Classes

CSE 142, Summer 2002

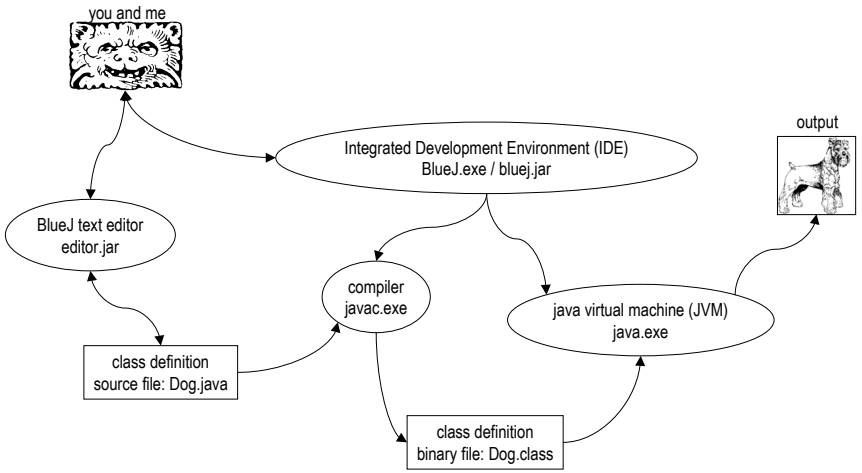
Computer Programming 1

http://www.cs.washington.edu/education/courses/142/02su/

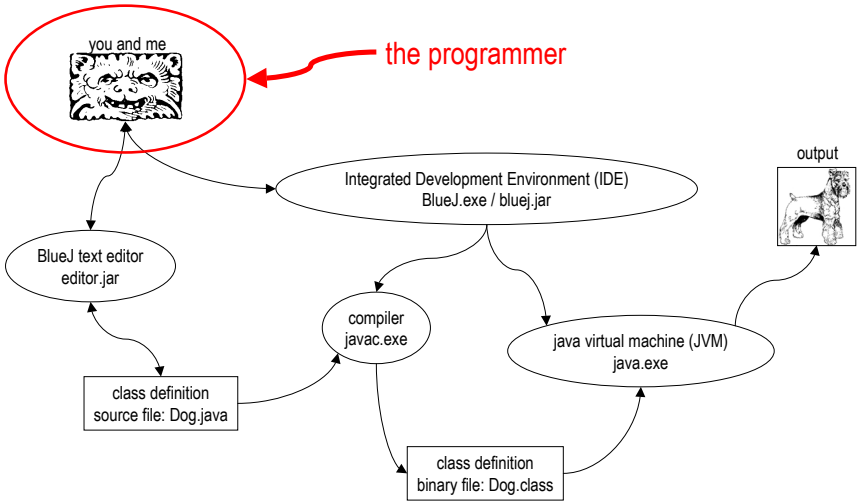
Readings and References

- Reading
  - Chapter 3, *An Introduction to Programming and Object Oriented Design using Java*, by Niño and Hosch
  - BlueJ Tutorial
- Other References

