Blogs Core java PageNo: Go << >>

--Tag-- Search

Swap two numbers without using third variable

```
public class SwapTwoNum {

 public static void main(String[] args) {
  int a = 10;
  int b = 5;

  a = a+ b;
  b = a - b;
  a= a - b;

  System.out.println("After swapping, Num1= " + a + " and Num2= " + b);
 }

}  Output After swapping, num1= 5 and num2= 10
```

Create Object

Q: How you can create an object without new keyword ?

Ans : Using reflection API Class.

//User u = new User(); is replaced by

User u = (User) Class.forName("User").newInstance();

by four ways :
1) using new keyword
2) by class.forName().newInstance() method
3) by object factory
4) cloning

Immutable class

Q: How you can make an Immutable class?

Ans : An immutable class is one whose state can not be changed once created.

An IMMUTABLE CLASS can be made by

1. Make all fields final and private.
2. Initialize final attributes with help of 'parametrized Constructor'.
3. Define only getter methods.
4. Don't provide "setter" methods — methods that modify fields or objects referred to by fields.
5. Don't allow subclasses to override methods. The simplest way to do this is to declare the class as final.

JDK 1.5 New features

Q What are the new features of JDK 1.5 ?

Answer:

1. Generics
It is the way to provide data type to a Collection. Data type is passed to Compiler at compile time. You do not need to typecast (type conversion) while retrieving data from collection

2. Autoboxing/unboxing:
It is the auto conversation of primitive data to Wrapper objects or Wrapper objects to primitive classes.

When primitive data is converted into Object, it is called Auto-boxing. When an Object is converted into its primitive data then it is called auto-unboxing. Both are called autoboxing-unboxing.

For Example

Integer ageObj = 25; // Autoboxing

int age = ageObj; //Autounboxing

3. Enumerations:
the enum keyword creates a typesafe, ordered list of values (such as day.monday, day.tuesday, etc.). It is used to define set of contacts.

4. Swing: New skinnable Look and Feel:
It is called synth are provided to visual object (widgets).

5. Var args:
Earlier only fix number of parameters were allowed to pass to a method as per method signature. But now method signature can allow you to pass dynamic number of parameters to a method at runtime.

The last parameter of a method can now be declared using a type name followed by three dots (e.g. Void drawtext(string... Lines)). In the calling code any number of parameters of that type can be used and they are then placed in an array to be passed to the method, or alternatively the calling code can pass an array of that type.

6. Enhanced for each loop:
the for loop syntax is extended with special syntax for iterating over each member of either an array or any iterable, such as the standard collection classesfix the previously broken semantics of the java memory model, which defines how threads interact through memory.

7. Automatic stub generation for RMI objects.
8. Concurrency utilities :
Static imports concurrency utilities in package java.util.concurrent.

9. Scanner Class :

for parsing data from various input streams and buffers.

10. Assertions:
An assertion is a conditional expression that should evaluate to true if and only if your code is working correctly. If the expression evaluates to false, an error is signaled.

11. java.util.StringBuilder class:It is just like StriingBuffer but StringBuffer is synchronous by String Builder is asynchronous.
12. Annotations (Metadata):Annotations are tags. These tags are used in Class definitions to pass additional data about Class, Methods and Attributes to Compiler. As per tags, additional processing is done by some Metadata-aware utilities at runtime.

Annotations are generally a way to add meta-data information to an element. An element can be a class, method, field, constructor, variables etc. This meta-data are processed by the tools i.e. compilers, javadoc, etc.

An annotation type definition takes an "at" (@)sign, followed by the interface keyword plus the annotation name.
Public @interface Auditor{

}

"SUNRAYS" Vs new String("SUNRAYS")

Q: What are differences between following statements ?

1. *String name = "SunilOS";*
2. *String name = new String("SunilOS");*

A : String object is assigned from 'String literal pool' in statement No#1. A new String object is created out of the 'String literal pool' in statement No#2 .

Examples

```
String str1 = "Hello";
String str2 = "Hello";
System.out.print(str1 == str2);
```

Output will be **true** because str1 & str2 are pointing to same string from String Literal Pool

```
String str1 = "Hello";
String str2 = new String("Hello");
System.out.print(str1 == str2 + " ");
System.out.print(str1.equals(str2));
```

Output will be **false** because str1 & str2 will be two different instances of String "Hello".

String literal Pool : To cut down the number of String objects created in the JVM, the String class keeps a pool of strings. Each time your code create a string literal, the JVM checks the string literal pool first. If the string already exists in the pool, a reference to the pooled instance returns. If the string does not exist in the pool, a new String object instantiates, then is placed in the pool. Java can make this optimization since strings are immutable and can be shared without fear of data corruption.

See http://home.sunrays.co.in/string-literal-pool (http://home.sunrays.co.in/string-literal-pool) for more details