

7. Expressions, Statements, and Blocks

Java

Fall 2009

Instructor: Dr. Masoud Yaghini

Outline

- Expressions
- Statements
- Blocks
- References

Expressions, Statements, and Blocks

- **Operators** may be used in building expressions
- **Expressions** are the core components of **statements**
- **Statements** may be grouped into **blocks**



Expressions



Expressions

- An **expression** is a construct made up of:
 - variables
 - operators
 - method invocations
- Expressions perform operations on data and move data around.
- An expression can have three kinds of result:
 - a value, such as the result of: $(4 * i)$
 - a variable, such as the result of: $i = 4$
 - nothing (in the case of an invocation of a method declared as void)

Expressions

- Examples of expressions, illustrated in bold below:

```
int cadence = 0 ;
```

```
anArray[0] = 100 ;
```

```
System.out.println("Element 1 at index 0: " + anArray[0]);
```

```
int result = 1 + 2 ; // result is now 3
```

```
if(value1 == value2)
```

```
    System.out.println("value1 == value2" );
```

Expressions

- The data type of the value **returned by an expression** depends on the elements used in the expression.
- The expression `cadence = 0` returns an `int` because the assignment operator returns a value of the same data type as its left-hand operand; in this case, `cadence` is an `int`.
- As you can see from the other expressions, an expression can return other types of values as well, such as `boolean` or `String`.

Compound expression

- Here's an example of a compound expression:
 - $1 * 2 * 3$
 - In this example, the order in which the expression is evaluated is unimportant.
- The following expression gives different results, depending on whether you perform the addition or the division operation first:
 - $x + y / 100$ // ambiguous
 - $(x + y) / 100$ // unambiguous, recommended
 - $x + (y / 100)$ // unambiguous, recommended

Compound expression

- When writing compound expressions, be explicit and indicate with **parentheses** which operators should be evaluated first.
 - This makes code easier to read and to maintain.



Statements



Statements

- A **statement** forms a complete unit of execution.
- Each statement in Java is terminated with a semicolon character (;).
- Kinds of statements:
 - **expression statements**
 - **declaration statements**
 - **control flow statements**

Expression statements

- The following types of expressions can be made into a **expression statement** by terminating the expression with a semicolon (;):
 - Assignment expressions
 - Any use of ++ or --
 - Method invocations
 - Object creation expressions
- Examples of expression statements:
 - `aValue = 8933.234; // assignment statement`
 - `aValue++; // increment statement`
 - `System.out.println("Hello World!"); // method invocation statement`
 - `Bicycle myBike = new Bicycle(); // object creation statement`

Declaration statements

- A **declaration statement** declares a variable.
- Examples of declaration statement:
 - `double aValue = 8933.234; // declaration statement`

Statements

- A programmer can put more than one statement on a line, and it will compile successfully, as in the following example:

```
int cadence = 0 ; anArray[0] = 100 ;
```

- Not recommended

Control flow statements

- The statements inside your source files are generally executed from **top to bottom**, in the order that they appear.
- **Control flow statements** break up the flow of execution and enabling your program to conditionally execute particular

Control flow statements

- **Control flow statements:**
 - **Decision-making statements**
 - if-then, if-then-else, switch
 - **Looping statements**
 - for, while, do-while
 - **Branching statements**
 - break, continue, return
- You'll learn about control flow statements later.



Blocks



Blocks

- Statements in Java are grouped using the opening brace ({) and closing brace (}).
- A group of statements organized between these characters is called a **block**.
- Example:
 - [BlockDemo.java](#)



References



References

- S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, **The Java Tutorial: A Short Course on the Basics**, 4th Edition, Prentice Hall, 2006. (Chapter 3)



The End