# CSCI 253

Object Oriented Design:
Object Oriented Design and Programming in Java
George Blankenship



## Outline

- Background
  - Rationale for the course
  - Why object oriented programming?
- OOP principles
  - Objects
  - Object oriented design
- Java



## Background

- Computer Science is not (just) programming
- Java is a tool to teach programming and problem solving concepts
- So if you know everything in Java you still need to stay awake
- Will be introducing many advanced topics along with the background material

## Rationale

- Programming is based up the use of a language just as any other human to exchange information between two intelligent entities
- The exchange must reflect an accurate intention of the source is a form that is intelligible and understandable by the recipient



## Study of Grammar

- Man has learned that the grammar of a language must be studied as a separate discipline
- Man has also learned that selected constructs of a language convey information more effectively than other constructs
- These two observations are the basis of writing classes in any language



## Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An object-oriented approach lends itself to this kind of solution decomposition
- We will dissect our solutions into pieces called objects and classes

## Object-Oriented Programming

- Objects can be used effectively to represent real-world entities
- We try to define all our data as objects, and define programs to work on those objects
- For instance, an object might represent a particular employee in a company
- Each employee object handles the processing and data management related to that employee



## Objects

- An object has:
  - *state* - descriptive characteristics
  - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state



## Reusability

- Object oriented design encourages the design of reusable components
- Vehicle is a general definition
  - A base object defines common attributes and behaviors
  - The definition is reused to define unique objects that have the common attributes and behaviors
- Mini-van as a more specific object

```
Public class miniVan {
        String manufacturer;
        String model;
        int year;
        Color color;
}
```

## Classes

- A class represents a concept, and an object represents the embodiment of that concept
- An object is defined by a *class* (blueprint)
- Multiple objects can be created from the same class
- Attributes define the state of the object
- Methods are the access point to effect the behavior of the object



## Attributes

- Internal to the class
  - Arguments for/against to make the attributes visible outside the class
- Define the state of an instantiation
- State changes are made when an instantiation receives a message
  - If attributes are hidden, messages can only be received via externalized methods



## Methods

- Define behavioral actions
- Action invoked by the receipt of a message
- The externalized methods of a class define the API of the class
- The parameters of an invocation are considered to be a message defining the details of the desired behavior.

## Instantiation

- Defined classes are used to create execution time object (instantiation of the class)
- The constructor method is responsible for initializing the object
- *new* creates an instance of the class
- The execution time name of the object is the handle to reference the object
  - name.attribute is a reference to an attribute
  - name. method(<message>) is an invocation of a method passing the message



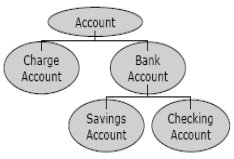
## Example

- Class Test
  - methods: Test, behaviorOne(parm1, parm2), behaviorTwo(parm1,parm3)
- Instantiation of an object
  - Test testing = new Test();
- Referencing the object
  - testing.behaviorOne(7,"testing");
  - testing.behaviorTwo(37,59);



## Inheritance

- One class can be used to derive another via *inheritance*
- Classes can be organized into hierarchies



## Encapsulation

- Attributes of the class are objects from other classes
- The methods of the class create a new API hiding the encapsulated objects' API
- Class MainGUI encapsulates the complex AWT classes
- MainGUI API presents append(), addMenu(), addMenuItem(), message(), getClock()



## Object Oriented Design

- Problem space broken into distinct components (classes)
- Each component solves a portion of the problem
- Communication protocol validation (SerialLink)
  - AnswerConnection
  - DialConnection
  - EchoConnection
  - ConfigurationPanel
  - SerialConnection



## Design Consideration

- Creation of objects
  - A class that just creates the objects used in the solution
  - Standardizes the creation of objects
- Structure of solution components
  - Templates for the organization of the solution objects to simplify, clarify and standardize the organization
- Behavior of the solution components
  - Templates for the behavioral properties of the solution objects to simplify, clarify and standardize the behavior

## Java Language

- A programming language specifies the words and symbols that we can use to write a program
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid program statements
- The Java programming language was created by Sun Microsystems, Inc. and introduced in 1995.



## Java Translation

- The Java compiler translates Java source code into a special representation called byte code
- Java byte code is not the machine language for any traditional CPU
- Another software tool, called an interpreter, translates Byte code into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine
- Java is considered to be architecture-neutral



## Java Code

- Basic unit of Java code is the object
  - Software bundle of related state and behavior
  - Programming problem is mapped into the interactions of objects
- Objectives
  - Modularity
  - Information-hiding
  - Code re-use
  - Pluggability and debugging ease
- Fundamental items of an object
  - Attributes (properties, fields) – object state
  - Methods - modification of object state

## API Documents

- Unlike other languages, Java has many libraries bundled by default
- Application Programming Interface (API) docs, are the view given to the programmer
- Encouragement of code re-use
- Example:
  - java.lang.String
  - java.util.StringTokenizer