Objects and Classes

Amirishetty Anjan Kumar

Computer Science and Engineering
Indian Institue of Technology
Bombay

November 27, 2004

What is Object Oriented Programming?

- Identifying objects and assigning responsibilities to these objects.
- Objects communicate to other objects by sending messages.
- Messages are received by the methods of an object

The two steps of Object Oriented Programming

- **Making Classes:** Creating, extending or reusing abstract data types.
- **Making Objects interact:** Creating objects from abstract data types and defining their relationships.

What are Software Objects?

- Building blocks of software systems
  - program is a collection of interacting objects
  - objects cooperate to complete a task
  - to do this, they communicate by sending messages to each other
- Object model **tangible** things
  - school
  - car
- Objects model **conceptual** things
  - meeting
  - date

What are Software Objects? (continued..)

- Objects model **processes**
  - finding path through a maze
  - sorting a deck of cards
- Objects have
  - capabilities: what they can do, how they behave
  - properties: features that describe the objects

Object Capabilities: Actions

- Objects have capabilities that allow them to perform specific actions
  - objects are smartthey know how to do things
  - an object gets something done only if some object tells it to use one of its capabilities
- *Capabilities can be:*
  - **constructors:** establish initial state of objects properties
  - **commands:** change objects properties
  - **queries:** provide answers based on objects properties

Object Properties: State

- Properties determine how an object acts
  - some properties may be constant, others variable
  - properties themselves are objects also can receive messages
  - trash cans lid and trash are objects
- Properties can be:
  - *attributes:* things that help describe object
  - *components:* things that are part of an object
  - *associations:* things it knows about, but are not parts
- **State:** collection of all objects properties

Object Instances

- Object instances are individual objects
  - made from class template
  - one class may represent an indefinite number of object instances
  - making an object instance is called instantiating that object
- Shorthand:
  - class: object class
  - instance: object instance (not to be confused with instance variable)

Object Instances (continued)

- Individual instances have individual identities
  - allows other objects to send messages to given object
  - each is unique, even though it has same capabilities
  - think of class of CS15 students
- A **reference** is just the address in memory where its instance is stored
  - also called pointer

Memory Revealed

- Every instance is stored in computers memory
  - memory is a set of consecutively numbered storage locations, each containing a byte
  - instance is stored in a series of contiguous bytes starting at a given location
- Instance is identified and referenced by unique address of its starting location when it is made
  - address looks like 0xeff8a9f4 (hexadecimal notation, base 16)
  - just like postal address represents actual home

Messages for Object Communication

- No instance is an island must communicate with others to accomplish task
  - properties allow them to know about other objects
- Instances send messages to one another to invoke a capability (i.e., to execute a task)
  - method is code that implements message
  - we say call a method instead of invoke capability
- Each message requires:
  - sender: object initiating action
  - receiver: instance whose method is being called
  - message name: name of method being called
  - optionally parameters: extra info needed by method to operate

Encapsulation

- Car **encapsulates** lots of information quite literally, under hood and behind dashboard
- So, you do not need to know how a car works just to use it
  - steering wheel and gear shift are the interface
  - engine, transmission, driver train, wheels, . . . , are the (hidden) implementation
- Likewise, you do not need to know how an object works to send messages to it
- But, you do need to know what messages it understands (i.e., what its capabilities are)
  - class of instance determines what messages can be sent to it

Views of a Class

- Objects separate interface from implementation
  - object is black box; hiding internal workings and parts
  - interface protects implementation from misuse
- Interface: **public view**
  - allows instances to cooperate with one another without knowing too many details
  - like a contract: consists of list of capabilities and documentation for how they would be used
- Implementation: **private view**
  - properties that help capabilities complete their tasks

Notes About Java Syntax

- **Reserved words**
  - certain words in Java have a particular meaning and cannot be used for any other purpose case-sensitive (always all lower case)
  - *class, public, new, private, extends*
- **dentifiers**
  - names used for classes, methods, and variables
  - first character must be a letter or underscore
  - rest may be any combination of letters, numbers, and underscores but no spaces

Constructors

- Next we need to have instances of our class to do something useful
- **Constructor** is special method that is called whenever class is instantiated (created)
  - another object sends a message that calls constructor
  - constructor is first message an object receives and cannot be called subsequently
  - establishes initial state of properties for instance
- If you do not define any constructors for class, Java writes one for you
  - called **default** constructor
  - initializes all instance variables for instance to their default values
  - default values should not be relied upon
    - ALWAYS write your own constructor for each class
    - ALWAYS give each instance variable an initial value in this constructor

Constructors(continued..)

- A constructor is always called when an object is created.
- We can define our own constructors (Note: a class can have more than one constructor).
- If an object is copied from another object then the copy constructor is called.

Multiple Constructors

- Sometimes want to initialize in a number of different ways, depending on circumstance.
- This can be supported by having multiple constructors having different input arguments.

Method Overloading

- Constructors all have the same name.
- Methods are distinguished by their signature:
  - name
  - number of arguments
  - type of arguments
  - position of arguments
- That means, a class can also have multiple methods with the same name : **method overloading.** This is a form of **polymorphism**.

Polymorphism

- Allows a single **method or operator** associated with different meaning depending on the type of data passed to it. It can be realised through:
  Method Overloading
- Defining the same method with different argument types (method overloading) - polymorphism.
- The method body can have different logic depending on the date type of arguments.

Automatic garbage collection

- The object does not have a reference and cannot be used in future.
- The object becomes a candidate for automatic garbage collection.
- Java automatically collects garbage periodically and releases the memory used to be used in the future. Q

Object Instantiation

- **Object instantiation** makes an object instance from a particular class
  - allows other instances to send messages to instance
  - constructor is first message: makes the instance

Summary

- A class is a blueprint for an object.
- Objects are created similar to other data types (int, char,).
- The construction of an object can be defined by the user.
- Messages are sent to an object by calling a method.