Multiple Inheritance in Java Contrast C++ with examples Rate Topic: 1 Votes

KYA
Posted 19 February 2009 - 07:35 PM

POPULAR

Multiple Inheritance in Java

What is multiple inheritance? It is the ability for a class to be derived from more then one class. To fully appreciate what Java can do in this regard we must take

A Quick Detour into C++

C++ allows multiple inheritance easily, depending on how the programmer sets up the base classes, etc... Consider the following example in C++:

```
01 #include <iostream>
02 #include <string>
03 using namespace std;
04
05 //base class
06 class Animal {
07 protected: //derived classes get these
08     int age;
09     string name;
10 public:
11     Animal() {};
12     string getName()        {return name;};
13     int getAge()            {return age;};
14 };
15
16 //Bird is a derived class from Animal
17 class Bird : public virtual Animal {
18 public:
19     void birdNoise()        { cout << "Bird
   Noise!\n";};
20     Bird() {};
21 };
22
23 //Horse also derives from Animal
24 class Horse : public virtual Animal {
25 public:
26     void horseNoise()       { cout << "Horse
   Noise!\n";};
27     Horse() {};
28 };
29
30 class Pegasus : public Bird, public Horse {
31     //reserved for future items
32 public:
33     Pegasus(string n, int a) {name = n; age =
   a;};
34 };
35
```


Follow & Share

3084 readers BY FEEDBURNER

Java Tutorials

- JDesktopPane and JInternalFrame
- Simple Client and Server Chat Program
- Separating the GUI and Logic with a StateManager
- A simple Chat program with Client/Server (GUI optional)
- Understanding Bit-wise Operators in Java
- A Look at Backtracking
- Design Patterns: Observer Pattern
- Design Patterns:

General Discussion

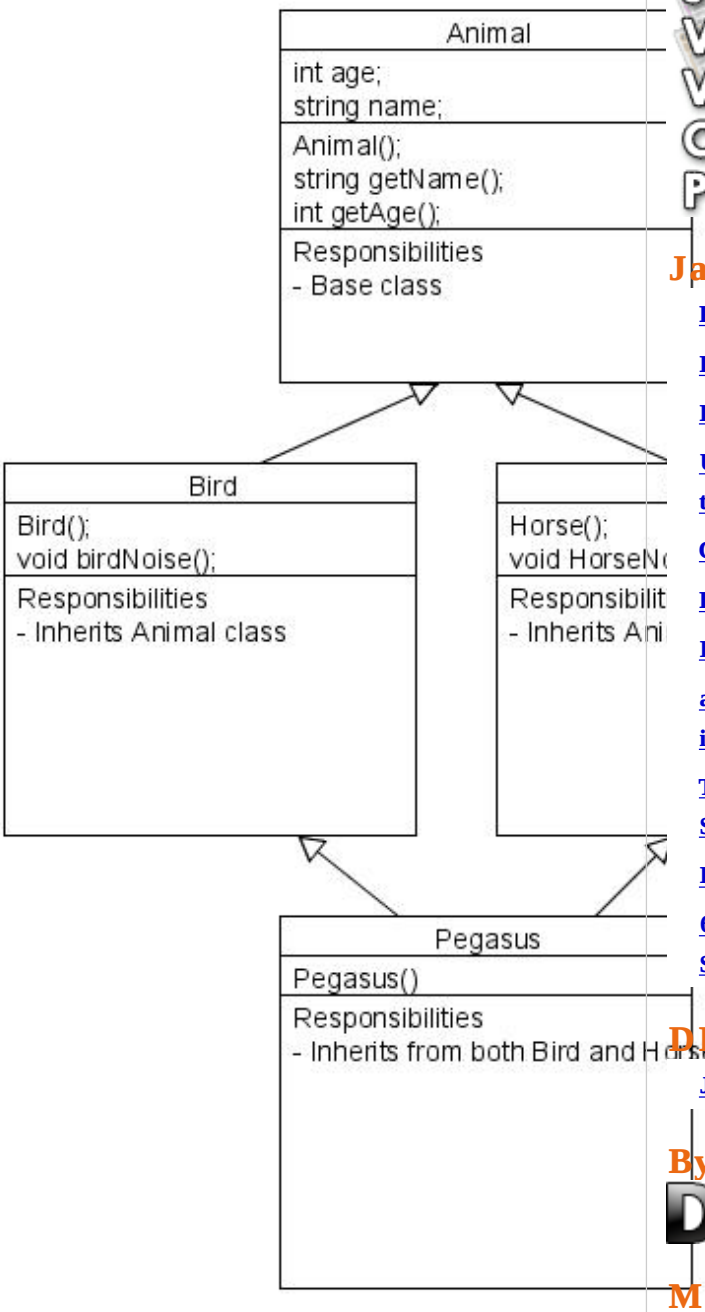
- Caffeine Lounge
- Corner Cubicle
- Student Campus
- Software Development
- Industry News
- Introduce Yourself

