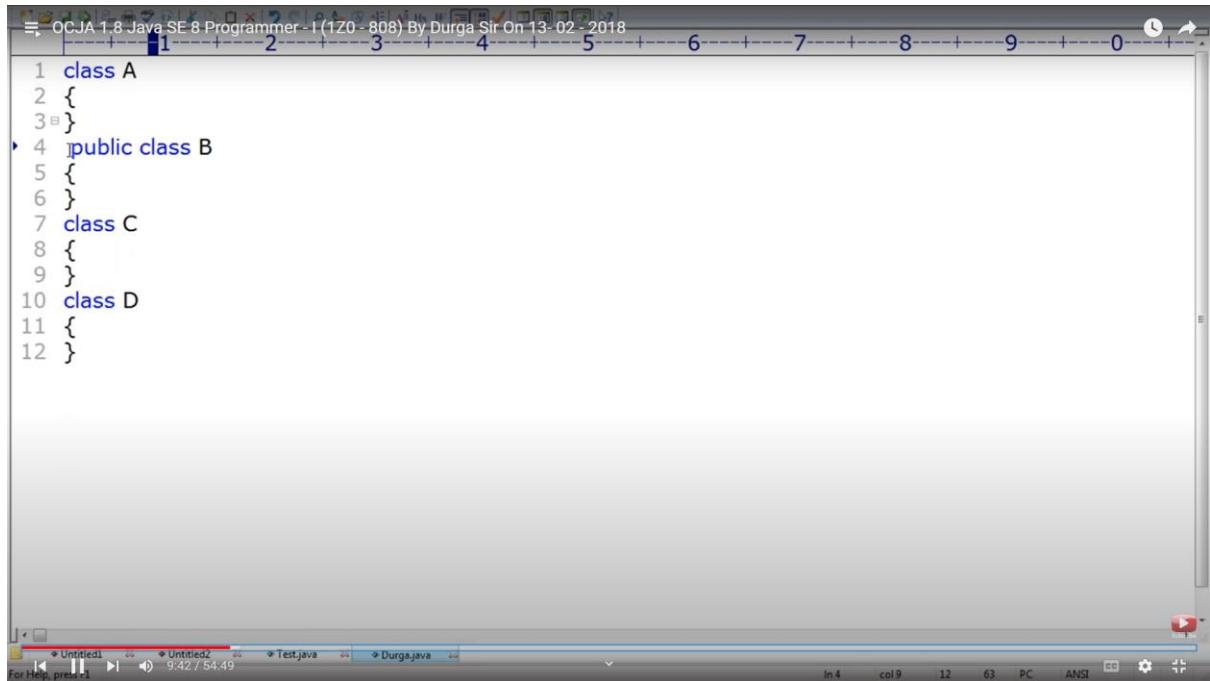
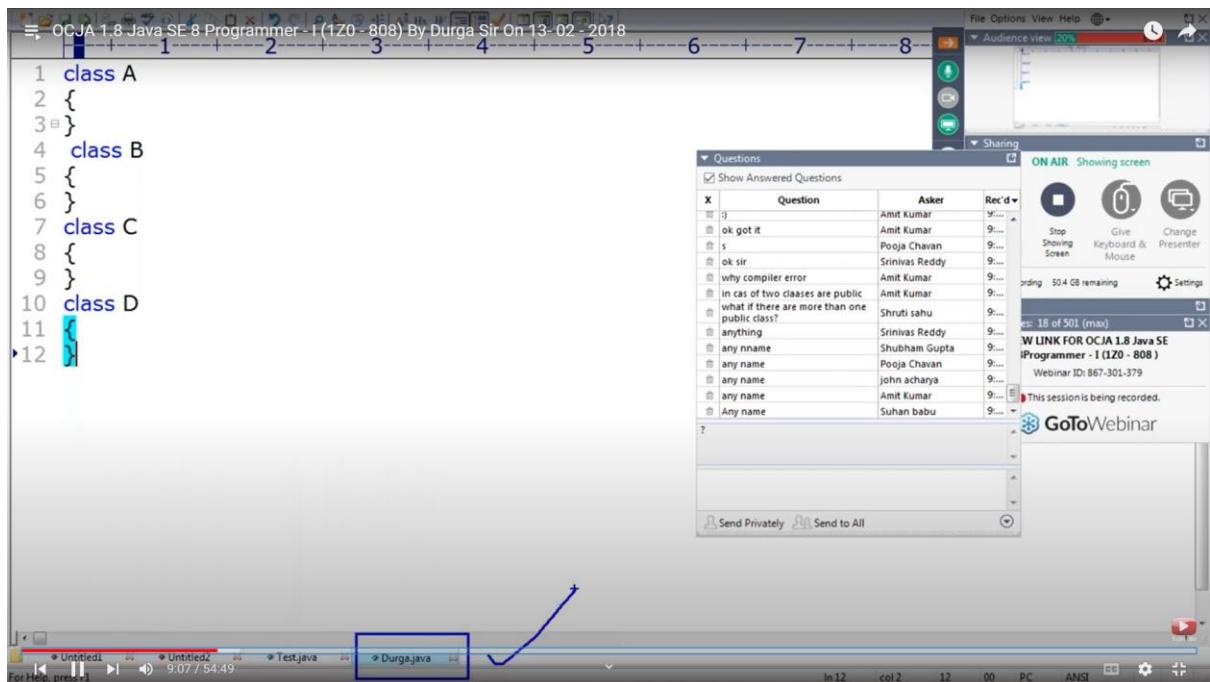


```
1 Java Source File Structure:  
2 -----  
3 import statements  
4 package statements  
5 access modifiers(public,private,protected and default)  
6 abstract modifier  
7 interface
```

