Uttara InfoSolutions

www.uttarainfo.com

Initializers & Statics - 8

1) 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.

See this below code only after following the instructions above.

X.java->

```
public class X
{
	int a = 10;
	static int b = 15;
	static
	{
		System.out.println("in static init 1 b = "+b);
		b = 25;
	}
	{
		System.out.println("in inst init 1 a = "+a);
```

```
        a = 20;
    }
    {
        System.out.println("in inst init 2 a = "+a);
        a = 30;
    }
    static // does order matter for init execution?
    {
        System.out.println("in static init 2 b = "+b);
        b = 35;
    }
    public X()
    {
        System.out.println("in constr of X a = "+a);
        a = 40;
    }
    public X(int p)
    {
        System.out.println("in constr of X a = "+a);
        a = p;
    }

}
```

Recollect in your mind when to use instance field init / instance init / constructor and ask us if you have doubts in this. Test all permutation / combinations pls.

2) Test wrapper class usage => do you know how to convert a primitive into an object & back?, do you know how to access constants in Wrapper classes? Test the working of parsing & Character static methods.

3) Create 3 static methods called process in class called X. Accept Integer i, long l & float f as params. Now test calling this method from main() by using X.test(5), X.test(5L), X.test(5.5), X.test(5.5f), X.test(<byte>), X.test(<Long ref>), X.test(<Short ref>), X.test(<short primitive>) and test your understanding of overloading.

3) Code a method to accept a string and return a string with all uppercase letters converted to lowercase and vice versa with every other char retained as is.
Ex: aBC$1 => Abc$1

4) *Challenge problem* Build a method to accept a string and parse it to an int without using library parse functions!

5) *Challenge problem* Build a method to accept 2 strings as param and check if first string content exists in another without using lib functionality.

6) *important* 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 names of 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()
{
        SOP(..);
}
public Item(String n, double p)
{
        SOP(..);
        name = n;
        price = p;
}
```

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]; // why are we creating the array object here?
}
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
}
```

7) Create an IntStack class. An IntStack can be used to push(int), int pop(). An IntStack can hold a max of 10 ints. Think and create IntStack class and test it for correct working. What should it have to hold all the ints pushed into it?