**Import statements**

Java imports flow naturally from packages. In Java, there are TONS of useful **built in classes and methods** that allow us to do things like:

* Read the contents of a file / Create a file and populate it with contents
* Compare dates with each other (i.e. see if one date is before or after another)
* Send emails to anyone

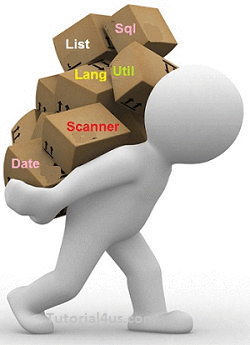
All of these classes are nicely organized in packages, and if you want to USE these classes, you’ll need to import them into your project. So, this means that before you can play around with Dates, you’ll need to import the Date object first!

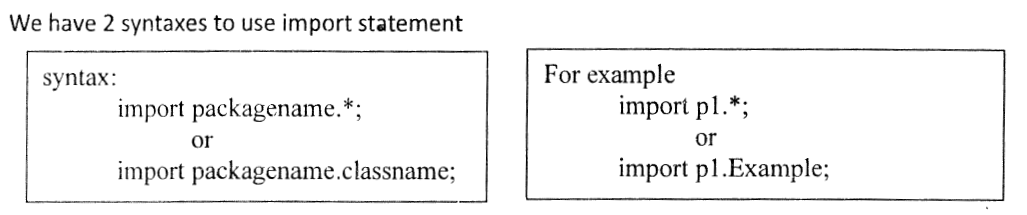
**Therefore,**

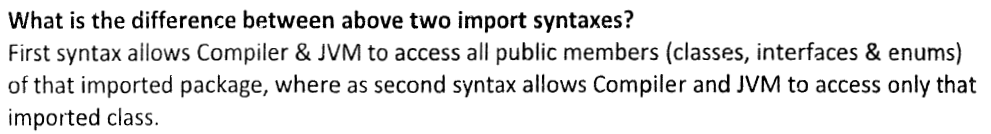
**Import** is a keyword in java language used to import the predefined properties of java API into current working java program.

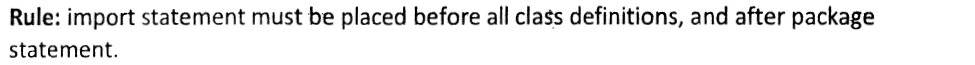
**Or**

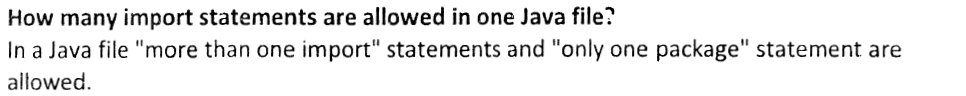
**Import** keyword is used to import built-in and user-defined packages into your java source file. So that your class can refer to a class that is in another package by directly using its name.

