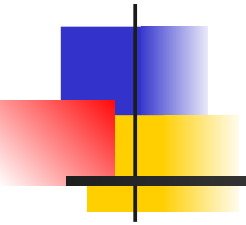

Review of Java Concepts

James Brucker

What Package are these in?

_____ Scanner

_____ String

_____ System

_____ Date

_____ InputStream

_____ FileInputStream

_____ Double

_____ ArrayList

_____ int

Core Packages

java.lang	Java language core classes. `Object, String, System, Integer, Double, Math, Thread` java compiler always imports this package, so you don't need to.
java.io **(java.nio)**	Classes for input and output `InputStream, BufferedReader, File, OutputStream`
java.util	Date/time classes, collections, & utilities `Calendar, Date, List, ArrayList, Set, Arrays, Formatter, Scanner`

What does "import" do?

```
package ske.oop;
import java.util.ArrayList;
```

a) Copy source code for ArrayList into your app.

b) Copy ArrayList.class to "`bin`" directory for app.

c) Add ArrayList to names known by compiler (the namespace).

d) Load ArrayList into memory at run time.

"import" and "fat app" syndrome?

```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.InputStream;
import java.io.FileInputStream;
```

Do many "imports" make your compiled application bigger?

Import Everything

You can import everything from a package. Use * (wildcard)

```
package lazyprogrammer;
import java.util.*;
class Person {
    private static Scanner console = ...;
    private Date birthday;
    private List<Person> friends;
    ...
```

Import Ambiguity

There are 2 Date classes: `java.util.Date` and `java.sql.Date`.
Suppose we import both classes using wildcards:

```
import java.util.*;
import java.sql.*;
/** a class using a Date */
public class Ambiguous {
    private Date today;
```

Which **Date** class will Java use?

Resolving Import Ambiguity

If a class matches more than one wildcard *, Java requires you to resolve the ambiguity using an import with no wildcard (import one class).

```
import java.util.*;
import java.sql.*;
import java.util.Date;

public class Ambiguous {
    private Date today;
```

Programming without Import

You can write code without ever using "import". How?

```
import java.util.Scanner;
public class Demo {
   private static Scanner console =
            new Scanner( System.in );
```

Without using import:

```
public class Demo {
   private static ____________________________
         console =
       new ____________________( System.in );
```

Java Naming Convention

class name begins with Uppercase: **`Coffee, String`**

method name uses camelCase: **`getMoreCoffee( )`**

variable name also uses camelCase: **`myCoffee`**

constants use UPPER_CASE and _: **`MAX_VALUE`**

package names are all lowercase (almost always):

`java.util  java.io`

`org.apache.commons.logging`

primitive type names are all lowercase:

`boolean, byte, char, int, long, float, double`

What are these?

Date

System

System.nanoTime( )

System.out

System.out.println( )

double

Double

"Hello nerd".length( )

java.lang.Double.MAX_VALUE

Comparable

java.util

java.util.ArrayList

java.util.*List*

Is it a ...

package

class

primitive type

attribute ("field")

method

(static or instance)

constant

(static final attribute)

interface *(more advanced)*

???