Classes and Objects
Part 01
• 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
• 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

Java Software Structure
Classes

- 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*
• 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*
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*
• 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*
• 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*
• 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*
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*

<table>
<thead>
<tr>
<th>Class Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cat</strong></td>
</tr>
</tbody>
</table>
| **Attributes**| • Name  
                 • Weight  
                 • Number of Legs |
| **Actions**   | • Eat   |
• 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**

<table>
<thead>
<tr>
<th>Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td></td>
</tr>
<tr>
<td>Number of Legs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eat</td>
<td></td>
</tr>
<tr>
<td>Sleep</td>
<td></td>
</tr>
</tbody>
</table>
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
Class Concept

Object Concept

Cat

Attributes
- Name
- Weight
- Number of Legs

Actions
- Eat
- Sleep
- Destroy your stuff

Cat

Attributes
- Name = “Mr. Flufferkins”
- Weight = 8.0
- Number of Legs = 4

Actions
- Eat
- Sleep
- Destroy your stuff

\[ \text{cat01} = \]
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
Class Concept

- **cat01**
  - Attributes:
    - Name
    - Weight = 8.0
    - Number of Legs = 4
  - Actions:
    - Eat
    - Sleep
    - Destroy your stuff

- **cat02**
  - Attributes:
    - Name = "Triscuit"
    - Weight = 9.2
    - Number of Legs = 4
  - Actions:
    - Eat
    - Sleep
    - Destroy your stuff

- **cat03**
  - Attributes:
    - Name = "Dr. Boots"
    - Weight = 8.6
    - Number of Legs = 4
  - Actions:
    - Eat
    - Sleep
    - Destroy your stuff

Object Concept

<table>
<thead>
<tr>
<th>Cat</th>
<th>Attributes</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name = &quot;Mr. Flufferkins&quot;</td>
<td>Eat</td>
</tr>
<tr>
<td></td>
<td>Weight = 8.0</td>
<td>Sleep</td>
</tr>
<tr>
<td></td>
<td>Number of Legs = 4</td>
<td>Destroy your stuff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cat</th>
<th>Attributes</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name = &quot;Triscuit&quot;</td>
<td>Eat</td>
</tr>
<tr>
<td></td>
<td>Weight = 9.2</td>
<td>Sleep</td>
</tr>
<tr>
<td></td>
<td>Number of Legs = 4</td>
<td>Destroy your stuff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cat</th>
<th>Attributes</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name = &quot;Dr. Boots&quot;</td>
<td>Eat</td>
</tr>
<tr>
<td></td>
<td>Weight = 8.6</td>
<td>Sleep</td>
</tr>
<tr>
<td></td>
<td>Number of Legs = 4</td>
<td>Destroy your stuff</td>
</tr>
</tbody>
</table>
Classes

• 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;
• 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

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

### Example

```java
r = new Random();
```
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!
Example
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

---

Define the Class

Defining a Class

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

Example

```java
public class Cat
{
   //Body of class
}
```
Defining Instance Variables

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

Properties

• 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

Example

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

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

**Example**

```java
public Cat()
{
    this.name = "none"
    this.weight = 1.0;
    this.numberOfLegs = 4;
}
```
Defining a Parameterized Constructor

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

• 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)

Example
public Cat(String aN, String aW, int aN)  
{  
   //Call mutators (Discussed later)  
}
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>NULL</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
//Declare the cat
Cat cat01;

//Construct the Cat
cat01 = new Cat();
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
Defining an Accessor

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

• 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”

Example

```java
public String getName()
{
    return this.name;
}
```
Defining a Mutator

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

- 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”

**Example**

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

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

• 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

Example

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

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

```java
// Example
public String toString()
{
    return this.name + " " +
            this.weight + " " +
            this.numberOfLegs;
}
```
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
Objects in Memory

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

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>cat01.name</td>
<td>“none”</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>1.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLeds</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>cat01.&lt;methods&gt;</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```java
//Declare the cat
Cat cat01;
//Construct the Cat
cat01 = new Cat();
```

### Objects in Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>cat01.name</td>
<td>“none”</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>1.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());

### Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>cat01.name</td>
<td>&quot;none&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>1.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>cat01.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
```

---

<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>cat01</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
</tr>
<tr>
<td>cat01.name</td>
</tr>
<tr>
<td>cat01.weight</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>More Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
// 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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//Declare the cat
cat01 = new Cat();
//Construct the Cat
cat01 = new Cat();
//Set properties
cat01.setName("Mr. Flufferkins");
cat01.setWeight(8.0);
//Declare and Construct another cat
cat02 = new Cat("Triscuit", 9.2, 4);
//Declare and Construct another cat
cat03 = new Cat("Mr. Flufferkins", 8.0, 4);
System.out.println(cat01.equals(cat02));
System.out.println(cat01.equals(cat03));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>cat01.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat02</td>
<td>96</td>
<td>64</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>cat02.name</td>
<td>&quot;Triscuit&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat02.weight</td>
<td>9.2</td>
<td>-</td>
</tr>
<tr>
<td>cat02.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

More Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>28</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat02</td>
<td>&quot;Triscuit&quot;</td>
<td>64</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat02.name</td>
<td>&quot;Triscuit&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat02.weight</td>
<td>9.2</td>
<td>-</td>
</tr>
<tr>
<td>cat02.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

More Memory
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat02</td>
<td>96</td>
<td>64</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>cat02.name</td>
<td>&quot;Triscuit&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat02.weight</td>
<td>9.2</td>
<td>-</td>
</tr>
<tr>
<td>cat02.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat03</td>
<td>120</td>
<td>78</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>120</td>
</tr>
<tr>
<td>cat03.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat03.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat03.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());

Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat01</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>cat01.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat01.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat01.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Even More Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cat03</td>
<td>120</td>
<td>78</td>
</tr>
<tr>
<td>Cat</td>
<td>-</td>
<td>120</td>
</tr>
<tr>
<td>cat03.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat03.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat03.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

More Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat02</td>
<td>96</td>
<td>64</td>
</tr>
<tr>
<td>cat03</td>
<td>120</td>
<td>78</td>
</tr>
<tr>
<td>cat02.name</td>
<td>&quot;Triscuit&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat02.weight</td>
<td>9.2</td>
<td>-</td>
</tr>
<tr>
<td>cat02.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

Even More Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Contents</th>
<th>Byte Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat03.name</td>
<td>&quot;Mr. Flufferkins&quot;</td>
<td>-</td>
</tr>
<tr>
<td>cat03.weight</td>
<td>8.0</td>
<td>-</td>
</tr>
<tr>
<td>cat03.numberOfLegs</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
//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));
cat03 = cat02;
cat02.setName("Dr. Boots");
System.out.println(cat03.getName());
• 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