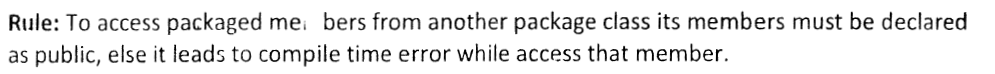


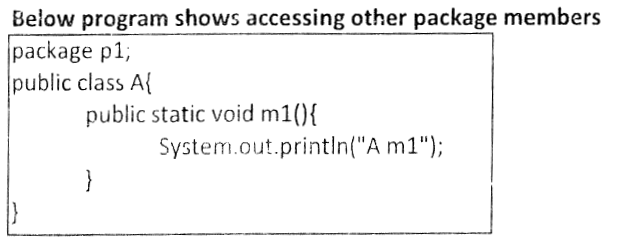
****

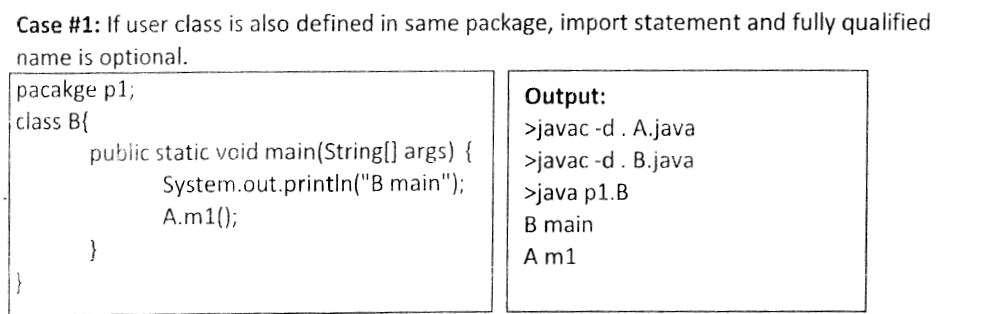
****

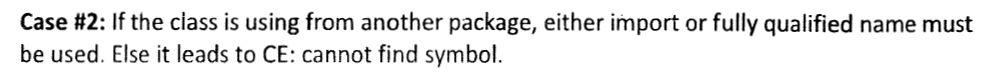
****

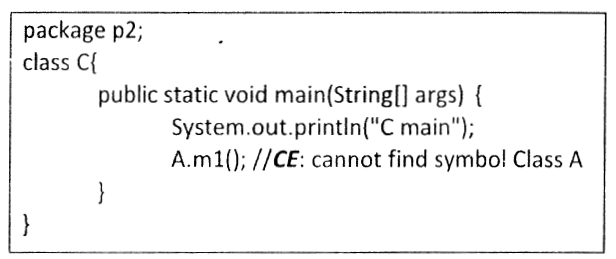
****

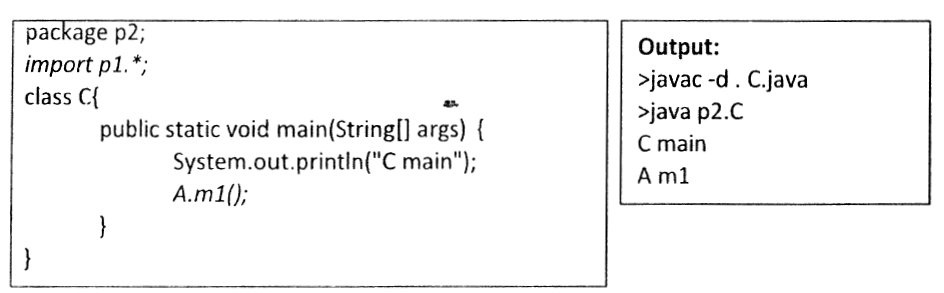
****

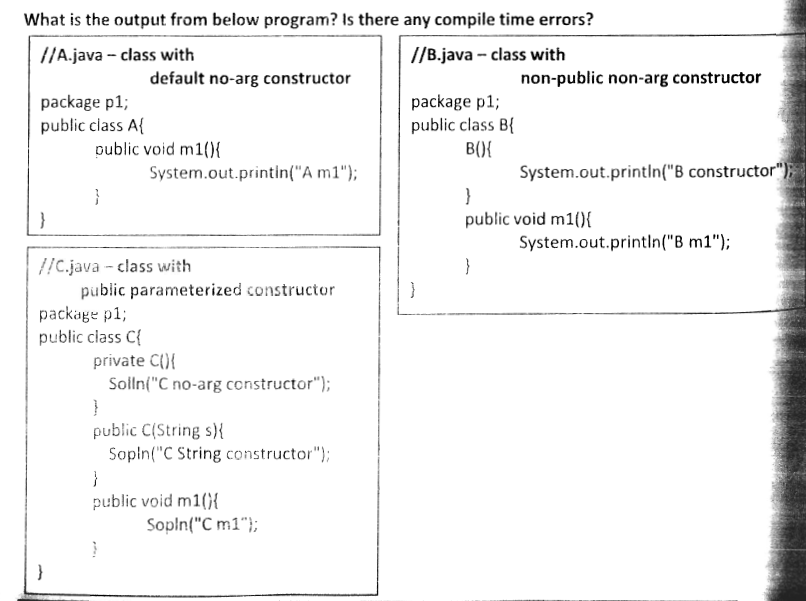
****

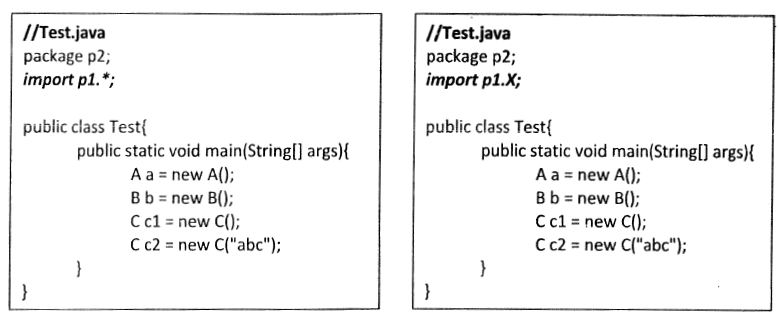
****

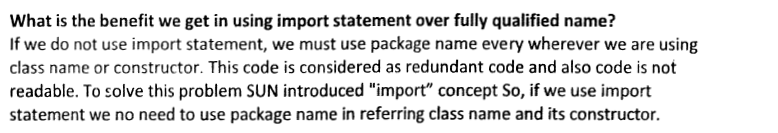
****

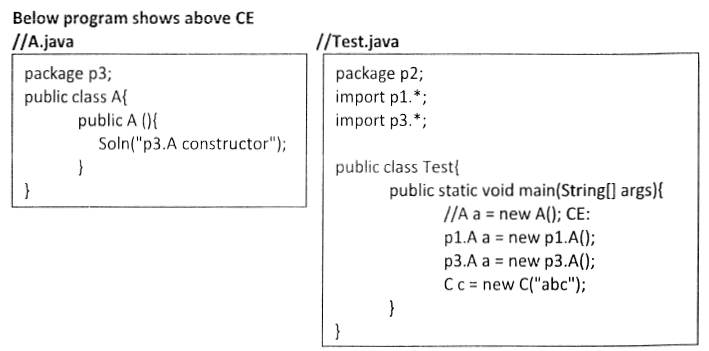
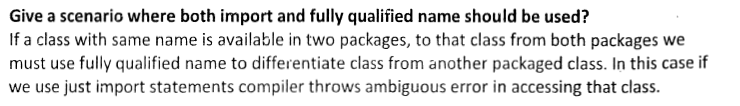
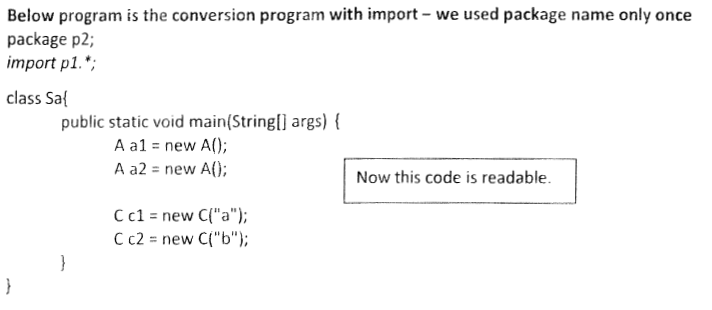
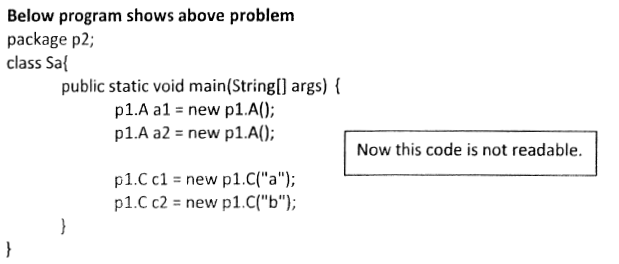
****

