Java

90-726 Section A

Lecture Three: Object Oriented Design and Development

January, 1997

Manu Kumar
(sneaker@sneaker.org)

Object Oriented?

- Object Oriented seems to be the buzzword for the 90s
  - But what is it?
  - What does it really do?
  - And is it worth it?
- In this session we will
  - talk about the basics of OO concepts
  - evaluate some of the advantages and disadvantages of using OO
  - take an in-depth tour of how Java exploits OO concepts in its design
  - Put on your Object Oriented seat-belts and lets get started…

Objects

- Any entity which mirrors the existence of a real world entity is an *Object*
  - Examples of Objects:
    - Person, Student, Chair, Desk
  - essentially any entity that exists in real-life and can be mirrored in a software system is an object
- Objects contain
  - attributes (variables)
  - functionality (methods)
- Object can have some properties or be *acted* upon
    - example:
      - a person has a name and social security number
      - a chair can be *sat* on, a desk can be *lift*ed

- A *description* of an *Object* is called a *class*
  - For example
    - A Person is a *class* which may have attributes
      - name
      - social security number
    - and may have functionality
      - eat
      - walk
- But in the previous slide we said a person was an object!?
  - In English a “person” can be an object
  - But objects in Computer Science are a specific occurrence (instance) of a class

Objects vs. Classes

- Person is a class
  - it has attributes
  - it has funtionality
- “Bart” is an *Object* of type Person
  - Bart has attributes:
    - name = Bart
    - ssn = 123-45-6789
  - Bart has functionality
    - eat - Bart eats only spaghetti
    - walk - Bart only walks to class
- Similarly “Lisa” is an *instance of* Person
  - name = Lisa, ssn = 012-34-5678,
  - eat - Lisa eats chocolates

Java

Variables & Methods

- Attributes are stored as *Variables*
  - In our previous example name and ssn were the two variables
- Functionailty is stored in *Methods*
  - In our previous example eat and walk were methods
- Another example:
  - Class Shape
    - variable: color, method: computeArea
  - Object Circle
    - color = red, computeArea = πr^2
  - Oject Rectangle
    - color = blue, computeArea = w*h

OO Buzzwords

- If you claim to know OO programming you should be able to define
  - Encapsulation
  - Inheritance
  - Polymorphism
- Encapsulation
  - Notice what what happened in our previous example
    - Our *Object* Bart had some attributes and some functionality. but all we need to know about Bart is that Bart is a *Person*
    - the information about Bart's name, his SSN and the fact that he can eat and walk (implementation) are hidden from us
    - The Person class could also define another variable called "secret" as an attribute, which need not ever be exposed to the outside world.

Java OO Buzzwords: Inheritance

- Inheritance
  - allows one Class to automatically “assume” the attributes of another class
  - defines an “is a” relationship for classes
- When you think of inheritance, think genetics
  - you have “inherited” some characteristics and behavior from your parents
    - characteristics are “variables”
    - behavior is “methods”
  - However at the same time you are an individual
    - you’ve developed your own characteristics and behaviors
      - modified your parent’s
      - added your own

Java OO Buzzwords: Inheritance

- Example
  - Class Person
    - variables: name, ssn
    - methods: eat, walk
  - Class Student inherits from (extends) Person
    - added variables: courses, grades, gpa
    - added methods: study, party
    - modified methods: walk
      - the implementation for walk may be replaced by running instead of walking
  - Bart is a Student, but Bart is ALSO a Person
  - Student is a *subclass* of Person
  - Person is the *superclass* of Student

Java OO Buzzwords: Polymorphism

- Polymorphism
  - the ability to do different things when called on different objects
- Example:
  - Class Shape
    - variable: color
    - method: area
  - Class Circle inherits from Shape
    - modifies (*overrides*) area to return πr^2
  - Class Rectangle inherits from Shape
    - modifies (overrides) area to return w*h
  - Object c is a Circle, but is also a Shape
  - Object r is a Rectangle, but is also a Shape
    - any call of the type shape.area will use the most restrictive method!

Java OO Buzzwords: Polymorphism

- Example continued
  - c.area will call Circle's method
  - r.area will call Rectangle's method
- More formally:
  - Polymorphism enables an object to send the same message to different receivers (Objects) without knowing how the receiver (Object) will implement the message.
- Do not confuse with same method with different parameter types
  - Class Student
    - method: eat (Pizza pizza)
    - method eat (NotPizza notPizza)

Invoking methods

- Methods are *invoked* by
  - objectName.methodName(parameter1, parameter2...)
    - objectName is an instance of a particular class
    - bart.eat(pizza)
- However!
  - Some times it makes sense to have a method on the “Class” rather than on the “Object”
    - these are called *static* methods
    - Static methods apply to the className
      - className.methodName(parameter1, parameter2…)
    - Static methods are used for functionality which applies to the type of the object rather than each instance of the object.
    - Static methods are useful since they can be called without instantiating an object of the class.

OO Design (1)

- Why *design* software?
  - Why do you design a building on paper before building it in concrete?
    - To make sure it won't come crumbling down!
    - To make sure the doors and windows fit and are the right size
  - Software which is designed has a much better chance of working right…
- What is OO Design?
  - OO Design is one of the most popular design methodologies for software
  - In OO design, you start by analyzing the real-world entities which exist in the environment
  - then add in the attributes and behavior for each of those entities

OO Design (2)

- Steps in OO design
  - Map real-world entities into Classes and Objects
  - Establish relationships between classes
    - Student inherits from person
  - Analyze all *actions* one object can perform on another object
    - create methods for these actions
  - Build wrapper around the objects to make hem interact

Java

Java and OO Concepts

- Everything in Java is a class
  - you are dealing with classes and objects
  - variables and methods
  - encapsulation, inheritance and polymorphism
- Especially when we get to writing applets and GUIs
- Java allows you to use OO concepts
  - you can still write spaghetti code in Java
  - but you will learn how to truly exploit the power of OO with experience

Java Packages

- A collection of related classes in Java can be bundled together in *Packages*
- For example
  - java.net contains all network related classes
  - java.awt contains all AWT (GUI) related classes
  - java.io contains all input output related classes

Java

Information Hiding in Java

- Java provides three levels of information hiding
  - public
  - protected
  - private
- Classes, variables and methods can all be preceded by one of the above keywords
- Public:
  - visible to ALL
- Protected:
  - visible to only the subclasses and classes within this package
- Private:
  - visible only to this class

An Idiom explained

- Remember this
  - public static void main(String args)
- first, it does not make sense to have every “object” have a main method
  - therefore it is defined as static
- the method is public so that it can be called from outside this class and outside this package

Java

So far in this lecture

- We have had a very cursory overview of OO principles
- In the next lecture we will
  - go hands on and do some OO design
  - do some OO development in class
- Any suggestions on a programming problem that you would like to see addressed?
  - send email

Java

Status update

- Assignment #1 is due soon
  - bring print out to class
- Quiz #2 will be simple
  - and cover some topics discussed in class today and some things from the readings
  - 10-15 minutes
- No more scheduled Quizzes
  - you should have enough fundamentals to move further
- Assignment #2 will be announced