Classes and Objects: Enum and Packages

CPSC 233: Introduction to Computer Science for Computer Science Majors II
Winter 2022

Jonathan Hudson, Ph.D.
Instructor
Department of Computer Science
University of Calgary

Wednesday, 10 November 2021

UNIVERSITY OF CALGARY

Static?

Static Methods

All Java methods must be in a class

A **static method** is not invoked on an object

Why write a method that does not operate on an object?

Common reason: encapsulate some computation that involves only numbers (or other immutable objects like Strings). Numbers aren't objects, you can't invoke methods on them.

E.g., int x; x.sqrt() can never be legal in Java (Math.sqrt(x) exists)

Static Methods

```
public static String canadaCountryCode(){
    return "1";
}
```

Call with class name instead of object:

String cc = **PhoneNo2.canadaCountryCode();**

main is static – one reason is that there aren't any objects yet

Static Fields

A static field belongs to the class, not to any object of the class. Also called class field.

```
public class BankAccount {
    private double balance;
    private int accountNumber;
    private static int lastAssignedNumber = 1000;
}
```

A common usage, counting last id number assigned so next one is one bigger

Static Fields

If **lastAssignedNumber** was not static, each instance of **BankAccount** would have its own value of **lastAssignedNumber**

```
public class BankAccount {
    private double balance;
    private int accountNumber;
    private static int lastAssignedNumber = 1000;

    public BankAccount() {
        lastAssignedNumber += 1;
        accountNumber = lastAssignedNumber;
    }
```

Minimize the use of static fields.

Static FINAL fields are constants.

Static Fields

Three ways to initialize:

1. Do nothing. Field is with 0 (for numbers), false (for boolean values), or null (for objects)
2. Use an explicit initializer, such as

```
public class BankAccount {
    private double balance;
    private int accountNumber;
    private static int lastAssignedNumber = 1000;    // Execute when class is loaded
```

3. Use a static initialization block

```
private static int getLastAssignedNumber;
static{
    lastAssignedNumber = 1000;
}
```

Static Fields

- Static constants, which may be either private or public

```
public static final double OVERDRAFT_FEE = 5;     // BankAccount.OVERDRAFT.FEE
```

Enum

enum

```
public enum State{
    ON, OFF;
}
```

```
State state = State.ON;
if (state == State.OFF) {

} else {

}
```

Sometimes we have a determinate (finite, known) listing of things, otherwise known for our purposes as an enumeration in Java

Often these are **states**

Stored in file **State.java** like a class but keyword **enum**

List as many options as needed as names (there is a hidden increasing **int ordinal** value created for each one you make)

Basically a managed set of associated **int** constants

enum

```
public class ObjectWithState {

    private int state;

    public ObjectWithState() {
        this.state = 0;
    }

    public int getState() {
        return state;
    }

    public void setState(int state) {
        this.state = state;
    }
}
```

Consider you want to code state tracking in an object

If you make this an int you will have to remember in your head or add quite a few comments to track what each integer value means and which are valid

enum

```
public class ObjectWithStateConstants {

    private int state;
    public static final int OFF = 0;
    public static final int LO = 1;
    public static final int MED = 2;
    public static final int HI = 3;

    public ObjectWithStateConstants() {
        this.state = 0;
    }

    public int getState() {
        return state;
    }

    public void setState(int state) {
        this.state = state;
    }
}
```

Well one solution is to make a set of **constants**

This is helpful as we can now use these constants to change state

But we still will have to add in error check in something outside of the valid constants are used

enum

```
public class ObjectWithStateEnum {

    private State state;
    public enum State{
        OFF, LO, MED, HI;
    }

    public ObjectWithStateEnum() {
        this.state = State.OFF;
    }

    public State getState() {
        return state;
    }

    public void setState(State state) {
        this.state = state;
    }
}
```