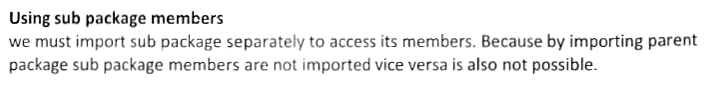
****

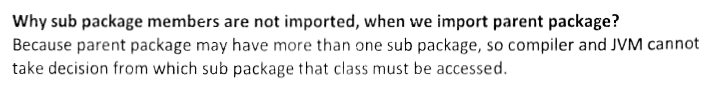
****

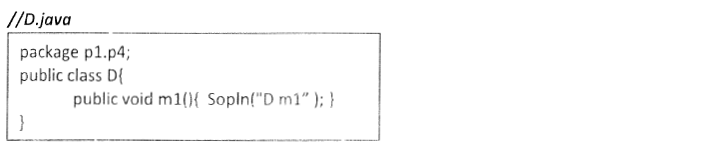
****

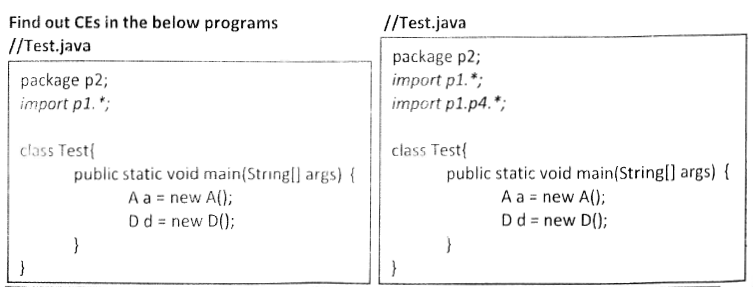
****

****

****

****

****

****

**Used to import specific class or interface in a java program.**

* Java API is a collection of package, package is a container which is collection of predefined classes and interfaces.
* If a class wants to use another class in the same package, the package name does not need to be used. Classes in the same package find each other without any special syntax.

***Java Import Classes: Advantages of Packages***

**1.** With a simple import statement, all the classes and interfaces can be imported.  
**2.** Java includes a provision to import only one class from a package.  
**3.** It avoids namespace problems (name conflicts). Two classes with the same name cannot be put in the same package but can be put in two different packages because a package creates its own namespace (folder).  
**4.** Access for the classes can be controlled.  
**5.** Classes and interfaces of same functionality can be grouped together.  
**6.** Because functionally all the classes are related, later their identification and determining the location become easier.  
**7.** Java packages are used to group and organize the classes.

**There are 3 different ways to refer to class that is present in different package**

1. **Using fully qualified name** (But this is not a good practice.)

***Example*** *:*

***class MyDate extends java.util.Date***

***{***

***//statement;***

***}***

1. **import the only class you want to use.**

***Example*** *:*

***import java.util.Date;***

***class MyDate extends Date***

***{***

***//statement.***

***}***

1. **import all the classes from the particular package**

***Example*** *:*

***import java.util.\*;***

***class MyDate extends Date***

***{***

***//statement;***

***}***

## *Example:*

Here, a class named Boss is added to the payroll package that already contains Employee. The Boss can then refer to the Employee class without using the payroll prefix, as demonstrated by the following Boss class.

***package payroll;***

***public class Boss***

***{***

***public void payEmployee(Employee e)***

***{***

***e.mailCheck();***

***}***

***}***

**What happens if the Employee class is not in the payroll package?**

The Boss class must then use one of the following techniques for referring to a class in a different package.

* The fully qualified name of the class can be used.

**Example:**

**payroll.Employee**

* The package can be imported using the import keyword and the wild card (\*).   
    
  **Example:**

**import payroll.\*;**

* The class itself can be imported using the import keyword.   
    
  **Example:**

**import payroll.Employee;**

**Note:**   
A class file can contain any number of import statements. The import statements must appear after the package statement and before the class declaration.

***Example :***

***package mypack;***

***import java.util.\*;***

But if you are not creating any package then import statement will be the first statement of your java source file.

***\*\*Caution*:**

The star form may increase compilation time—especially if you import several large packages. For this reason it is a good idea to explicitly name the classes that you want to use rather than importing whole packages. **However, the star form has absolutely no effect on the run-time performance or size of your classes.**

**Example** :

The package java.util contains many useful classes, such as **ArrayList** and **LinkedList**. If you need to use these, you can write declarations such as

***java.util.ArrayList myArrayList = new java.util.ArrayList();  
     java.util.LinkedList myLinkedList = new java.util.LinkedList();***

or you can import these classes

***import java.util.ArrayList;  
     import java.util.LinkedList;***

and simplify your declarations to:

***ArrayList myArrayList = new ArrayList();  
     LinkedList myLinkedList = new LinkedList();***

In addition, you can use an asterisk (**\***) as a "wild card" and import all the classes in a package at once:

**import java.util.\*;**

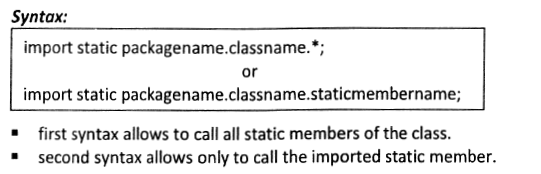
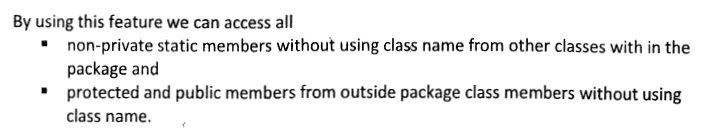
**Note:**