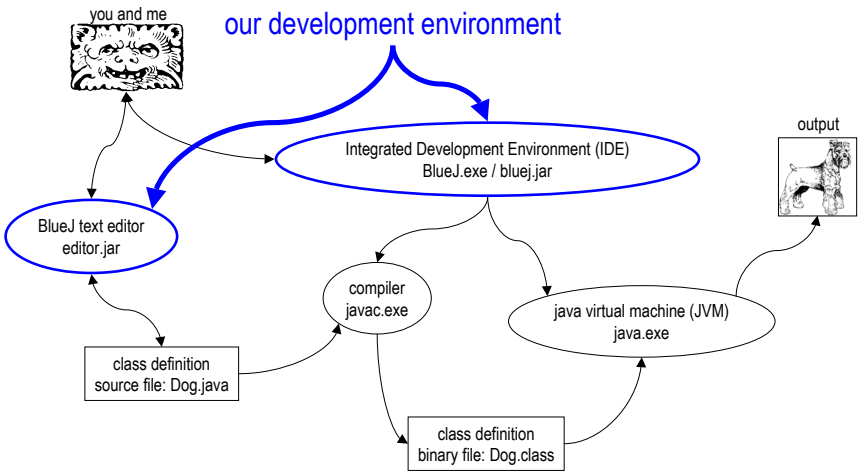
Our Environment



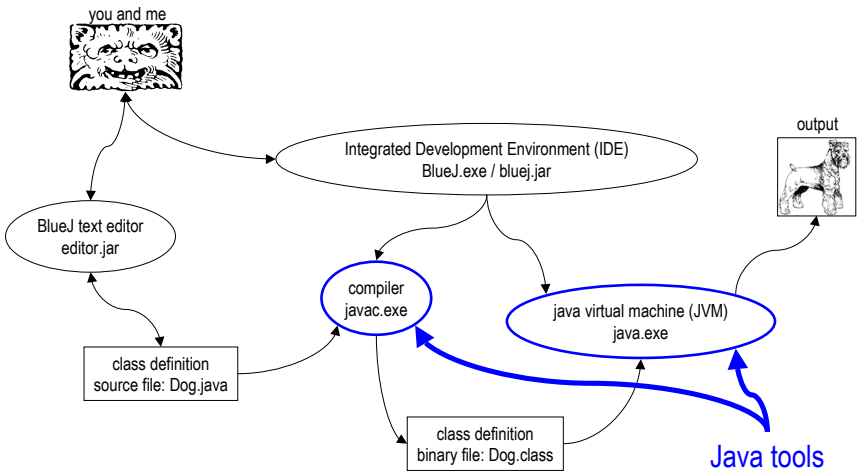
Us



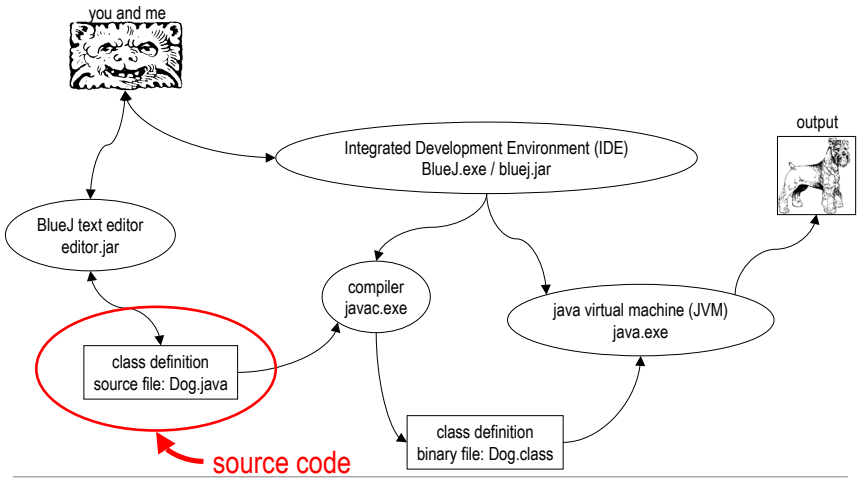
Integrated Development Environment (IDE)