A java program can contain any no of classes. If there is a public class, then the name of the prog should be public class name. Else it can be any class name.

If there are two public, then comp error. It can be 0 or 1 public class.

```
1 class A  
2 {  
3 }  
4 class B  
5 {  
6 }  
7 class C  
8 {  
9 }  
10 class D  
11 {  
12 }  
13  
14
```



A screenshot of a Java IDE interface. The code editor shows the following Java code:

```
1 class A
2 {
3 }
4 public class B
5 {
6 }
7 class C
8 {
9 }
10 class D
11 {
12 }
```

The line "public class B" is highlighted with a blue box and has a blue checkmark drawn next to it. Below the code editor, the taskbar shows several tabs: Untitled1, Untitled2, Test.java, and Durgajava. The Durgajava tab is selected. A red circle highlights the Durgajava tab. The status bar at the bottom shows "For Help, press F1".

To the right of the code editor is a "Questions" panel showing a list of user questions and answers. At the top right of the screen is a "GoToWebinar" interface with various control buttons.

A screenshot of a Java IDE interface. The code editor shows the same Java code as the previous screenshot, but the "Durgajava" tab now displays a compilation error message:

```
D:\durgaclasses>javac Durga.java
Durga.java:4: error: class B is public, should be declared in a file named B.java
      ^
1 error
```

The line "public class B" is highlighted with a red box. A large red arrow points from this error message down to the "B.java" tab in the taskbar, which is also highlighted with a red box. The status bar at the bottom shows "For Help, press F1".

To the right of the code editor is a "Questions" panel showing a list of user questions and answers. At the top right of the screen is a "GoToWebinar" interface with various control buttons.

A screenshot of a Java IDE interface. The code editor shows the following Java code:

```
1 class A
2 {
3 }
4 public class B
5 {
6 }
7 public class C
8 {
9 }
10 class D
11 {
12 }
```

The lines "public class B" and "public class C" are highlighted with red boxes and underlined. The line "public class C" has a large red arrow pointing from it towards the terminal window below. The terminal window shows the command "javac B.java" being run twice, resulting in an error message:

```
D:\durgaclasses>javac B.java
D:\durgaclasses>javac B.java
B.java:7: error: class C is public, should be declared in a file named C.java
  public class C
                  ^
1 error
```

Class which contains the main method should be the class name. No main method has no control on class name

The screenshot shows a YouTube video player with a terminal window open. The terminal window displays the following Java code:

```
3  public static void main(String[] args)
4  {
5      System.out.println("A class main method");
6  }
7 }
8 class B
9 {
10    public static void main(String[] args)
11    {
12        System.out.println("B class main method");
13    }
14 }
15 class C
16 {
17    public static void main(String[] args)
18    {
19        System.out.println("C class main method");
20    }
21 }
22 class D
```

The video player interface includes a search bar, a sign-in button, a progress bar at 19:43 / 54:49, and a control bar with play, pause, and volume buttons. Below the video player, there's a navigation bar with tabs for Untitled1, Untitled2, Test.java, B.java, and Untitled5.java. At the bottom, there's a status bar showing In 22, col 8, 25, 00, PC, ANSI, and a help key indicator.

