Tinker Academy

AP Computer Science Prep (Java Programming)
Lecture 3 - Java Fundamentals 1
(OOP Part 2)

Winter 2015

Introduction to
Object Oriented Programming

Lecture 3 - Java Fundamentals 1

Introduction to Object Oriented Programming (OOP)

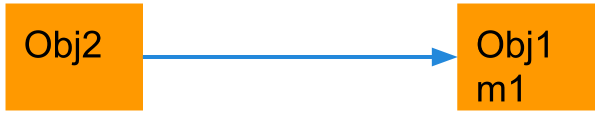
- OOP is a programming “model”.
- Many languages support OOP, such as C++ and Java.
- Many do not, such as C, ML,and Pascal.

Lecture 3 - Java Fundamentals 1

Introduction to Object Oriented Programming (OOP)

- In OOP, everything is an “object”*
- The entire running program is just a bunch of objects
- An object can communicating with another object by invoking its "method"

Lecture 3 - Java Fundamentals 1



Obj1, Obj2 are objects

m1 is Obj1’s method

Obj2 communicates with Obj1 by invoking Obj1's method m1

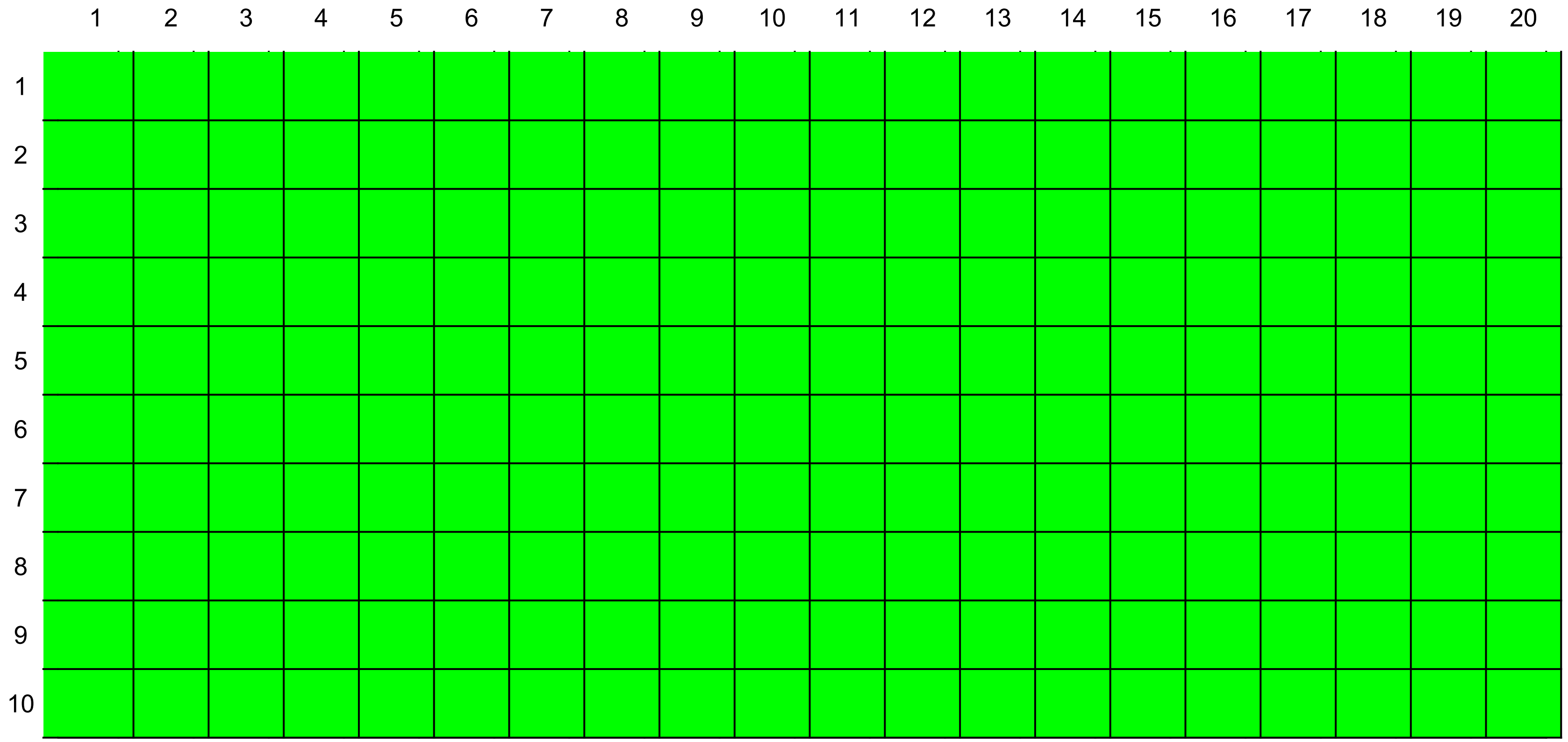
What is an Object?

