Uttara InfoSolutions

www.uttarainfo.com

Uttara Practicals 3:

Theory: Instance variables (variables declared outside method and inside class) can be accessed by instance methods and we use this for implementing state impacts behaviour (will discuss in complete detail tomorrow in class). First go through Hippo.java, Duck.java, compile and run them and understand the control flow.

0) Create a Duck. A Duck has a tailSize (int). A Duck can swim. When you ask a Duck to swim, it says so (print to monitor). Design and test Duck class usage as we did for Hippo. Create 2 Duck objects in tester class (TestDuck->main()), with 2 references pointing to it (lets say d1,d2). Set the tailSize states to 10 and 20. Invoke d1.swim() & d2.swim(). Compile, run and test your understanding so far. Now make d1.size = d2.size. SOP d1.size and d2.size to monitor. What has changed and why? Now make d1.size = 30. SOP d1.size and d2.size to monitor. What has changed and why? Now make d1=d2. SOP d1.size and d2.size to monitor. What has changed and why? Now make d1=d2=null. Can you access any of the 2 objects now? Now do d1= new Duck(); How many objects are reachable? Do you understand the answers to all the questions? If not, please ask.

1) There are Cows. A Cow has a name and can moo. When you ask a Cow to moo, it shouts out its name. Design a class for Cow and the tester code. After testing if cows work correctly, create 2 references and then point to same object. Verify what happens if you change the state of the object using one ref when you invoke behaviour using the second.

```
Cow
        String name;
        public void moo()
        {
                SOP("I am cow and I am moo mooing…"+name);
        }
```

(See example code Cow.java and TestCow.java. Code on your own after understanding the same).

2) Check if instance variables of type string, int and boolean are given default values. Check if local variables are given default values.
How to do this? Create a class with 4 instance variables with String, boolean, int and boolean as datatypes. Create a Tester class with main where you create an object and using reference print the instance variable values to monitor using SOP.
Ex:

```
class X
{
        int i;
        String str;
        public static void main(String[] args)
        {
                X obj = new X();
                System.out.println(obj.i);
                System.out.println(obj.str);
        }
}
```

3) A Pen has inkQty (int), colour (string) and can be used to write and refill. A text must be given for it to write. A quantity must be given to refill. If there is ink then the pen will write the text given to it (SOP). Refill works by taking in the int qty to add to the existing inkQty. First as a class designer, on paper apply OOAD and arrive at the class design. Then create the class implementation and create a tester class to create 2 pen objects, give it inkQty and ask it to write. First code this on your own. If you cannot get it correctly, then see Pen.java and TestPen.java. Then fix the code and rerun. Do not see the code first.

4) There are Persons. A person has a age, name. Persons can eat, sleep. A person sleeps more if his age is < 40 and eats less. If the

persons age is >= 40, he sleeps less but eats more. Design a Person class and test it.

```
Person
	String name;
	int age;
	public void eat() {
		if(age < 40)
			sdfsdkf
		else
			adfkgldafgk
	}
	public void sleep() {
		….
	}
```

5) There are Dogs. Every Dog has a name and a size. Dogs can bark. If the size of the dog is > 10, it "meows". If the size <=10, then as many times, it "bow wow" its name to the monitor. Test Dog design. After testing the same, make the size variable private and then add setSize()/ getSize() method. See how this impacts your tester class. What check should you add in bark() to ensure that even if the class user has not set size and invokes bark, he gets scolded with a message?

6) *Important problem* Cars can be started, driven, reversed or stopped. You have to start the car to drive/reverse/stop it. When car is being driven / reversed, the fuel reduces. Once the car has no fuel, the car stops. Every car has a name. Write a tester class to test cars. How to know whether the car has started and then only being asked to be driven? Try to design on paper first and then see the below design. Understand how state can impact behaviour.

```
Car
	- name
	- fuelQty
	- isStarted : boolean
	- start()
		// if fuelQty > 0
		//	set isStarted = true;
	- drive()

		// check if the car has started, only if
		yes,
		if(isStarted)
			// check if fuelQty>0, then only drive
```

```
                // reduce fuelQty by 1 unit
                // if fuelQty==0
                        stop();
```

- reverse()
- stop()
        // isStarted = false!

7) There are Accounts. Every Account has a number (string), owner (string) and balance (double). You can withdraw(double amt), debit(doubt amt) and checkBalance(). When you withdraw, the balance should reduce accordingly. When you debit, the balance will increase accordingly. Code an Account class and test it by creating 2 account objects? How do you stop over withdrawal?

8) There are TVs. A TV has a name and channel that is being displayed. You can increment/decrement channel. You can change the channel to a given number as well. You can ask the TV to display. When a TV is asked to display, it will print the channel num, the volume.TV has volume (int). You have to switch on the TV first before you can operate the channels or increase or decrease the volume. Design and test TV working.

9) In TestPerson -> main(),

Person p = null;

invoke p.eat(); What happens and why?

In a for loop, create 10 person objects and set it to p reference. How many objects are reachable after the control comes out of the for loop?