We can name the class any name cos there are no public class

The screenshot shows a YouTube video player with a terminal window open. The terminal window displays the same Java code as the previous screenshot:

```
3  public static void main(String[] args)
4  {
5      System.out.println("A class main method");
6  }
7 }
8 class B
9 {
10    public static void main(String[] args)
11    {
12        System.out.println("B class main method");
13    }
14 }
15 class C
16 {
17    public static void main(String[] args)
18    {
19        System.out.println("C class main method");
20    }
21 }
22 class D
```

The video player interface is identical to the first one, including the search bar, sign-in button, progress bar, and terminal window content. The navigation bar and status bar are also present.

For every class .class file will be generated. Durga.class will not be generated. But though the class will be compiled. At run time A.java , main method in A Class will be executed. If there is no main method then error Main method is not available

```
3  public static void main(String[] args)
4  {
5      System.out.println("A class main method");
6  }
7 }
8 class B
9 {
10    public static void main(String[] args)
11    {
12        System.out.println("B class main method");
13    }
14 }
15 class C
16 {
17    public static void main(String[] args)
18    {
19        System.out.println("C class main method");
20    }
21 }
22 class D
```

The video title 'javac Durga.java' is overlaid in red text on the code editor area. The browser interface shows a sidebar titled 'Questions' with a list of user posts and their askers.

```
D:\durgaclasses>java A
A class main method
D:\durgaclasses>java B
B class main method
D:\durgaclasses>java C
C class main method
D:\durgaclasses>java D
Error: Main method not found in class D, please define the main method as:
  public static void main(String[] args)
```

```
D:\durgaclasses>java C
Error: Could not find or load main class Durga
```

Compiler does not know array list, based on our code it finds if it's a class or method

```
3 public static void main(String[] args)
4 {
5     ArrayList l = new ArrayList();
6 }
7 }
```

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 13-02-2018
or a JavaFX application class must extend javafx.application.Application

1 class Test
2 {
3     public
4     {
5         Arr
6     }
7 }

D:\durgaclasses>java Durga
Error: Could not find or load main class Durga

D:\durgaclasses>javac Test.java
Test.java:5: error: cannot find symbol
        ArrayList l = new ArrayList();
                           ^
symbol:   class ArrayList
location: class Test
Test.java:5: error: cannot find symbol
        ArrayList l = new ArrayList();
                           ^
symbol:   class ArrayList
location: class Test
2 errors
```

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         ArrayList();
6     }
7 }

I
```

```
D:\durgaclasses>javac Test.java
Test.java:5: error: cannot find symbol
        ArrayList();
               ^
symbol:   method ArrayList()
location: class Test
1 error
```

```
D:\durgaclasses>javac Test.java
Test.java:5: error: cannot find symbol
        System.out.println(ArrayList);
                           ^
symbol:   variable ArrayList
location: class Test
1 error
```

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         ArrayList l = new ArrayList();
6     }
7 }
8
9 world.asia.india.ts.hyd.ameerpet.maitrivanam.Durgasoft
```

Fully Qualified Name

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         java.util.ArrayList l = new java.util.ArrayList();
6     }
7 }
8
```

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         java.util.ArrayList l = new java.util.ArrayList();
6         java.util.ArrayList l = new java.util.ArrayList();
7         java.util.ArrayList l = new java.util.ArrayList();
8         java.util.ArrayList l = new java.util.ArrayList();
9         java.util.ArrayList l = new java.util.ArrayList();
10        java.util.ArrayList l = new java.util.ArrayList();
11    }
12 }
```

```
1 import java.util.ArrayList;
2 class Test
3 {
4     public static void main(String[] args)
5     {
6         ArrayList l = new ArrayList();
7     }
8 }
```

After package semi colon is invalid, After class * is invalid no more classes after ArrayList