Lecture 3 - Java Fundamentals 1

So what is an object anyway?

- An object is a central concept in a OOP
- An object represents some entity that can be identified
- An object **should be uniquely identifiable**

Lecture 3 - Java Fundamentals 1



Lecture 3 - Java Fundamentals 1

What can it do?

- An object can store useful data
- An object can point to other objects

Lecture 3 - Java Fundamentals 1

Object Bugatti1	
Fields	
color1	"Gray"
color2	"Red"
imageObj	

Object Bugatti2	
Fields	
color1	"Titanium"
color2	"Black"
imageObj	

Object Bugatti3	
Fields	
color1	"Green"
color2	"Ebony"
imageObj	

Object ShowRoom1	
Fields	
address	999 Van Ness Avenue, San Francisco, CA 94109
cars	[ref1 , ref2 , ref3]

Lecture 3 - Java Fundamentals 1

What can it do?

- Objects can do things
- Behavior are the things the objects can do
- Behavior is defined by "methods"

Lecture 3 - Java Fundamentals 1

Bugatti 1	
Fields	
color1	"Gray"
color2	"Red"
imageObj	
showroom	ref1
Methods	
start()	
setSpeed(newSpeed)	
stop()	

Bugatti 2	
Fields	
color1	"Titanium"
color2	"Black"
imageObj	
showroom	ref2
Methods	
start()	
setSpeed(newSpeed)	
stop()	

Bugatti 3	
Fields	
color1	"Green"
color2	"Ebony"
imageObj	
showroom	ref3
Methods	
start()	
setSpeed(newSpeed)	
stop()	

Lecture 3 - Java Fundamentals 1

So what is an object anyway?

- Every object is created from a certain “class”
- The class forms the blueprint
- The blueprint indicates the kind of data its objects can store
- The blueprint indicates the behavior of the objects

Lecture 3 - Java Fundamentals 1

BLUEPRINT 1

BugattiBlueprint1	
Fields	
Field1	color1
Field2	color2
Field3	imageObj
Methods	
Method1	start()
Method2	stop()
Method3	setSpeed(newSpeed)

Lecture 3 - Java Fundamentals 1

So who designs this blueprint? You

You get to

- Choose the data fields
- Choose the objects behavior

Lecture 3 - Java Fundamentals 1

BLUEPRINT 2

BugattiBlueprint2	
Fields	
Field1	color
Field2	yearOfManufacture
Field3	maxSpeed
Methods	
Method1	start()
Method2	stop()
Method3	setGear(gearLevel)

What is a Class?

Lecture 3 - Java Fundamentals 1

What is a class?

- The “class” is a central concept in Java OOP
- A class is a “blueprint” or “datatype”
- The blueprint defines the data fields
- The blueprint defines the methods
- The blueprint defines the constructors
- Every object is created from a “class”

Lecture 3 - Java Fundamentals 1

What is a class?

- In OOP, each class is a small program
- In java, each class is in a separate source file
- The simplest OOP program is a single class
- Your Java program is nothing but a bunch of classes
- The JDK provide 4000+ classes
- These classes in the JDK are called the Java API.
- You get to define your own classes (or blueprints)

Lecture 3 - Java Fundamentals 1

Class BugattiBlueprint1	
Fields	
Field1	color1
Field2	color2
Field3	imageObj
Methods	
Method1	start()
Method2	stop()
Method3	setSpeed(newSpeed)

```
public class BugattiBlueprint1 {

        private String color1;
        private String color2;
        private Image imageObj;

        public void start() {
            // body of the start() method
        }

        public void stop() {
            // body of the start() method
        }

        public void setSpeed(int newSpeed) {
            // body of the start() method
        }
}
```

Lecture 3 - Java Fundamentals 1

Class BugattiBlueprint2	
Fields	
Field1	color
Field2	yearOfManufacture
Field3	maxSpeed
Methods	
Method1	start()
Method2	stop()
Method3	setGear(gearLevel)

```
public class BugattiBlueprint2 {

        private String color1;
        private int yearOfManufacture;
        private int maxSpeed;

        public void start() {
            // body of the start() method
        }

        public void stop() {
            // body of the start() method
        }

        public void setGear(int gearLevel) {
            // body of the start() method
        }
}
```

What is a field?

Lecture 3 - Java Fundamentals 1

What is a field?

