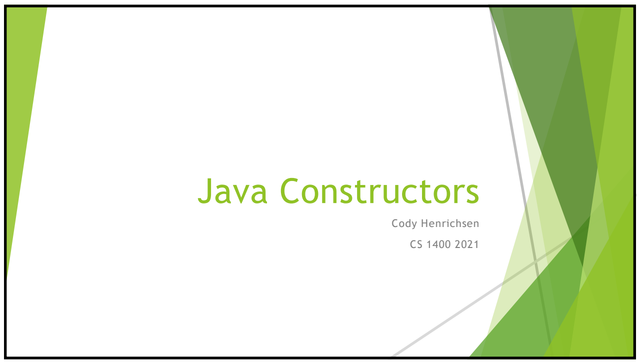
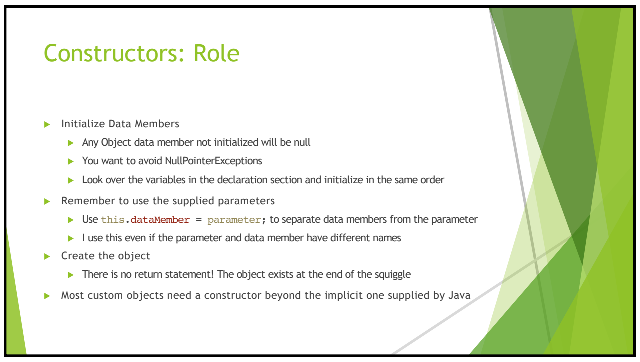


Java Constructors

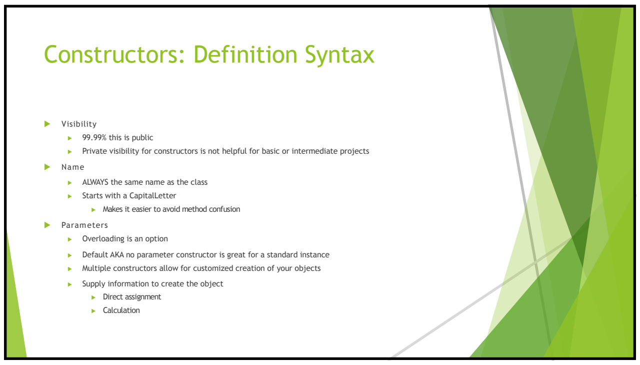
Cody Henrichsen

CS 1400 2021



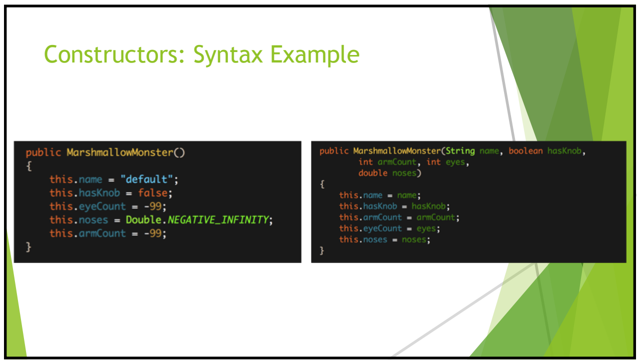
Constructors: Role

- Initialize Data Members
  - Any Object data member not initialized will be null
  - You want to avoid NullPointerExceptions
  - Look over the variables in the declaration section and initialize in the same order
- Remember to use the supplied parameters
  - Use `this.dataMember = parameter;` to separate data members from the parameter
  - I use this even if the parameter and data member have different names
- Create the object
  - There is no return statement! The object exists at the end of the squiggle
- Most custom objects need a constructor beyond the implicit one supplied by Java



Constructors: Definition Syntax

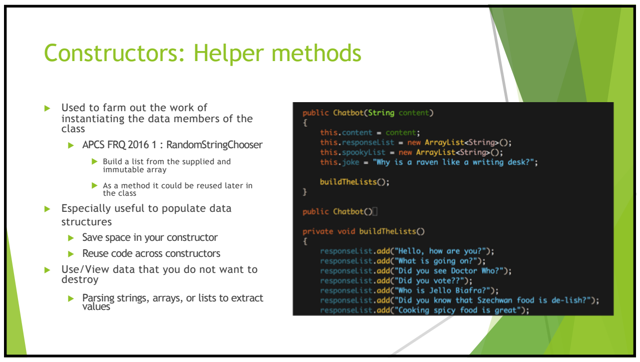
- Visibility
  - 99.99% this is public
  - Private visibility for constructors is not helpful for basic or intermediate projects
- Name
  - ALWAYS the same name as the class
  - Starts with a CapitalLetter
    - Makes it easier to avoid method confusion
- Parameters
  - Overloading is an option
  - Default AKA no parameter constructor is great for a standard instance
  - Multiple constructors allow for customized creation of your objects
  - Supply information to create the object
    - Direct assignment
    - Calculation



