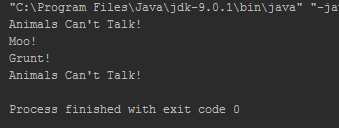
CIS 400 HW3

General Information:

* Adam Giaccaglia
* Windows 7 64 bit
* Interpreter type:
  + Java: IntelliJ IDEA and SDK 9.0
  + C++ and C#: Visual Studio 2013

Question 1

/\*  
 Java  
 Question1  
 Adam Giaccaglia  
  
 \*/  
  
public class Question1 {  
 public static void main( String[] args ) {  
  
 Animal[] animals = new Animal[4];  
  
 animals[0] = new Animal();  
 animals[1] = new Cow();  
 animals[2] = new Pig();  
 animals[3] = new Snake();  
  
 for (int i = 0; i < animals.length; i++)  
 {  
 animals[i].talk();  
 }  
  
 }  
}  
  
class Animal  
{  
 private int Leg;  
 public Animal()  
 {  
 Leg = 4;  
 }  
  
 public void talk()  
 {  
 System.*out*.println("Animals Can't Talk!");  
 }  
}  
  
class Cow extends Animal {  
 // functions are virtual by default  
 public void talk() {  
 System.*out*.println("Moo!");  
 }  
}  
  
class Pig extends Animal {  
 public void talk() {  
 System.*out*.println("Grunt!");  
 }  
}  
class Snake extends Animal {  
  
}



/\*

\* C#

\* Question 1

\* Adam Giaccaglia

\*

\*/

namespace CSharp\_Versions

{

class Question1

{

static void Main(string[] args)

{

Animal[] animals = new Animal[4];

animals[0] = new Animal();

animals[1] = new Cow();

animals[2] = new Pig();

animals[3] = new Snake();

for (int i = 0; i < animals.Length; i++)

{

animals[i].talk();

}

//pause

Console.ReadKey();

}

}

public class Animal

{

private int Leg;

public Animal()

{

Leg = 4;

}

public virtual void talk()

{

Console.WriteLine("Animals Can't Talk!");

}

}

public class Cow : Animal

{

public Cow()

: base()

{

}

public override void talk()

{

Console.WriteLine("Moo!");

}

}

public class Pig : Animal

{

public Pig()

: base()

{

}

public override void talk()

{

Console.WriteLine("Grunt!");

}

}

public class Snake : Animal

{

public Snake()

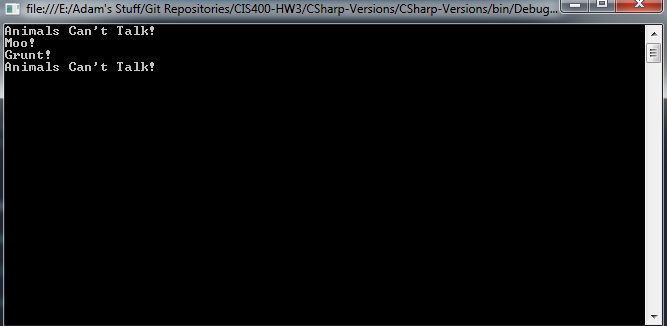
: base()

{

}

}

}



/\*

Question 1

C++

Adam Giaccaglia

\*/

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

class Animal

{

private:

int Leg;

public:

Animal()

{

Leg = 4;

}

virtual void talk()

{

cout << ("Animals Can't Talk!");

}

};

class Cow : public Animal

{

public:

void talk()

{

cout << ("Moo!");

}

};

class Pig : public Animal

{

public:

void talk()

{

cout << ("Grunt!");

}

};

//public to override default private inheritance

class Snake : public Animal

{

public:

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

//pointer for dyanmic binding and allocation

Animal\* animals[4];

animals[0] = new Animal();

animals[1] = new Cow();

animals[2] = new Pig();

animals[3] = new Snake();

for (int i = 0; i < 4; i++)

{

animals[i]->talk();

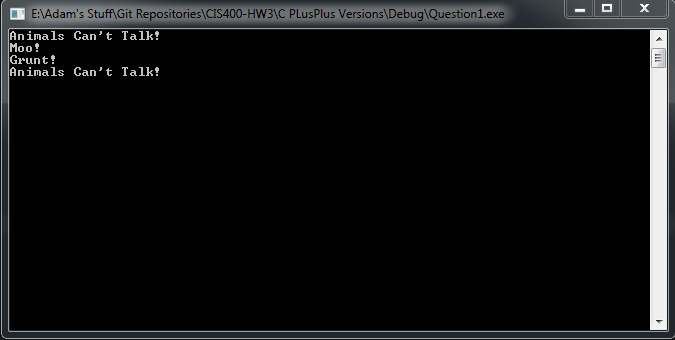
cout << endl;

delete animals[i];

}

cin.get();

}



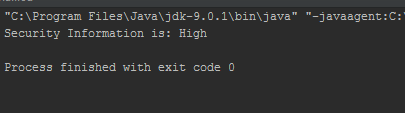
The difference between C# and Java is that functions are virtual by default in Java. This means no need to do anything special for dynamic binding. In C# you need the virtual and override key word. For C++, I had to do the same thing with C# and make the function virtual but no need for override. In C++ I had to use an Animal pointer to enable dynamic binding whereas C# and Java are smart enough to know the correct class.

