University of Technology, Jamaica
School of Computing and Information Technology

OOP Principles Lab Practical Four - Inheritance Part One

Objective:
The objectives of this lab practical are to allow students to be able to:

- implement inheritance relationships in code using C++ and Java
- demonstrate how inherited attributes are accessed by sub classes
- instantiate base class and derived class objects
- invoke local and inherited methods of a derived class

This lab demonstrates how to allow one class to inherit from another class, and how to access the inherited protected attributes. You will create objects of both the parent and child classes, and call methods from the objects created.

Exercise One

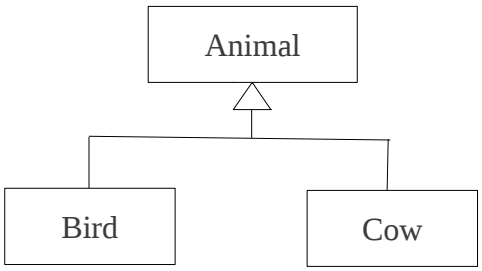
Enter and run both implementations (in C++ and/or Java) of the classes depicted in the UML class diagrams below. Note carefully how inheritance is accomplished in both languages. There is a simple problem preventing both programs (in C++ and Java) from compiling successfully – which has to do with how you access the inherited attributes. Examine the error messages and see if you can figure out the error and correct it. The solution is on the page 12.

Design using UML

Animal
- Age : float
+ Animal() + Eat() : void

Bird
- NumberEggsLaid : int
+ Bird(int, float) + Fly() : void + Display()

Cow
- MilkProduced : float
+ Cow(float,float) + Walk() : void + Display() : void



C++ Implementation

```
//Animal.h

//these two lines and the #endif at the end
//prevent redifinition of the Animal class
#ifndef Animal_Header
#define Animal_Header

#include <iostream>
using namespace std;

class Animal {
	private:
		float age;

	public:
		//Default constructor
		Animal()
		{
			age = 0;
			cout << "\nA new animal was just created" << endl;
		}
		//Eat method
		void Eat()
		{
			cout << "Like every animal, this one can eat" << endl;
		}
};
#endif
```

```
//Bird.h

#include "Animal.h"
#include <iostream>
using namespace std;

//this is a derived class of the base class Animal
class Bird : public Animal {
	private:
		int NumberEggsLaid;

	public:
		//primary constructor
		//expects initial value for NumberEggsLaid attribute which is in this
		// class plus the age attribute which is in the parent class
		Bird(int  NEL, float Ag)
		{
			NumberEggsLaid = NEL;
			age = Ag;
		}
		//Fly method
		void Fly()
		{
			cout << "This bird is now flying" << endl;
		}
		//Display the value of the attributes
		void Display()
		{
			cout << "This bird has laid " << NumberEggsLaid << " eggs"
				<< endl;
			cout << "This bird is " << age << " years old" << endl;
		}
};
```

```
//Cow.h

#include "Animal.h"
#include <iostream>
using namespace std;

//This is a child class of Animal
class Cow : public Animal {
	private:
		float MilkProduced;

	public:
		//primary constructor
		//expects initial value for MilkProduced attribute which is in this
		// class plus the age attribute which is in the parent class
		Cow(float  MP, float Ag)
		{
			MilkProduced = MP;
			age = Ag;
		}
		//Walk method
		void Walk()
		{
			cout << "This cow is now walking" << endl;
		}
		//Display the value of the attributes
		void Display()
		{
			cout << "This cow has produced " << MilkProduced <<
				" litres of milk" << endl;
			cout << "This cow is " << age << " years old" << endl;
		}
};
```

//Driver.cpp

```
#include "Animal.h"
#include "Bird.h"
#include "Cow.h"

//main
int main (int argc, char *argv[])
{
	//create a new Animal object
	Animal A;
	//Invoke its eat method
	A.Eat();

	//Create a new Bird object
	//Note the 3.5f indicates a float value
	Bird B(4, 3.5f);
	//Invoke its eat method
	B.Eat();
	//Invoke its eat method
	B.Fly();
	//Invoke its eat method
	B.Display();

	//Create a new Bird object
	//Note the 50.0f and 8.0f indicates float values
	Cow C(50.0f, 8.0f);
	//Invoke its eat method
	C.Eat();
	//Invoke its eat method
	C.Walk();
	//Invoke its eat method
	C.Display();

	return 0;
}
```

Java Implementation

```java
//Animal.java

public class Animal {
	private float age;

	//Default constructor
	public Animal()
	{
		age = 0;
		System.out.println("\nA new animal was just created");
	}
	//Eat method
	public void Eat()
	{
		System.out.println("Like every animal, this one can eat");
	}
}
```

```java
//Bird.java

//this is a derived class of the base class Animal
public class Bird extends Animal {
	private int NumberEggsLaid;

	//primary constructor
	//expects initial value for NumberEggsLaid attribute which is in this class
	//plus the age attribute which is in the parent class
	Bird(int  NEL, float Ag)
	{
		NumberEggsLaid = NEL;
		age = Ag;
	}
	//Fly method
	public void Fly()
	{
		System.out.println("This bird is now flying");
	}
	//Display the value of the attributes
	public void Display()
	{
		System.out.println("This bird has laid "+NumberEggsLaid+" eggs");
		System.out.println("This bird is "+age+" years old");
	}
}
```

```java
//Cow.java

//This is a child class of Animal
public class Cow extends Animal {
	private float MilkProduced;

	//primary constructor
	//expects initial value for MilkProduced attribute which is in this class
	//plus the age attribute which is in the parent class
	Cow(float  MP, float Ag)
	{
		MilkProduced = MP;
		age = Ag;
	}
	//Walk method
	public void Walk()
	{
		System.out.println("This cow is now walking");
	}
	//Display the value of the attributes
	public void Display()
	{
		System.out.println("This cow has produced "+MilkProduced+
			" litres of milk");
		System.out.println("This cow is "+age+" years old");
	}
}
```

//Driver.java

```
public class Driver {

	//main method
	public static void main(String[] args) {
		//create a new Animal object
		Animal A = new Animal();
		//Invoke its eat method
		A.Eat();

		//Create a new Bird object
		//Note the 3.5f indicates a float value
		Bird B  = new Bird(4, 3.5f);
		//Invoke its eat method
		B.Eat();
		//Invoke its eat method
		B.Fly();
		//Invoke its eat method
		B.Display();

		//Create a new Bird object
		//Note the 50.0f and 8.0f indicates float values
		Cow C  = new Cow(50.0f, 8.0f);
		//Invoke its eat method
		C.Eat();
		//Invoke its eat method
		C.Walk();
		//Invoke its eat method
		C.Display();
	}

}
```

Exercise Two

Shark
- NumberOfTeet : int
+ Shark(int,float) + Swim() : void + Display() : void

Assume the Shark class above is also a sub class of the Animal class.

a) Add code to your program to implement the Shark class. The Swim method should display “This shark is swimming”. The Display method should print the value of all the attributes in the Shark object, including any inherited ones.

b) Add code to your main() in the Driver file to create a new Shark object, then invoke the Eat(), Swim() and Display() methods for that object.

Exercise Three

a) Add default, primary and copy constructors to all four classes in your program and instantiate new objects of each class in such a way that each of the constructors for each class gets called in main().

b) Code mutators and accessors for each attribute (including the inherited ones) in each class, then demonstrate how to use these methods on objects created in main().

Exercise Four – homework

Cover the code for the four classes in the UML diagrams above, then see if you can code the entire program by hand on paper. Next, start a new Java or C++ project and enter your hand-coded program. Does it work? Compare it with the solution on these pages and adjust your version if necessary until you get it right.

Solution:

Do not reach this section until you have completed exercise one.

Change the private access specifier to protected in the Animal class so that the child classes Bird and Cow (and later Shark) can have access to the inherited attribute. Remember, all attributes from the parent class are inherited by the children, but they can only be accessed by the children if those attributes have an appropriate access specifier such as protected.

The corrected code follows for both languages:

C++

```
class Animal {
	protected:
		float age;

	public:
		//Default constructor
		Animal()
		{
			age = 0;
			cout << "\nA new animal was just created" << endl;
		}
		//Eat method
		void Eat()
		{
			cout << "Like every animal, this one can eat" << endl;
		}
};
```

Java

```
public class Animal {
	protected float age;

	//Default constructor
	public Animal()
	{
		age = 0;
		System.out.println("\nA new animal was just created");
	}
	//Eat method
	public void Eat()
	{
		System.out.println("Like every animal, this one can eat");
	}
}
```