```
222  
223  
224 import java.util.ArrayList;  
225 import java.util.*;  
226 import java.util.ArrayList.*;  
227 import java.util.*;  
228  
229  
230  
231 1. Explicit class import  
232     import java.util.ArrayList;  
233  
234 2. Implicit class import  
235     import java.util.*;
```

Memory and performance both are the same. At compile time no class will be loaded. Readability wise Explicit is recommended

The screenshot shows a Java IDE interface with a code editor and a terminal window.

Code Editor:

```
1 import java.sql.*;
2 import java.util.*;
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         Date d = new Date();    I
8     }
9 }
10
```

Terminal Window:

```
1 import D:\durgaclasses>javac Test.java
2 import Test.java:7: error: reference to Date is ambiguous
3 class Date d = new Date();
4 ^
5 both class java.util.Date in java.util and class java.sql.Date in java.sql match
6 Test.java:7: error: reference to Date is ambiguous
7         Date d = new Date();
8 ^
9 } both class java.util.Date in java.util and class java.sql.Date in java.sql match
10 2 errors
```

Explicit date specified so no error

The screenshot shows a Java IDE interface with a code editor and a terminal window.

Code Editor:

```
1 import java.sql.*;
2 import java.util.Date;
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         Date d = new Date();
8     }
9 }
```

Terminal Window:

```
10 D:\durgaclasses>javac Test.java
11 D:\durgaclasses>
```

Code Editor (Continued):

```
1 import java.sql.*;
2 import java.util.Date;
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         Date d = new Date();
8         System.out.println(d.getClass().getName());
9     }
10 }
```

The screenshot shows a Java IDE interface with a code editor and a terminal window.

Code Editor:

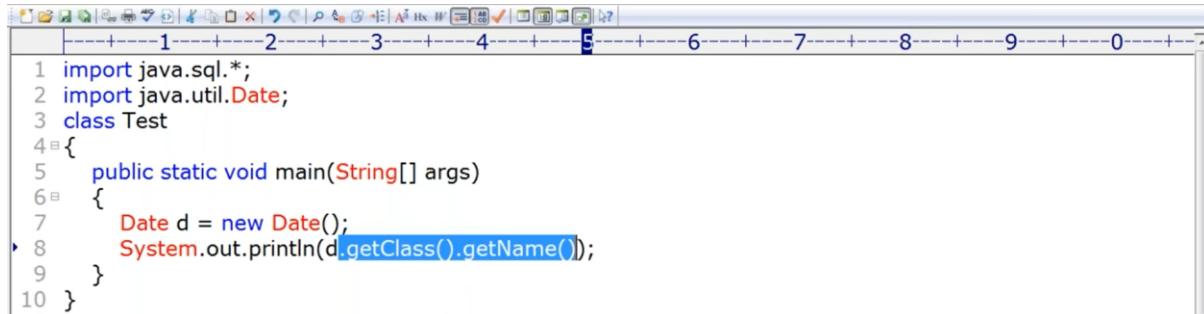
```
7
8
9
10 }
```

Terminal Window:

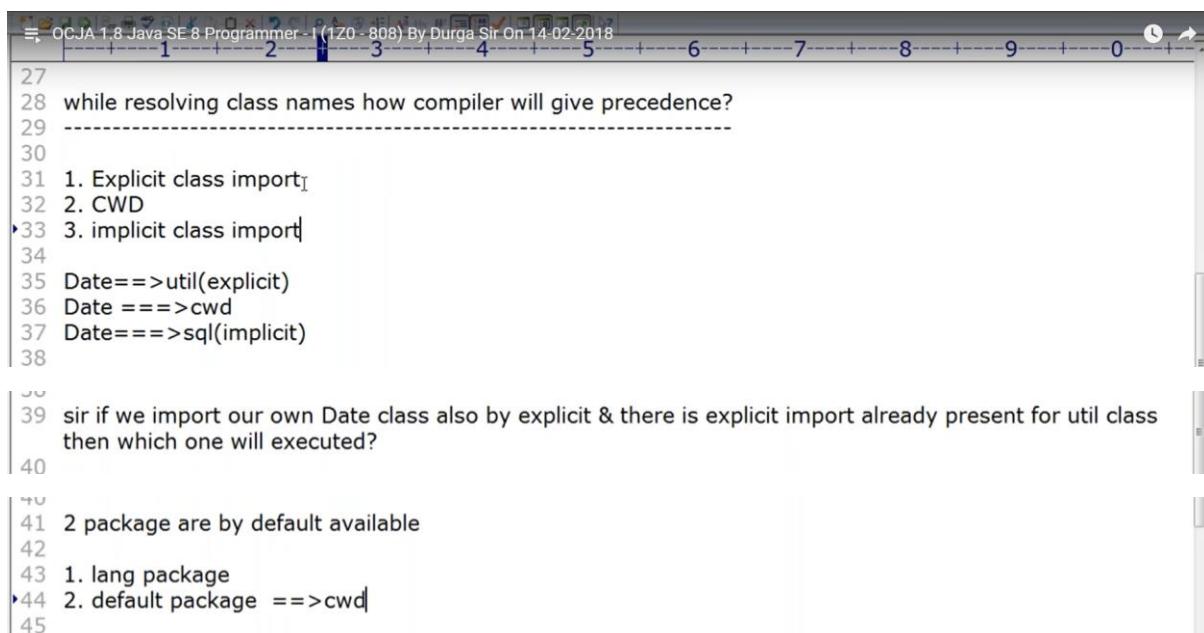
```
7 D:\durgaclasses>javac Test.java
8
9 D:\durgaclasses>java Test
10 java.util.Date
```

Three dates class are available. Highest priority?

If you have a date class in current working Directory.



