Objects

- The term object is not easily defined
- According to Webster:
  - Object: a visible or tangible thing of relative stable form; A thing that may be apprehended intellectually; A thing to which thought or action is directed
- In this class, we will use the following definition:
  - An object has state, behavior, and identity (Booch)

State

- The state of an object encompasses all of the (static) properties of the object plus the current (dynamic) values of each of these properties
- A property is an inherent or distinctive characteristic, trait, quality, or feature that contribute to making an object uniquely that object
- We will use the word *attribute*, or *data member*, to refer to the state of an object
Examples

• Properties
  – Elevators travel up or down
  – Vending machines accept coins
  – Clocks indicate the current time
• Values
  – Current floor
  – Number of coins deposited
  – The number of minutes since the last hour

Behavior

• Behavior is how an object acts and reacts, in terms of state changes and interactions with other objects.
• An operation is some action that one object performs upon another in order to elicit a reaction.
• We will use the word method to describe object behavior in java.
• Invoking a method causes the behavior to take place.
Types of Methods

• There are 4 basic types of methods:
  – Modifier (sometimes called a mutator)
    • Changes the value associated with an attribute of the object
  – Accessor
    • Returns the value associated with an attribute of the object
  – Constructor
    • Called once when the object is created (before any other method will be invoked)
  – Destructor
    • Called when the object is destroyed

Identity

• Is a river still the same river from one day to the next, even if the same water never flows through it?
Identity

• Identity is the property of an object that distinguishes it from all other objects.
• The failure to recognize the difference between the name of the object and the object itself is the source of many errors in object-oriented (OO) programming.

Assignment and Equality

• What does it mean to assign one object to another?
  – Copy the name only (*shallow copy*)
  – Duplicate the object, creating a different object (with a different name) whose state and behavior is the same as the original (*deep copy*)
• Equality like assignment, can mean two things
  – Two names designate the same object
  – Two objects are different but their state and behavior are the same
Relationships

• Objects contribute to the behavior of a system by collaborating with one another
  – An airplane is a collection of parts having an inherent tendency to fall to the earth, and requiring constant effort and supervision to stave off that outcome

• The relationship between any two objects encompasses the assumptions that each makes about the other, including what operations can be performed and what behavior results

Relationships

• There are two kinds of relationships that are of particular interest
  – Using relationship
  – Containing relationship
Class

• According to Webster
  – A group, set, or kind marked by common attributes or a common attribute

• A class is a set of objects that share a common structure and a common behavior
  – An object is a concrete entity that exists in space and time, an instance of a class
  – A class represents only an abstraction, the essence of an object from the class

Class

• A class can be thought of as a cookie cutter, form which objects can be instantiated
  – A class is to an object, as a blueprint is to a building

• The class mammal represents the characteristics common to all mammals
  – Live birth, nurse young, have hair, …

• “Paul”, “PJ”, “Lisa”, “Heidi”, and “James” are specific instances from the class mammal
Class Relationships

• Consider for a moment the following classes
  – Flowers
  – Daisies
  – Red roses
  – Yellow roses
  – Petals
• What observations can you make?

Kinds of Relationships

• Classes, like objects, do not exist in isolation.
• Relations between classes can be made for one of two reasons.
  – To indicate some sort of sharing.
    • A yellow rose and a red rose are both roses and have petals, roots, leaves, thorns, etc.
  – Some kind of semantic connection.
    • Daisies and roses are both flowers that are pollinated in the same way.
Kinds of Class Relationships

- There are three basic kinds of class relationships
  - Generalization ("kind of")
    - A rose is a kind-of a flower
    - Generalization provides the ability to create subclasses
    - Subclasses share the structure of the parent class
  - Aggregation ("part of")
    - A petal is part-of a rose
    - Aggregation allows one to construct new classes from existing one
  - Association

Inheritance

- The term, inheritance, is used in many object oriented (OO) programming languages to describe the generalization relationship
- Inheritance is a relationship where one class shares the structure or behavior defined in one class (single inheritance) or more (multiple inheritance)
Types of Inheritance

<table>
<thead>
<tr>
<th>Form of Inheritance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>The superclass defines behavior that is implemented in the subclass but not in the superclass. Provides a way to guarantee that subclass implement the same behavior.</td>
</tr>
<tr>
<td>Specialization</td>
<td>The subclass is a specialized form of the superclass but satisfies the specifications of the parent class in all relevant aspects.</td>
</tr>
<tr>
<td>Extension</td>
<td>The subclass adds new functionality to the parent class, but does not change any inherited behavior.</td>
</tr>
<tr>
<td>Limitation</td>
<td>The subclass restricts the use of some of the behavior inherited from the superclass.</td>
</tr>
<tr>
<td>Combination</td>
<td>The subclass inherits features from more than one superclass (i.e. multiple inheritance).</td>
</tr>
</tbody>
</table>

Benefits of Inheritance

- One view of inheritance is that it provides a way to specify some properties/behaviors that all subclasses must exhibit
- Inheritance can be used to re-use code
- Inheritance also provides the ability to generalize
  - A method can be written to work with the super-class but subclasses can be passed as arguments
Views of a Class

- A class can be viewed as a sort of contract that specifies what instances of the class can, and cannot do
- It is possible to distinguish between the outside and inside view of a class
- The interface of a class provides its outside view and emphasizes the abstraction
- The implementation of a class is its inside view

Access

- Most classes provide three levels of access to their members (state and behavior):
  - Public
    - The part of the class of the class that is visible to all clients of the class
  - Protected
    - The part of the class that is only visible to subclasses of the class
  - Private
    - A part of the class that is not visible to any other classes