Building New Robots

D Barnette
https://canvas.vt.edu/

Working with objects

- **Class** - a definition for an object (`LightBot`)
- **Instantiation** - process to create a new object
- **Declaration** - process to create a new name/variable
- **Object** - in this case, andy is an instance/object of the class

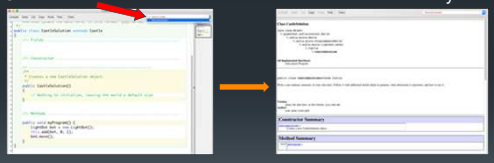
```
LightBot andy = new LightBot();

//alternatively
LightBot andy;
andy = new LightBot();
```

CS@VT

Why Documentation?

- Gives you and others a place to answer questions
  - What's this thing do?
  - How can I use it?
- Comments are turned into HTML documents automatically



CS@VT

Creating new methods

```
public <ReturnType> <methodName>() {
  // Method body here
}
```

```
public void moveTurnLightOn() {
  // Method body here
}
```

- The `public` access modifier indicates anyone can call the method
  - We'll talk about other access modifiers later
- The <ReturnType> indicates what will be returned
  - If nothing will be returned, use `void`
- The method's name should reflect what it will do
- Methods are the modularization of a problem solution
- Methods avoid code duplication
- Document your new methods

CS@VT

Making our bot patrol

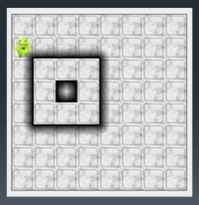
We could simply tell the bot to move four times, turn right, move four times, turn right...

```
bot.move();
bot.move();
bot.move();
bot.move();
bot.turnRight();

bot.move();
bot.move();
bot.move();
bot.move();
bot.turnRight();

bot.move();
bot.move();
bot.move();
bot.move();
bot.turnRight();

bot.move();
bot.move();
bot.move();
bot.move();
bot.turnRight();
```



CS@VT

Suppose we want to extend the methods our robot understands

- Without new methods, code becomes very long
  - Harder **to read and understand**
  - Harder to find our logic errors
  - Lots of repetitive code
- There *has* to be a better way …
- Suppose we **built** a robot that knew how to do more?

Introducing class inheritance

Could we make our bot smarter, so he could patrol on his own?
Two problems with doing that…

Not all LightBots will patrol (cohesion)
We don't have the source for `LightBot` anyways :(

```
public class <NewClass> extends <ParentClass> {
  // New stuff goes here
}
```

- The `NewClass` will inherit all methods that were on `ParentClass`

CS@VT

Defining a new class

```
public class <new-class-name>
    extends <old-class-name>
{
    <new methods>
}
```

visisbilty modifier: public, private, protected

public: everyone can see it

(Direct Relationship)

inheritance : Java

Grand Parent → Parent → child

Move() move() Move()

Turn() Turn()

flop

extends :

→ Move

Grand child

→ child inherits all the capabilities of the parents, not the other around

Extends LightBot?

```
public class PatrolBot extends LightBot
```

- The **extends** keyword indicates the PatrolBot **inherits** all the capabilities of LightBot
  - In other words, PatrolBot understands all the methods of LightBot already
- LightBot is the **base class** of PatrolBot
  - Also called **superclass** or **parent class**
- PatrolBot is a **derived class** of LightBot
  - Also called **subclass** or **child class**

The PatrolBot

```
public class PatrolBot extends LightBot {

  /**
   * Perform a single patrol around the castle
   */
  public void patrolCastle() {
    walkOneWall();
    walkOneWall();
    walkOneWall();
    walkOneWall();
  }
```

```
  public void walkOneWall() {
    move();
    move();
    move();
    move();
    turnRight();
  }

}
```

- Created a helper method named `walkOneWall`. It's also a public method, so can be invoked by anything

CS@VT