Classes Part 1

Forest Agostinelli
University of South Carolina

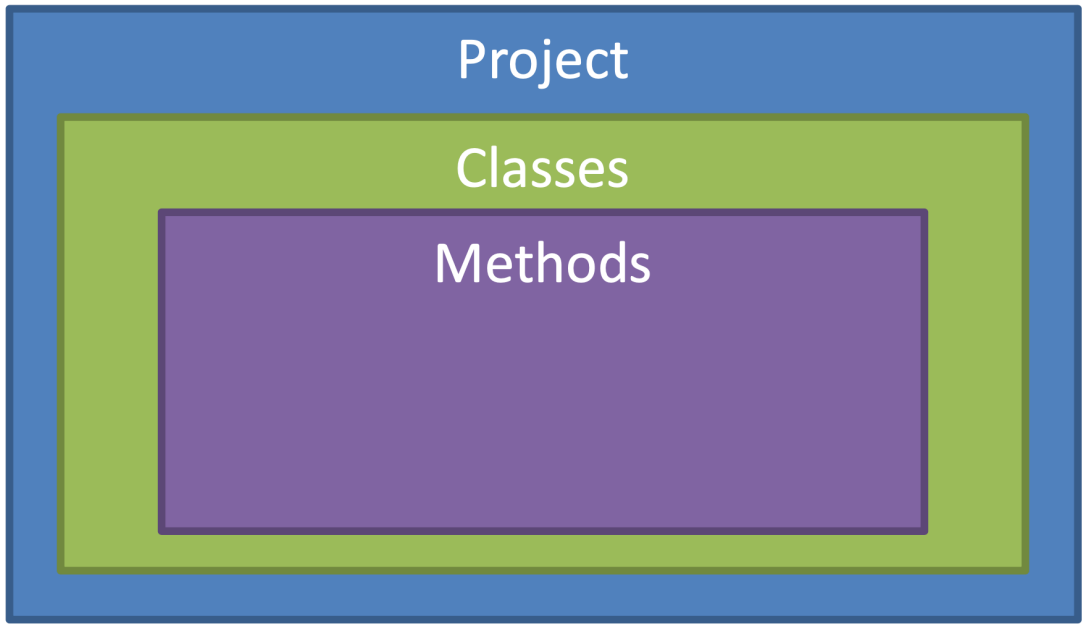
Outline

- Overview
- Quick Quiz
- Example

Overview

- Organized and structured code helps to:
  - Reuse parts of code, so you use less statements
  - Quickly find bugs or errors
  - Easily add or extend functionality
- Java Organizes Software
  - First in Projects
  - Then in Classes
  - Then in Methods

Java Software Structure



Overview

- Classes are a way that we can create *classifications* of “objects”
- Instances of a class are referred to as “objects”
- Classes provide a “blueprint” of a class of objects
  - Shared Qualities
  - Shared Characteristics
- Classes combine
  - Data (Attributes / Properties)
  - Methods (Actions)
- Think of Classes as *nouns*

Class Concept

Cat	
Attributes	• Name • Weight • Number of Legs
Actions	• Eat • Sleep • Destroy your stuff

Overview

Class Concept

Cat	
Attributes	• Name • Weight • Number of Legs
Actions	• Eat • Sleep • Destroy your stuff

Object Concept

- `cat01 =`

Cat	
Attributes	• Name = “Mr. Flufferkins” • Weight = 8.0 • Number of Legs = 4
Actions	• Eat • Sleep • Destroy your stuff

- `cat02 =`

Cat	
Attributes	• Name = “Triscuit” • Weight = 9.2 • Number of Legs = 4
Actions	• Eat • Sleep • Destroy your stuff

- `cat03 =`

Cat	
Attributes	• Name = “Dr. Boots” • Weight = 8.6 • Number of Legs = 4
Actions	• Eat • Sleep • Destroy your stuff

Overview

- Classes become “types” that are defined by programmers
  - Data like variables and constants require a type followed by an identifier
- Class identifiers hold a reference (memory address) to the contents in memory
- Classes allow objects to be created *dynamically* in memory
- Generally in the “heap” section of memory

Declaring a Class Type

```
<<Class Type>> <<id>>;
```

Example

```
Random r;
```

Overview

- Dynamic allocation replicates code written in a class in memory
- Using the word “new” to *construct/create* an instance of a class replicates the code for that instance
- An object that has not been constructed has the “null” memory address
  - If an object is NULL then it has not been created in memory
  - Before calling methods from objects make sure to check if that object is NULL
  - Not checking can result in a runtime error called a “NullPointerException”

Constructing a Class Type

```
<<id>> = new <<Constructor>>;
```

Example

```
r = new Random();
```

Overview

Creating a Class in 7 Easy Steps!

1. Define the class
2. Create Properties
   1. Instance Variables
   2. Constants
3. Define Constructors
   1. Default
   2. Parameterized
4. Create Accessors for every Instance Variable
5. Create Mutators for every Instance Variable
6. Create other Methods
   1. equals()
   2. toString()
7. Use the Class to create Objects!