```
1 import java.sql.*;
2 import java.util.Date;
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         Date d = new Date();
8         System.out.println(d.getClass().getName());
9     }
10 }
```



27
28 while resolving class names how compiler will give precedence?
29 -----
30
31 1. Explicit class import
32 2. CWD
33 3. implicit class import
34
35 Date==>util(explicit)
36 Date ==>cwd
37 Date==>sql(implicit)
38
39 sir if we import our own Date class also by explicit & there is explicit import already present for util class then which one will executed?
40
41 2 package are by default available
42
43 1. lang package
44 2. default package ==>cwd
45

You don't need to import the classes in CWD by default its imported. Even then the Explicit will take high priority.

then which one will executed?

```

40
41 2 package are by default available
42
43 1. lang package
44 2. default package ==> cwd
45
46
47 java
48   |-util
49     |-regex
50       |-Pattern
51
52
53
54
55
56
57
58
59
60
61

```

For Help, press F1 In 52 col1 68 00 PC ANSI

```

45
46
47 java
48   |-util
49     |-regex
50       |-Pattern
51
52
53 To use Pattern class in our program
54
55 1. import java.*;
56 2. import java.util.*;
57 3. import java.util.regex.*;
58 4. No import is required b'z Pattern is available for all programs
59
60
61

```

For Help, press F1 In 57 col 2 73 2E PC ANSI

1. If you use `java.*` - all packages in java package will be imported but not the subpackage so wrong

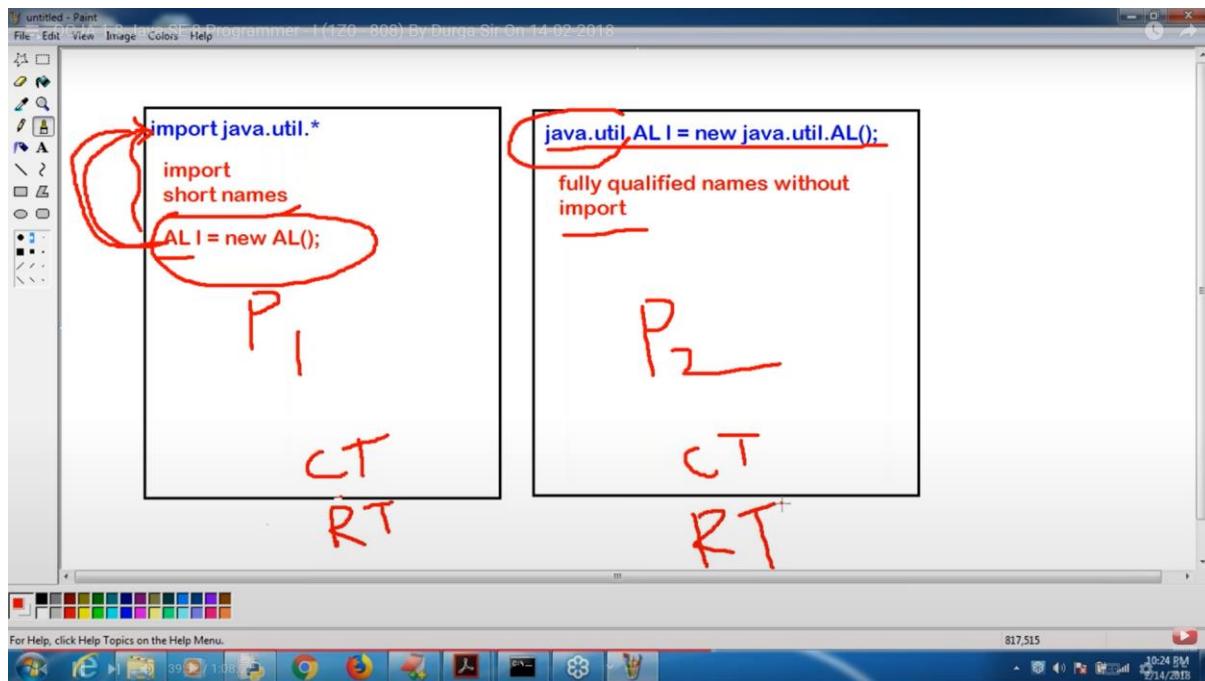
```