Question 2:

C++ supports multiple inheritance so having musician inherit from pianist and violinist is simply adding a comma to add another base class.

Question 3:

/\*  
 Question 3  
 Java  
 Ada Giaccaglia  
  
 \*/  
  
  
package Corporation;  
public class Question3 {  
 public static void main( String[] args ) {  
 Bank bank1 = new Bank();  
 Manager boss = new Manager();  
 boss.securityAccess(bank1);  
 }  
}  
  
class Bank  
{  
 private String name;  
 protected String securityInfo;  
 //so securityInfo is not null  
 Bank(){  
 securityInfo = "High";  
 }  
 public void display(){  
 System.*out*.println("This is a bank!");  
 }  
  
}  
class Manager  
{  
 private int id;  
 public void display()  
 {  
 System.*out*.println("I am a manager!");  
 }  
 public void securityAccess(Bank x){  
 System.*out*.println("Security Information is: " + x.securityInfo);  
 }  
}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

/\*

\* Question3

\* C#

\* Adam Giaccaglia

\*/

namespace Question3

{

class Program

{

class Bank

{

private string name;

protected internal string securityInfo;

public Bank()

{

securityInfo = "High";

}

public void display(){

Console.WriteLine("This is a bank!");

}

}

class Manager

{

private int id;

public void display()

{

Console.WriteLine("I am a manager!");

}

public void securityAccess(Bank x){

Console.WriteLine("Security Information is: " + x.securityInfo);

}

}

static void Main(string[] args)

{

Bank bank1 = new Bank();

Manager boss = new Manager();

boss.securityAccess(bank1);

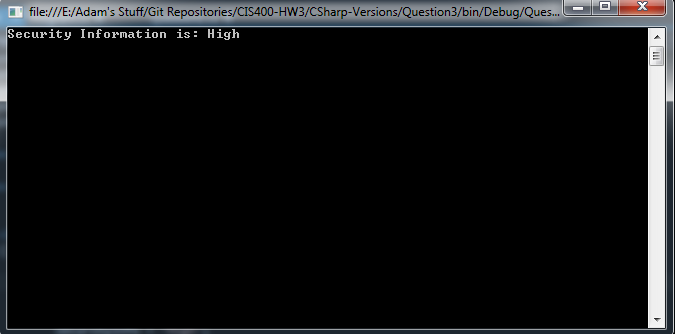
//pause

Console.ReadKey();

}

}

}



// Question3.cpp : Defines the entry point for the console application.

//

/\*

Question3

C++

Adam Giaccaglia

\*/

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

class Bank

{

friend class Manager;

private:

string name;

protected:

string securityInfo;

public:

Bank(){

securityInfo = "High";

}

void display(){

cout <<("This is a bank!");

}

};

class Manager

{

private:

int id;

public:

void display()

{

cout <<("I am a manager!");

}

void securityAccess(Bank x){

cout <<("Security Information is: " + x.securityInfo);

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

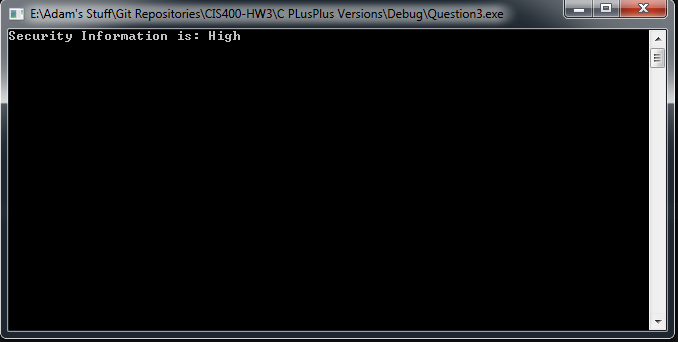
Bank bank1;

Manager boss;

boss.securityAccess(bank1);

return 0;

}



**Finish**

C# it won’t compile unless securityInfo is also internal as protected stops a nonderived class from accessing. Making the classes internal changes northing. SecurityInfo and name are also never set. In Java, the complier didn’t complain about protected and ran normally. For C++, I made the Bank class a friend of the Manager class giving the Manager class access to the Bank class’s protected and private members

Friend class only works for specific classes labeled friend where as package and internal work on a more general level.