- A field is a named placeholder
- Can hold a value

Assigning and Accessing Values

Lecture 3 - Java Fundamentals 1

Assigning values

- Assigning a value to a field is called an **assignment**
- Uses the special symbol =
- Left Hand side of the = is the field name
- Right Hand side of the = is the field value

Lecture 3 - Java Fundamentals 1

Reading values

- Reading a value from a field is called an **access**
- Uses the special dot (.) operator if accessing through object

What is a method?

Lecture 3 - Java Fundamentals 1

What is a method?

- A method is a small program defined as part of the class
- Accepts input, does something useful and usually returns output
- Made up of statements and blocks

Lecture 3 - Java Fundamentals 1

Class BugattiBlueprint	
Fields	
Field1	color1
Field2	color2
Field3	imageObj
Methods	
Method1	start()
Method2	stop()
Method3	setSpeed(newSpeed)

```
public class BugattiBlueprint1 {

        private String color1;
        private String color2;
        private Image imageObj;

        public void start() {
            // body of the start() method
        }

        public void stop() {
            // body of the start() method
        }

        public void setSpeed(int newSpeed) {
            // body of the start() method
        }
}
```

Lecture 3 - Java Fundamentals 1

Methods have Signatures

Method name	Signature
start	public void start(
stop	public void stop()
setSpeed	public void setSpeed(int newSpeed)

What is an Interface?

Lecture 3 - Java Fundamentals 1

What is an interface?

- Classes can have common methods
- The interface **is a description of set of common methods**
- Shows **what** the behavior is
- Does **not** show **how** the behavior is implemented. **Why?**

Lecture 3 - Java Fundamentals 1

Class BugattiBlueprint1	
Fields	
Field1	color1
Field2	color2
Field3	imageObj
Methods	
Method1	start()
Method2	stop()
Method3	setSpeed(newSpeed)

BugattiBlueprint1 is a type of Car

Interface Car	
Methods	
Method1	start()
Method2	stop()
Method3	setSpeed(newSpeed)

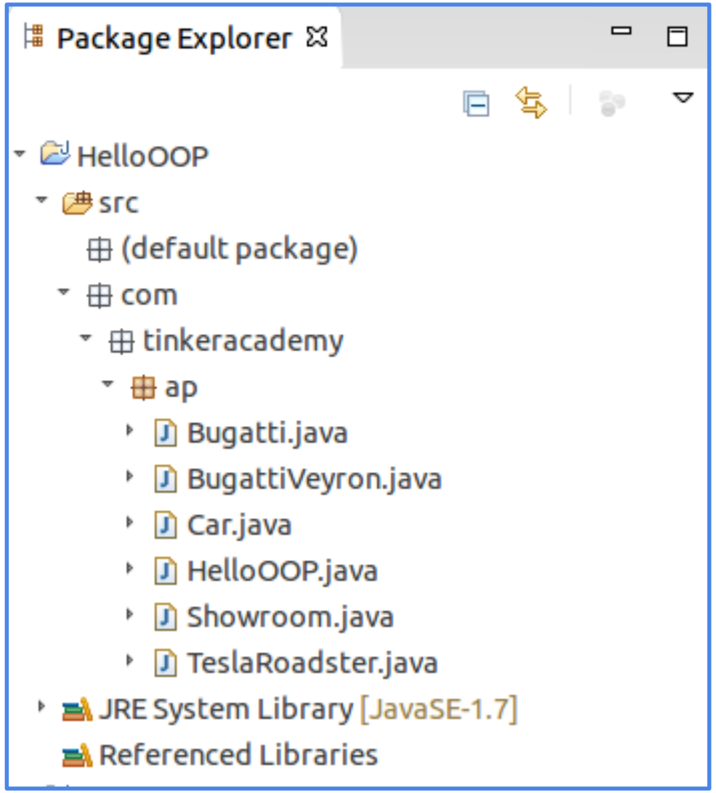
Package

Lecture 3 - Java Fundamentals 1

What is a package?

- Used to organize classes
- A class can belong to only 1 package
- A package can have many classes or other packages

Lecture 3 - Java Fundamentals 1



Package Explorer

```
HelloOOP
  src
    (default package)
    com
      tinkeracademy
        ap
          Bugatti.java
          BugattiVeyron.java
          Car.java
          HelloOOP.java
          Showroom.java
          TeslaRoadster.java
  JRE System Library [JavaSE-1.7]
  Referenced Libraries
```

Superclass and Subclass

Lecture 3 - Java Fundamentals 1

What is a superclass?

- Used to organize classes
- **Allows code reuse**
- A class can be a special form of another class
- The special form is a Subclass
- Every class in Java is a special form (subclass) of the **Object** class
- The **Object class** is called the superclass

Lecture 3 - Java Fundamentals 1

$$\begin{tikzcd}
 & \text{Object} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Number} \arrow[d] & \text{Bugatti} \arrow[d] & \text{String} \\
\text{Integer} & \text{BugattiVeyron} &
\end{tikzcd}$$

Lecture 3 - Java Fundamentals 1

What is a superclass?

