Manu Technologies

CORE JAVA TRAINING MANUAL

Author :Manohar S

Mob No:+91 9986781724

Volume :1

DAY 15(14-Nov-2015):-

JDK 5 New Features:-

1. **Assertion:** Assertion can be used to test your assumptions about the program. While executing assertion, it is believed to be true. If it fails, JVM will throw an error named AssertionError. It is mainly used for testing purpose.
   Syntax :
   assert condition1 : expression2;

   Example:

```
package com.box;

public class AssertionDemo {
public static void main(String[] args) {
int marks = 34;
assert marks >= 35 : "Student is not eligible ";
System.out.println("Student is eligible: " + marks);
}
}
```

OutPut:

```
Exception in thread "main" java.lang.AssertionError: Student is not eligible
at com.box.AssertionDemo.main(AssertionDemo.java:6)
```

Note: By default JVM will not raise assertion errors,we need to instruct JVM by passing –ea command as part JVM argument.

2. **Boxing & Unboxing:**
   By using **Autoboxing** feature we can eliminates the need for conversion between primitive data types such as int, float etc. and their respective wrapper classes Integer, Float, etc.
   Autoboxing Examples:
   (1) Integer marks = 31; // Autoboxing: 31 => Integer.valueOf(31)

   The implicit automatic conversion of wrapper class objects to primitive values is called '**Unboxing'**.

Unboxing Examples:

(3) if (marks > 25) { // Unboxing: age => age.intValue()
// do something
}

3. **Enums:** A Java Enum is a special Java type used to define collections of constants. More precisely, a Java enum type is a special kind of Java class. An enum can contain constants, methods etc.
   Example:-

```
package com.box;
```

```
public class EnumDemo {
public static void main(String[] args) {

EnumDemo en = new EnumDemo();
Gender gend = en.getGender(Gender.Male);//M,F
System.out.println("Gen:"+gend);
Gender all[] = Gender.values();
for(Gender g:all)
{
System.out.println("Content:"+g);
}
}

public Gender getGender(Gender gen)
{
return gen;
}
}
enum Gender
{
Male,Female
        }
```

OutPut:-

```
Gen:Male
Content:Male
Content:Female
```

4. **Annotations:** **Annotations** provide a little extra information about the classes we write. This feature is very useful because we can attach extra information to our code that may determine how it is used.
   Example:-

```
public class MyClass extends Object {
@Override
public String toString() {
return "My overridden toString() method!";
}
}
```

5. **Generics:** By using Generics we can achieve **compile-time type safety** to the Collections Framework and eliminates the necessity for type casting.We can also provide this feature to our own class also.
   Example 1:

```
package com.box;

import java.util.ArrayList;
import java.util.List;

public class GenericDemo {
public static void main(String[] args) {
List li = new ArrayList();
```

```
li.add(12);
li.add("10");
li.add(12.89);
li.add('c');

List<Integer> geList = new ArrayList<Integer>();
geList.add(122);
// geList.add("Abc");
// geList.add('c');

List<String> geList1 = new ArrayList<String>();
geList1.add("Abc");
// geList1.add(12);
}
}
```

Including Generics to our own class:-

```
Student<String> s = new Student<String>();
System.out.println("Out:"+s.getName("Manohar"));

Student<Integer> s1 = new Student<Integer>();
        System.out.println("Out1:"+s1.getName(908));

class Student<T>
{
public T getName(T name)
{
return name;
}
        }
```

6. **Static imports:**By using static imports static member of a class can be referred with out refering class name.
   Example:

```
package com.box;

import static java.lang.Math.*;
import static java.lang.System.*;

public class StaticImports {
public static void main(String[] args) {
long num = round(12.4);
out.println("Num:" + num);
}
}
```

7. **Var args:**If we want to pass n number of similar data type element values then we can use var arg parameter as mentioned below.Internally a variable argument is maintained as an array that can hold zero or one or more arguments of the same type.
   Example:-

```
package com.box;

public class VarArgDemo {
public static void main(String[] args) {
add("abc", 'k', 10, 12, 11, 45, 55, 66, 77, 88, 99, 100);
}

public static void add(String str, char c, int... i) {
System.out.println(str);
for (int j : i) {
System.out.println(j);
}
}
}
```

Output:-

```
abc
10
12
11
45
55
66
77
88
99
100
```

Note: Var arg parameter must be last argument in the method signature.

8. **Enhanced for loop:-** be used when you wish to step through the elements of the array in first-to-last order, and you do not need to know the index of the current element. In all other cases, the "standard" for loop should be preferred.

Syntax of enhanced for loop is:

```
for (data_type variableName: array_name)
```

Example:-

```
package com.box;

public class EnhancedForLoopDemo {
public static void main(String[] args) {

int all[] = {12,34,55,66,77,88,6,2,3,445,567};
for (int i : all) {
System.out.println(i);
}
}
```

```
}
```

OutPut:

```
12
34
55
66
77
88
6
2
3
445
567
```

9. **Scanner:** By using Scanner we can provide inputs from the keyboard.

Example:

```
package com.box;

public class Scanner {
public static void main(String[] args) {
java.util.Scanner sc = new java.util.Scanner(System.in);

System.out.println("Enter your Credit card num?");
int cardnum = sc.nextInt();
System.out.println("Enter CVC?");
int cvc = sc.nextInt();
System.out.println("Enter your Amount");
double amount = sc.nextDouble();
System.out.println("Credit card num:" + cardnum + " CVC:" + cvc
+ " Amount:" + amount);
sc.close();
}
}
```

OutPut:-

```
Enter your Credit card num?
12541252
Enter CVC?
125
Enter your Amount
1253.364
Credit card num:12541252 CVC:125 Amount:1253.364
```