**JAVA1.4/1.5**

## Java 4/5 Features

* [Java Assertion](https://www.javatpoint.com/assertion-in-java)
* [Java For-each Loop](https://www.javatpoint.com/for-each-loop)
* [Java Varargs](https://www.javatpoint.com/varargs)
* [Java Static Import](https://www.javatpoint.com/static-import-in-java)
* [Java Autoboxing](https://www.javatpoint.com/autoboxing-and-unboxing)
* [Java Enums](https://www.javatpoint.com/enum-in-java)
* [Java Annotations](https://www.javatpoint.com/java-annotation)
* [Java Generics](https://www.javatpoint.com/generics-in-java)

**1.Varargs:**

Allow methods to accept one or more arguments

While using the varargs, you must follow some rules otherwise program code won't compile. The rules are as follows:

There can be only one variable argument in the method.

Variable argument (varargs) must be the last argument.

**void** method(String... a, **int**... b){}//Compile time error

**void** method(**int**... a, String b){}//Compile time error

Example: static void display(String... val){

for(String str:val){

Sop(str);

}

}

**-------------------------------------------------------------**

**Java Static Import:**

The static import feature of Java 5 facilitate the java programmer to access any static member of a class directly. There is no need to qualify it by the class name.

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

**class** StaticImportExample{

**public** **static** **void** main(String args[]){

       out.println("Hello");//Now no need of System.out

    out.println("Java");

 }   }

**----------------------------------------------------**

**Generics in Java:**

The **Java Generics** programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time.

A generic class represents a class that is type safe. Before Generics we can store any type of object in collection i.e. non generic. Now generic forces the programmer to store specific type of objects.

**Advantage of Java Generics**

1. Type-Safety

2. Compile Time Checking

3. Type Casting is not require

Examples

Class MyClass<T>{

// generic class

}

class GenericMethods{

public static <E> void printArray(E[] inputArray){

int i = 1;

for(E e:inputArray){

System.out.printf("%S", e);

if( i < inputArray.length)

System.out.printf(",");

i++;

}

}

}

Generic Class: That is refer to any type is called generic class.

Generic methods: That can accept any type of argument.

Type Parameters: Naming convention of generic are important to learn

T-Type, E-Element, N-Number, K-Key, V-value

Generic Interface:

interface Test<T>{

void hello(T fruit)

}