Constructors: Syntax Example

```
public MarshmallowMonster()
{
    this.name = "default";
    this.hasKnob = false;
    this.eyeCount = -99;
    this.noses = Double.NEGATIVE_INFINITY;
    this.armCount = -99;
}
```

```
public MarshmallowMonster(String name, boolean hasKnob,
        int armCount, int eyes,
        double noses)
{
    this.name = name;
    this.hasKnob = hasKnob;
    this.armCount = armCount;
    this.eyeCount = eyes;
    this.noses = noses;
}
```



Constructors: Helper methods

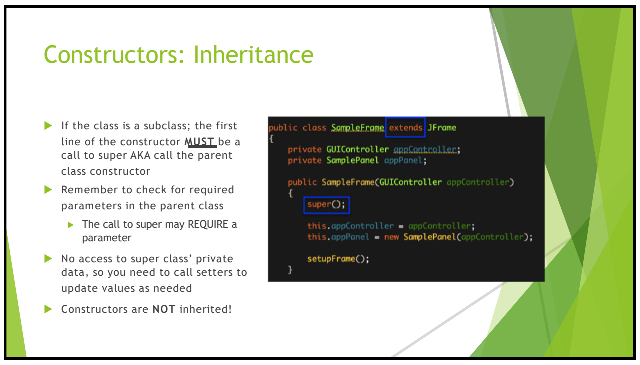
- Used to farm out the work of instantiating the data members of the class
  - APCS FRQ 2016 1 : RandomStringChooser
    - Build a list from the supplied and immutable array
    - As a method it could be reused later in the class
- Especially useful to populate data structures
  - Save space in your constructor
  - Reuse code across constructors
- Use/View data that you do not want to destroy
  - Parsing strings, arrays, or lists to extract values

```
public Chatbot(String content)
{
    this.content = content;
    this.responseList = new ArrayList<String>();
    this.spookyList = new ArrayList<String>();
    this.joke = "Why is a raven like a writing desk?";

    buildTheLists();
}

public Chatbot()

private void buildTheLists()
{
    responseList.add("Hello, how are you?");
    responseList.add("What is going on?");
    responseList.add("Did you see Doctor Who?");
    responseList.add("Did you vote??");
    responseList.add("Who is Jello Biafra?");
    responseList.add("Did you know that Szechwan food is de-lish?");
    responseList.add("Cooking spicy food is great");
```



Constructors: Inheritance

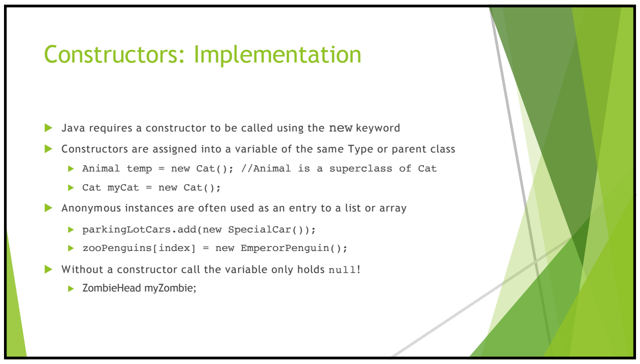
- If the class is a subclass; the first line of the constructor MUST be a call to super AKA call the parent class constructor
- Remember to check for required parameters in the parent class
  - The call to super may REQUIRE a parameter
- No access to super class' private data, so you need to call setters to update values as needed
- Constructors are **NOT** inherited!

```
public class SampleFrame extends JFrame
{
    private GUIController appController;
    private SamplePanel appPanel;

    public SampleFrame(GUIController appController)
    {
        super();

        this.appController = appController;
        this.appPanel = new SamplePanel(appController);

        setupFrame();
    }
```



Constructors: Implementation

- Java requires a constructor to be called using the `new` keyword
- Constructors are assigned into a variable of the same Type or parent class
  - `Animal temp = new Cat(); //Animal is a superclass of Cat`
  - `Cat myCat = new Cat();`
- Anonymous instances are often used as an entry to a list or array
  - `parkingLotCars.add(new SpecialCar());`
  - `zooPenguins[index] = new EmperorPenguin();`
- Without a constructor call the variable only holds `null`!
  - ZombieHead myZombie;