

Object-Oriented Concepts

ISE 208: Intermediate Programming

SUNY at Stony Brook

Modeling Information

- *Model*: simplified description
 - Used to make decisions, make predictions, and simulate processes
- Models are an *abstraction*
 - They reduce complexity by only considering relevant data



Models and Software

- Models are used to maintain information and answer queries about something
- We can implement models using software
- An *object* is a block of code that represents one element of a model
 - Objects track information about some part of the problem

Object-Oriented Programming

- An object-oriented program uses objects to model the **problem domain** (the things we have to work with)
 - Objects are constructed from data and program code
- Topics to consider:
 - Designing objects
 - Implementing objects
 - Advanced concepts: access modifiers and `static`

What is an Object, Anyway?

- Objects are a very natural way to view the world around us
 - Ex. People, things, even non-physical stuff (i.e., a bank account)
- Objects are defined in terms of **attributes** and **behaviors**
 - Attribute: properties that an object has (e.g., eye color, height, weight)
 - Behavior: actions that an object can perform (e.g., walk, speak, pick up)
- An object is an entity that contains **both** data **and** behavior

Object Queries

- Query: question to which an object can respond
- Queries are always directed to a specific object, and answered by that object
 - Answers are based on the object's attributes
 - Answers can take any form: T/F, integer, etc.
- An object can only respond to pre-programmed queries

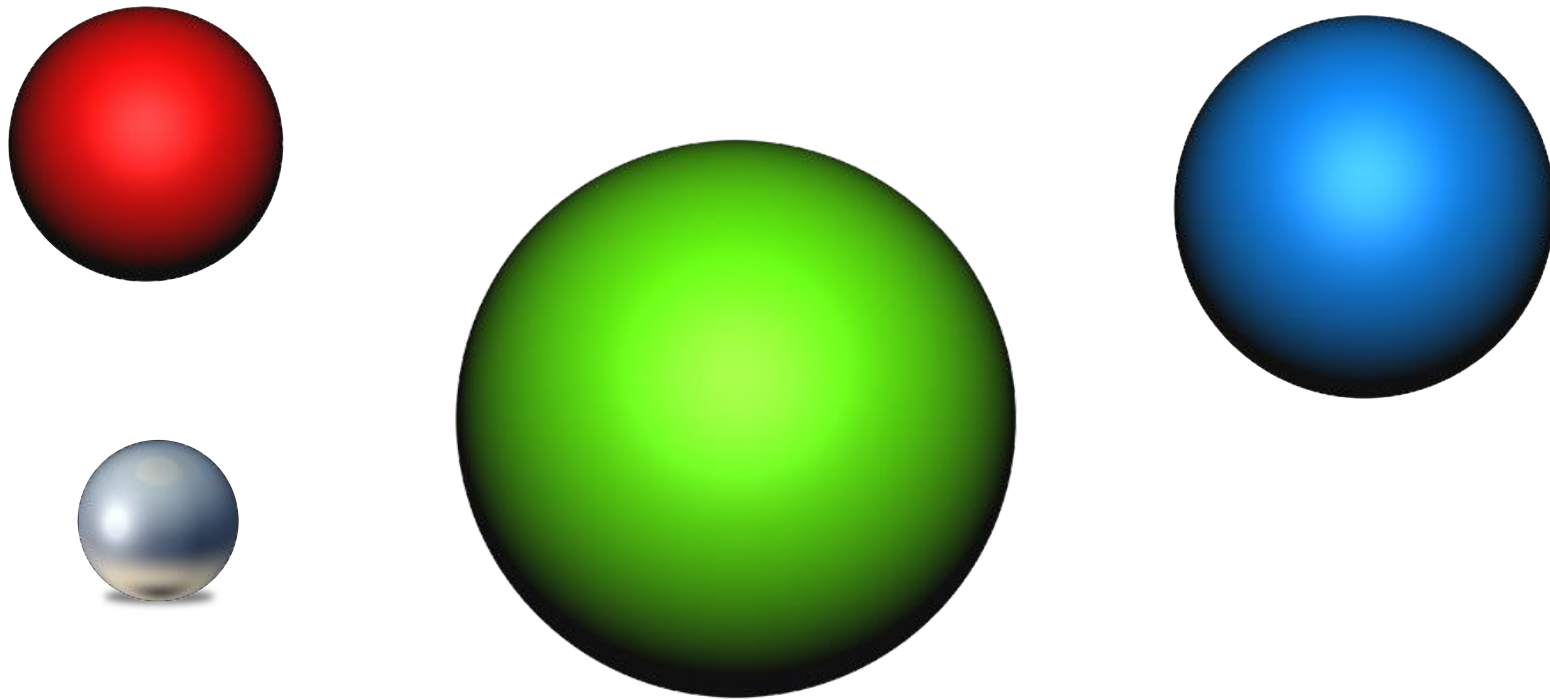
Objects Are Self-Managing

- An object contains all of the functions (methods, representing “behavior”) that need to operate on its data (“attributes”)
- This is called **encapsulation**, which means that an object is responsible for managing its own data
- An object may hide its contents from the rest of the program
- This protects our data from mischief and can conceal how we represent and manipulate data inside an object

