Core Java

2. Basics of Object-Oriented Programming

Procedural Programming

- Simple Structure
- A program to add 2 numbers, written in *C*
- Functions can provide in-code reusability
- Software are programs which are products
- Not very great for large pieces of software
- Key issue: maintainability

Modular Programming

- More Complex Structure
- *Allegro Game Programming Library*, *Yasm Modular Assembler*, *Winamp*, *Linux Kernel*
- Application Binary Interfaces provides reusability and extensibility
- In-source or Runtime
- Key issue: Typing, Boilerplate Code

Object-Oriented Programming

- Everything is an object
- Make references to the real world
- Each Object has a lifetime
- A program is a set of objects which pass messages between each other and keep changing during their lifetime
- The analogy is life: living beings interacting with each other, changing states

Properties of Objects

- Objects have attributes: Animals have hearts, brains, ears
- Objects have behaviors: Human beings walk, think, talk
- Objects with similar behaviors can be grouped together
- A tall man is a man, A short man is a man

Inheritance

- Objects with similar behaviors can be further grouped together
- A man is an animal, A cow is an animal
- An animal is a living being, A plant is a living being
- This is often referred to as inheritance

Encapsulation

- An animal is a group of animal cells
- A switchbox is a set of switches, wires, fuses, regulators, LEDs
- A capsule contains tiny bits of medicines in it
- This abstraction of objects is often referred to as encapsulation
- An object can contain objects: Composites

Polymorphism

- For a particular behavior, an object can interact differently with different objects
- Analogy of a man talking to an instructor and colleagues
- A very useful feature which allows a variety of behaviors to be grouped into one

Okay, So What Now?

- When we write code in Java, we write objects
- We write objects with coherent, cohesive functionality
- Each object should ideally do only a few things
- Keep it simple

Examples of Objects

- Main Program: *HelloWorld*, *sun.tools.javac.Main*
- Processors: *java.lang.Thread*
- Entities: *Student*, *Manager*, *Juice*
- Virtual Entities: *Account*, *java.util.Date*
- Collections of Entities: *java.util.ArrayList*, *javax.swing.JPanel*
- Meta-Objects: *java.lang.Class*, *java.lang.Interface*

Classes

- Classes are ways to logically group a set of Objects
- Objects can be thought of as instances of Classes
- I am an instance of man and You are too!
- Sometimes, classes need not have instances, or a shared instance
- Our *HelloWorld* Program is such an example

Questions?