

A+ Computer Science

Abstract

classes



Abstract Classes

Abstract classes are used to define a class that will be used only to build new classes.

No objects will ever be instantiated from an abstract class.

Abstract Classes

Mammal (abstract class)



Human



Whale



Cow

Abstract Classes

Any sub class that extends a super abstract class must implement all methods defined as abstract in the super class unless the extending class is an abstract class.

Abstract Classes

Abstract classes are typically used when you know quite a bit about an Object and what you want the Object to do, but yet there are still a few unknowns.

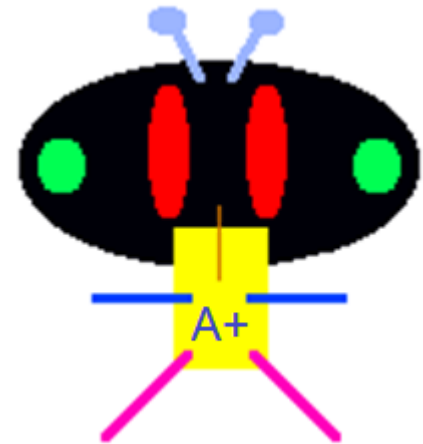
```
public abstract class Monster
{
    private String name;

    public Monster( String nm )
    {
        name = nm;
    }

    public abstract String talk( );

    public String toString()
    {
        return name + " says " + talk();
    }
}
```

Abstract Classes



Abstract Classes

Why define talk as abstract?

```
public abstract String talk( );
```

**Does each Monster say
the exact same thing?**

```
public class Vampire extends Monster  
{  
  public Vampire( String name )  
  {  
    super(name);  
  }  
  
  public String talk()  
  {  
    return "\"I want to drink your blood!\"";  
  }  
}
```



```
public class Ghost extends Monster  
{  
  public Ghost( String name )  
  {  
    super(name);  
  }  
  
  public String talk()  
  {  
    return " \\"Where did I go?\\\"\\n\\n";  
  }  
}
```

Abstract Classes

Mammal (abstract class)



Human



Whale



Cow

Abstract Classes

Monster (abstract class)



Vampire



Ghost



Witch

monster.java
ghost.java
ghostrunner.java

Polymorphism

Polymorphism - the ability of one general thing to behave like other specific things.

Polymorphism

```
//instance variable
```

```
private Monster[] monsters;
```

```
//ask for the number of monsters
```

```
//get the number of monsters
```

```
for ( int j=0; j < monsters.length; j++ )  
{  
    out.print("Enter Monster " + j + " Name :: ");  
    int r = (int)( Math.random() * 3 );  
    if(r==0)  
        monsters[j] = new Vampire(kb.nextLine());  
    else if(r==1)  
        monsters[j] = new Witch(kb.nextLine());  
    else  
        monsters[j] = new Ghost(kb.nextLine());  
}
```

Polymorphism

```
public String monstersTalk( )  
{  
    String out = "";  
    for ( int i=0; i<monsters.length; i++ )  
        out += monsters[i].talk();  
    return out;  
}
```

Polymorphism

```
public String toString( )  
{  
    String output="";  
    for ( int i=0; i<monsters.length; i++ )  
        output+=monsters[i].toString();  
    return output;  
}
```


monsterpack.java
packrunner.java

Interfaces vs Abstract Classes

Description	Interface	Abstract Class
Can contain abstract methods?	Yes	Yes
Can contain non-abstract methods?	No	Yes
Can contain constructors?	No	Yes
Can be instantiated?	No	No

Description	Interface	Abstract Class
Can be extended?	Yes	Yes
Can be implemented?	Yes	No

Description	Interface	Abstract Class
Can contain instance variables?	No	Yes
Can contain final instance variables?	No	Yes
Can contain final class variables?	Yes	Yes
Can contain class variables?	No	Yes

Extends / Implements Rules

Classes extend Classes

Interfaces extend Interfaces

SAME extends SAME

Classes implement Interfaces

CLASS implements INTERFACE

classextends.java
interfaceextends.java

Types of Binding

Static Binding

Method calls are locked down at compile time based on the type of reference used.

```
Object a = "apluscompsci";  
int x = a.length();           //syntax error  
System.out.println( x );  
  
x = ((String)a).length();    //add a cast
```

staticbinding.java

Dynamic Binding

Specific types of objects associated with method calls are determined at run time, creating polymorphic behavior.

```
public void monstersTalk( )
{
    out.print("monstersTalk\n\n");
    for ( int i=0; i<monsters.length; i++ )
        out.println( monsters[i].talk() );
}
```

Dynamic Binding

```
public double processList( List<Integer> list )
{
    double sum = 0;
    for( int i = 0; i < list.size(); i++ )
        sum += list.get(i);
    return sum / list.size();
}
```

Calls to processList() could be made with an ArrayList, LinkedList, Vector, or Stack as all four classes implement the List interface, sharing a common set of methods.

dynamicbinding.java

static

Static

Static is a reserved word use to designate something that exists as part of a class, but not part of a specific object.

Static variables and methods exist even if no object of that class has been instantiated.

Static

Static means one!

All Objects will share the same static variables and methods.

Static variables are also called class variables.

Static

```
class Monster
```

```
{
```

```
    private String myName;
```

```
    private static int count = 0;
```

all Monster share count

```
    public Monster() {
```

```
        myName = "";
```

```
        count++;
```

```
    }
```

```
    public Monster( String name ) {
```

```
        myName = name;
```

```
        count++;
```

```
    }
```

```
}
```



static.java

Work on Programs!

Crank

Some Code!

A+ Computer Science

Abstract

classes