55
56
57
58
59
60
61
62 import java.lang.*;
63 import java.lang.reflect.Method
64
65

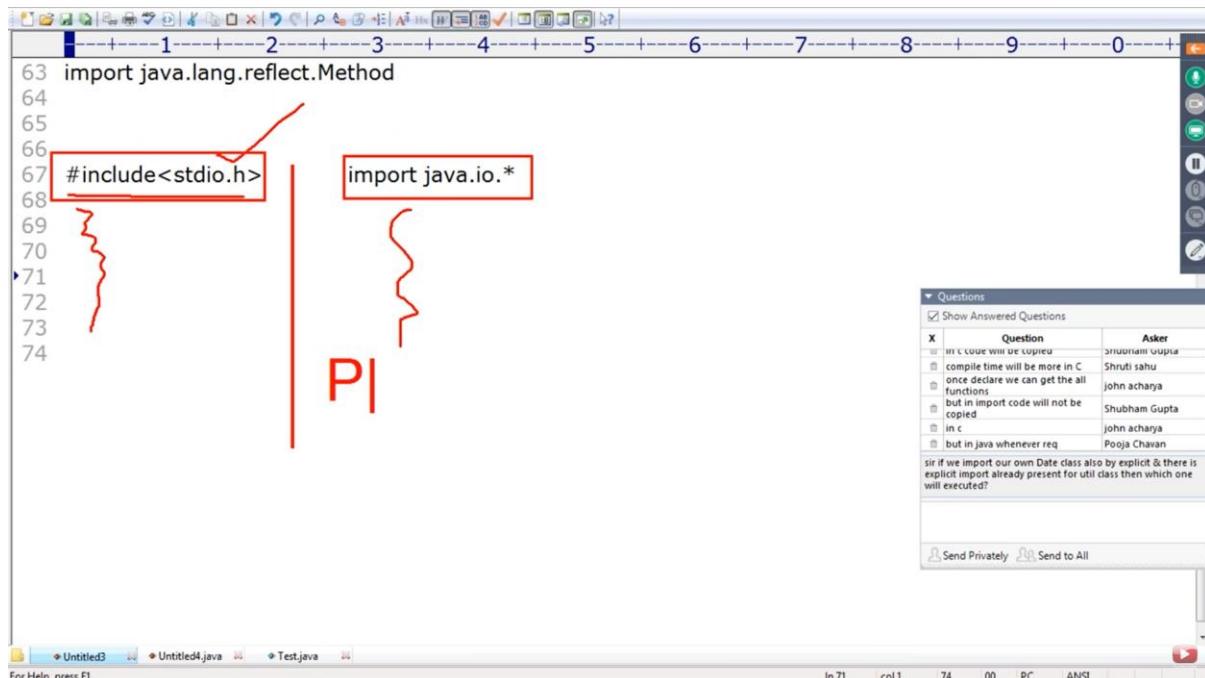
```

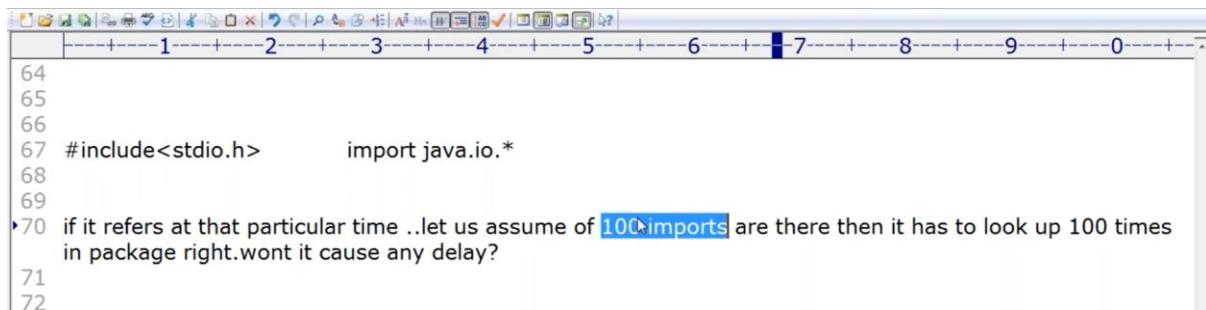
If I want to use Method we need to either us `reflect.*` or `reflect.Method`