- The **Object class** is the superclass of ALL classes in Java
- Number is a subclass of Object and superclass of Integer
- Integer is a subclass of ?
- String is a subclass of ?
- Bugatti is a subclass of ?
- Bugatti is a superclass of ?
- BugattiVeyron is a subclass of ?

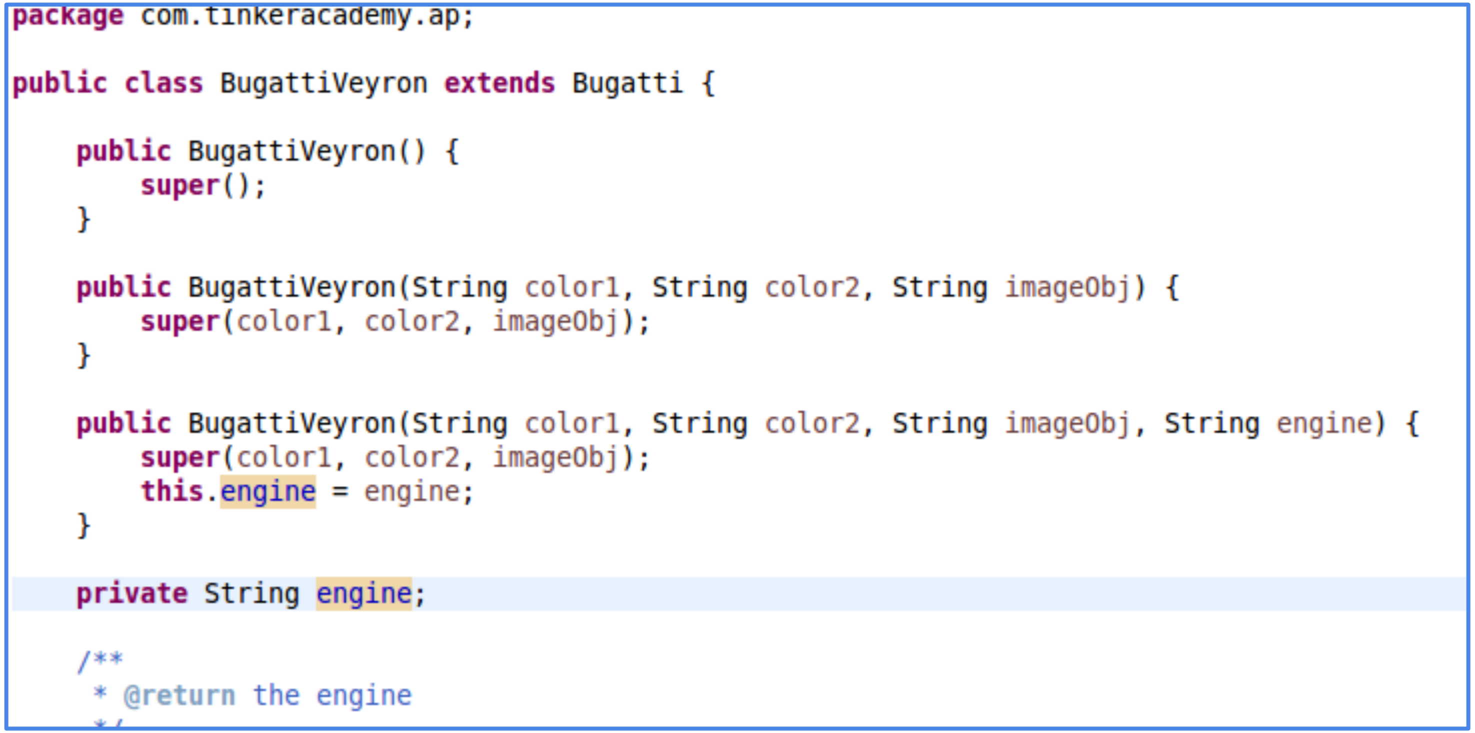
Constructor

Lecture 3 - Java Fundamentals 1

What is a constructor?

