Java Basics

CSE 373, Fall 2015

Megan Hopp

Definitions

- **Object**
  - Objects have states and behaviors. Objects are instances of classes.
- **Class**
  - A class can be defined as a template/ blueprint that describes the behaviours/states that object of its type support
- **Method**
  - A method is basically a behaviour. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variable**
  - Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.
  - Also referred to as a “field” or “member”

Style: Naming Conventions

- **Class Name**
  - Start with capital letter, then camel case (ex. ‘MyClassName’)
- **Method Name**
  - Start with lower-case letter, then camel case (ex. ‘myMethodName’)
- **Program File Name**
  - Should exactly match class name with ‘.java’ appended (ex. ‘MyClassName.java’)

Style: Comments

Should sufficiently comment all classes, methods, and important variables (typically fields in a class or any important variables that are not obvious)

- What is the function of each?
- Information on parameters/returns
- Try to avoid implementation details

Comment Style Convention:

- Large block comments should use: /* .... */
- Single line comments should use: // ....

Style: JavaDoc

Not required, but can use JavaDoc if you like.

Javadoc convention for writing specifications:

- Method signature: Text description of method
- @param: description of what value gets passed in
- @return: description of what value gets returned
- @throws: description of what exceptions may occur and why

Style: JavaDoc Example

```
/**
 * This method does something, and this describes it.
 *
 * @param firstParameter this describes the boolean parameter
 * @throws MyException when XYZ condition is not met
 * @return some int value that should be returned
 */
 public int method1(boolean firstParameter) {
   ...
```

Java Variables: Scope

They have "scope" which is essentially the realm in which they exist.

- **Method-specific variables** (declared within the method or passed in as a parameter)
- **Instance variables** (non-static: declared within class)
- **Class variables** (static variables that apply to all instances of a given class)

Higher-level variables have larger scope, for example a class variable/field is visible to the class, any methods within that class, and even smaller scopes

Inheritance

Java objects can use **inheritance**.

Avoids redundant code/logic by allowing subclasses to use their superclass's code or behavior (public or protected fields, methods, etc.)

Inheritance: Example

```
public class Dog {

  public void bark() {

    System.out.println(“woof”);

  }

}

public class Husky extends Dog {

  // empty class, no implemented methods

}
```

```
Dog dog = new Husky();

dog.bark(); // prints out “woof”
```

Interfaces

Java classes that specify the required methods that MUST be present/implemented in an implementation of that interface

```
interface Human {
   public void eat();
   public void sleep();
   public void breathe();
}

class Kevin implements Human {
 public void eat() {
     System.out.println("nom");
 }
 public void sleep() {
     System.out.println("zzz");
 }
 public void breathe() {
      System.out.println("skadoosh");
 }
}
```

Loops

For Loops

- for finite/known number of iterations

```
for (int i = 0; i < 5; i++) {
    // do something
}
```

While Loops

- for unknown number of iterations
- we don't know on which iteration the loop will terminate, only that it will terminate when the condition is not met

```
while (input.hasNextLine()) {
    // do something
}
```

Algorithm Analysis

Big-Oh Notation:

Worst-case bound on algorithm's performance

Big-Omega Notation:

Best-Case bound on algorithm's performance

*Note: we will cover this in detail in the course, but it's a good idea to refresh your memory from 143

Recursion

A method or function calling itself.

Base case:

- Logic to return a value without any recursive calls. Indicated by a condition to ensure that we don’t enter an infinite series of recursive calls
- Possible to have more than 1 base case

Recursion case:

- Logic to kickstart a recursive call, updating value(s) usually toward converging with the base case

Helpful Links

Style Guide (found here)

Past CSE 143 Course Websites (found here)

Practice-It! (found here)