Tinker Academy

AP Computer Science Prep (Java Programming)
Lecture 3 - Java Fundamentals 1
(Intro. to OOP Part 1)

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
- These objects can call each others “methods”

Lecture 3 - Java Fundamentals 1

Obj1, Obj2 are objects

m1 is Obj1's method
m2 is a Obj2's method

m2 calls m1

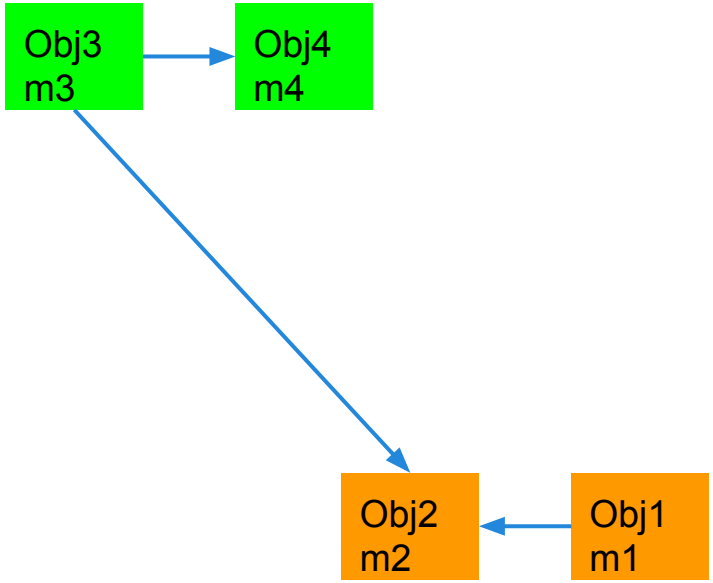
What is an Object?

Lecture 3 - Java Fundamentals 1

So what is an object anyway?

- An object is a central concept in a OOP
- An object can keep references to other objects
- Every object has a unique identity
- Every object is created from a certain “class”
- Every object has some associated data in fields
- Every object provide methods.
- Methods of other objects can invoke these methods.

Lecture 3 - Java Fundamentals 1



Obj1, Obj2, Obj3, Obj4 are objects

m1,m2,m3,m4 are methods

m1 and m3 call m2
m3 also calls m4

What is a Class?

Lecture 3 - Java Fundamentals 1

What is a class?

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

Lecture 3 - Java Fundamentals 1

What is a class?

- In OOP, each class is a small program
- The simplest OOP program is a single class
- Your program is nothing but a bunch of classes created in source files (.java files)
- OOP Languages such as Java provide various classes (4000+).
- These classes form the Java Application Programming Interface or Java API.
- You can also define new classes.

What is an Interface?

Lecture 3 - Java Fundamentals 1

What is an interface?

- Several classes can have common behavior
- This means they share a common methods
- The interface is a common set of methods.
- More than one class can share the same interface

What is a data field?

Lecture 3 - Java Fundamentals 1

What is a data field?

- A placeholder to keep a reference to something useful
- Fields are defined as part of the class “blueprint”
- Objects can optionally keep references using the field

What is a method?

Lecture 3 - Java Fundamentals 1

What is a method?

- A service is a small program
- A method takes inputs, does something useful and produces some output
- Methods are called functions in other languages
- A method is made up of statements which do something useful

Class Activity

Lecture 3 - Java Fundamentals 1

Import Java Project

- Open File Manager
- Navigate to OOP under starterpack2
- Right Click, extract HelloObjects1.zip
- Double click to Open eclipse (under Tinker Apps)
- Right click over Package Explorer
- Import...
- Select extracted folder HelloObjects1

Lecture 3 - Java Fundamentals 1

Create a new Class

- Click on HelloObjects1 in Package Explorer
- File->New->Class
- Type in Car
- Click Finish
- This generates a new Java Source File called Car.java
- Congratulations! you just created a new class
- File->Save All to save your changes

Lecture 3 - Java Fundamentals 1

Create new Objects

- Navigate to HelloObjects1.java under src
- Double click to open
- Type in the following code on line 5,6 between the { and }

```
Car roadster = new Car();
Car bugatti = new Car();
```

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

Lecture 3 - Java Fundamentals 1

Create new Objects

- The code within the main method should look like this

```
public static void main(String[] args) {
        Car roadster = new Car();
        Car bugatti = new Car();
}
```

- The car objects are created from the Car class
- The objects are created using the expression **new Car()**

Lecture 3 - Java Fundamentals 1

Import Java Project

- Open File Manager
- Navigate to OOP under starterpack2
- Right Click, extract HelloObjects2.zip
- Double click to Open eclipse (under Tinker Apps)
- Right click over Package Explorer
- Import...
- Select extracted folder HelloObjects2

Lecture 3 - Java Fundamentals 1

Create new data fields

- Navigate to the Car.java under src
- Double click to open

Type in the following code under // declare fields...

```
private String name;
private int speed;
```

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

Lecture 3 - Java Fundamentals 1

Create new methods

- Right Click in Editor, Source->Generate Getters and Setters…
- Select All, Click OK
- Eclipse will generate 4 methods
  - getSpeed
  - setSpeed
  - getName
  - setName

Lecture 3 - Java Fundamentals 1

Create new methods

- Type in the following code under // declare fields…

```
public void startRace() {
    System.out.println(name + " now racing at " + speed + " mph");
}
```

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

Lecture 3 - Java Fundamentals 1

Add code to invoke methods

- Within the main method in HelloObjects2

```
roadster.setName(“Roadster Razor”);
bugatti.setName(“Bugatti Blazer”);

roaster.setSpeed(125); // electronic controlled speed limit
bugatti.setSpeed(267); // top speed limit

roadster.startRace();
bugatti.startRace();
```

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

Lecture 3 - Java Fundamentals 1

Run the program

- Run HelloObjects2

```
Roaster Razor now racing at 125 mph
Bugatti Blazer now racing at 267 mph
```