Java compiler and virtual machine

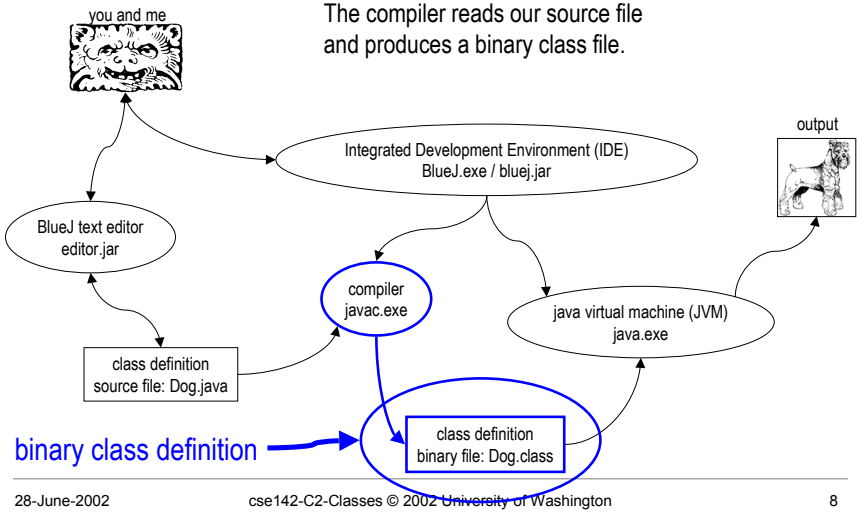


Source code (we write this)

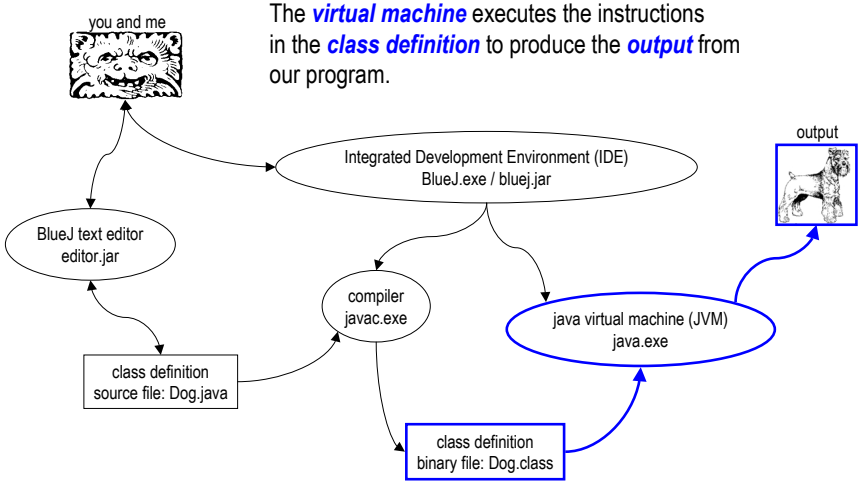


Class file (the compiler produces this file)

The compiler reads our source file
and produces a binary class file.



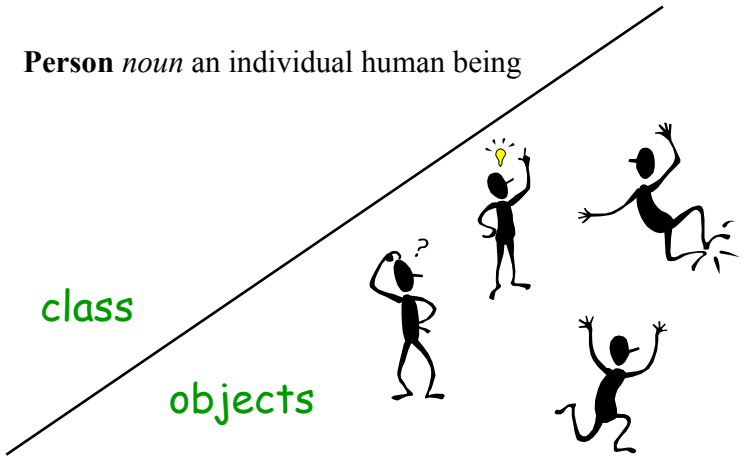
Running the program (finally!)



