Sum2 {
    public static void main (String [] args) {
        double x, y;
        double sum;
        x = Input.readDouble("Enter the first number: ");
        y = Input.readDouble("Enter the second number: ");
        sum = x + y;
        Output.showValue("The first number is ", x);
        Output.showValue("The second number is ", y);
        Output.showValue("The sum is ", sum);
    } // method main
} // class Sum
```

Formula Programs

- the Sum example and Lab 3 are similar
- they all involve writing programs to calculate a mathematical formula
- EX: \( c = a + b \)
- we can write a general algorithm for solving such problems

Algorithm

- an algorithm is a step-by-step procedure for solving a problem
- like a recipe for baking a cake
- algorithms are written in pseudocode
Algorithm for Mathematical Formulas
1. declare variables for the variables on the right-hand side of the formula
2. declare a variable for the result (on the left-hand side of the formula)
3. get values for all right-hand side variables
4. calculate the result using the formula; store the result in your result variable
5. output all values

Using the Algorithm
✴ EX: apply our algorithm to write a program that calculates the mathematical formula:

\[ c = a + b \]

1. public class Sum
2. {
3.     public static void main (String [] args)
4.     {
5.         double x, y;
6.         double sum;
7.         x = Input.readDouble("Enter the first number:'");
8.         y = Input.readDouble("Enter the second number:'");
9.         sum = x + y;
10.        Output.showValue("The first number is ', x);
11.        Output.showValue("The second number is ', y);
12.        Output.showValue("The sum is ', sum);
13.    } // method main
14. } // class Sum

Module 7 Vocabulary

method return
algorithm
Questions?

Email Kevin at sahrk@sou.edu