

Ch8: Packages

14 February 2007

CMPT167

Dr. Sean Ho

Trinity Western University

Quiz 2: 10 minutes

- Name the Java **interface** we use that requires the **actionPerformed()** method [2]
- Write Java **declarations** reflecting the following: [4]
 - ◆ A **Quadrilateral** is a **kind** of **Polygon**.
 - ◆ A **Square** is a **kind** of **Quadrilateral**.
 - ◆ Each **Polygon** has a **cornerList**.
 - ◆ **myNameTag** is a **Square**.
- Contrast **scope** and **duration** of variables. [4]
- Contrast **init()** and **start()** in **JApplet**. [4]
- Write a Java code block that creates an **array** of 20 **random** doubles and calculates the **sum**. [6]

Quiz 2: answers #1

- Name the Java **interface** we use that requires the **actionPerformed()** method
 - **ActionListener**
 - ◆ **class** MyApplet **extends** JApplet
implements ActionListener { ...

Quiz 2: answers #2

- Write Java **declarations** reflecting the following:
 - A **Quadrilateral** is a **kind** of **Polygon**.
 - ◆ `class Quadrilateral extends Polygon {};`
 - A **Square** is a **kind** of **Quadrilateral**.
 - ◆ `class Square extends Quadrilateral {};`
 - Each **Polygon** has a **cornerList**.
 - ◆ `class Polygon { Point cornerList[]; }`
 - **myNameTag** is a **Square**.
 - ◆ `Square myNameTag = new Square();`
 - Only declaration is needed

Quiz 2: answers #3-4

- Contrast **scope** and **duration** of variables.
 - **Scope**: **lexical** extent (lines of code) where variable can be referenced
 - **Duration**: **time** intervals when the variable exists in memory as the program runs
- Contrast **init()** and **start()** in **JApplet**.
 - **init()**: when applet is **loaded** into memory
 - **start()**: when applet starts **running**, e.g., upon page refresh

Quiz 2: answers #5

- Write a Java code block that creates an **array** of 20 **random** doubles and calculates the **sum**.

```
double randList[] = new double[20];
double sum = 0.0;
for (int i=0; i<20; i++) {
    randList[i] = Math.random();
    sum += randList[i];
}
```

Packages



- **Group** related classes and interfaces
- Avoids name **collision**
- Package **declaration** at top of each file:
 - ◆ `package mypackage;`
- Popular convention: use reverse **domain** name
 - ◆ `com.sun.java.awt...`
 - ◆ `ca.twu.cmpt167.lab3.seanho.FractalTree`
- Pass “-d” option to `javac` to create **directories** when compiling:
 - ◆ `javac -d . FractalTree.java`

Using packages

- Every **file** should specify what **package** it belongs to in the **first line** of code in the file
- Each file should still have only **one public** class
 - **Non-public** classes have **package** scope
 - ◆ Useful for **internal** helper classes
- **Import** from a package as normal
 - **Classpath** specifies where to search for packages
 - ◆ **Default** classpath includes “.”
 - ◆ **Override** with **java -classpath ./other/path**



Access modifiers

- **Access modifiers** limit who can read/write variables and methods:
 - **public**: **anyone** who imports this class
 - **private**: only methods within this **class**
 - **protected**: **subclasses** of this class

	Class	Package	Subclass	World
Private	Y	N	N	N
(none)	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

jar

- Wrap up a **collection** of related classes/packages into one file with **jar** (Java **AR**chiver)
 - Like **ZIP**, Unix **tar**
- Syntax:
 - **Create** a jar file: `jar cvf mypackage.jar <files>`
 - **Unpack** a jar: `jar xvf mypackage.jar`
 - ◆ **C**: create
 - ◆ **X**: extract
 - ◆ **V**: verbose
 - ◆ **F**: specify jar file



TODO

- Midterm next week Wed 21Feb
 - Through today's lecture
- Lab3 due next week Wed 21Feb:
 - Recursion (fractal tree)