Welcome

Java Programming I
CIS 325

Week 6 – Java Classes

Inheritance and Polymorphism

Christopher K. Burns

Agenda

Tonight's agenda

- Classes
  - Inheritance
  - Polymorphism
- MidTerm Review
- Homework Review

Home Work

Read text Chapters – up to 12

Demonstrate Inheritance and Polymorphism:

- Create a Java class to act as a base class; it must define at least two methods.
- Create two other Java classes that inherit from the base class and add at least one additional method of their own.
- Create a Java class with a main method that will demonstrate inheritance and polymorphism.

- Due on the 25th of May.

Schedule

Week		Content	
1	4/6	Chapter 1 Intro to Computers, the Internet and the Web Chapter 2 Intro to Java Applications Chapter 3 Java Classes and Objects: Part 1 *Homework 1 Assigned*	
2	4/13	Chapter 4 Control Structures: Part 1 Chapter 5 Control Structures: Part 2	
3	4/20	Chapter 6 Methods Chapter 7 Arrays *Homework 2 Assigned*	HW 1 DUE
4	4/27	Chapter 8 Java Classes and Objects: Part 2 Chapter 1-8 Review	
5	5/4	***MID-TERM EXAMINATION***	HW 2 DUE
6	5/11	Chapter 9 Object-Oriented Programming: Inheritance Chapter 10 Object-Oriented Programming: Polymorphism *Homework 3 Assigned*	PROJECT IDEA DUE
7	5/18	**No Class Tonight**	
8	5/25	Chapter 11 GUI Components: Part 1 Chapter 12 Graphics and Java2D *Homework 4 Assigned*	HW 3 DUE
9	6/1	Chapter 13 Exception Handling Chapter 29 Strings, Characters and RegEx	
10	6/8	Chapter 20: Java Applets Chapter 23 Multithreading	HW 4 DUE
11	6/15	Files, JDBC, Networking, Servlets, and JSP *Class lab time for review and assistance with final project* ***FINAL PROJECT DUE***	PROJECT DUE

Inheritance

- Definition -

in·her·i·tance (P) **Pronunciation Key** (n-hr-tns) *n.*

The act of inheriting.

Something inherited or to be inherited.

Something regarded as a heritage: *the cultural inheritance of Rome.* See Synonyms at heritage.

Biology.

The process of genetic transmission of characteristics from parents to offspring.

A characteristic so inherited.

The sum of characteristics genetically transmitted from parents to offspring.

from *dictionary.com*

Relationships

- Two ways of looking at items related to a class:

  IS – A

  and

  HAS – A

- Some class "is a" something
- Some class "has a" somthing

Relationships

- **Some class “is a” something**
  - This is an example of inheritance
  - A Mac “is a” Computer, so if you were creating a class hierarchy, Mac would be based on Computer
- **Some class “has a” something**
  - This is an example of containment or a member item of a class. It is a property of a class/object.
  - A Computer “has a” processor, so your computer class would define a processor property.

Simple Inheritance

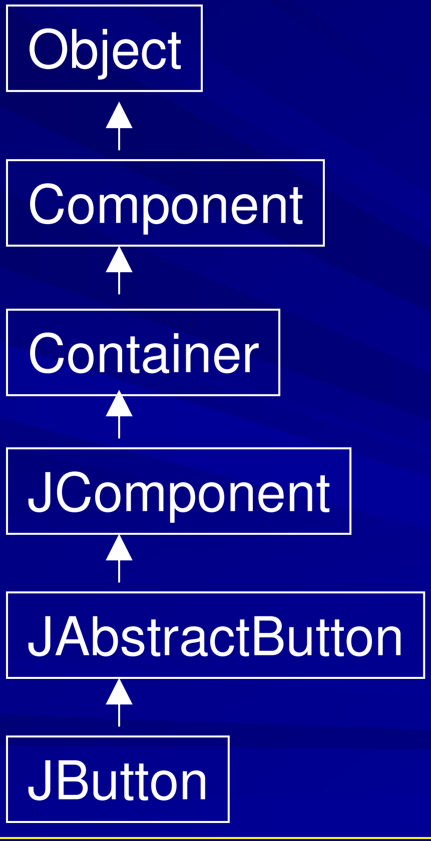
```
public class A
{
  private String name;
  public String getName() { return name; }
  public void setName( String name ) { this.name = name; }
}

public class B extends A
{
  private String stuff4B;
  public void setStuff4B( String stuff ) { stuff4B = stuff; }
}
```

Simple Inheritance

```
public class Test
{
  public static void main( String[] args )
  {
     B b = new B();
     b.setName( "I inherited this from A" );
  }
}
```