Overview

- Defining a class creates the type
- The scope of a class should be “public” in most cases
- The identifier follows the same general rules as variable and method identifiers
  - Only exception is it is good programming practice to **capitalize the first letter** in Class identifiers
- In Java, the class name must match the filename
- In Java, one *external class* per file

Defining a Class

```
<<scope>> class <<class id>>
{
        <<Body of the class>>
}
```

Example

```
public class Cat
{
        //Body of class
}
```

Overview

- Properties are the data each object contains
  - Constants
  - Variables called *Instance Variables*
- Instance variable's scope should be "private"
- Encapsulation
  - Information hiding
  - We want to protect information to preserve its integrity

Defining Instance Variables

```
//Inside of class
private <<type>> <<id>>;
```

Example

```
//Inside of class Cat
private String name;
private double weight;
private int numberOfLegs;
```

Overview

- Special kinds of methods used to create an instance (construct) of the class called an object
- Does not have a return type
- The constructor's identifier must match the class' name
- Default Constructors
  - No parameters
  - Set all instance variables to a *default* value
- Parameterized Constructors
  - Parameter for each instance variable
  - Parameters assign instance variable values as long as they are valid (discussed later)

Defining a Default Constructor

```
public <<class id>>()
{
        <<Body of the constructor>>
}
```

Example

```
public Cat()
{
        this.name = "none"
        this.weight = 1.0;
        this.numberOfLegs = 4;
}
```

Overview

- Methods that provide access to properties outside of the class
  - Instance variables are defined as “private” so we cannot directly access this information
- Sometimes called “Getters”
- Create an accessor for every instance variable (“InstV”)
- The reserved word “this” is a way to access a class’ variables and methods
  - “Self-referential pointer”

Defining an Accessor

```
public <<InstV type>> get<<InstV id>>()
{
      return this.<<iv>>;
}
```

Example

```
public String getName()
{
      return this.name;
}
```

Overview

- Methods that allows properties to be modified from outside of the class
  - Instance variables are defined as “private” so we cannot directly modify this information
- Sometimes called “Setters”
- Create a mutator for every instance variable (“InstV”)
- Make sure to check for errors!
- The reserved word “this” is a way to access a class’ variables and methods
  - “Self-referential pointer”

Defining a Mutator

```
public void set<<InstV id>>(<<parameter>>)
{
	if(<<Error check>>)
	{
		this.<<iv>> = <<parameter>>;
	}
}
```

Example

```
public String setName(String aN)
{
	if(aN != null)
	{
		this.name = aN;
	}
	…
}
```

Overview

- The ".equals" method checks to see if another instance of this same type has the same properties
- Make sure the other instance is not NULL first
- Use Accessors to compare data from another instance with *this* instance
- Very useful to write these for every class

Defining equals

```
public boolean equals(<<another instance>>)
{
      return <<all properties equal
            another instance's properties>>
}
```

Example

```
public boolean equals(Cat aC)
{
     return aC != null &&
     this.name.equals(aC.getName()) && …
}
```

Overview

- The “toString” returns a String will the values of all of the properties concatenated together.
- Called by “System.out.print” or “System.out.println”
- Useful for debugging
- Good idea to write these for every class

Defining toString

```
public String toString()
{
	return <<all properties equal
		another instance’s properties>>
}
```

Example

```
public String toString()
{
	return this.name + " " +
		this.weight + " " +
		this.numberOfLegs;
}
```

Outline

- Overview
- Quick Quiz
- Example

Cat Class

- Has two different constructors
- Parameterized constructor calls “setters” or “mutators”

```
public class Cat {
    private String name;
    private double weight;
    private int numberOfLegs;
    public Cat() {
        this.name = "none";
        this.weight = 1.0;
        this.numberOfLegs = 4;
    }
    public Cat(String name, double weight, int numberOfLegs) {
        this.setName(name);
        this.setWeight(weight);
        this.setNumberOfLegs(numberOfLegs);
    }
```

Cat Class

- Getters and setters (mutators)

```
public String getName() {
	return this.name;
}
public void setName(String name) {
	this.name = name;
}
public double getWeight() {
	return this.weight;
}
public void setWeight(double weight) {
	this.weight = weight;
}
public int getNumberOfLegs() {
	return this.numberOfLegs;
}
public void setNumberOfLegs(int numberOfLegs) {
	this.numberOfLegs = numberOfLegs;
}
```

Cat Class

- Other methods

```
public void eat() {
        this.weight += 1;
}
public String toString() {
    return this.name + " " + this.weight + " " +
    this.numberOfLegs;
}
public boolean equals(Cat other) {
    return (this.getName().equals(other.getName())) &&
    (this.getWeight() == other.getWeight()) &&
    (this.getNumberOfLegs() == other.getNumberOfLegs());
}
```

