Marco Seman

Programming Language CIS 400

Professor Jie Shen

December 14, 2018

ASSIGNMENT 3 QUESTION 10

# **Question 10 (30 points).** Polymorphism is one cornerstone of Object-Oriented Programming. This question has the following setting of a polymorphism problem:

**Class Animal:**

**Attributes:**

Leg, an integer, *private* 🡪 The number of legs

**Behaviors:**

Animal(), *public* 🡪 default constructor sets Leg=4

talk(), *public* 🡪 Takes no input parameter, prints out a string

“Animal can’t talk !”

**End Class Animal**

**Class Cow:**

**Attributes:**

An object of type Cow has all the attributes of an

object of type Animal

**Behaviors:**

An object of type Cow has all the behaviors of an object of

type Animal

Additionally, Cow overrides the talk( ) method:

talk(), *public* 🡪 Takes no input parameter, prints out a string

“Moo !”

**End Class Cow**

**Class Pig:**

**Attributes:**

An object of type Pig has all the attributes of an

object of type Animal

**Behaviors:**

An object of type Pig has all the behaviors of an object of

type Animal

Additionally, Pig overrides the talk( ) method:

talk(), *public* 🡪 Takes no input parameter, prints out a string

“Grunt !”

**End Class Pig**

**Class Snake:**

**Attributes:**

An object of type Snake has all the attributes of an

object of type Animal

**Behaviors:**

An object of type Snake has all the behaviors of an object of

type Animal

**End Class Snake**

In a main( ) routine, do the following things:

* Create a one-dimension array of type Animal with four elements
* Create an instance of each class type, i.e., Animal, Cow, Pig and Snake
* Assign each instance to an element of the array we just created
* Do a loop over each array element, and call the talk( ) method of that array element to demonstrate how the polymorphism works.

This question consists of four following sub-questions:

1. Implement the above polymorphism problem in C++, and test the results in you main( ) routine;

**C++: (**Used Visual Studios as my IDE)

#include <iostream>

using namespace std;

class Animal {

    int legs;

public:

    Animal() {legs = 4;}

    virtual void talk() { cout << "Animal Can't Talk!\n"; }

};

class Cow : public Animal {

public:

    void talk() { cout << "Moo!\n"; }

};

class Pig : public Animal {

public:

    void talk() { cout << "Grunt!\n"; }

};

class Snake : public Animal { };

int main()

{

    Animal Arr[4];

    Cow cow;

    Pig pig;

    Snake snake;

    Arr[1] = cow;

    Arr[2] = pig;

    Arr[3] = snake;

    Arr[0].talk();

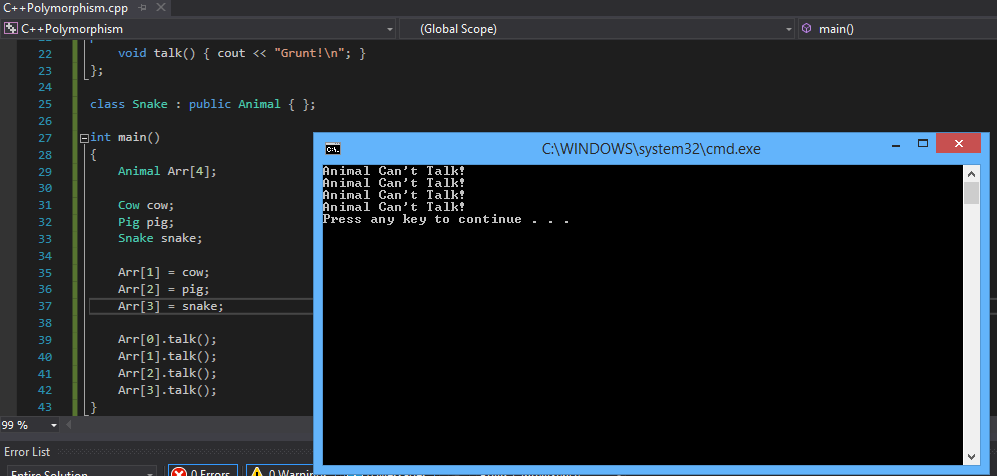
    Arr[1].talk();

    Arr[2].talk();

    Arr[3].talk();

}

A)



JAVA:

# (b) Implement the above polymorphism problem in Java, and test the results in you main( ) routine;

**class** Animal{

// set variable for static binding

**private** **int** leg = 4;

**void** talk()

{

System.***out***.println("Animal Can't Talk!");

}

**public** **void** print()

{

System.***out***.println("Animal has " + leg + " legs");

}

}

// This is dynamic binding

**class** Cow **extends** Animal

{

**void** talk()

{

System.***out***.println("MOO!");

}

}

**class** Pig **extends** Animal

{

**void** talk()

{

System.***out***.println("Grunt!");

}

}

**class** Snake **extends** Animal

{

**public** **void** print()

{

System.***out***.println("Snake has: ");

**super**.print();

}

}

**class** Polymorphism {

**public** **static** **void** main(String[] args)

{

Animal a[]= **new** Animal[3];

a[0] = **new** Cow();

a[1] = **new** Pig();

a[2] = **new** Snake();

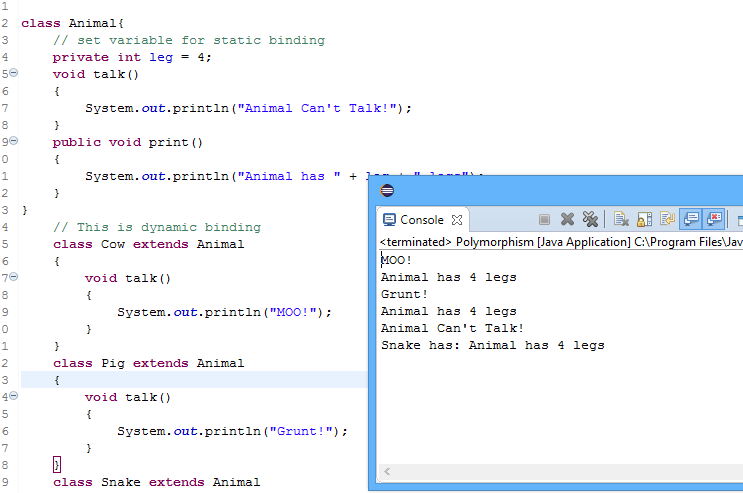
**for**(**int** i = 0; i<a.length;i++) {

a[i].talk();

}

}

}



# (c) Implement the above polymorphism problem in C#, and test the results in you Main( ) routine;

C#: (Used <https://repl.it/repls/DeadDampDesignmethod>) as my IDE

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PolymorphismCSharp{

public class Animal

{

  private int legs; // static binding

  public Animal(){legs=4;}

  public virtual void talk()

  {

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

  }

  public virtual void printLegs(){

    Console.WriteLine(legs + " legs");

  }

}

  public class Cow : Animal

{

    public override void talk()

    {

      Console.WriteLine("Moo!");

    }

    public override void printLegs(){

    Console.WriteLine("Cow has: ");

    base.printLegs();

}

}

  public class Pig : Animal

{

    public override void talk()

    {

      Console.WriteLine("Grunt!");

    }

    public override void printLegs(){

    Console.WriteLine("Pig has: ");

    base.printLegs();

}

}

  public class Snake : Animal

  {

    public override void printLegs(){

    Console.WriteLine("Snake has: ");

    base.printLegs();

  }

  }

class Test {

public static void Main(){

Animal[] a = new Animal[3];

a[0] = new Cow();

a[1] = new Pig();

a[2] = new Snake();

foreach(Animal draw in a)

{draw.talk();}

foreach(Animal draw in a)

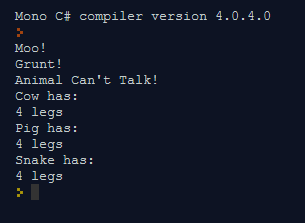
{draw.printLegs();}

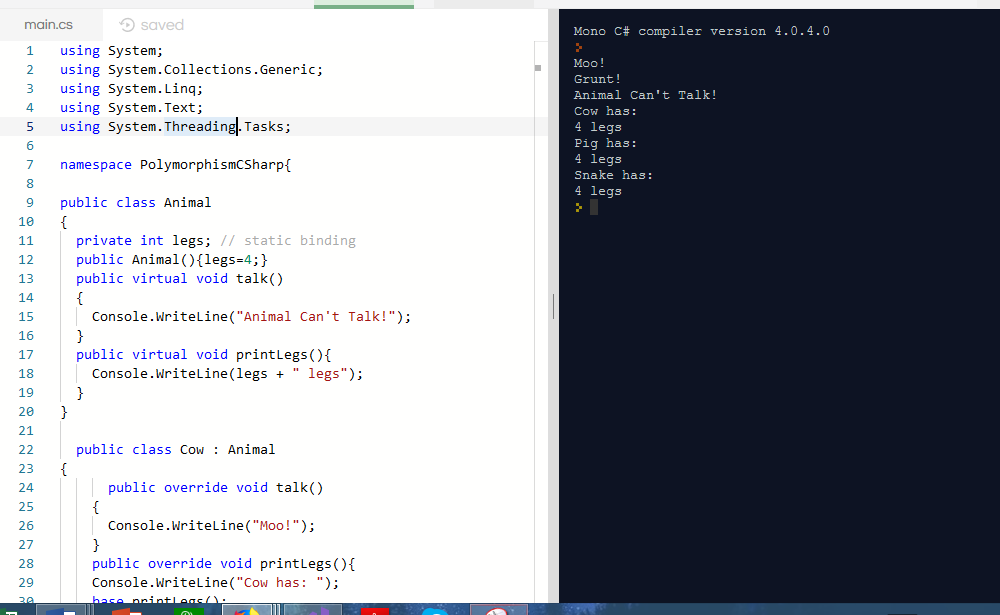
Console.ReadKey();

}

}

}





# (d) Summarize what are your findings about the polymorphism of C++, Java, and C#. What are the differences between them ?

C# and Java both have dynamic binding. So, whatever is in the class of that array element is what is printed by the talk function. C++ does not have dynamic binding. So, whatever is in the base class ends up being what is printed by the talk() function.