- Used to create objects from a class
- Looks like a method but cannot return anything
- Has the **same name** as the class
- Can have any number of inputs
- A class can define multiple constructors
- A constructor can invoke the superclass constructor
- A constructor cannot be invoked directly

Lecture 3 - Java Fundamentals 1



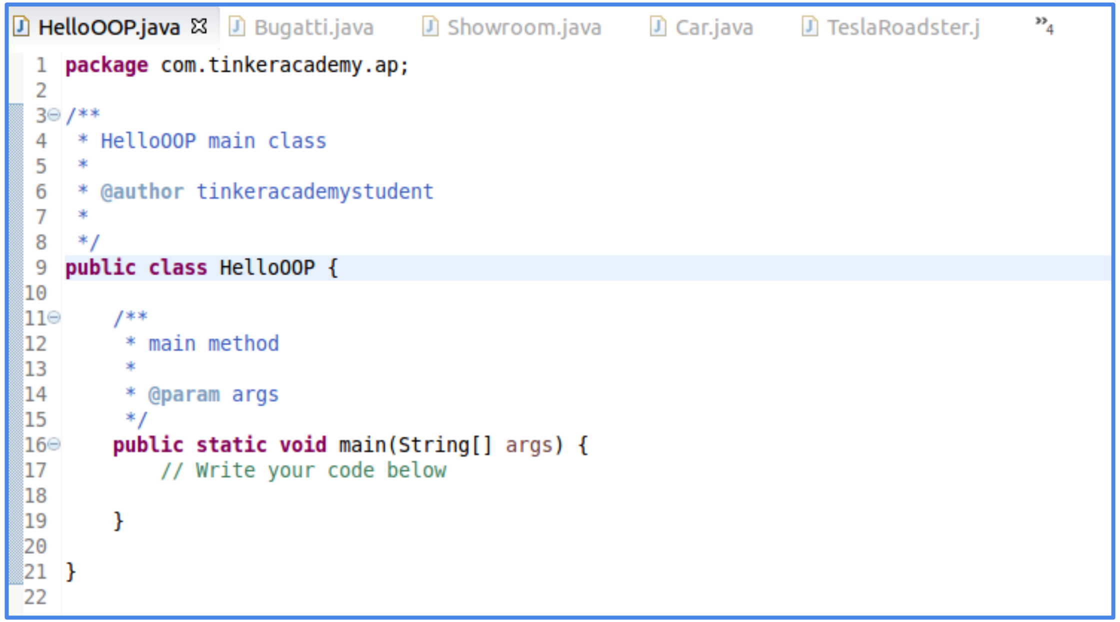
```
package com.tinkeracademy.ap;

public class BugattiVeyron extends Bugatti {

    public BugattiVeyron() {
        super();
    }

    public BugattiVeyron(String color1, String color2, String imageObj) {
        super(color1, color2, imageObj);
    }

    public BugattiVeyron(String color1, String color2, String imageObj, String engine) {
        super(color1, color2, imageObj);
        this.engine = engine;
    }

    private String engine;

    /**
     * @return the engine
```

Lecture 3 - Java Fundamentals 1

What is a constructor?

- 3 constructors
- Each constructor takes different inputs (or no inputs)
- Each constructor does something different

Lecture 3 - Java Fundamentals 1



```java
package com.tinkeracademy.ap;

/**
 * HelloOOP main class
 *
 * @author tinkeracademystudent
 *
 */
public class HelloOOP {

    /**
     * main method
     *
     * @param args
     */
    public static void main(String[] args) {
        // Write your code below

    }

}

```

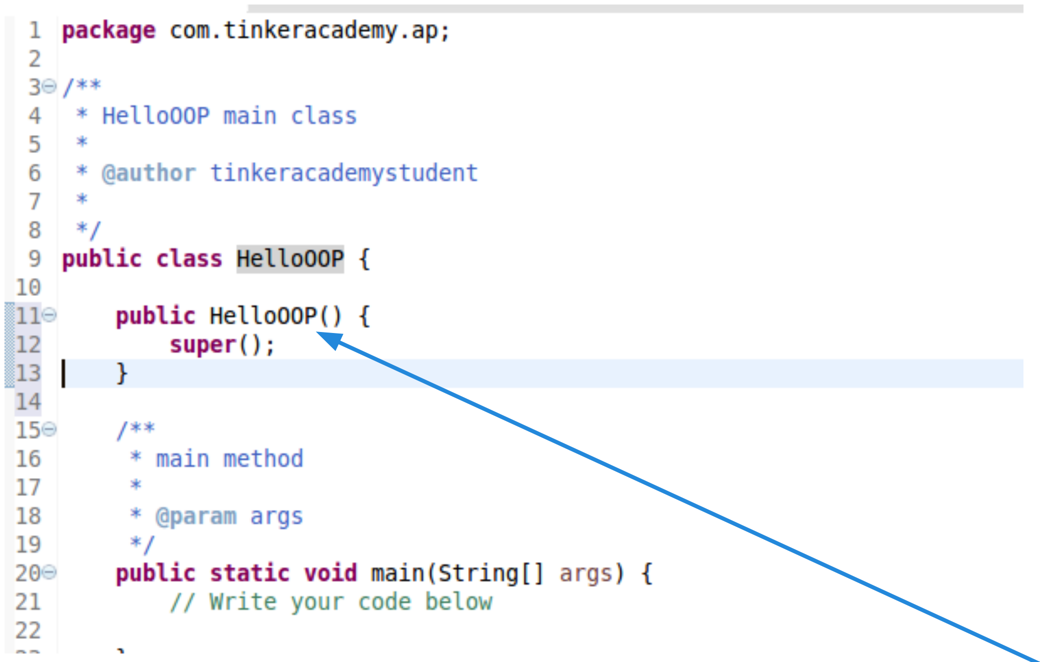
Where is the constructor?