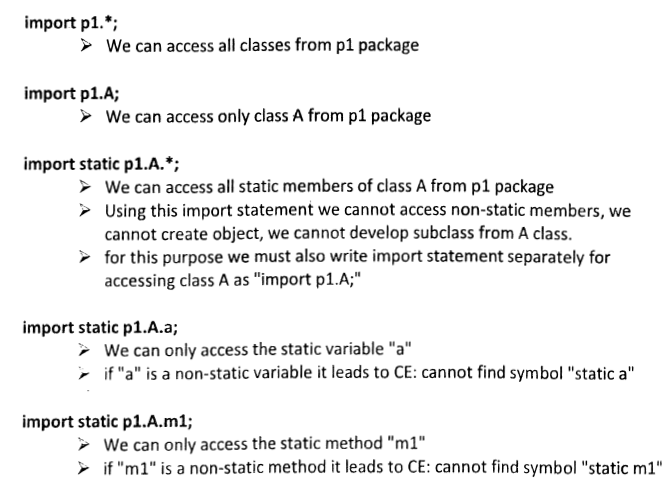
One important package, **java.lang**, is automatically imported for you (auto-imported for us for convenience).

**Static Import**



In java applications, if we declare any class or interface in a particular package then to use the class or interface in a java file either we have to use fully qualified name at each and every time of writing class or interface or we have to use import statements.

****

****

**Example:**

***A.java:***

***package*** *com.srini;*

***public******class*** *A {*

***public******void*** *m1(){*

*System.out.println("Hello");*

*}*

*}*

***B.java:****//Fully Qualified Name*

***class B {***

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

***com.srini.A a=new com.srini.A();  
a.m1();***

***}***

***}***

***Output:*** *Hello*

In order to eliminate the fully qualified name, we can use import statement.

***B.java://Using import statement***

***import com.srini.\*;***

***class B {***

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

***A a=new A();***

***a.m1();***

***}***

***}***

**Output::** Hello

***In java we are able to access static members either by using class name directly or by creating object for the respective class.***

***Example:***

***A.java:***

***package com.srini;***

***public class A {***

***public static int i=0;***

***public static void m1(){***

***System.out.println("Hello");***

***}***

***}***

***B.Java: //By using class name directly***

***import com.srini.A;***

***class B {***

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

***System.out.println(A.i);//0***

***A.m1();//Output: Hello***

***}***

***}***

*B.Java: //By using class name directly*

***import com.srini.A;***

***class B {***

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

***A a=new A();***

***System.out.println(a.i);//0***

***a.m1();//Output: Hello***

***}***

***}***

In the above program to eliminate class name while accessing static members we have to go for **“STATIC IMPORTS”.**

**Syntax:**

***import static package\_path.Entity\_Name.Member\_Name;***

It will import only the specified static members from the specified entity.

***import static package\_path.Entity\_Name.\*;***

It will import all the static members from the specified entity.

**Example 1:**

**A.java:**

***package com.srini;***

***public class A {***

***public static int i=0;***

***public static void m1(){***

***System.out.println("Hello");***

***}***

***}***

**B.java:**

***import static com.srini.A.\*;***

***class B {***

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

***System.out.println(i);//0***

***m1();//Output: Hello***

***}***

***}***

**Output:**0Hello

**Example 2:**

**A.java:**

***package com.srini;***

***public class A {***

***public static int i=0;***

***public int j=0;***

***public static void m1(){***

***System.out.println("Hello");***

***}***

***public void m2(){***

***System.out.println("Hello");***

***}***

***}***

**B.java:**

***import static com.srini.A.\*;***

***class B {***

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

***System.out.println(i);//0***

***System.out.println(j);  
//Compile Error: j cannot be resolved***

***m1();//Output: Hello***

***m2();  
//Compile Error: The method m2() is undefined for the type B***

***}***

***}***

**Static import**  **concept was introduced in 1.5 version.**

**Static import** is a feature that expands the capabilities of **import** keyword. It is used to import **static** member of a class. We all know those static members are referred in association with its class name outside the class. Using **static import**, it is possible to refer to the static member directly without its class name.

**There are two general form of static import statement.**

1. **import only a single static member of a class**

**Syntax**

**import static *package.class-name.static-member-name;***

**Example**

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

***//importing static method sqrt of Math class***

1. **imports all the static member of a class**

**Syntax**

***import static package.class-type-name.\*;***

**Example**

***import static java.lang.Math.\*;   
//importing all static member of Math class***

#### Example without using static import

***public class Test{***

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

***System.out.println(Math.sqrt(144));***

***}***

***}***

**Output**

12

#### Example using static import

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

***public class Test{***

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

***System.out.println(sqrt(144));***

***}***

***}***

**Output**

12

**Note:**

* According to SUN, Static import improves the readability of the code, but according to worldwide java experts (like us), static imports creates confusion and reduces readability of the code. Hence, if there is no specific requirement never recommended to use a static import.
* It is not recommended to use static import.
* Usually we can access static members by using class name but whenever we are using static imports, we can directly access static members (i.e. Fields and methods) i.e, it is not required to use class name, while accessing static members.

**Without Static import:**

***class Test{***

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

***System.out.println(Math.sqrt(4));***

***System.out.println(Math.max(10, 2));***

***System.out.println(Math.random());***

***}***

***}***

**Output:**  
2.0

10

0.37590688583144705

**With Static import:**

**Example 1:**

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

***class Test{***

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

***System.out.println(sqrt(4));***

***System.out.println(max(10, 2));***

***System.out.println(random());***

***}***

***}***

**Output:**

2.0

10

0.06849117853674602

**Example 2:** *import static java.lang.Math.sqrt;*

*class Test{*

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

*System.out.println(sqrt(4));*

*//****Output****: 2.0*

*System.out.println(max(10, 2));*

***//Compile Error****:   
//The method max(int, int) is undefined for the type Test*

*System.out.println(random());*

***//Compile Error:***  *//The method random() is undefined for the type Test*

*}*

*}*

**Explain about System.out.println statement?**

**Example 1:**

***class Test{***

***static String name1="Srini";***

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

***System.out.println(Test.name1.length());//5***

***}***

***}***

Here, Test.*name1*.length() represents different things.

