Object-Oriented Programming: Static Methods & Variables

Ewan Klein

Inf1 :: 2008/09

What are Static Methods?

1. What are static methods?

What are Static Methods?

1. What are static methods?
2. You've already seen them.

What are Static Methods?

1. What are static methods?
2. You've already seen them.

Calling a method from Maths

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

What are Static Methods?

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

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
```

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, 1

- 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() {
    }
}
```

Classes with Static Methods, 2

Static Method in Sim

```
public static Sim getInstance() {
    if (instance == null) {
        instance = new Sim();
    }
    return instance;
}
```

Classes with Static Methods, 2

Static Method in Sim

```
public static Sim getInstance() {
    if (instance == null) {
        instance = new Sim();
    }
    return instance;
}
```

Calling the Method

```
public class AnimalCell extends BaseCell {
    Sim world = Sim.getInstance();
    ...
}
```

Classes with Static Methods, 3

- Classes with static methods may be instantiated.
- And static and non-static methods can be mixed.

Mixed methods

```
public class Cell1 {
    public static void main(String[] args) {
        Cell1 c = new Cell1();
        c.hello();
    }
    public String hello() {
        return "I'm a cell!";
    }
}
```

Classes with Static Methods, 4

- `main()` is a static method.
- You can do the following (though it's a bit pointless!):

Calling a `main()` method

```
public class CellLauncher {
    public static void main(String[] args) {
        String[] arglist = ;  // Make an empty array
        Cell1.main(arglist);     // prints "I'm a cell!"
    }
}
```

Static Methods and Instance Variables, 1

- A static method cannot use an instance variable.
- Think about it.

Mixed methods

```
public class Cell2
    int loc = 0;
    public static void main(String[] args)
        System.out.println("I'm at: " + loc ); // Wrong!

    public int getLoc()
        return loc ;                          // OK
```

Static Methods and Instance Variables, 2

Output

```
Exception in thread "main" java.lang.Error: Unresolved com-
pilation problem:
            Cannot make a static reference to the non-
static field loc
```

Static Methods calling Non-Static Methods, 1

Doesn't Work

```
public class Cell3 {
    int loc = 0;
    public static void main(String[] args) {
        System.out.println("I'm at: " + getLoc());
    }
    public int getLoc() {
        return loc;
    }
}
```

Output

```
Exception in thread "main" java.lang.Error: Unresolved com-
pilation problem:
            Cannot make a static reference to the non-
static method getLoc() from the type Cell3
```

Static Methods calling Non-Static Methods, 2

Does Work

```
public class Cell3 {
    int loc = 0;
    public static void main(String[] args) {
        Cell3 c = new Cell3();
        System.out.println("I'm at: " + c.getLoc());
    }
    public int getLoc() {
        return loc;
    }
}
```

Static Variables, 1

Static variable:

- one value per class,
- not one value per instance.
- Also called: static field, class variable

Static Variables, 2

- Variable `printsMade` is initialized when class is loaded, not when an instance is created.
- But incremented each time a new instance is created.
- Resulting value is shared by all instances.

How many prints?

```
public class Lithograph {
    private static int printsMade = 0;
    private int copyNo;

    public Lithograph() {
        printsMade++;
        copyNo = printsMade;
    }
}
```

Static Variables, 2

How many prints?

```
public class Lithograph {
    static int printsMade = 0;
    private int copyNo;
    public static void main(String[] args) {
        ArrayList<Lithograph> prints = new ArrayList<Lithograph>();
        for (int i = 0; i < 6; i++) {
            prints.add(new Lithograph());
         }
        for (Lithograph l : prints) {
            l.howMany();
        }
    }
    public Lithograph() {
        printsMade++;
        copyNo = printsMade;
    }
    public void howMany() {
        System.out.printf("Copy %d of %d%n", copyNo, printsMade);
    }
```

Static Variables, 3

Output

```
Copy 1 of 6
Copy 2 of 6
Copy 3 of 6
Copy 4 of 6
Copy 5 of 6
Copy 6 of 6
```

Static Final Variables, 1

Final variable:

- Once initialized, value cannot be changed.
- Value stays the same as long the class stays loaded.
- This is how constants are created.
- Convention: write the variable all in capitals

Some constants

```
public static final double PI = 3.141592653589793;
public static final int DAYS_PER_WEEK = 7;
public static final String BULGARIAN_NATNL_DAY = ”3 March”;
```

Finally, ...

- Instance and local variables can be declared `final` — value cannot be subsequently changed.
- Methods can be `final` — cannot be overriden.
- Classes can be `final` — cannot be extended.

Reading, etc

- Mock exam — sign up soon! Most places left are for today.
- Chapter 10. Read for yourself the material on
  - `Math` methods
  - Autoboxing (less complicated than it sounds)
  - Formatting (we've already done some of this)
  - `Date` and `Calendar`, if you're interested.
- Next up: Chapter 11, Exceptions
- `examreadonly` directory: I haven't put reference sheet there yet, or the dump of web pages. Will do this before real exam.