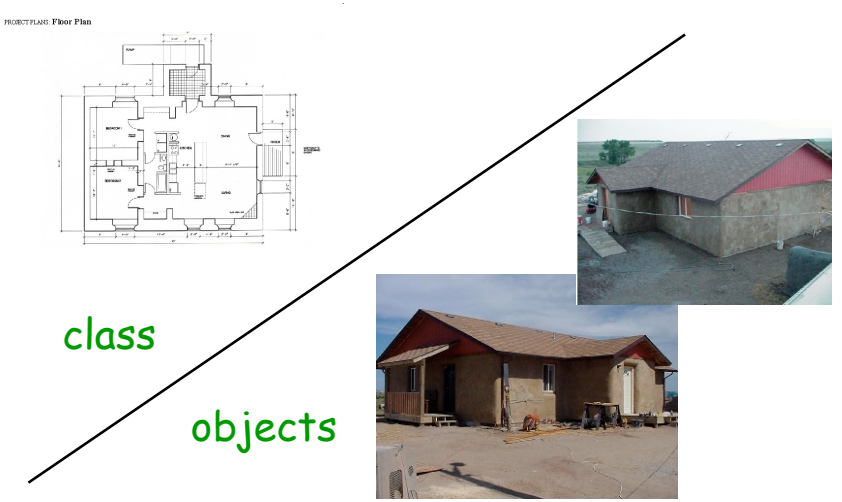
What is a Java class?

- A class is a *template* or *blueprint* for building objects
- A class is like a dictionary definition, while objects are like things in the real world that "are" whatever is defined
- A class definition generally resides on disk long term
  - the original class definition is written in Java (the .java file) then translated into a more compact form (the .class file) by the compiler
  - the class definition can be used over and over to create more objects, just like a blueprint can be used over and over to build more houses
- An object resides in memory and is discarded during or at the end of a program run

Individuals are instances of class "Person"



Houses are instances of blueprints



http://vcourses.caup.washington.edu:8900/public/CM599/index.html

Instantiate

- Once we create a class definition using an editor and the compiler, we can *instantiate* it with the "**`new`**" operator
  - to *instantiate* means to create objects based on the class definition
  - `Oval moon = new Oval(100,100,20,20,Color.gray,true);`
- We can then manipulate these objects to do the work that needs to be done
- Note that many classes have already been defined for us
  - There are 2723 classes defined in the standard Java libraries from Sun - see the JavaAPI documentation
  - There are several classes defined in the UWCSE.jar file - see docs

Example: Committee and Person

- Example: a Committee object is composed of Person objects, each of which has a vote
- When the Committee has to decide an issue, it "asks" each of its Person objects to cast its vote
- When we design the Committee class, we will instantiate and use Person objects to get the work done

Class Concepts

- Class definitions have two important components:
  - state
  - behavior or interface
- State is expressed using fields in the class definition
- Behavior is expressed using methods
- Together, fields and methods are called class members

Class Concepts: State

- State is a complete description of all the things that make a class a class.
- For example, part of the state of class Employee is the Employee's UWNetID. All objects of class Employee will have a UWNetID specified.
- State is stored in data members
  - also known as fields, member variables, instance variables, properties

Class Concepts: Behavior

- Behavior of a class defines how other classes may interact with it. It indicates the capabilities of the class to "do" things.
- For example, a BaseballPlayer class might define such behavior as hit, pitch, stealBase, etc.
- Behavior is defined in methods
  - Methods look like functions in C, subroutines in Fortran, etc

Class Concepts: *get* and *set* methods

- Part of a class' behavior is simply to return information about state
- The convention in Java is to use accessor and mutator methods to allow other classes to query and (possibly) alter the state of a class' objects
- The conventional accessor method is of the form get<fieldname>. For example, if a field is named age, the corresponding accessor method would be getAge().
- Mutator methods are of the form set<fieldname>

Java Class Syntax

- Basic form:

```
[modifiers] class name { [body]  }
```

- Classes often written like:

```
class myClass {
  // public features
  // private features
}
```