**Test** 🡪 It is a class.

**name1** 🡪 Static variable of string present in Test class.

**length**() 🡪 It is a method present in String class.

**Example 2:**

***import*** *java.io.\*;*

***class*** *System{*

***static*** *PrintStream out;*

*}*

Here, System.out.println() represents different things.

**System**🡪 It is a class present in java.lang package.

**out**🡪 It is a static variable of type PrintStream present in System class.

**Println()**🡪It is a method present in PrintStream class.

**Example 3:  
  
*class*** *Test{*

***static*** *String name1="Srini";*

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

***static*** *String name2="Srini";*

*//****Compiler Error:*** *//Illegal modifier for parameter name1; only final is permitted*

*String name3="Srini";  
//****Compiler Error:*** *//Static cannot be resolved to a type*

*}*

*}*

**Example 4:**

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

***class System{***

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

***out.println("Hello");***

***out.println("Hi");***

***}***

***}*Output:**Hello

Hi

**Example 5:**

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

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

***class Test{***

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

***System.out.println(MAX\_VALUE);***

***}***

***}***

**Compile Error:**

Reference to MAX\_VALUE is ambiguous, both variable MAX\_VALUE in java.lang.Integer and variable MAX\_VALUE in java.lang.Byte match System.*out*.println(MAX\_VALUE);

**Note:**

Two packages contains a class or interface with the same is very rare. Hence, ambiguity problem is very rare in normal imports.

- But, 2 classes or interfaces can contain a method or variable with the same name is very common. Hence, ambiguity problem is very common in static import.

- While resolving static members compiler will give the precedence in the following order.

**a) Current class static members.**

**b) Explicit static import.**

**c) Implicit static import.**

**Example:**

***//import static java.lang.Integer.MAX\_VALUE;🡪Line 2***

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

***class Test{***

***//static int MAX\_VALUE=999;🡪 Line 1***

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

***System.out.println(MAX\_VALUE);//999***

***}***

***}***

**Output:**

a) If we comment line one, then we will get Integer class MAX\_VALUE 2147483647.

b) If we comment lines one and two, then Byte class MAX\_VALUE will be considered 127.

c) 1st priority, Current class static members, If we don’t comment any line, we will get 999.

**Which of the following import statements are valid?**

a) import java.lang.Math.\*;**//False**

b) import static java.lang.Math.\*;**//True**

c) import java.lang.Math; **//True**

d) import static java.lang.Math; **//False**

e) import static java.lang.Math.sqrt.\*; **//False**

f) import java.lang.Math.sqrt.\*; **//False**

g) import static java.lang.Math.sqrt();**//False**

h) import static java.lang.Math.sqrt; **//True**

**Diagram:**

Math;

**Normal import starts with**

Util.\*;

Math.sqrt;

**static import starts with**

Math.\*;

Usage of static import reduces readability and creates confusion. Hence, if there is no specific requirement never recommended to use static import.

**What is the difference between general import and static import?**

* We can use normal imports to import classes and interfaces of a package. Whenever we are using normal import we can access class and interfaces directly by their short name, it is not required to use fully qualified names.
* We can use static import to import static members of a particular class. Whenever we are using static import, it is not required to use class name. We can access static members directly.

**Example 1:**

***Line 1: package MyMsg;***

***Line 2: public class Msg {***

***Line 3: public static final int UPPER=1;***

***Line 4: public static final int LOWER=2;***

***Line 5: public static final int MIXED=3;***

***Line 6: private String msg;***

***Line 7: //Display a message in specific case.***

***Line 8: public void showMsg(int how){***

***Line 9: String str;***

***Line 10: switch(how){***

***Line 11: case UPPER:***

***Line 12: str=msg.toUpperCase();***

***Line 13: break;***

***Line 14: case LOWER:***

***Line 15: str=msg.toLowerCase();***

***Line 16: break;***

***Line 17: case MIXED:***

***Line 18: str=msg;***

***Line 19: break;***

***Line 20: default:***

***Line 21: System.out.println("Invalid command.");***

***Line 22: return;***

***}***

***Line 23: System.out.println(str);***

***}***

***Line 24: public Msg(String string) {   
 msg=string;   
 }***

***}***

**Example 2:**

***//Static import user-defined static fields.***

***Line 25:import MyMsg.\*;***

***Line 26:import static MyMsg.Msg.\*;***

***Line 27:class Test{***

***Line 28: public static void main(String[] args){***

***Line 29: Msg m=new Msg("Testing Static Import.");***

***Line 30: m.showMsg(MIXED);***

***Line 31: m.showMsg(LOWER);***

***Line 32: m.showMsg(UPPER);***

***}***

***}***

**Output**:

Testing Static Import.

testing static import.

TESTING STATIC IMPORT.

**Explanation:**

* In line 3,4,5 we are declaring static constants(final).
* We are using these constants for defining case labels in the switch statement at line 11,14,17.
* In Example 2, we are directly using those constants in line 30, 31, 32. i.e. by passing these constants as a method arguments, because we are using Static import statement at Line 26 i.e. **import static MyMsg.Msg.\*;**
* By using this statement all the static variables and methods are defined in the Msg class will available in the current program i.e. in the Example 2.

**Example 3:**

***Line 1:****import static java.lang.System.out;*

***Line 2:****class Test{*

***Line 3:****public static void main(String args[]){*

***Line 4:****out.println(“from print method”);*

***}***

**Output:**

from print method.