Lecture 3 - Java Fundamentals 1

What is a constructor?

- Every class **requires** a constructor
- If you don't provide, the Java Compiler will **create** one for you
- The created constructor will take no arguments

Lecture 3 - Java Fundamentals 1



```
package com.tinkeracademy.ap;

/**
 * HelloOOP main class
 *
 * @author tinkeracademystudent
 *
 */
public class HelloOOP {

    public HelloOOP() {
        super();
    }

    /**
     * main method
     *
     * @param args
     */
    public static void main(String[] args) {
        // Write your code below

```

The created constructor will look like this

Access Permission

Lecture 3 - Java Fundamentals 1

Private vs Public

- Indicates which method can access
- private field
  - Only methods in this class
- public field
  - ANY method in ANY class
- private method
  - Only other methods in this class
- public method
  - ANY method in ANY class

Lecture 3 - Java Fundamentals 1

Static vs non static

- static field
  - Any data is shared by ALL objects of that class
- non static field
  - Data is owned by the object
- static method
  - Invoked the method through the class
- non static method
  - Invoke the method through the object

Whew!
That was a lot of stuff!

We will absorb it
Over this class and some of next

Class Activity

Lecture 3 - Java Fundamentals 1

Getting Started

- Open File Manager
- Navigate to starterpack/starterpack3
- Click on HelloOOP.zip
- Right Click, Extract Here
- Start Eclipse
- Import HelloOOP into Eclipse

Lecture 3 - Java Fundamentals 1

Concepts

- Class
- Object
- Field
- Constructor
- Assignment and Access
- Method
- Interface
- Package
- Superclass and Subclass
- Access Permission

Concept #1
Class

Lecture 3 - Java Fundamentals 1

Create a new Class

- Click on OOP in Package Explorer
- File->New->Class
- Type in Bugatti
- Click Finish
- This generates a new Java Source File called Bugatti.java
- Congratulations! you just created a new class
- File->Save All to save your changes

Concept #2
Object

Lecture 3 - Java Fundamentals 1

Create new Objects

- Navigate to HelloOOP.java under src
- Double click to open
- Type in the following code within the main method { and }

```
Bugatti bugattiGrayRed = new Bugatti();
Bugatti bugattiTitaniumGray = new Bugatti();
```

- Edit->Select All, Right Click in Editor
- Source->Correct Indentation, File->Save All
- Run, Test Program

Concept #3
Field

Lecture 3 - Java Fundamentals 1

Create new data fields

- Navigate to the Bugatti.java under src
- Double click to open
- Type in the following code within the class { and }

```
private String color1;
private String color2;
private String imageObj;
private int speed;
```

- Edit->Select All, Right Click in Editor
- Source->Correct Indentation, File->Save All
- Run, Test Program

Lecture 3 - Java Fundamentals 1

Create new data fields

- Fields are part of the Class
- Every time an object is created it gets placeholders for the fields based on class (blueprint) and a default value
- The default value for speed is 0, the others is **null**

Object 1	
color1	null
color2	null
imageObj	null
speed	0

Object 2	
color1	null
color2	null
imageObj	null
speed	0

Concept #4
Constructor

Lecture 3 - Java Fundamentals 1

Constructor

- Navigate to the Bugatti.java under src
- Double click to open
- Type in the following code within the class { and }

```
public Bugatti(String color1, String color2, String imageObj) {


}
```

- Code adds a constructor with 3 inputs
- Lets use the new constructor

Lecture 3 - Java Fundamentals 1

Constructor

- Navigate to HelloOOP.java under src
- Double click to open
- Type in the following code within the main method { and }

```
Bugatti bugattiGrayRed = new Bugatti("Gray", "Red", "GrayRedBugatti.jpg");
Bugatti bugattiTitaniumGray = new Bugatti("Titanium", "Gray", "TitaniumGrayBugatti.jpg");
```

- Edit->Select All, Right Click in Editor
- Source->Correct Indentation, File->Save All
- Run, Test Program

Lecture 3 - Java Fundamentals 1

Constructor

- What do the objects look like now?

Object 1	
color1	?
color2	?
imageObj	?
speed	?

Object 2	
color1	?
color2	?
imageObj	?
speed	?