Quick Quiz

- What are the outputs of the print statements?

```
public class Cat {
    private String name;
    private double weight;
    private int numberOfLegs;
    public Cat() {
        this.name = "none";
        this.weight = 1.0;
        this.numberOfLegs = 4;
    }
    public Cat(String name, double weight, int
    numberOfLegs) {
        this.setName(name);
        this.setWeight(weight);
        this.setNumberOfLegs(numberOfLegs);
    }
    public void eat() {
        this.weight += 1;
    }
```

```
Cat cat01;
cat01 = new Cat();
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);

Cat cat02 = new Cat("Triscuit", 9.2, 4);
Cat cat03 = new Cat("Mr. Flufferkins",
8.0, 4);

System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);

cat01.eat();
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);

cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
System.out.println(cat03.equals(cat02));
System.out.println(cat03 == cat02);
```

Quick Quiz

- What are the outputs of the print statements?

```
Cat cat01;
cat01 = new Cat();
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);

Cat cat02 = new Cat("Triscuit", 9.2, 4);
Cat cat03 = new Cat("Mr. Flufferkins",
8.0, 4);

System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);

cat01.eat();
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);

cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
System.out.println(cat03.equals(cat02));
System.out.println(cat03 == cat02);
```

```
false
true
false
false
false
Dr. Boots
true
true
```

Outline

- Overview
- Quick Quiz
- Example

Example

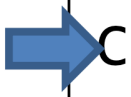
```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
```

Memory

Identifier	Contents	Byte Address
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
```

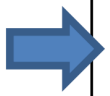


Memory

Identifier	Contents	Byte Address
…	…	…
cat01	NULL	28
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
```



Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	“none”	-
cat01.weight	1.0	-
cat01.numberOfLegs	4	-
cat01.<<methods>>	-	-
…	…	…

- Will omit .methods sometimes for brevity

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	**"Mr. Flufferkins"**	-
cat01.weight	1.0	-
cat01.numberOfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	**"Mr. Flufferkins"**	-
cat01.weight	**8.0**	-
cat01.numberOfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	**"Mr. Flufferkins"**	-
cat01.weight	**8.0**	-
cat01.numberOfLegs	4	-
…	…	…
…	…	…

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Triscuit"**	-
cat02.weight	**9.2**	-
cat02.numberOfLegs	**4**	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Triscuit"**	-
cat02.weight	**9.2**	-
cat02.number OfLegs	**4**	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	120	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.number OfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	**"Mr. Flufferkins"**	-
cat01.weight	**8.0**	-
cat01.numberOfLegs	4	-
…	…	…
…	…	…

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Triscuit"**	-
cat02.weight	**9.2**	-
cat02.numberOfLegs	**4**	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	**"Mr. Flufferkins"**	-
cat01.weight	**8.0**	-
cat01.numberOfLegs	4	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	120	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.numberOfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

Memory

Identifier	Contents	Byte Address
…	…	…
cat01	40	28
…	…	…
Cat	-	40
cat01.name	**"Mr. Flufferkins"**	-
cat01.weight	**8.0**	-
cat01.numberOfLegs	4	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	120	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.numberOfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Triscuit"**	-
cat02.weight	**9.2**	-
cat02.number OfLegs	**4**	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	120	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.number OfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Triscuit"**	-
cat02.weight	**9.2**	-
cat02.number OfLegs	**4**	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	96	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.number OfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Dr. Boots"**	-
cat02.weight	**9.2**	-
cat02.number OfLegs	**4**	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	96	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.number OfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Dr. Boots"**	-
cat02.weight	**9.2**	-
cat02.number OfLegs	**4**	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	96	78
…	…	…
Cat	-	120
cat03.name	**"Mr. Flufferkins"**	-
cat03.weight	**8.0**	-
cat03.number OfLegs	4	-
…	…	…
…	…	…

Example

```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
Cat cat02 = new Cat("Triscuit",9.2,4);
//Declare and Construct another cat
Cat cat03 = new Cat("Mr. Flufferkins", 8.0,4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
System.out.println(cat01 == cat03);
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

More Memory

Identifier	Contents	Byte Address
…	…	…
cat02	96	64
…	…	…
Cat	-	96
cat02.name	**"Dr. Boots"**	-
cat02.weight	**9.2**	-
cat02.number OfLegs	**4**	-
…	…	…
…	…	…

Even More Memory

Identifier	Contents	Byte Address
…	…	…
cat03	96	78
…	…	…

- Data from no longer referred to will eventually get garbage collected

Recap

- Identifiers for class types only contain a memory address
- Multiple identifiers can reference the same address
- New instances of classes can only be created by using Constructors
  - Assigning one identifier of a class to another does not create clones
- Memory managed languages, like Java, have a mechanism called a “Garbage Collector”
- When the reference to an object is lost the “Garbage Collector” removes that information from memory

Java Software Structure