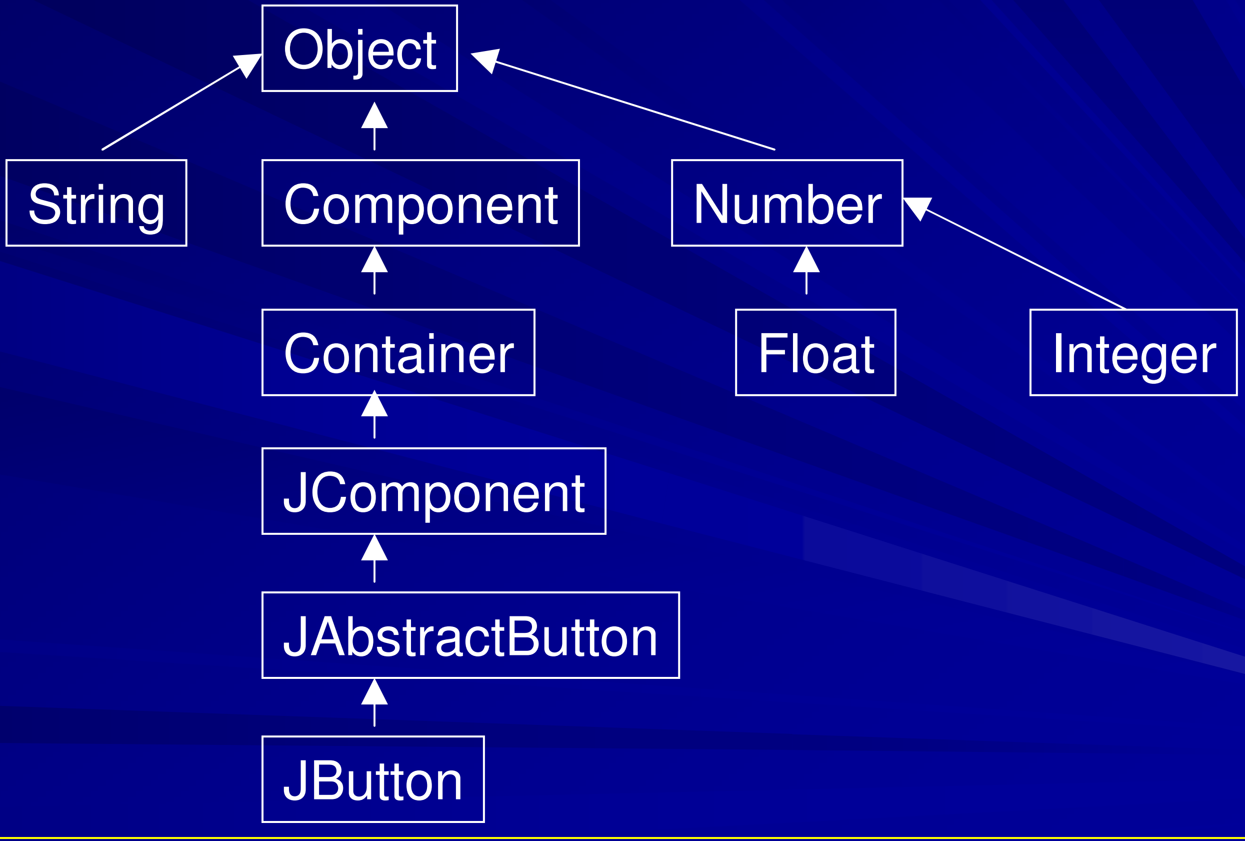
Class Hierarchy

- Much like inheritance of family traits, it is easiest to look at inheritance in a tree style graph.