Concept #5
Assignment

Lecture 3 - Java Fundamentals 1

Assignment

- Navigate to the Bugatti.java under src
- Double click to open
- Type in the following code within the constructor { and }

```
public Bugatti(String color1, String color2, String imageObj) {
        this.color1 = color1;
        this.color2 = color2;
        this.imageObj = imageObj;
}
```

- Code adds a constructor with 3 inputs and assigns the values to the fields

Lecture 3 - Java Fundamentals 1

Assignment

- What do the objects look like now?

Object 1	
color1	?
color2	?
imageObj	?
speed	?

Object 2	
color1	?
color2	?
imageObj	?
speed	?

Concept #6
Method

Lecture 3 - Java Fundamentals 1

Method

- Need a way to access the value of color1 from the main method in HelloOOP
- Cannot access speed directly. Why?
- Need a method to access the speed
- Methods that are used to access objects own fields are called accessors

Lecture 3 - Java Fundamentals 1

Method

- Navigate to the Bugatti.java under src
- Double click to open
- Type in the following code within the constructor { and }

```
public int getSpeed() {
     return this.speed;
}
```

- **this is a special keyword** refers to this object
- return is a keyword used to return a value
- the code will return the value stored for the speed

Lecture 3 - Java Fundamentals 1

Method

- What values will getSpeed return for Object 1 ?
- What values will getSpeed return for Object 2 ?

Object 1	
speed	0

Object 2	
speed	0

Lecture 3 - Java Fundamentals 1

Method

- Navigate to HelloOOP.java under src
- Double click to open
- Type in the following code within the main method { and }

```
Bugatti bugattiGrayRed = new Bugatti("Gray", "Red", "GrayRedBugatti.jpg");
Bugatti bugattiTitaniumGray = new Bugatti("Titanium", "Gray", "TitaniumGrayBugatti.jpg");
System.out.println("bugattiGrayRed speed is " + bugattiGrayRed.getSpeed());
System.out.println("bugattiTitaniumGray speed is " + bugattiTitaniumGray.getSpeed());
```

- Edit->Select All, Right Click in Editor
- Source->Correct Indentation, File->Save All
- Run, Test Program

Lecture 3 - Java Fundamentals 1

Method Signature

- Need a way to access the value of color1 from the main method in HelloOOP
- Cannot access color1 directly. Why?
- Need a method to access color1
- Such methods that are used to access fields are called accessors

Concept #7
Interface

Lecture 3 - Java Fundamentals 1

Create a new Class

- Click on OOP in Package Explorer
- File->New->Class
- Type in Tesla
- Click Finish
- This generates a new Java Source File called Tesla.java
- File->Save All to save your changes

Lecture 3 - Java Fundamentals 1

- Add a new field to the class Tesla

```
private int speed;
```

- Add a new accessor to the class Tesla

```
public int getSpeed() {
    return speed;
}
```

Lecture 3 - Java Fundamentals 1

Identify common behavior

- We can identify some common behavior between Bugatti and Tesla
- For example, we might identify both classes provide a way to get the speed

Lecture 3 - Java Fundamentals 1

Behavior	Bugatti's method signature	Tesla's method signature
Get Speed	public int getSpeed()	public int getSpeed()

Lecture 3 - Java Fundamentals 1

Create an interface for common behavior

- We will call the new interface Car
- Interface Car will describe 1 methods, but will NOT implement them

Lecture 3 - Java Fundamentals 1

Create a new Interface

- Click on OOP in Package Explorer
- File->New->Interface
- Type in Car
- Click Finish
- This generates a new Java Source File called Car.java
- Congratulations! you just created a new interface
- File->Save All to save your changes

Lecture 3 - Java Fundamentals 1

- Interface reside in their own source files
- The source file name is the name of the interface (with the . java extension)

Lecture 3 - Java Fundamentals 1

- Add the method description to the interface within the { and }

```
public int getSpeed() {
    return speed;
}
```

Lecture 3 - Java Fundamentals 1

Modify both class

- Modify both classes to indicate that they implement the interface

```
public class Tesla implements Car {
```

```
public class Bugatti implements Car {
```

Lecture 3 - Java Fundamentals 1

Modify both class

- Tesla implements Car => Tesla behaves like a Car
- Bugatti implements Car => Bugatti behaves like a Car
- This is an extremely powerful concept called polymorphism which we will cover in the next lecture

Concept #8
Package

Lecture 3 - Java Fundamentals 1

Create a new package

- Click on OOP in Package Explorer
- File->New->Package
- Type in com
- Click Finish
- This generates a new Java package called "com"
- File->Save All to save your changes

Lecture 3 - Java Fundamentals 1

Create packages within this package

- Click on OOP in Package Explorer
- File->New->Package
- Type in com.tinkeracademy.ap
- Click Finish
- This generates a new Java package called "tinkeracademy" under "com" and a package "ap" under "tinkeracademy"
- File->Save All to save your changes

Lecture 3 - Java Fundamentals 1

Packages are folders!