Classes and Objects

- Every object is defined by a **class**
 - A class is a template for objects of that type — it describes the types of data and the behaviors that each object of that type has
 - If two objects have identical behaviors and attribute types, they belong to the same class
- A specific object is called an **instance** of its class
 - e.g., you and your neighbor are instances of the Student class (same kinds of data, different actual data)

Classes and Instances



Each ball has has the same shape, but a unique size and color

The Anatomy of a Class

- A class contains two types of elements:
 - Instance data — variables representing an object's attributes
 - Instance methods — methods (code) that define an object's behaviors
- Each class is defined using the Java keyword “class”
- Curly braces enclose the data and method declarations that make up the class

Class Definition Example

```
class Example // defines a new data type called "Example"  
{  
    instance variable declarations go here...  
  
    instance method definitions go here...  
}
```

- Normally, every class is defined in a separate source file named for the class
 - If this is done, the keyword “public” must be written before “class”
- If several classes are defined in the same file, exactly one must be declared “public” and share the name of the file.

Working With Objects

- Before we can use an object in our program, we must create a new *instance* of that object
- We need to create a *variable* that represents that object
- After we create the object, we can send messages to it
 - These messages tell the object to do various things
 - Every object has its own set of possible actions

Variables

- Variables are the “nouns” of a program
 - They represent pieces of information
- Different types for different kinds of data:
 - int stores integers (whole numbers)
 - float and double store decimal values
 - String stores character sequences
 - We can also create variables for objects

Declarations

- Java requires you to declare a variable before you use it
 - This tells Java what behavior to expect from that variable
- Declare a variable (tell the compiler that you're going to use it) by writing the type, followed by the name:

`int x; // Declares an integer variable x`

`double pi; // pi holds a fractional number`

- A variable's name can contain any combination of letters and digits, but must begin with a letter

Storing Values in Variables

- Use a *single* '=' to store a value in a variable
 - “=” translates to “is assigned the value”
 - The left gets the value on the right side:

```
x = 5; // Stores the integer value 5 in x
```
- We can combine assignments with declarations:

```
int value = 75;
```

Creating An Object

- To create (instantiate) an object:
 - Declare a new *variable* to hold the object
 - Use the keyword **new** to create the object:

```
MyClass foo = new MyClass( );
```

- “new” invokes an object’s *constructor*
- A constructor is a special behavior (method) that creates a new object of a specific type
 - It has the same name as the type

Speaking To Objects

- A *message* tells an object to do something
- A message consists of the object's name, followed by a "dot" (period), followed by the behavior name and argument list
- Messages are terminated by a semicolon



Introduction to BlueJ

What is BlueJ?

- BlueJ is a simple IDE (integrated development environment) designed for new programmers
 - An IDE combines a source code editor and the compiler
- BlueJ is written in Java, and runs on all platforms
- You can download BlueJ for free from

