Uttara InfoSolutions

www.uttarainfo.com

Uttara Inheritance Practicals 1

I want you to go through the below problem and then follow the steps I have mentioned in detail below (do not directly start coding). In the unzipped contents, the Animal and Vehicle discussed example have been coded. You can compile them all in one shot and go through the code of TestAnimal.java and TestVehicle.java.

a) A device has a name and can do something. TV, Printer and Microwave are devices. Using TV, you can watch movie; using Printer, you can print; using Microwave, you can cook (all these are doSomething() implementations - so override them in the respective classes).When a device is made to dosomething, it shouts out its name as well. An Electrician is one who can test any device. When he is asked to test a device, he will make it do something and test it. A TV can switchChannel as well (delta functionality). So if a tv is given to be tested to an electrician, he will make the tv dosomething and switch channel as well. Write a tester class to test how devices, electricians work.

If Device has a private String name, do Printer, TV, etc subclasses have a name? If yes, how to access it in overridden doSomething() method? Try it out.

Follow these steps one by one to implement the above problem. Do not copy paste the code but type it yourself. Follow each step:

1) Frist build the Device.java class in source folder with the Device class definition.

```
Device
        String name;
        public void doSomething()
        {
                SOP(name+" doing something");
        }
```

Compile the .java file from source to classes.

2) Build the TestDevice tester class with main(), Test device working:

```
Device d = new Device();
d.name = "Davy";
d.doSomething();
```

Compile, execute TestDevice and verify working.

3) Add 2 constructors to Device - one a no-arg constructor and another which takes a String as parameter (in this set the passed param to name state).
Put SOPs in the constructors. Recompile Device.java

4) Re-execute the TestDevice program without recompilation. Do you see the SOP from the no-arg constructor? How come even without re-

compiling TestDevice, the latest Device.class was used? (class loa...happens at ru..)

5) Build the TV class and make it extend Device. Put a no-arg constructor in TV and put an SOP. In tester class, plug in Device d = new TV(); and compile, execute. Do you see 2 constructors getting executed? (constructor chaining. which constructor of parent Device is getting called and why?). Verify in the tester class, if d.doSomething() is called, parent implementation is getting picked up. Do you understand now that TV can do everything that Device can do? Print the name and see the output as well.

6) Override doSomething in TV.java and plug in a SOP. Now execute the tester. This time the overridden implementation of TV`s doSomething should be executed (as method call to method body linking happens at runtime depending on the object on which the method is called). Will this change if you had TV t1 = new TV(); ?? Check and verify your understanding that method call to method body linking happens only at runtime depending on type of object.

7) Add another method called switchChannel() in the TV with a SOP. Verify in TestDevice, whether you can invoke d.switchChannel()? Why not?

```
TV t = (TV) d;
t.switchChannel();

// parent ref can be downcasted to subclass data type but it will work
// only if d is pointing to TV object
```

Is this working?

8) Now create Printer.java and Microwave.java and make them become IS-A Device. Put no-arg constructors with SOPs. Override doSomething() methods in them with SOPs. Compile them.

9) In tester class, comment out earlier code in main(). Check the following:
Device d = new TV(); // (verify 2 constructors are called)
d.doSomething();
d = new Printer(); // (verify 2 constructors are called again - constructor chaining happens per object creation)

//can I use the same d ref to point to new obj?
//what happens to TV obj now?

d.doSomething();
d = new Microwave(); // (verify 2 constructors are called to create Microwave object)
d.doSomething();

Compile and run the tester class. Now are you able to understand that depending only on object, the JVM will pick method implementation at runtime?

Code this in tester class after the above code now (d is now pointing to Microwave obj):

System.out.println("d instance of Microwave? "+ (d instance of Microwave));
System.out.println("d instance of Printer? "+ (d instance of Printer));
System.out.println("d instance of TV? "+ (d instance of TV));
System.out.println("d instance of Device? "+ (d instance of Device));
System.out.println("d instance of String? "+ (d instance of String));

Now compile. Will all statements compile? Why doesn't last statement compile? Remember you can use instanceof operator only between rela…da..ty.. Now comment the last statement. Recompile. Check mentally what output should be printed. Then run and verify. Remem-

ber instanceof operator checks if left hand ref is pointing to an object that IS-A right hand operand.
Now make d point to TV object and verify instanceof working. Now make d point to Device object and verify instanceof working. Do you now correctly understand the working of instanceof?