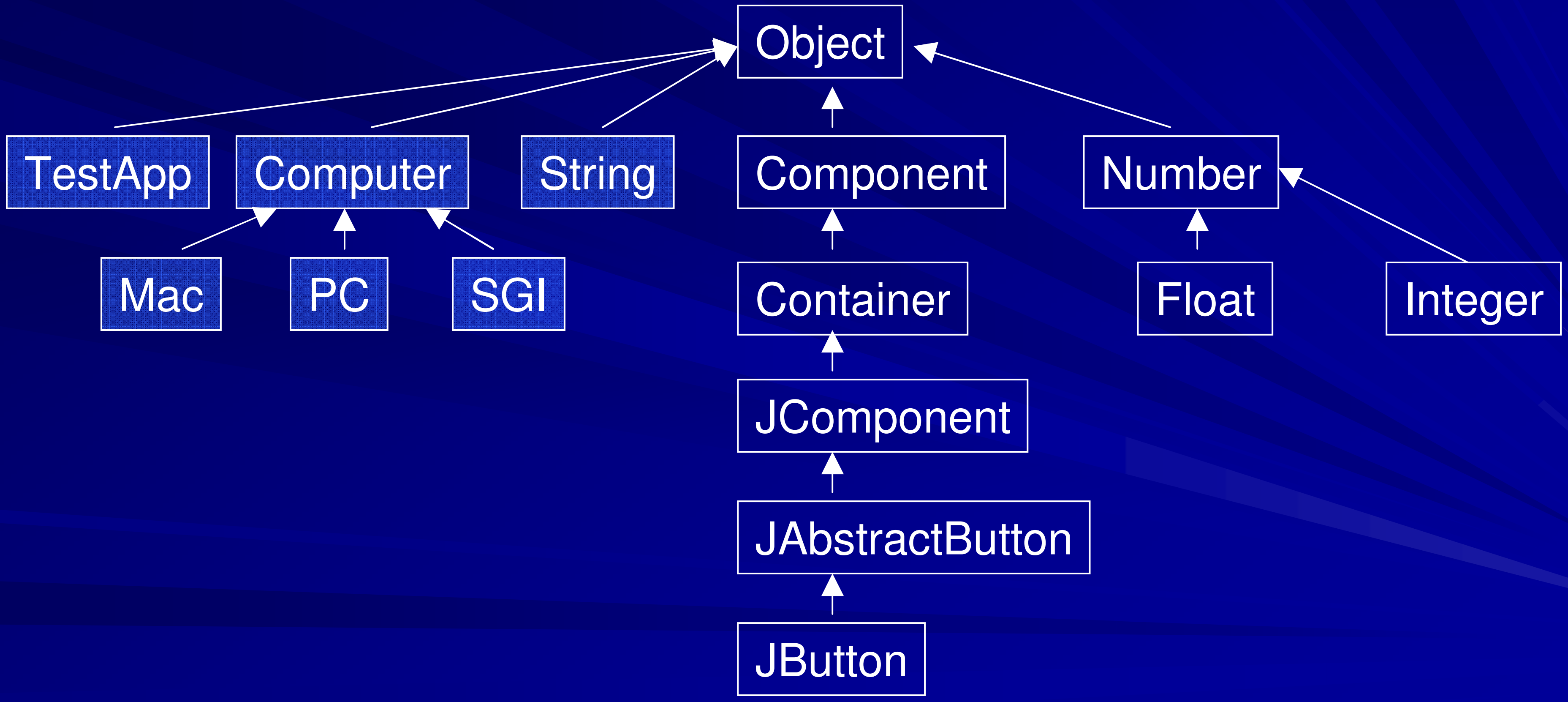
Class Hierarchy

- **All** Classes in java inherit from Object (java.lang.Object)