**Explantion**:

* In the Line 1, we are importing static variable out which is defined in predefined class **System,** by using static import.
* Now the static variable **out** available in **Test** class.So, we can directly use that variable in the **Test** class.
* In Line4, we are directly caling **println()** which is existed in the static variable **out.**

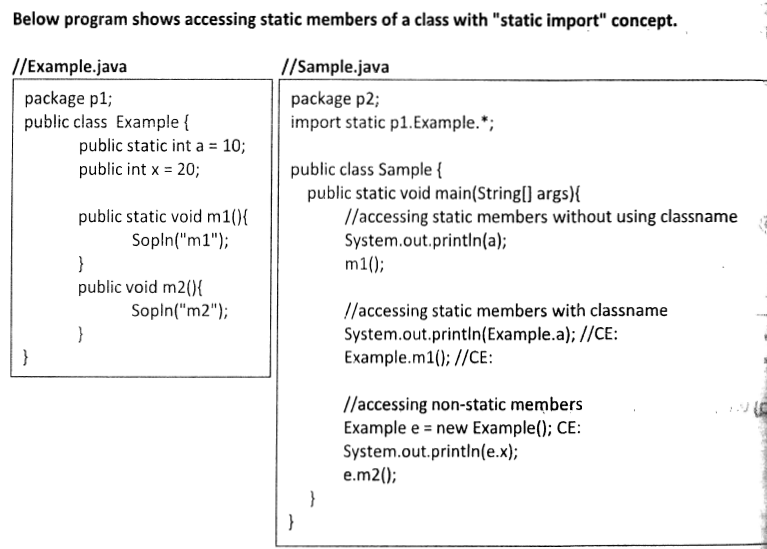
Because, it is available in the Test class.

## Difference between Inheritance and package

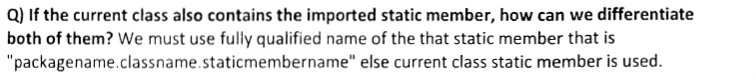
Inheritance concept always used to reuse the feature within the program between class to class, interface to interface and interface to class but not accessing the feature across the program.  
Package concept is to reuse the feature both within the program and across the programs between class to class, interface to interface and interface to class.

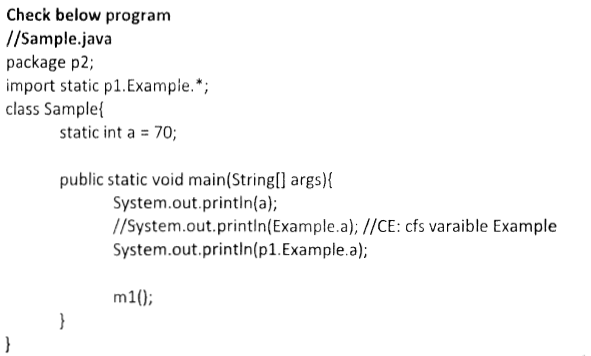
**Difference between package keyword and import keyword**

Package keyword is always used for creating the undefined package and placing common classes and interfaces.  
import is a keyword which is used for referring or using the classes and interfaces of a specific package.

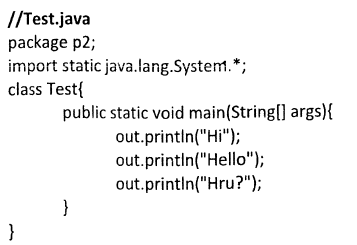
****

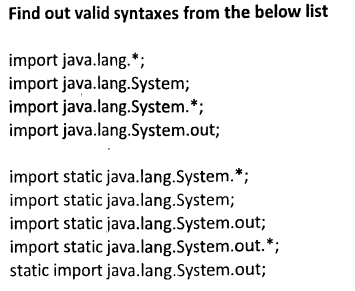
****

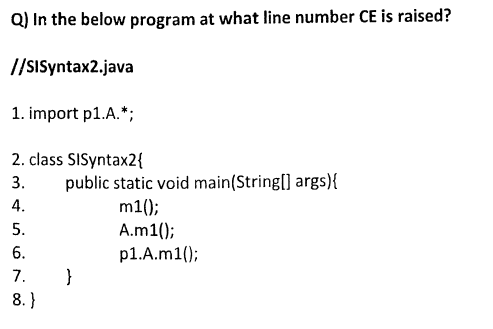
****

****

****

****

****

****

**SCJP :**

*class Test{*

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

*ArrayList l=new ArrayList();*

*}*

*}*

***Output:***

*Compile time error.*

*D:\Java>javac Test.java*

*Test.java:3: cannot find symbol*

*symbol : class ArrayList*

*location: class Test*

*ArrayList l=new ArrayList();*

* We can resolve this problem by using fully qualified name ***"java.util.ArrayList l=new java.util.ArrayList();".***

But problem with using fully qualified name every time is it increases length of the code and reduces readability.

* We can resolve this problem by using import statements.

***Example:***

*import java.util.ArrayList;*

*class Test{*

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

*ArrayList l=new ArrayList();*

*}*

*}*

**Output:**

*D:\Java>javac Test.java*

Hence whenever we are using import statement it is not require to use fully qualified names we can use short names directly. This approach decreases length of the code and improves readability.

***Case 1: Types of Import Statements:***

**There are 2 types of import statements.**

**1) Explicit class import**

**2) Implicit class import.**

**Explicit class import:**

**Example:**

*Import java.util.ArrayList*

* This type of import is highly recommended to use because it improves

readability of the code.

* Best suitable for Hi-Tech city where readability is important.

**Implicit class import:**

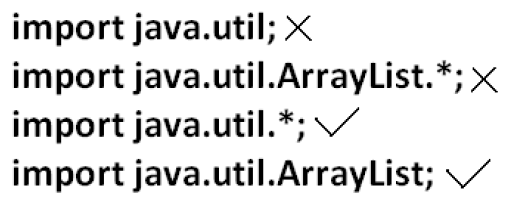
**Example:**

*import java.util.\*;*

* It is never recommended to use because it reduces readability of the code.
* Best suitable for Ameerpet where typing is important.

