Java Classes
&
Object Oriented Programming

Introduction

Object Oriented Programming

- One of the first applications of modern computing was modeling and simulation.
- Scientists soon realized that functions alone were insufficient to model systems intuitively
- If we are going to model a planet we would like to actually create a virtual planet, define how it behaves in our simulated universe, and then just observe it.

Object Oriented Programming

- Programmers quickly realized that the idea of creating virtual “things” made software engineering simpler to think about.
- If we create within our programs agents and objects then we can assign duties and tasks to them.
- This is really just another way applying decomposition to our software.
- Break up the problem to be solved into logical parts and assign each part to an object.

Object Oriented Programming

- Even engineers are social animals - we evolved to think about the world in terms of agents and objects (not recursion).
- In many situations we solve large problems by delegation. That is we have workers who specialize in solving a particular problem.
- Those specialists have specific skills that they can apply to a specific class of problems.

Object Oriented Programming

- We can pattern software after a group of specialists at a company working on a problem.
- For example, there are two objects we have used – `System.out` and `System.in`.
- `System.in` is the name of an object who knows all about reading data from the keyboard and putting it into a variable.
- It is easier to ask `System.out` to do the work than write a program to do it ourselves.

Object Oriented Programming

- Important: we don't have to have any idea *how* `System.out` does its job. We just trust that it does.
- Just like we don't question the US Mail about how our letter gets from here to Seattle.
- We only care that it arrives within certain tolerances – not how it got there.
- This is called *abstraction*, *information-hiding*, and *encapsulation* and we like it!

Object Oriented Programming

- When we mail a letter all we have to worry about is following the correct protocol to ensure our letter gets to the right place.
- We have to know where to go, how to pay, the format expected for the destination address and return address, etc.
- In software this protocol is called the interface.
- All objects have to have an interface that clearly defines how we can interact with the object.

Object Oriented Programming

- Almost any problem can be broken up into objects.
- Objects are defined by three things:
  - Their *state* – this is the information they contain.
  - Their behavior or capabilities – these are the functions they have access to.
  - Their *interface* – the rules describing how they interact with other objects in the system.

Object Oriented Programming

- Programmer thinks about and defines the attributes and behavior of objects.
- Often the objects are modeled after real-world entities.
- Very different approach than *function-based* programming (like C, C++, Fortran, Lisp,...).

Reasons for OOP

Abstraction

Encapsulation

Information hiding

Inheritance

Software Engineering Issues

Class: Object Types

- Java uses *classes* and *structures* to define objects
- A Java *class* is an object type.
- When you create the definition of a class you are defining the attributes and behavior of a new type.
  - Attributes are data members.
  - Behavior is defined by methods.

Creating an object

- The interface acts as a contract specifying how the object will behave – as long as the code fulfills the contract we don't care how it works.
- Defining a class does not result in creation of an object.
- Declaring a variable of a class type creates an object. You can have many variables of the same type (class).

This is called *instantiation* of the class, i.e. we create an *instance* of the object.

Information Hiding

- The *interface* to a class is the list of public data members and methods.
- The interface defines the behavior of the class to the *outside world* (to other classes and functions that may access variables of your class type).
- The implementation (the code that makes the class work) doesn't matter outside the class.

Information Hiding (cont.)

- This is good because it allows us to change the underlying code without forcing everyone who uses our objects to change their code.

- You can change the implementation and nobody cares! (as long as the interface is the same).

Private vs. Public

- Classes define certain parts of the object to be public, private, or protected.
- *Public* parts of the object can be used by anyone who has access to the object.
- The *private* parts of the object are for the objects internal use only.
- *Protected* parts are accessible from outside the object only under certain circumstances.
- Try to make as much private as possible.

Special Member Functions

- Constructors: called when a new object is created (instantiated).
  - can be many constructors, each can take different arguments
- Garbage Collection. This is why Java is so popular.

Anatomy of a Class

```
public class Dog
{
    Dog( String dog_name ){
       name = dog_name;
    }
    public void bark(){
        System.out.println("woof");
    }
   public string getName() { return name }
   private String name;
}
```

Put all this in Dog.java

Using a Class and an Obeject

```
public static void main(String[] args){
    String my_dogs_name = “Fido”;

    // Create object of type “Dog”
    Dog mydog = new Dog( my_dogs_name );
    // Access data and call methods in “mydog”
    System.out.println(
                    mydog.getName()
                    + “: ”
                    + mydog.bark() );
    return 0;
}
```

Accessing Data Members

- Data members are available within each method (as if they were local variables).
- Public data members can be accessed by other functions using the member access operator ".".

Accessing class methods

- Within other class methods, a method can be called just like a function.
- Outside the class, public methods can be called only when referencing an object of the class.

Classes and Files

Each class definition goes in it's own .java file.

Give the file the same name as the class.

Java can automatically find the class definition this way.

Classes and Objects

- It is essential to understand the difference between a class and the corresponding object.
- A class defines the properties of the object (methods, data members, name).
- When you use the "new" keyword you instantiate the class.
- This means you tell java to reserve some memory in the computer to store data required by the class.

Creating Objects

Class Definition:
"Dog.java"

Code to create object

Object

Class:
Name: Dog

Data Member:
String name

Methods:
String getName(),
void bark()

Dog pet1 = new Dog("Fido")

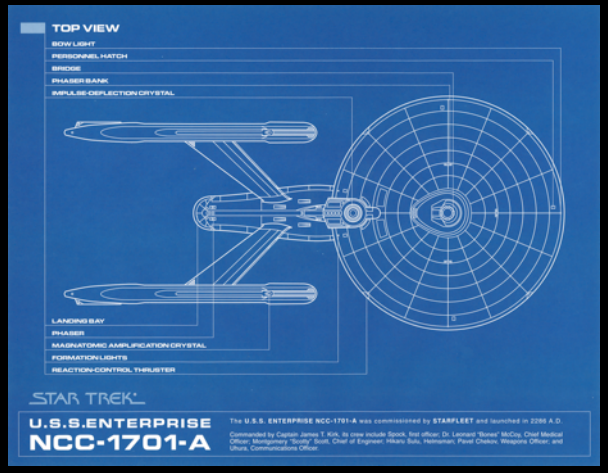
Object:
Name is "Fido"

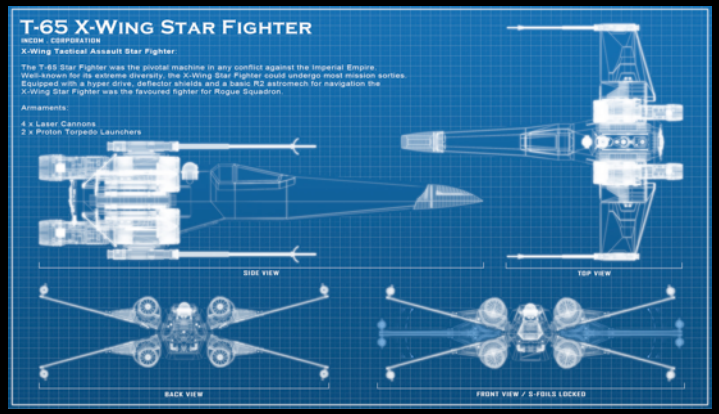
Dog petA = new Dog("Rover")

Object:
Name is "Rover"

Dog dog1 = new Dog("Hick")

Object:
Name is "Hick"





- Imagine when you are writing a class that it is a blueprint.
- Instantiating a class is building the object described by the blueprint.