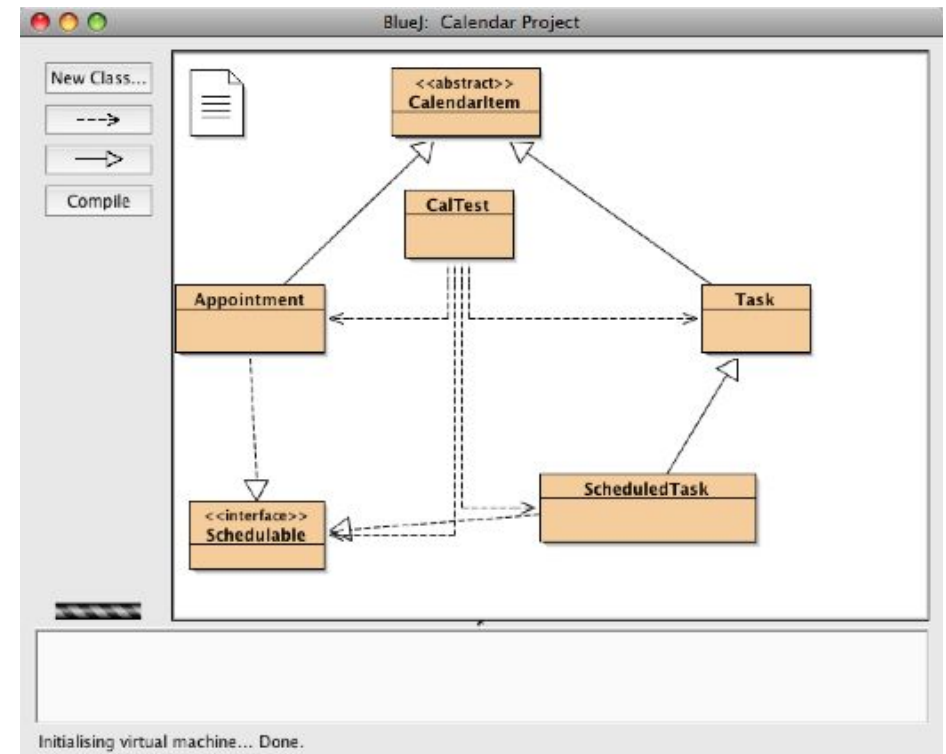
<http://www.bluej.org>

Projects

- A *project* is the set of files that make up a program
- Before you can do *anything* in BlueJ, you must create a new project (or open an existing one)
- BlueJ creates a new folder on disk with the same name as the project
- A BlueJ project contains all of your .java and .class files, along with a few special BlueJ-specific files

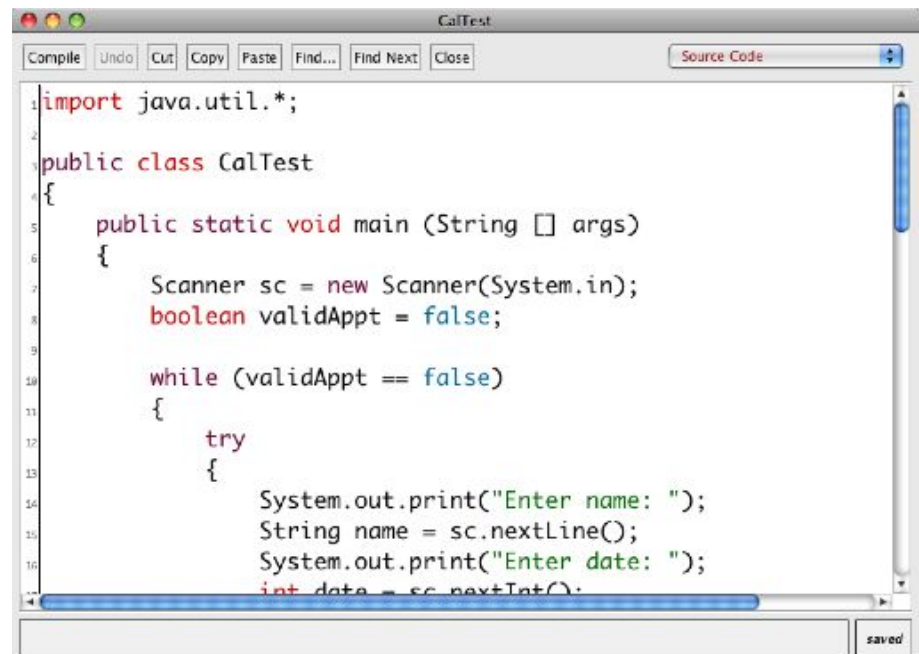
The Main BlueJ Window

- The project window has two main sections:
 - The top area shows all of the classes in your current project (arrows indicate relationships)
 - The object workbench is at the bottom



Editing Source Code

- To open a source code file for editing, double-click on the box for that class
- BlueJ includes a basic text editor
- BlueJ has auto-indent and syntax coloring



```
import java.util.*;

public class CalTest
{
    public static void main (String [] args)
    {
        Scanner sc = new Scanner(System.in);
        boolean validAppt = false;

        while (validAppt == false)
        {
            try
            {
                System.out.print("Enter name: ");
                String name = sc.nextLine();
                System.out.print("Enter date: ");
                int date = sc.nextInt();
            }
            catch (Exception e)
            {
                System.out.println("Invalid input");
            }
        }
    }
}
```

Executing Your Program

- To execute your program, right-click on the class in the main window
- Select the main() method from the menu that appears
- Unlike most IDEs, BlueJ actually allows you to start execution from any method in your program
- This lets you test a specific method without having to run your entire program

The Object Workbench

- BlueJ allows you to work with your classes without actually running your entire program
- Right-click on a class and select “New...” to create a new instance of that class
 - Instances appear in the object workbench at the bottom
- Right-click on an object to call its methods
- Double-click on an object to inspect its data

Next Time

- Defining methods
- More on variables