10) Create an Electrician.java (not IS-A Device!).Now create a testDevice(Device d) -> coding to the parent ref method. You should call d.doSomething() in its body. Compile.

```
public class Electrician
{
	// this is polymorphic method. do you know why so?
	public void doSomething(Device d)
	{
		d.doSomething();
		// do you know which body of doSomething() will
		// be picked by seeing this code ??
	}
}
```

11) In TestDevice->main()-> remove all earlier code (simply comment it), create an Electrician object, create a TV object, Printer object, Microwave object and ask the electrician to test each of the devices by passing their reference, one at a time. Put proper SOPs before asking electrician to test each device in main(). Verify each time by recompiling and executing is he testing the correct device and is the polymorphic code of testDevice() is working correctly. Now do you understand how the testDevice() is polymorphic and how it works is not fixed but totally depends on which object ref is passed?
Try invoking testDevice(null) and see what happens. Remember null is a literal that can be passed wherever a ref is expected. What is NullPointerException? How to not get this runtime failure in testDevice()?

12) Now add instanceof check in testDevice() with respect to TV and then downcast and invoke t.switchChannel() in the method. Verify if the Electrician is switching channel only if you pass him a TV by re-running your tester class. Only when TV is given, he should switch the channel.

13) Now make your name state variable private and setter/getters in Device. Now change code accordingly in the subclasses and verify if all subclasses have a name (remember each time you need to access name now you need to invoke getName() in subclasses directly and in tester class, ref.getName()).

14) Every time you create an object of TV / Microwave / Printer, no -arg constructor of Device is getting called. I want you to code over-loaded constructors in all the 3 classes to accept a string n as para-meter. Does TV have a name? (remember name is private in Device) Can you initialise n to name directly in TV parameterised constructor? If not, how can you? Put super(n); does what? Which statement this must be in TV constructor? Put SOP before super(n) and see if it compiles. When and all should we chain constructors explicitly?

14) For any doubts, ask Lab Instructors or me.

b) (do this if you have not done it in the previous lab)
Create a class X. Create an int field a and assign 10 to it.
Create an instance initializer and put in SOP("in inst init 1 a = “+a); and change a value to 20.
Create another instance initializer and put in SOP("in inst init 2 a = “+a); and change a value to 30.
Create a no-arg constructor, print value of a and then assign 40 to it.
In a Tester class, in main()-> create object of X and then print obj.a to the monitor.
Do you understand how the initialisation is happening?
Now add a static variable in X named b = 15. Create 2 static init where you print the value of b and change them like earlier. Add printing of

X.b in Tester class and then run it. Do you understand now how the initialisation is occurring. Create one more object of X and see if the static initializers are getting fired.
Now create a subclass of X named Y with same class defn (copy paste) except change the name of instance var to c and d and SOPs to indicate Y class.
Now in Tester class, create object of Y. Think what should be the order of execution of init and constructors first. Verify if that is correct.

Ask doubts if you have any. It is important you understand order of steps during object creation!

X.java (do not copy paste code. type it) ->

```
public class X
{
	int a = 10;
	static int b = 15;
	static
	{
		System.out.println("in static init 1 of X b = "+b);
		b = 25;
	}
	{
		System.out.println("in inst init 1 of X a = "+a);
		a = 20;
	}
	{
		System.out.println("in inst init 2 of X a = "+a);
		a = 30;
	}
	static // does order matter for init execution?
	{
		System.out.println("in static init 2 of X b = "+b);
		b = 35;
```

```
	}
	public X()
	{
		System.out.println(“in constr of X a = "+a);
		a = 40;
	}
}

Y.java->
public class Y extends X
{
	int c = 10;
	static int d = 25;
	static
	{
		System.out.println("in static init 1of Y d = "+d);
		d = 35;
	}
	static
	{
		System.out.println("in static init 2 of Y d = "+d);
		d = 45;
	}
	{
		System.out.println("in inst init 1 of Y c = "+c);
		c = 20;
	}
	{
		System.out.println("in inst init 2 of Y c = "+c);
		c = 30;
	}
	public Y()
	{
		System.out.println(“in constr of Y c = "+c);
		c = 40;
```

```
    }
}
```

Order of execution when an object of Y is created for first time:
1) Class loading for parent.class
    a) class Class object created for parent
    b) all static members allocated mem and
    created in class Class object
    c) static field initializers executed
    d) static initializers gets executed of parent
2) Class loading for child.class
    a) class Class object created for child
    b) all static members allocated mem and
    created in class Class object
    c) static field initializers executed
    d) static initializers gets executed of child
3) allocates mem, creates object, creates all
instance variables of child and parent (all parent classes),
irrespective of access specifiers!
4) Enter constructor. Execute first statement super() -> to reach parent most class in class hierarchy
5) inst intializers of parent executed
6) constructor body of parent executed
7) inst initalizers of child executed
8) constructor body of child executed!

