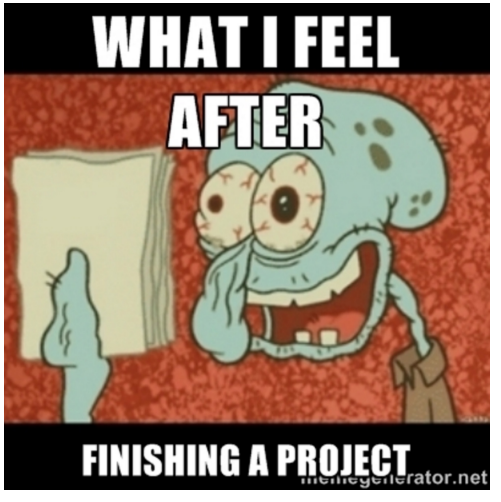
CS 11114

Introduction to Software Design

Spring 2017 - Michael Irwin



How's Program #2?

Class Structure

- **Fields** - store values/data for an object
- **Constructors** - used to initialize/setup the class
- **Methods** - operations/behaviors the class exposes

```
public class LazyCleaningJeroo extends CleaningJeroo {
    // Fields
    // Constructors
    // Methods
}
```

Fields

- Also known as **instance variables**
- Define the *state* of the object
- Values can change frequently, or rarely

```
public class LazyCleaningJeroo extends CleaningJeroo {
    private int numFlowersCollected;
    private int numFlowersAllowed;

    // Other methods go here
}
```

Setting Fields using Constructors

- Constructors are perfect place to set up or initialize an object
- Recipe for a constructor
  - Have same name as the class
  - No return type
- Can setup fields not passed as parameters

```
public class LazyCleaningJeroo extends CleaningJeroo {
    private int numFlowersCollected;
    private int numFlowersAllowed;

    public LazyCleaningJeroo(int flowerLimit) {
        this.numFlowersAllowed = flowerLimit;
        this.numFlowersCollected = 0;
    }
}
```

Retrieving Field State using Methods

- Methods can be used to get the current state of an object
- Commonly called a **getter**
- Can also use the inspector in Greenfoot

```
public class LazyCleaningJeroo extends CleaningJeroo {
    private int numFlowersCollected;
    private int numFlowersAllowed;

    public LazyCleaningJeroo(int flowerLimit) {
        this.numFlowersAllowed = flowerLimit;
        this.numFlowersCollected = 0;
    }

    public int getNumFlowersAllowed() {
        return numFlowersAllowed;
    }

    public int getNumFlowersCollected() {
        return numFlowersCollected;
    }
}
```

Why they're called instance variables...

```
public class LazyCleaningJeroo
        extends CleaningJeroo {
    private int numFlowersCollected;
    private int numFlowersAllowed;

    public LazyCleaningJeroo(int flowerLimit) {
        this.numFlowersAllowed = flowerLimit;
        this.numFlowersCollected = 0;
    }

    public int getNumFlowersAllowed() {
        return numFlowersAllowed;
    }

    public int getNumFlowersCollected() {
        return numFlowersCollected;
    }
}
```

```
LazyCleaningJeroo jeroo1 = new LazyCleaningJeroo(10);
LazyCleaningJeroo jeroo2 = new LazyCleaningJeroo(25);

jeroo1.getNumFlowersAllowed();
    // returns 10

jeroo2.getNumFlowersAllowed();
    // returns 25
```

Updating Field State using Methods

- Methods can be used to update the current state of an object
- If simply setting, called a **setter**

```
public class LazyCleaningJeroo extends CleaningJeroo {
    private int numFlowersAllowed;

    // ...

    public void setNumFlowersAllowed(int numAllowed) {
        this.numFlowersAllowed = numAllowed;
    }

}
```

Problem Scenario

- Let's actually code `LazyCleaningJeroo`, who will only pick up flowers 1-*N* where *N* is the limit we specify
- Then, write tests for it!