- Open File Manager
- Navigate to starterpack3/HelloOOP
- Open the src folder
- Notice that src now contains the com folder
- The com folder contains the tinkeracademy folder
- The tinkeracademy folder now contains the ap folder
- Eclipse created these folders automatically when you created the packages

Lecture 3 - Java Fundamentals 1

Move the files

- Click on each file
- Right Click, Refactor, Move...
- Select com.tinkeracademy.ap
- Select OK
- Do this for each of the source files **except HelloOOP.java**

Lecture 3 - Java Fundamentals 1

Move the files

- Open File Manager
- Navigate to starterpack3/HelloOOP
- Notice that Eclipse has moved the files to under src/com/tinkeracademy/ap
- In addition each class has the following code

```
package com.tinkeracademy.ap;
```

Concept #9
Superclass and Subclass

Lecture 3 - Java Fundamentals 1

Create a new Subclass

- Click on OOP in Package Explorer
- File->New->Class
- Name should be **BugattiVeyron**
- Package should be **com.tinkeracademy.ap**
- Superclass, Browse..., type in Bugatti and click OK
- Check constructors from superclass
- Click Finish
- This generates a new Java Class called BugattiVeyron

Lecture 3 - Java Fundamentals 1

Create a new Subclass Class

- BugattiVeyron subclasses Bugatti
- The extends keyword indicates a subclass

```
public class BugattiVeyron extends Bugatti {
```

Lecture 3 - Java Fundamentals 1

Constructor

- Navigate to HelloOOP.java under src
- Double click to open
- Type in the following code within the main method { and }

```
Bugatti bugattiGrayRed = new Bugatti("Gray", "Red", "GrayRedBugatti.jpg");
Bugatti bugattiTitaniumGray = new Bugatti("Titanium", "Gray", "TitaniumGrayBugatti.jpg");
Bugatti bugattiGreenEbony = new Bugatti("Green", "Ebony", "GreenEbonyBugatti.jpg");
System.out.println("bugattiGrayRed speed is " + bugattiGrayRed.getSpeed());
System.out.println("bugattiTitaniumGray speed is " + bugattiTitaniumGray.getSpeed());
System.out.println("bugattiEbonyGreen speed is " + bugattiEbonyGreen.getSpeed());
```

- Edit->Select All, Right Click in Editor
- Source->Correct Indentation, File->Save All

Lecture 3 - Java Fundamentals 1

Subclass inherits method from superclass

- BugattiVeyron extends Bugatti => BugattiVeyron behaves like a Bugatti
- Bugatti has a method getSpeed, BugattiVeyron automatically **inherited** that method

Concept #10
Access Permission

Lecture 3 - Java Fundamentals 1

Access Permission

- The data field speed in class Bugatti cannot be accessed directly from HelloOOP. Why?
- The method getSpeed in class Bugatti can be accessed directly from HelloOOP. Why?

Lecture 3 - Java Fundamentals 1

Access Permission

- The data field speed in class Bugatti cannot be accessed directly from HelloOOP.
  - data field speed is private to the class Bugatti
  - private fields or methods cannot be accessed by any method that is not part of the class
- The method getSpeed in class Bugatti can be accessed directly from HelloOOP
  - public method getSpeed is accessible from any method in any class anywhere else in the program

Lecture 3 - Java Fundamentals 1

Access Permission

- Add a new method in class Bugatti within the { and }

```
protected void start() {
}

protected void stop() {
}
```

Lecture 3 - Java Fundamentals 1

Access Permission

- Add a new method in class BugattiVeyron within the { and }

```
public void startAndStop() {
        start();
        stop();
}
```

Lecture 3 - Java Fundamentals 1

Constructor

- Navigate to HelloOOP.java under src
- Double click to open
- Type in the following code within the main method { and }

```
System.out.println("bugattiGrayRed speed is " + bugattiGrayRed.getSpeed());
System.out.println("bugattiTitaniumGray speed is " + bugattiTitaniumGray.getSpeed());
System.out.println("bugattiEbonyGreen speed is " + bugattiEbonyGreen.getSpeed());
bugattiGrayRed.start();
bugattiGrayRed.stop();
bugattiGrayRed.startAndStop();
```

- Compile Errors. Why?

Lecture 3 - Java Fundamentals 1

Access Permission

- Compile Errors. Why?
  - The methods start() and stop() in Bugatti are protected
  - Protected methods can be accessed by methods in any subclass or methods in a class in the same package
  - The class HelloOOP is not a subclass and does not reside in the same package