Create 2 objects of Y in tester class and verify what SOPs are not printed and why?

Now in in X, code a parameterised constructor like this

```
public X(int x)
{
    SOP("in X param constr with x = "+x);
    a = x;
}
```

Now in in Y, code a parameterised constructor like this

```
public Y(int y)
{
        SOP("in Y param constr with y = "+y);
        c = y;
}
```

When you create new Y(99) in tester class, which constructor of X is being called by default? Now add super(y) as first statement in above constructor and see which constructor of X is called. Now do you understand constructor chaining correctly?

c) Animals can eat,sleep and dance. Hippo,tiger,croc etc are animals. Every animal has name. An animal snores when it sleeps. A vet can treat animals. When an animal is treated, it is made to dance, eat and sleep. A Croc can swim, a hippo can smoke. When a Vet is given a croc, he will make it swim as well. When he is given a hippo to treat, he will make it smoke as well.
Test the working of Vets polymorphic code as per the steps followed in the earlier example. Put all classes in relevant packages.

d) (do this if you have not done it in previous lab)
Person has a name and age. A person has a number of pet names(20max) which he obtains over a period of time. He can dance; if his age is less than 25 he can do chacha. If his age is greater than 25, he does the waltz. He can sing too and when he is asked to sing, he uses his petnames to form the song (randomly). Write a tester program to test persons.

Design:

Person

```
        private String name;
        private int age;
```

```
	private String[] petNames = new String[20]; // why are we using
the array here?
	int count=0;

	public void sing()
	{
		String song = "";
		for(int i = 0; i < petNames.length; i++)
		{
			int n = (int)(20 * Math.random());
			song = song + petNames[n];

		}
		SOP(song);
	}
	public void addPetName(String n)
	{
		if(count < petNames.length)
			petNames[count++] = n;
		else
			SOP(..);
	}
	public boolean searchPetName(String n)
	{
		// search in the petNames array whether a name equal to n
exists...and if yes, return true, else return false;

		for(String s : petNames)
		{
			if(s.equals(n))
				return true;
		}
		return false;

	}
```

Important Polymorphism usage problems:

e) There are Bags. You can use the bag to store items (for which the user of the bag will give max number of items at construction time). You can then retrieve items from the bag. An item has a name and a price. Caps, notebooks, pens, lipstick are all items. A bag can be used to search for a given item. You can request the bag to give you the total of prices of all the items in the bag. Write a tester class to test creation of bags, items, then add items into the bags and invoke the various methods of bag to test how to search, retrieve, get total, get individual price of items.

```
Item
        String name;
        double price;
        public Item()
        {

        }
        public Item(String n, double p)
        {
                name = n;
                price = p;
        }
```

Pen, NoteBook are subclasses of Item with a similar param constr where we explicitly invoke super.

```
Pen extends Item
```

```
Bag
        String name;
```

```
Item[] items; // create Item.java and compile it first
// bag has-a item(s) -> 1..n multiplicity

int count = 0; // to keep track of how many items added

public Bag(String n, int size)
{
        name = n;
        items = new Item[size];
}
public void addItem(Item it)
{

}
public boolean searchItem(String n)
{
        //loop over the items array and compare each items name
        // with n and if equal, return true, else after comparing
        // the entire array, return false
}
public double getItemPrice(String n)
{

}
public double getTotal()
{
        // add all the items prices to a sum var and return it
}
public Item getItem(int pos)
{
        // return item ref from the position
}
```

f) An Account has a balance (double) and can be used to checkBalance (return the balance), deposit (add to the balance by taking in a

double), withdraw (reduces the balance). BankAccount is an account. A BankAccount will have a maxBalance (balance should not go higher than this) and minBalance (balance should not go lower than this). CreditCardAccount is an Account as well. In CCA, there is a minBalance that can be negative. A Person has a Account (only 1), a name and can be asked to buy item (given a string name and double amount). When asked to buy an item, the person will first check the balance in the account, then withdraw as much is required to buy the item and SOPs it. A Person can be asked to debit money as well by passing a double amount which he will deposit into the account he is linked with. Which type of an Account should a Person be linked with? Should we hardcode? How to create the Person class in such a way that a Person object can be linked to either of the accounts. Create 2 Person objects with different accounts and see how they behave. This is an important problem.