**Case 2:**

Which of the following import statements are meaningful ?

**

**Case 3:**

**consider the following code.**

*class MyArrayList extends java.util.ArrayList*

*{*

*}*

* The code compiles fine even though we are not using import statements because we used fully qualified name.
* Whenever we are using fully qualified name it is not required to use import

statement. Similarly whenever we are using import statements it is not require to use fully qualified name.

**Case 4:**

**Example:**

***import java.util.\*;***

***import java.sql.\*;***

***class Test{***

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

***Date d=new Date();***

***}***

***}***

**Output:**

***Compile time error.***

***D:\Java>javac Test.java***

***Test.java:7: reference to Date is ambiguous,***

***both class java.sql.Date in java.sql and class java.util.Date in java.util match***

***Date d=new Date();***

**Note:**

Even in the List case also we may get the same ambiguity problem because it is available in both util and awt packages.

**Case 5:**

While resolving class names compiler will always gives the importance in the following order.

**1. Explicit class import**

**2. Classes present in current working directory.**

**3. Implicit class import.**

**Example:**

import java.util.Date;

import java.sql.\*;

class Test{

public static void main(String args[]){

Date d=new Date();

}

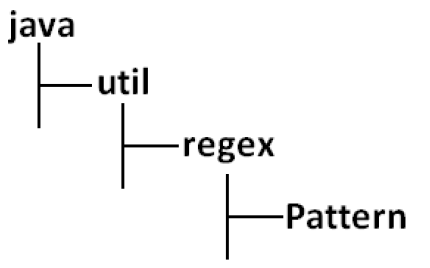
}

The code compiles fine and in this case util package Date will be considered.

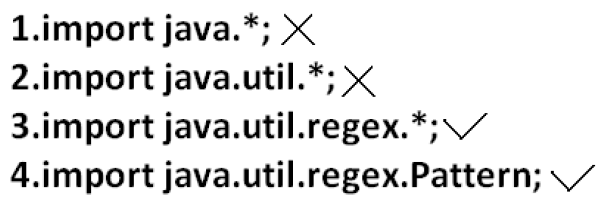
**Case 6:**

Whenever we are importing a package all classes and interfaces present in that package are by default available but not sub package classes.

**Example:**

**

**To use pattern class in our Program directly which import statement is required ?**

**

**Case7:**

In any java Program the following 2 packages are not require to import because these are available by default to every java Program.

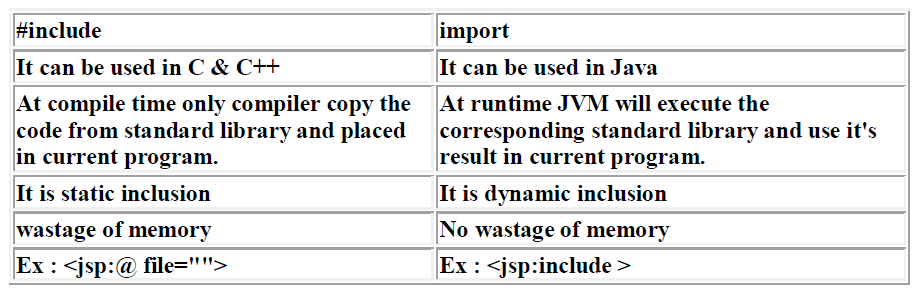
**1. java.lang package**

**2. default package(current working directory)**

**Case 8:**

**"Import statement is totally compile time concept"** if more no of imports are there then more will be the compile time but there is "**no change in execution time".**

**Difference between C language #include and java language import ?**

**

* In the case of C language #include all the header files will be loaded at the time of include statement hence it follows static loading.
* But in java import statement no ".class" will be loaded at the time of import statements in the next lines of the code whenever we are using a particular class then only corresponding ".class" file will be loaded. Hence it follows "**dynamic loading" or "load-on -demand" or "load-on-fly".**

**1.5 versions new features :**

***1. For-Each***

***2. Var-arg***

***3. Queue***

***4. Generics***

***5. Auto boxing and Auto unboxing***

***6. Co-varient return types***

***7. Annotations***

***8. Enum***

***9. Static import***

***10. String builder***

***Static import:***

This concept introduced in 1.5 versions. According to sun static import improves readability of the code but according to worldwide Programming exports (like us) static imports creates confusion and reduces readability of the code. Hence if there is no specific requirement never recommended to use a static import.

Usually we can access static members by using class name but whenever we are using static import it is not require to use class name we can access directly.