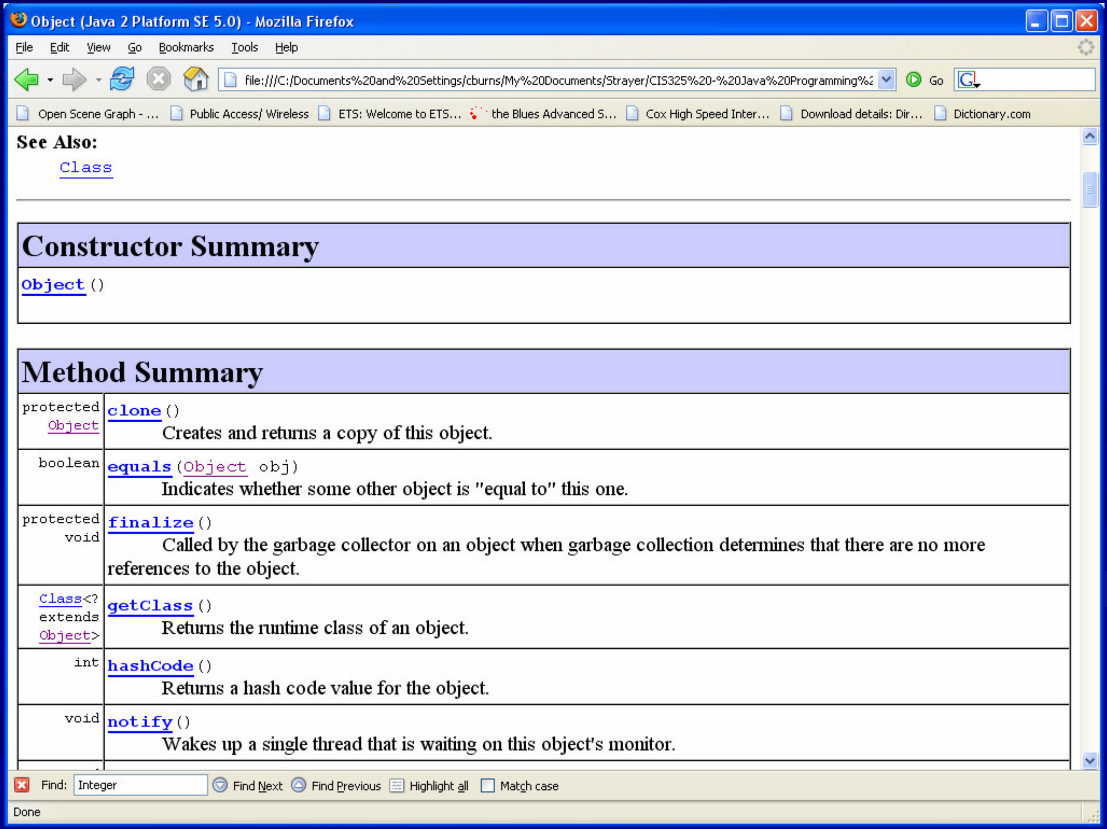
Class Hierarchy

- **All** Classes in java inherit from Object, even classes we create ourselves.



Class Hierarchy

- What do we get by having all classes in Java inheriting from Object?


Object (Java 2 Platform SE 5.0) - Mozilla Firefox

file:///C:/Documents%20and%20Settings/cburns/My%20Documents/Strayer/CIS325%20-%20Java%20Programming%2

See Also:
Class

Constructor Summary

Object()

Method Summary

protected Object	clone() Creates and returns a copy of this object.
boolean	equals(Object obj) Indicates whether some other object is "equal to" this one.
protected void	finalize() Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
Class<? extends Object>	getClass() Returns the runtime class of an object.
int	hashCode() Returns a hash code value for the object.
void	notify() Wakes up a single thread that is waiting on this object's monitor.

Find: Integer

Done

using base class object

```
public class A
{
  private String name;
  public String getName() { return name; }
  public void setName( String name ) { this.name = name; }

  // override Object toString
  public String toString() { return ("My name is: " + name); }
}
```

using base class object

```
public class A
{
  private String name;
  public String getName() { return name; }
  public void setName( String name ) { this.name = name; }

  // override Object toString
  public String toString() { return ("My name is: " + name); }
}

public class Test
{
  public static void main( String[] args )
  {
     B b = new B();
     b.setName( "I inherited this from A" );

     System.out.println( b );  // our toString() method is called here
  }
}
```

using base class object

Wait, how did println(b) know that b was of type B that inherited from A and was able to call it’s toString() method?

```
public class Test
{
  public static void main( String[] args )
  {
     B b = new B();
     b.setName( "I inherited this from A" );

     System.out.println( b );  // our toString() method is called here
  }
}
```

using base class object

Wait, how did println(b) know that b was of type B that inherited from A and was able to call it's toString() method?

Because B inherits from A who inherits from Object… B can be treated as A because it has all the characteristics of A, and B can also be treated as Object because it has all the characteristics of Object.

This is polymorphism

Polymorphism

- Definition -

poly·mor·phism
Pronunciation: "pä-lE-'mor-"fi-z&m
Function: *noun*
: the quality or state of being able to assume different forms: as **a** : existence of a species in several forms independent of the variations of sex
b : the property of crystallizing in two or more forms with distinct structure
– **poly·mor·phic** /-fik/ *adjective*
– **poly·mor·phi·cal·ly** /-fi-k(&-)lE/ *adverb*

from *m-w.com*

Group Lab

Let’s further investigate and learn about these concepts through hands-on examples.

We shall create a basic inventory control system that has a few classes to represent types we will be tracking and we shall use inheritance and polymorphism to make our lives easier.

We will also look at a few additional features of the Java API so that we can make a semi-usable application out of this demo/lab.

Group Lab

(note: the completed lab will be passed out and available for download)

- We shall create three classes:

  1. TestApp, Item, and Computer
  2. TestApp class shall contain main, and create instances of Computer
  3. Item shall have some basic properties
  4. Computer shall inherit from Item

Group Lab

5. Calling functions in our super (super constructor)
6. Add a few more classes to lab (e.g. Monitor, Software)
7. If you inherit from something you can be treated like that something. Let's see how this works.
8. Collections of objects
   - **ArrayList** – look at it in Java API doc
   - Implement simple array of Items
9. More fun with **JOptionPane**
   - Lists
   - Confirmations