Programming Help

- C and C++
- VB6
- Java
- VB.NET
- C#
- ASP.NET
- .NET Framework
- PHP
- Ruby
- Python
- ColdFusion
- Databases

```
36  int main()
37  {
38      Pegasus* peggi = new Pegasus("Pegasus", 5);
    //create the pegasus object
39      cout << peggi->getName() << "'s age is " <<
    peggi->getAge() << endl;
40      peggi->birdNoise();
41      peggi->horseNoise();
42      delete peggi;
43      return 0;
44  }
```

A possible UML diagram based on the following:



We set up all of that infrastructure to allow Pegasus to be both a bird and a horse, which is important because he's a flying horse. While on a small scale this seems unnecessary and overcomplicated, as a project grows in

Decorator Pattern
Beginning With For Each Loops
JAVA CLASS DESIGN "E = MC^2" TUTORIAL
191 More Java Tutorials...

both size and scope, determining how objects relate to each other will become more apparent.

But wait, you say, this is the Java tutorial section...you are absolutely right, let's not digress:

Back to Java

We can't do the above directly with Java (indirectly we can though), as it only supports single direct inheritance, such as:

```
01 public class Animal {
02
03 //animal stuff
04
05 }
06
07 public class Bird extends Animal {
08
09 //Bird stuff
10 }
11
12 //However we can't have another class extend
   Bird vis a vis extending Animal as well
```

What we can do is use an interface to simulate multiple inheritance:

```
01 //Base class
02 public class Animal {
03
04     protected int age;
05     protected String name;
06
07     public Animal()             {}
08
09     public String getName()     {return name;}
10     public int getAge()         {return
   age;}
11 }
```

Now, instead of having Bird and Horse classes extend Animal (single inheritance), we'll make them interfaces. Any class that implements an interface is required to override the methods presented in them. This is for a programmer's own protection:

```
01 //***************SYNTAX**********************
02 public interface interfaceName {
03      void abstractMethod();
04       int overrideMe();
05     //etc...
06 }
07 //*******************************************
08 public interface Bird {
09     //all implementers have to overwrite methods
10     void birdNoise();
11 }
12
13 public interface Horse {
14     //all implementers have to overwrite methods
15     void horseNoise();
16 }
```

The java syntax for inheritance and interfaces:

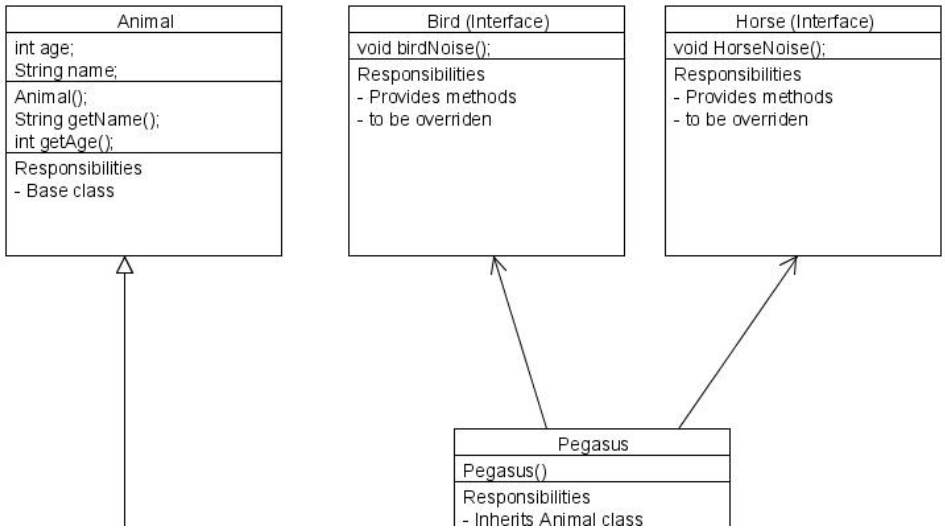
```
1  public class className extends superClass
   implements interfaceOne, interfaceTwo, ...
```

Here is the Pegasus class (from above, converted to Java) with a main() example:

```
01 import java.io.*;
02                      //single inheritance,
   multiple interfaces
03 public class Pegasus extends Animal implements
   Bird, Horse {
04     public Pegasus(String n, int a)
05     {
06         name = n;
07         age = a;
08     }
09
10     //overriding interface methods
11     public void horseNoise()
12     {
13         System.out.println("Horse Noise!");
14     }
15
16     public void birdNoise()
17     {
18         System.out.println("Bird Noise!");
19     }
20
21
22     public static void main(String[] args)
23     {
24         Pegasus peggi = new Pegasus("Pegasus",
   5);
25         System.out.println(peggi.getName() + "'s
   age is " + peggi.getAge());
26         peggi.birdNoise();
27         peggi.horseNoise();
28     }
29 }
```

A possible UML diagram based on the java implementation:

Resized to 67% (was 750 x 550) - Click image to enlarge



A visual is often the best learning aid. Quick synopsis: Based on interfaces we can "fake" multiple inheritance in java since only direct single inheritance is supported. We therefore extend to the base class we plan on using and then use interfaces to allow the object to have access to as many functions as we want. **Remember, we must write each class's own implementation for each method in the interface.** In large projects, using interfaces allows us to control the flow of code and data access, which is crucial. Hopefully you found this tutorial helpful. You are free use the code/images in any manner that you see fit. Happy coding!

--KYA

Replies To: Multiple Inheritance in Java

programstudent
Posted 04 February 2010 - 04:24 AM

hi,
Good Explanation very much helpful
In the uml diagram for java there is no connection to Animal from Bird and horse why?
Is it necessary to use the implement the same method in the derived class and why
[code]void birdNoise();
[code] void horseNoise();
why in the Peagus class
[code]public void horseNoise()
[code] {
[code] System.out.println("Horse Noise!");
[code] }

[code] public void birdNoise()
[code] {
[code]System.out.println("Bird Noise!");
[code] }
why this must be there

Why "Remember, we must write each class's own implementation for each method in the interface. " reason?
Thank for this good explanation
Thank you
This post has been edited by **programstudent**: 04 February 2010 - 04:25 AM

programstudent
Posted 04 February 2010 - 04:38 AM

hi,
Please expalne the same with a java project
For example a login program for example in swing

forget the part of gui creation just explain the concept of inheritence in login section means the base class for the database connection -class animal,class bird -count the login,class horse-check valid user, class peagus accepting the value from the gui

Thank you

Related Java Topics beta

- Multiple Inheritance In Java
- Inheritance In Java - Inheritance
- Multiple Inheritance Problem
- To Calculate Depth Of Inheritance - How To Calculate Depth Of Inheritance In Java Programs
- Using Inheritance In Java
- Writing JAVA Interface. Need HELP!
- Why Do We Need Interfaces?
- One Class Inheriting From Two
- What Is The Difference In OOP Concepts Between Java And C#? - Oop Concepts Like Inheritance, Polymorphism,abstraction And Encapsulat
- Java Threads - An Introduction To Threads In Java Tutorial