- Be consistent, not religious about structure

Example class

```
public class Dog  {
 public Dog(int rate) {
      consumptionRate=rate;
      weight = 20;
 }
 public void bark() { . . . }
 public int getRate() { . . . }
 public void eat(int pounds) { . . . }

 private int consumptionRate;
 private int weight;
}
```

dog.java

Tools - BlueJ

- The primary development tool we will use this quarter is a program called BlueJ
- BlueJ is a simple “Integrated Development Environment” or IDE
- BlueJ uses the regular Java compiler from Sun to convert our Dog.java source file into Dog.class class files
- Then we can create new objects (*instantiate* them) using the class definition and manipulate them
  - BlueJ lets us do slowly and visibly what our code can do very quickly

Tools - Documentation

- There is a lot to know when programming!
  - and there is a lot of information provided to help the programmer
- Your web browser is the key tool for documentation!
  - Many people use Internet Explorer, but any modern browser will do
- Spend the time now to find and bookmark the index pages for each of the key documents that you will need
  - JavaAPI - all the standard Java library classes
  - UWCSE packages - the CSE library classes for 142/143
  - JavaDoc - documentation tags
  - JavaSpec - the Java language specification
  - references can be found on our **software** and **otherlinks** pages

Tools - File System

- It is important that you be able to find the source files that you write and understand what they are and how they relate
- It’s confusing at first, but it is not magic and you can figure it out with a little work
  - YOU are the programmer, the computer will do what you tell it to
  - If you know the language of software development, you can tell the computer to do lots of interesting things
- Go look at the directories
  - find the .java files and the .class files
  - understand where BlueJ is putting your files
- The QuickLaunch bar is a handy place to store shortcuts

Tools - UW accounts on Dante

- The University of Washington has a very good set of systems that provide you with file storage, email,web publishing, etc
- Know how to back up your files to your account on Dante
- The labs have a shortcut on the desktop for this
- You can log on to Dante directly using SSH Secure Shell
  - available from the University for loading on your own machine
  - http://www.washington.edu/computing/software/

Appendix

Dog.java

```
/**
 * Sample class for demonstrating class structure.
 */
public class Dog {
    /**
    * Create a new Dog.  This constructor supplies a default weight of 20 pounds.
    * @param rate the rate at which this dog eats, specified in pounds/fortnight.
    */
    public Dog(int rate) {
        consumptionRate = rate;
        weight = 20;
    }
    /**
    * Provide this dog with a voice.
    */
    public void bark() {
        System.out.println("Woof! Woof!");
    }
    /**
    * Provide this dog with a way to rest his bones.
    */
    public void sleep() {
        System.out.println("Snrf ... woof ... snrf ...");
    }
```


Dog.java

```
/**
 * Eat some goodies.  There is some weight gain after eating.
 * @param pounds the number of pounds of food provided.
 */
public void eat(int pounds) {
    double coverage = (double)pounds/(double)consumptionRate;
    String foodUnits = (pounds == 1) ? "pound" : "pounds";
    String timeUnits = (coverage == 1) ? "fortnight" : "fortnights";
    System.out.println(
        Integer.toString(pounds)+" "+foodUnits+
            " lasted "+coverage+" "+timeUnits+".");
    weight += (double)pounds/2;
    System.out.println("Weight is now "+weight+" pounds.");
}
/**
 * Retrieve the rate value for this Dog.
 * @return the consumption rate specified for this Dog.
 */
public int getRate() {
    return consumptionRate;
}
```


Dog.java

```
/**
 * Run this guy through his day.  This is a simple test harness for this class.
 * @param args ignored
 */
public static void main(String[] args) {
    Dog rover = new Dog(28);
    rover.sleep();
    rover.bark();
    rover.eat(2);
    rover.bark();
    rover.eat(14);
}
//---------------------------------------------------------------------
/** the consumption rate specified for this dog.  Given in
 * pounds per fortnight.  A fortnight is two weeks or 14 days.
 */
int consumptionRate;

/** the weight of this dog.  Assumed to be in pounds. */
int weight;
}
```