


CSE1710

Week 01, Lecture 02

Click to edit this slide
Second level
Third level
Fourth level
Fifth level

Fall 2013 ◆ Thursday, Sept 12, 2013



Checklist

What we are reinforcing with the exercises this class...

- being able to identify key features (classes and packages, headers and bodies, statements)
- being able to identify the lexical elements of Java
- being able to identify the key features of Java programs
- being able to describe the edit-compile-run process
- being able to recognize code that adheres to correct coding style
- recognizing Java's *primitive types*
- describing what is meant by a *literal*

Checklist for next lecture

What you should be doing to prepare for what comes next...

- read (**again**) the document “JBA-CompanionNotes-Sec1_2_3” (on webpage)
- read sections 1.3
- review Ch 1 KC’s 14-17
- do RQ’s 18-21
- do Ex’s 1.17-1.22
- ensure you are ready for first lab (know your section, have your EECS account ready)

3



Anatomy of a Java class

Identify the preamble, class header, class body:

```
import java.util.Date;

public class Lect02Ex01 {

    public static void main(String[] args) {
        int theValue = 89;
        // this app doesn't do very much
        // it declares a variable, assigns a value, and then stops!
    }

}
```

4



Anatomy of a Java class

Identify the preamble, class header, class body:

```
import java.io.PrintStream;
public class Area
{public static void main(String[] args)
{PrintStream output;output = System.
    out;int width;width = 8;int height = 3;int
area = width * height;output.println(area);}}
```

5



Anatomy of a Java class

Identify the preamble, class header, class body:

Identify what is inside the class body (method header, method body)

```
package samplePkg;
import java.io.PrintStream;

public class Lect02Ex02 {

    public static void main(String[] args) {
        int theValue = 89;
        // this app doesn't do very much
        // it declares a variable, assigns a value, and then stops!
    }

}
```

6



Anatomy of a Java class

Identify what is inside the class body (method header, method body)

```
package samplePkg;
import java.io.PrintStream;

public class Lect02Ex03 {

    public static void main(String[] args) {
        int theValue = 89;
        // this app doesn't do very much
        // it declares a variable, assigns a value, and then stops!
    }

    public void myMethod() {
        int someOtherValue = 9;
    }
}
7 }
```



Anatomy of a Java class

Identify the preamble, class header, class body:

Notice that the variable `theValue` occurs twice. Is this a problem? Why or why not?

```
package samplePkg;

public class Lect02Ex04 {

    public static void main(String[] args) {
        int theValue = 89;
    }

    public void myMethod() {
        int theValue = 89;
    }
}
8 }
```



Anatomy of a Java class

Identify the preamble, class header, class body:

Notice that the variable `theValue` occurs twice. Is this a problem? Why or why not?

```
package samplePkg;
```

```
public class Lect02Ex04 {
```

```
    public static void main(String[] args) {
        int theValue = 89;
    }
```

```
    public void myMethod() {
        int theValue = 89;
    }
```

```
}
```

9

NOT a problem
Different method bodies
means different scopes

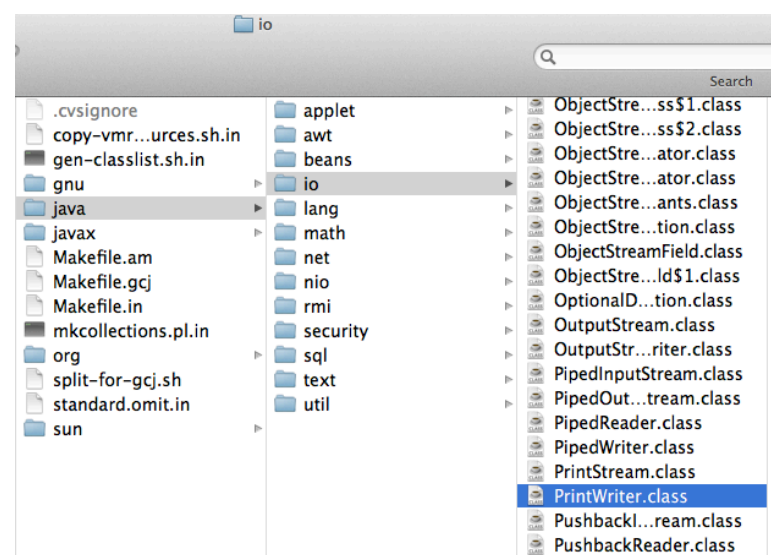
**A variable is available for
usage starting from the
statement that follows the
declaration and ending at the
end of the block that encloses
the declaration**
p.16



Anatomy of Java

RQ1. What is a package and how is it different from a subpackage?

What does a package/subpackage actually look like on the file system?



10



Anatomy of Java: Statements,

Classify all *statements* in terms of (1) **declaration**, (2) **assignment**, combo declaration/assignment, (3) **usage of other classes**, (4) **flow control**, (5) **other**

JD 1.3 (p. 10)

```
import java.lang.System;

public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

11



Anatomy of Java: Variable Declaration

RQ 6. What does **strongly typed** mean?

Java is strongly typed. Other languages, such as Perl, are weakly typed. They only require that an *identifier be declared as a variable*, they don't require that the type of the variable be declared in advance.

Java enforces strong typing discipline so that the compiler can do a lot of checking before producing bytecode.

This results in code that has certain "safety" features (see sec 2.3.1)

12



Lexical Elements

Ex 1.7 (riff) Create a legend of the 5 types of lexical elements (keywords, identifiers, literals, operators, separators) and identify each in the example below

```
double balance = 34e3;
boolean isValid = false;
long count = System.out.readLong();
booleanIsValid = 'F';
String address = new String("Toronto");
number = Integer.parseInt(input);
public static void main(String[] commmand)
System.out.printf("%.2f", amount)
```

13



Lexical Elements

RQ.10. How do you know whether a word is a Java reserved word?

RQ 13. Can you determine the type of a literal just by looing at one? ...Do this for each literal below

```
34e3
false
'F'
"Toronto"
34.6
1L
2f
3e6f
16
```

14



Syntax (and Semantic) Rules

RQ.11. What are the syntactical rules for naming identifiers?

***What's an example of a **semantic** rule for naming identifiers?

17



Coding Style

RQ.12. What are the style guidelines for naming identifiers?

What sorts of identifiers are there?

18



Primitive Types

RQ.17. Name Java's primitive types. Why are they called primitive?

Are there literal values for every primitive type in Java?