**Without static import:**

***class Test{***

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

***System.out.println(Math.sqrt(4));***

***System.out.println(Math.max(10,20));***

***System.out.println(Math.random());***

***}***

***}***

***Output:***

***javac Test.java***

***java Test***

***2.0***

***20***

***0.841306154315576***

**With static import:**

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

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

***class Test{***

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

***System.out.println(sqrt(4));***

***System.out.println(max(10,20));***

***System.out.println(random());***

***}***

***}***

***Output:***

***javac Test.java***

***java Test***

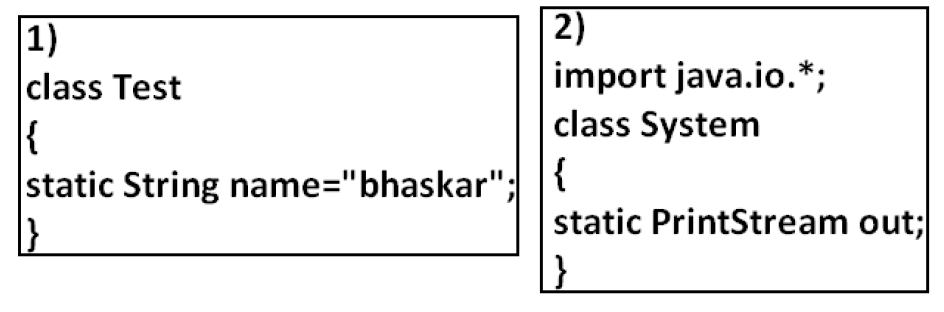
***2.0***

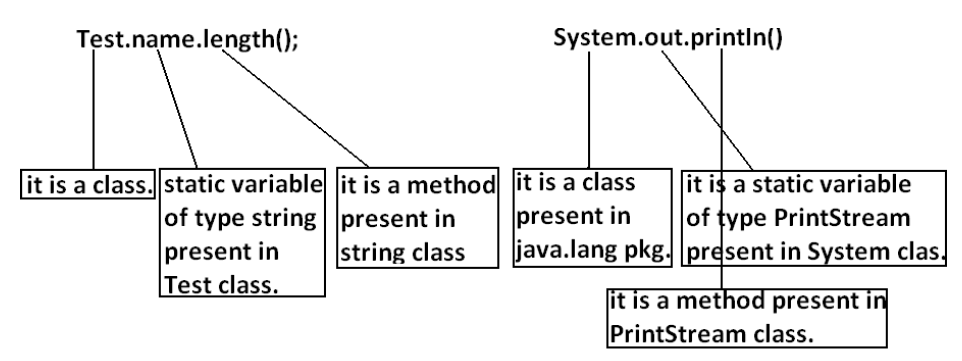
***20***

***0.4302853847363891***

**Explain about System.out.println statement ?**

**Example 1:**

******

******

**Example:**

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

*class Test{*

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

*out.println("hello");*

*out.println("hi");*

*}*

*}*

**Output:**

*javac Test.java*

*java Test*

*hello*

*hi*

**Example 4:**

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

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

*class Test{*

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

*System.out.println(MAX\_VALUE);*

*}*

*}*

***Output:***

*Compile time error.*

*D:\Java>javac Test.java*

*Test.java:6: reference to MAX\_VALUE is ambiguous, both variable MAX\_VALUE in java.lang.Integer and variable MAX\_VALUE in java.lang.Byte match*

*System.out.println(MAX\_VALUE);*

**Note:** Two packages contain a class or interface with the same is very rare hence

ambiguity problem is very rare in normal import.

But 2 classes or interfaces can contain a method or variable with the same name is very common hence ambiguity problem is also very common in static import.

While resolving static members compiler will give the precedence in the following order.

**1. Current class static members**

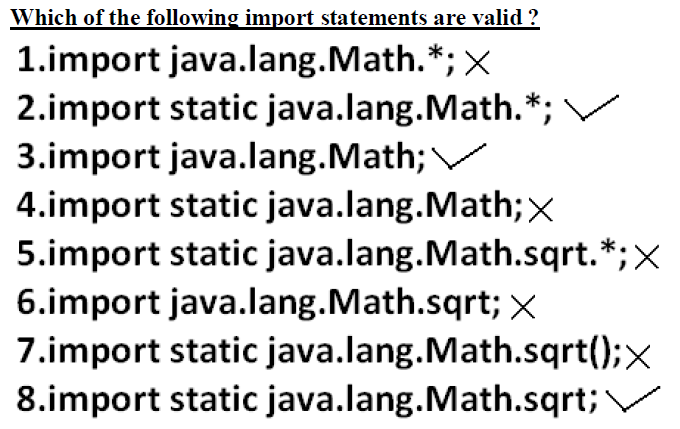
**2. Explicit static import**

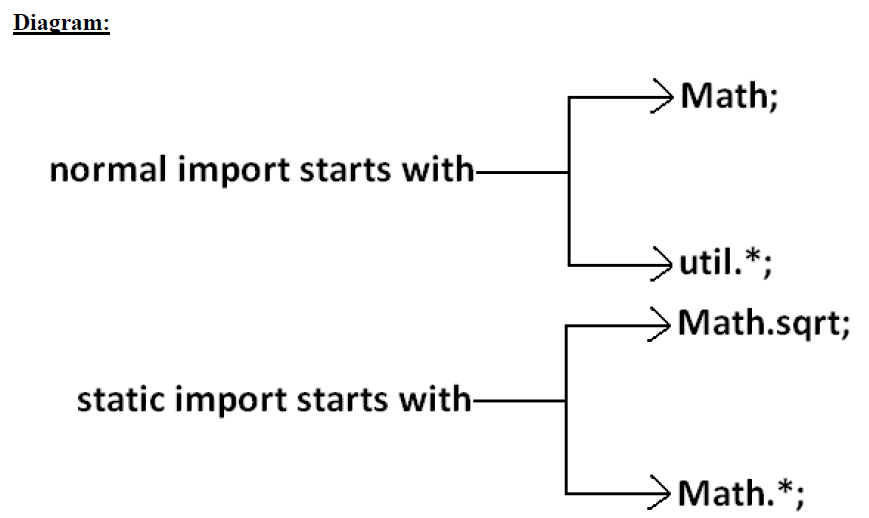
**3. implict static import.**

**Example:**

**

* If we comment line one then we will get Integer class MAX\_VALUE 2147483647.
* If we comment lines one and two then Byte class MAX\_VALUE will be considered 127.

**

**

Usage of static import reduces readability and creates confusion hence if there is no specific requirement never recommended to use static import.

***What is the difference between general import and static import ?***

* We can use normal imports to import classes and interfaces of a package.

whenever we are using normal import we can access class and interfaces directly by their short name it is not require to use fully qualified names.

* We can use static import to import static members of a particular class.

Whenever we are using static import it is not require to use class name we can access static members directly.