Packages and Sharing Code

June 26, 2017

Reading Quiz

Which is **True** about Packages?

A. Packages create namespaces in Java

B. Packages exist to reduce the disk space needed by third party code

C. A package is a class created by third party code

D. Packages can only be created by the Java Virtual Machine

Which is **True** about `import`?

A. `import` allows local code to use third party code

B. `import` aliases full package paths to shorter names

C. `import` converts binary code to text code

D. `import` changes private variables to public variables

Package Paths

```
// Where should I look for this code on disk?
import edu.cs342.somepackage.SomeClass;
```

A. class definition in edu/cs342/somepackage/SomeClass.java

B. defined variable in edu/cs342/somepackage.java

C. interface definition in edu.cs342.somepackage.java

D. enum definition in edu-cs342-somepackage/SomeClass.java

What is significant about the java.lang package?

A.It became deprecated in Java 7, and became the javax package

B. It contains the definition of the syntax of the Java language

C. Its contents are automatically imported into every program

D. Only trusted / verified code can access it

Packages

```
// What does this line of code do
package edu.cs342.otherpackage;
```

A. Instructs the compiler to create a edu.cs342.otherpackage.jar file

B. Imports the contents of the edu.cs342.otherpackage package

C. Declares a variable of type package

D. Declares the contents of the file as part of the edu.cs342.otherpackage package

Done!

House Keeping

- Homework 3 due Wednesday before class
- Mid-term next week
- "How ya'll doing" questions

"Just checking in and seeing how everyone is doing" questions

How do you find the pace of class?

A. The pace of class is too slow, this is boring and / or we should be covering more material

B. The pace of class feels about right, I'm regularly picking up new material and learning it

C. The pace of class is too fast, I'm having trouble keeping up with new material

Class Coding?

A. Its helpful or engaging when I do coding examples on the laptop / board. More coding on the laptop would be useful.

B. Its helpful to have the coding examples on the laptop, but at the current level.

C. Some on-the-laptop coding examples would not helpful, but less would be better.

D. The on-the-laptop coding examples are not helpful, please no more

Homeworks

A. They're too easy

B. They're just right

C. They're too hard

Coding in Groups

A. The in-class, as-groups coding parts are helpful, working through code in class is useful

B. The in-class, as-groups coding parts are not useful; more time covering the material in lecture would be better

C. The in-class, as-groups coding parts are not useful; more new material would be better

Done!

Quick Word on Coding

(or, why Java requires so much typing…)*

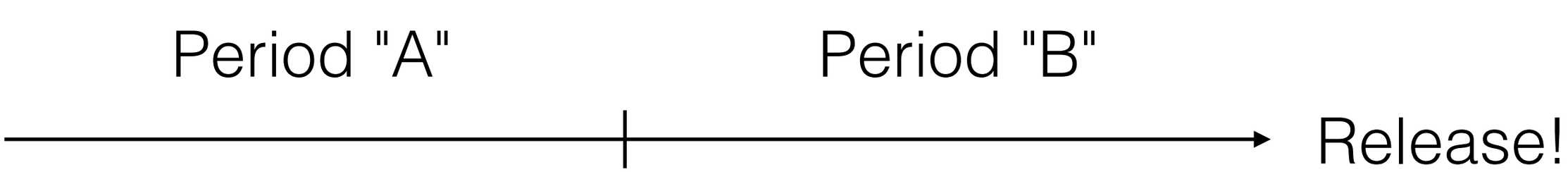
*even if it doesn't really need to…

Writing "Real World" Code

- As students, you type more than you read code
- As a developer / professional, you read code more than you write it

- If your code is crap, you debug code **way** more than you write it

Time Spent Developing



Douglas Crockford: Programming Style & Your Brain
https://www.youtube.com/watch?v=_EANG8ZZbRs

Packages, Namespaces and `import`

The Problem

- Sharing code is valuable
  - Helps in structuring programs
    - Division of labor
    - Solve hard problems once
- Naming makes this difficult…

The Problem: Names

Our Code

```
/**
 * Class to represent neat shoes.
 */
public class Sneaker {

    private String shoeName;

    private Integer shoeSize;
}
```

Their Code

```
/**
 * Class to represent a sneaky thief.
 */
public class Sneaker {

    private String criminalName;

    private Integer yearsInPrison;
}
```

{Main, Sneakers}.java

->

Unique Names: Naive Approach

- Prefix names with something "unique"
    - **Author's Initials**
      "PESSneaker"
    - **Vendor / Company's name**
      "UICSneaker"
- Tedious to type
- Still could collide

Java's Three Part Solution

1. Require REALLY long prefixes (e.g. edu.uic.cs342)

2. Provide syntax so that library authors don't need to type so much (`package`)

3. Provide syntax so that library authors don't need to type so much (`import`)

Really Long Names

- Start with "reversed Internet domain name" (e.x. edu.uic)

- Add name that describes the functionality being implemented
  (ex edu.uic + crimestuff = edu.uic.cs.crimestuff)

- Use this as our code prefix to ensure we're unique
  (ex "public class edu.uic.crimestuff.Sneakers")

{Main, Sneakers}.java

->

`package`:
Library Author Convenience

- Typing **public class edu.uic.crimestuff.Sneakers** is tedious
- **package** tells Java that the file uses the long prefix
- **package** must go at the top of the file
- Place the code in the corresponding directory (ex edu/uic/crimestuff)

{Main, Sneakers}.java

->

`import`:
Library User Convenience

- Typing **edu.uic.crimestuff.Sneakers sneakers = new edu.uic.crimestuff.Sneakers()** is horrible
- **import** says "use shortcut for this long name"
- **import** statements must go after **package** statement, if one exists
- **import** is only an alias

{Main, Sneakers}.java

->

Package and Import Remainders

- **import** accepts wildcards
  ex "import java.util.*"

- **import** can't always be used
  multiple packages can use the same names

- **import** is only an alias
  Full names can still be used

File System Example –>

JARs

Problem

- Using third party code can be tricky
  - Versioning
  - Overlapping Namespaces
    ex: edu.uic vs edu.uillinois vs. edu.uchicago
  - Entire applications?

Java's Solution

- JAR: Java Archive Format
  - One or more **packages** of .class files
  - Text file telling Java how to use this code
  - Optional specification of entry point
  - Zipped up (ex. ".zip")

Checkstyle.jar –>

Group Programming

- Recreate `ls`
- Implement it in the `edu.uic.cs342.fs` package
- Package as jar
- Send jar to me :)
- Suggestion to start: java.io.File https://docs.oracle.com/javase/8/docs/api/java/io/File.html