Object-Oriented Programming: Constructors & Interfaces

Ewan Klein

Inf1 :: 2008/09

Current Topics

HFJ, Chs 8, 9

Static Fields: .

Static Methods: .

Constructors: .

What are Static Methods?

Calling a method from Maths

```
int randomNum = (int)( Math.random() * 5)
```

Wrong

```
Math m = new Math();
m.random();
```

Output

```
Exception in thread "main" java.lang.Error: Unresolved com-
pilation problem:
        The constructor Math() is not visible
```

- Methods from `Math` not dependent on the state of instances of the `Math` class.

Static vs. Non-Static

- All 'regular' methods are non-static.
- A static method can be run without instantiating the class.
- Also called class methods.

Calling a Static Method

```
Math.max(7, 22);
```

Calling a Non-Static Method

```
Cell c = new Cell();
c.move(6);
```


Classes with Static Methods

- Classes with static methods often not meant to be instantiated.
- How to stop instantiation?
  1. Make the class abstract.
  2. Make the constructor private.

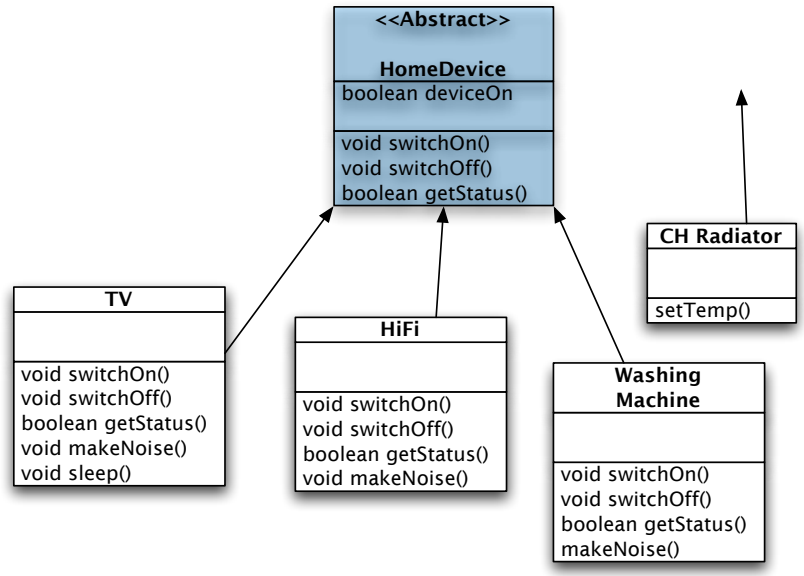
Private Constructor

```
public class Sim {
    private Sim() {
    }
}
```


Static Fields

Device API

Some Home Devices



In Breach of Contract

You broke the contract!

So, Abstract Classes

- Use an abstract class when you have several similar classes that:
  - have a lot in common — the implemented parts of the abstract class
  - have some differences — the abstract methods.

Hanging the Radiator

Where does `CH Radiator` belong in the class hierarchy?

Central Heating Radiator

```
public class CHRadiator {extends Object {
    private int temperature;
    public void setTemp(int temp) {
        temperature = temp;
    }
```

- `Object` is the superclass of every class in Java!
- If a class doesn't explicitly extend some superclass, then it implicitly extends `Object`.

`Object` as a Polymorphic Type

Javadoc for `ArrayList`'s `contains()` method

`public boolean contains(Object elem)`

Returns `true` if this list contains the specified element.

Parameters: `elem` - element whose presence in this List is to be tested.

Returns: `true` if the specified element is present; `false` otherwise.

- Since every class in Java is a subclass of `Object`, the `contains()` method can take any object as an argument.

Some Methods of `Object`

`Object` defines methods that are available to every class. E.g.,

- `equals(Object o)` — test whether two objects are equal.
- `hashCode()` — numerical ID; equal objects must have equal hash codes.
- `toString()` — returns a textual representation of an object; automatically invoked by methods like `System.out.println()`; default implementation is not very useful.

Overriding `toString()`, 1

Number Six

```
public class NumberSix {
    public String toString() {
        return "I am not a number - I am a free man!";
    }
}
```

Number Two

```
public class NumberTwo {
    public static void main(String[] args) {
        NumberTwo n2 = new NumberTwo();
        NumberSix n6 = new NumberSix();
        System.out.println("You are Number Two: " + n2);
        System.out.println("You are Number Six: " + n6);
    }
}
```


Overriding `toString()`, 2

Output

```
You are Number Two: NumberTwo@10d448
You are Number Six: I am not a number - I am a free man!
```

Revisiting Constructors

Instantiating `Bicycle`

```
Bicycle bike = new Bicycle();
```

Default Constructor

```
public Bicycle() {
}
```

- Java compiler adds a default constructor.
- Name (e.g. `Bicycle`) is same as class name.
- No return type.

Make Your Own Constructor, 1

Declared Constructor

```
public class Bicycle {
    private int gears;
    private String brand;
    public Bicycle() {
        gears = 15;
        brand = "Cannondale";
        System.out.printf("Bike: gears=%d, brand=%s",
                          gears,
                          brand);
    }
}
```


Make Your Own Constructor, 2

BikeLauncher

```
public class BikeLauncher {
    public static void main(String[] args) {
        Bicycle myBike = new Bicycle();
    }
}
```

Output

```
Bike: gears=15, brand=Cannondale
```


Using Setters to Initialize Instance Variables, 2

Rectangle1

```
public class Rectangle1 {
    private int height;
    private int width;
    int area = height * width; // Doesn't work!
    public void setHeight(int newHeight) {
        height = newHeight;
    }
    public void setWidth(int newWidth) {
        width = newWidth;
    }
}
```


Using Setters to Initialize Instance Variables, 2

RectangleLauncher

```
public class RectangleLauncher {
    public static void main(String[] args) {
        Rectangle1 r1 = new Rectangle1();
        r1.setHeight(3);
        r1.setWidth(6);
        System.out.println("Area of r1: " + r1.area);
    }
}
```

Output

```
Area of r1: 0
```

Using a Constructor to Initialize Instance Variables, 1

Rectangle2

```
public class Rectangle2 {
    private int height;
    private int width;
    int area;
    // Add a two-parameter constructor
    public Rectangle2(int startHeight, int startWidth) {
        height = startHeight;
        width = startWidth;
        area = height * width;
    }
}
```


Using a Constructor to Initialize Instance Variables, 2

Rectangle1

```
public class RectangleLauncher {
    public static void main(String[] args) {
        Rectangle1 r1 = new Rectangle1();
        r1.setHeight(3);
        r1.setWidth(6);
        System.out.println("Area of r1: " + r1.area);

        Rectangle2 r2 = new Rectangle2(3, 6);
        System.out.println("Area of r2: " +r2.area);
    }
}
```

Output

```
Area of r1: 0
Area of r2: 18
```


Reading, etc

- I can't cover all the material in Chapters 8 & 9 of *Head First Java* in the lectures, so do read them carefully.
- No lecture this Friday!
- Next Tuesday, guest lecture by Stuart Anderson on software engineering and software testing.
- Exam environment: normal DICE environment except firewalled down. All applications should work exactly as normal. You won't have your normal home directory however —- so any local configuration and setup will not be available in the exam.
- Reminder — Revision Tutorial / Lab Sessions:
  - Today, 3-4pm, AT 4.12
  - Thursday 26th Feb, 2-3pm, AT 4.12