

COMPUTER PROGRAMMING IN JAVA

COLUMBIA UNIVERSITY HIGH SCHOOL SCIENCE HONORS PROGRAM

Basic Java, Part 3
Lecture
2007 Feb 23 Sat

Methods

In Java, the correct terminology for a function is a method. Methods allow us to organize code in a useful and powerful way.

A method is defined in the following way:

```
public <RETURN_TYPE> <METHOD_NAME>(<ARGS>)  
{  
    <CODE>  
}
```

where:

- <RETURN_TYPE> is one of
 - void
 - a data type (int, boolean, String, etc)
- <ARGS> can be nothing, or a comma-separated list

main is a special method.

Code blocks, Semicolons, and Scope

Now that we know about methods, we can talk about code blocks and scope.

A code block is either

- a single statement, terminated with a semicolon, OR
- zero or more statements enclosed with curly braces {}

Note that a statement can be empty:

```
// These are all statements  
int a = 1;  
System.out.println("Statement.");  
;
```

Code blocks are important because they help define scope. Scope can be thought of as where a variable is visible.

- Scopes can enclose other scopes.
- All variables are defined in some scope.
- Each scope can access the variables defined in all of its parent scopes.
- It is an error to define a new variable that already exists in that scope or any parent scope.

Example:

```
// These are all statements
int a = 1;
System.out.println("Statement.");
;
```

- A for loop defines a new scope.
- A method defines a new scope.
- Defining a variable outside of all methods (including main) means it's accessible to all methods in that class.

```
String s = "method1(), method2(), and main method can all access me.";
```

```
public void method1()
{
    // No relation to int i in method2!
    int i = 3;
}
```

```
public void method2()
{
    // No relation to int i in method1!
    int i = 2;
}
```

```
public static void main(String[] args)
{
    for (int i = 0; i < 10; i++)
    {
        // this is a new scope, which contains the variable int i
    }

    // i no longer exists here, so it's okay to define a new int i
    int i = 5;
}
```

Going back to control structures, remember that if/else, for, and while all expect code blocks. So that means the following are valid:

```

if (true)
{
    System.out.println("ok");
}
else
{
    System.out.println("also ok");
}

for (int i = 0; i < 10; i ++)
{
    System.out.println(i);
}

while (false)
{
    System.out.println("This will never execute, but it's valid.");
}

```

In particular:

```

if (false);
{
    System.out.println("This always prints");
}

```

Comments & Style

Comments are very important!

- //
 - Single line comment
- /* */
 - Multiline comment

```

// This is a single line comment. Note that the comment goes ABOVE
// what it's commenting!
// Also note that each line needs to start with two forward slashes.
// This creates a new String called "s" that contains the value
// "I love CS!"
String s = "I love CS!";

```

```

/* Alternatively, you can use a forward slash plus asterisk (no
spaces!) to start a multiline comment and an asterisk plus forward
slash to close it. This is useful for commenting out code you don't
want to execute but don't want to delete. */
/*
int a = 1;
boolean block = false;
String of = "code";
*/
String t = "This is the code I want."

```

Style is also very important!

- Java is a “free-form” language
- Variable name conventions
 - someVariable – correct
 - SomeVariable – incorrect
 - somevariable – incorrect
 - some_variable – incorrect
- Method name conventions
 - same as for variable names
- Later we'll learn about classes, and their naming conventions
- Tabs
 - braces and blocks

Common Compiler Errors

```
A.java:5: cannot find symbol
symbol   : variable a
location: class A
```

- This means you forgot to declare variable “a”, or it might mean that the variable was declared in another scope which is not accessible from the current one.

```
A.java:6: cannot find symbol
symbol   : method printn(int)
```

- This probably means you have a typo in the method name.

Random numbers

```
import java.util.Random;

Random random = new Random();
int x = random.nextInt(100);
```

Note the range! [0, 100)