Identifiers

- the words used in writing a program are called **identifiers**
- valid identifiers can consist only of letters, numbers, and the underscore character _
- identifiers can **not** begin with a number
  - valid identifiers EX: *Output, num3, my_ball*
  - invalid identifiers EX: *3dog, my#ball*

Keywords

- **keywords** are part of the Java language
- EX: *class, public, static*
- can only be used as intended by Java
- color-coded in JGrasp editor
- list available on Resources page of class website

Identifiers Chosen by Someone Else

- some identifiers you use are specified by other programmers
- EX: class name *Output* and method name *showMessage* were chosen by Kevin
Identifiers You Choose

✴ you choose the names of the classes, methods, and variables in your programs
✴ must be valid identifiers
✴ can not be keywords
✴ must be descriptive
  • EX: very confusing if I’d named the class that performs output the Elvis class

Programs and Data

✴ programs operate on data values
✴ a fixed data value used in a program is known as a literal
✴ EX: our programs have used specific fixed-value text strings like “Hello World”
✴ fixed value text strings like “Hello World” are known as string literals
  • delimited by “ ”’s

Limitations

✴ can only write limited programs with literals
✴ more interesting programs need to be able to:
  • get data via input
  • store/retrieve data in memory
  • change the value of data

Variables

✴ a variable is a named location in memory
✴ must have:
  • a unique name
  • a data type that specifies the kind of data that will be stored in that memory location
Our First Data Type

✿ variables with a data type of **double** are used to store/represent real numbers
  • real numbers EX: 3.14, -2.6, 0.0

Variable Declaration

✿ variables are created using a **variable declaration statement**
✿ **syntax:**

\[
\text{dataType variableName;}
\]

the data type \quad valid identifier

✿ **EX:**

\[
\text{double width;}
\]

• assigns the label **width** to a memory location
• tells Java that real numbers will be stored at that location

✿ can declare multiple variables of the same data type with a single declaration statement
✿ separate variable names with commas
✿ **EX:**

\[
\text{double x, y, z;}
\]

• declares three real number variables named \textit{x}, \textit{y}, and \textit{z}
Variable Declaration

✴ a variable must be declared before it can be used!

Assignment

✴ we store a value in a variable using an assignment statement

✴ syntax:

```
variableName = value;
```

left-hand side
right-hand side

✴ stores value in the memory location labeled variableName

Right-Hand Side

✴ the right-hand side value can be a literal number

✴ EX:

```c
double pi;
pi = 3.14;
```

✴ stores the value 3.14 in the memory location labeled pi

Multiple Assignments

✴ assignment statements replace the current variable value

✴ a variable will have the value of the most recent assignment

✴ EX:

```c
double var;
var = 88.88;
var = 5.6;
```

✴ the final value of var is 5.6
Right-Hand Side

- the right-hand side value can be an arithmetic expression using operators \(+\), \(-\), \(*\), and \(/\).

**EX:**

```java
double num;
num = 3.0 / 4.0;
```

- the arithmetic expression is evaluated and the result (.75) is stored in `num`

Arithmetic Gotcha

**EX:**

```java
double x;
x = 5.0 + 1.0 * 10.0;
```

- what is the result? depends on whether the + or the * is performed first
- in Java * and / are performed before + and -
- so `x` would be 15.0

Changing the Order

- you can override the default order using parenthesis

**EX:**

```java
double x;
x = (5.0 + 1.0) * 10.0;
```

- `x` would be 60.0

Right-Hand Side

- the right-hand side can contain variables
- right-hand side variables are replaced with their current values

**EX:**

```java
double a, b;
a = 5.0;
b = a * 2.0;
```

- `b` would be 10.0
Both Sides

✴ the same variable can occur on both the left- and right-hand sides
✴ the right-hand side variable is replaced with its current value
✴ EX:

double x;
x = 20.0;
x = x + 3;
• final value of x would be 23.0

Variable Output

✴ the Output class provides the showValue method for outputting the value of a variable
✴ showValue takes two arguments:
• a text string label for the value
• the variable name
✴ separate multiple arguments using commas

✴ syntax:

Output.showValue("label", variableName);

✴ EX:

double x;
x = 66.7;
Output.showValue("x is ", x);
✴ outputs: x is 66.7

showValue

✴ syntax:

Output.showValue("label", variableName);

✴ EX:

double x;
x = 20.0;
x = x + 3;
• final value of x would be 23.0

Example

public class Sum
{
    public static void main (String [] args)
    {
        double x, y;
double sum;
x = 32.5;
y = 9.4;
sun = 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
### Module 6 Vocabulary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifiers</td>
<td>variable declaration</td>
</tr>
<tr>
<td>keywords</td>
<td>assignment statement</td>
</tr>
<tr>
<td>literals</td>
<td>right-hand side</td>
</tr>
<tr>
<td>string literal</td>
<td>left-hand side</td>
</tr>
<tr>
<td>variable</td>
<td>arithmetic expression</td>
</tr>
<tr>
<td>data type</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td></td>
</tr>
</tbody>
</table>

### Questions?

Email Kevin at sahrk@sou.edu