Or we could use an enum type

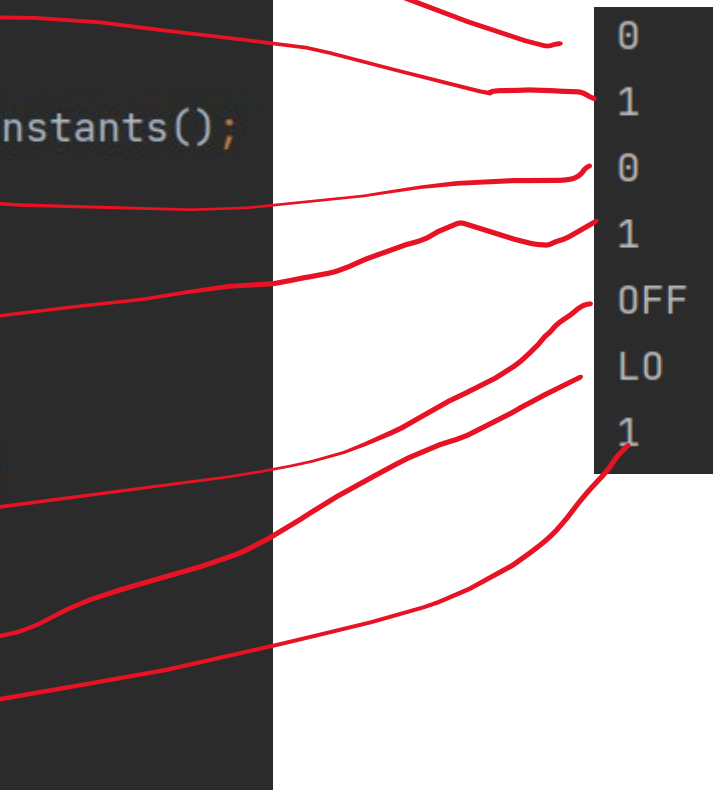
The type is now enum and we can only use valid enums to change the state

enum

```
ObjectWithState s0 = new ObjectWithState();
System.out.println(s0.getState());
s0.setState(1);
System.out.println(s0.getState());

ObjectWithStateConstants s1 = new ObjectWithStateConstants();
System.out.println(s1.getState());
s1.setState(ObjectWithStateConstants.LO);
System.out.println(s1.getState());

ObjectWithStateEnum s2 = new ObjectWithStateEnum();
System.out.println(s2.getState());
s2.setState(ObjectWithStateEnum.State.LO);
System.out.println(s2.getState());
System.out.println(s2.getState().ordinal());
```



Packages

Organizing Related Classes into Packages

- Package: Set of related classes
- To put classes in a package, you must place a line
  package packageName;
  as the first instruction in the source file containing the classes
- Package name consists of one or more identifiers separated by periods
- For example, to put the **TicTacToe** class into a package named **ca.ucalgary.jwhudson**, the **TicTacToe.java** file must start as follows:

```
package ca.ucalgary.jwhudson.cpsc231w22.a1;

public class TicTacToe {
```

- Default package has no name, no package statement

Importing Packages

Can always use class without importing

```
java.util.Scanner in = new java.util.Scanner(System.in);
```

Tedious to use fully qualified name, Import lets you use shorter class name

```
import java.util.Scanner;
Scanner in = new Scanner(System.in)
```

Can import all classes in a package

```
import java.util.*;
```

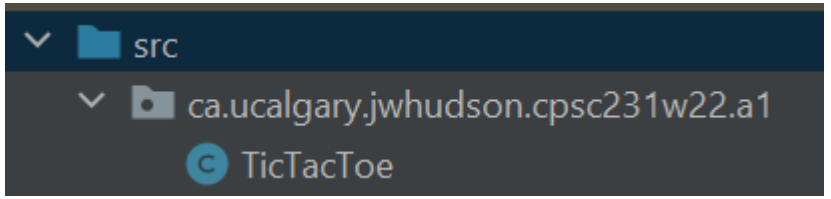
Package Names and Locating Classes

Recommendation: start with reversed domain name **ca.ucalgary.jwhudson**

```
package ca.ucalgary.jwhudson.cpsc231w22.a1;

public class TicTacToe {

}
```



Package name will match folder structure in src folder

ca/ucalgary/jwhudson/cpsc233w22/a1/TicTacToe.java

Usage

import ca.ucalgary.jwhudson.cpsc233w22.a1.TicTacToe;

You can see ex java 7 src file structure here https://github.com/openjdk-mirror/jdk7u-jdk/tree/master/src/share/classes

Onward to … Encapsulation

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY