University of Technology, Jamaica
School of Computing and Information Technology

OOP Principles Tutorial Six - Inheritance Part Two

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

- write both abstract and concrete methods and classes in an inheritance hierarchy
- create concrete child classes which inherit from abstract parent classes

Class Exercise

Class A has no attributes but has one default constructor and a abstract method called Display( ). The constructor in A prints “constructor in class A” on the screen.

1

Class B has no attributes of itself, but inherits from class A. It has one default constructor which prints "constructor in class B" on the screen, and one concrete method Display( ) which prints “Display from class B” on the screen.

Class C has no attributes of itself, but inherits from class B. It has one default constructor which prints "constructor in class C" on the screen, and one concrete method Display( ) which prints “Display from class C” on the screen.

a) Write code to implement class A above.

b) Write code to implement class B above.

c) Write code to implement class C above.

d) Write a driver file to display “Testing inheritance concepts from inheritance lecture two” on the screen. In the driver, create a pointer (C++) or reference (Java) for each of the three classes. Can this be done? Comment out any line which is illegal in the programming language (C++ or Java) you are using.

e) Using pointers (C++) or references (Java), instantiate objects for each of the three classes. Can this be done? Comment out any line which is illegal in the programming language (C++ or Java) you are using.

f) Using the pointer (C++) or reference (Java) created for the object of class C from question e) above, invoke the Display( ) method. Is this allowed? Comment out any line which is illegal in the programming language (C++ or Java) you are using.

Homework

g) Change the Display( ) method in class B to an abstract one.
h) What would have to change in your driver file from question d) above, to make the program still run?

<u>Answers to tutorial questions</u>

a) Write code to implement class A above.

<u>C++</u>
```
class A
{
	public:
	A()
	{
		cout << "constructor in class A" << endl;
	}

	virtual void Display() = 0;
};
```

<u>Java</u>
```
public abstract class A
{
	public A()
	{
		System.out.println("constructor in class A");
	}

	public abstract void Display();
}
```

b) Write code to implement class B above.

<u>C++</u>

```
class B : public A
{
	public:
	B()
	{
		cout << "constructor in class B" << endl;
	}

	void Display()
	{
		cout << "Display from class B" << endl;
	}
};
```

Java

```
public class B extends A
{
	public B()
	{
		System.out.println("constructor in class B");
	}

	public void Display()
	{
		System.out.println("Display from class B");
	}
}
```

c) Write code to implement class C above.

C++

```
class C :  public B
{
	public:
	C()
	{
		cout << "constructor in class C" << endl;
	}

	void Display()
	{
		B::Display();
		cout << "Display from class C" << endl;
	}
};
```

Java

```
public class C extends B
{
	public C()
	{
		System.out.println("constructor in class C");
	}

	public void Display()
	{
		super.Display();
		System.out.println("Display from class C");
	}
}
```

d) Write a driver file to display “Testing inheritance concepts from inheritance lecture two” on the screen. In the driver, create a pointer (C++) or reference (Java) for each of the three classes. Can this be done? Comment out any line which is illegal in the programming language (C++ or Java) you are using.

C++
```
int main ()
{
	cout << "Testing inheritance concepts from inheritance lecture two" << endl;
	A *ref1;	//ok even though A is abstract because no object is created
	B *ref2; //ok for concrete class too - no object created
	C *ref3;	//ok for concrete class too - no object created

	return 0;
}
```

Java

```
public class Driver2
{
	public static void main(String[] args)
	{
		System.out.println(
			"Testing inheritance concepts from inheritance lecture two");
		A ref1;		//ok even though A is abstract because no object is created
		B ref2; //ok for concrete class too - no object created
		C ref3;		//ok for concrete class too - no object created

		obj3.Display();
	}
}
```

e) Using pointers (C++) or references (Java), instantiate objects for each of the three classes. Can this be done? Comment out any line which is illegal in the programming language (C++ or Java) you are using.

C++
```
int main ()
{
	cout << "Testing inheritance concepts from inheritance lecture two" << endl;
	A *ref1;	//ok even though A is abstract because no object is created
	B *ref2; //ok for concrete class too - no object created
	C *ref3;	//ok for concrete class too - no object created

	//A *obj1 = new A; //will cause an error because A is abstract
	B *obj2 = new B;  //ok for object of class B to be created
	C *obj3 = new C;  //ok for object of class C to be created

	return 0;
}
```

Java

```
int main ()
{
	cout << "Testing inheritance concepts from inheritance lecture two" << endl;
	A *ref1;    //ok even though A is abstract because no object is created
	B *ref2; //ok for concrete class too - no object created
	C *ref3;    //ok for concrete class too - no object created

	//A obj1 = new A(); //will cause an error because A is abstract
	B *obj2 = new B;  //ok for object of class B to be created
	C *obj3 = new C;  //ok for object of class C to be created

	return 0;
}
```

f) Using the pointer (C++) or reference (Java) created for the object of class C from question e) above, invoke the Display( ) method. Is this allowed? Comment out any line which is illegal in the programming language (C++ or Java) you are using.

C++

```
int main ()
{
	cout << "Testing inheritance concepts from inheritance lecture two" << endl;
	A *ref1;    //ok even though A is abstract because no object is created
	B *ref2; //ok for concrete class too - no object created
	C *ref3;    //ok for concrete class too - no object created

	//A obj1 = new A(); will cause an error because A is abstract
	B *obj2 = new B;  //ok for object of class B to be created
	C *obj3 = new C;  //ok for object of class C to be created

	obj3->Display();
	return 0;
}
```

Java

```
public class Driver2
{
	public static void main(String[] args)
	{
		System.out.println(
			"Testing inheritance concepts from inheritance lecture two");
		A ref1;     //ok even though A is abstract because no object is created
		B ref2; //ok for concrete class too - no object created
		C ref3;     //ok for concrete class too - no object created

		//A obj1 = new A(); will cause an error because A is abstract
		B obj2 = new B(); //ok for object of class B to be created
		C obj3 = new C(); //ok for object of class C to be created

		obj3.Display();
	}
}
```