1 st one take more compile time. Run time takes equal time



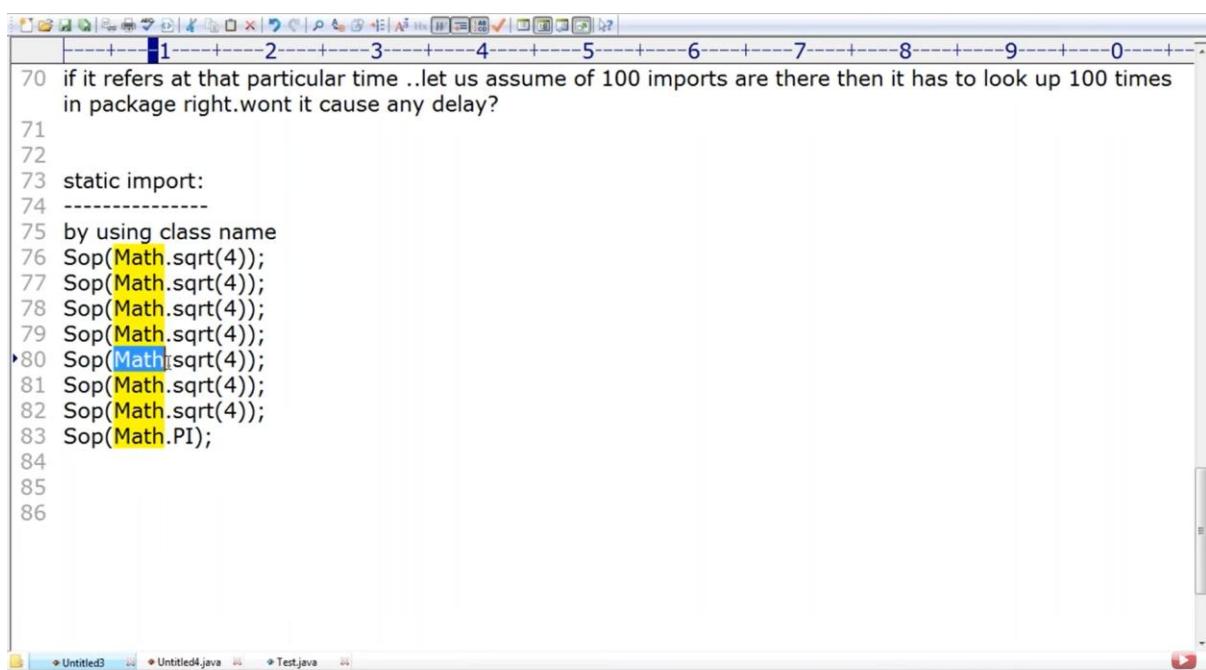
In C all classes will be loaded first before it moves to the next line. In Java nothing will be loaded only when the classes are used. Load on claim



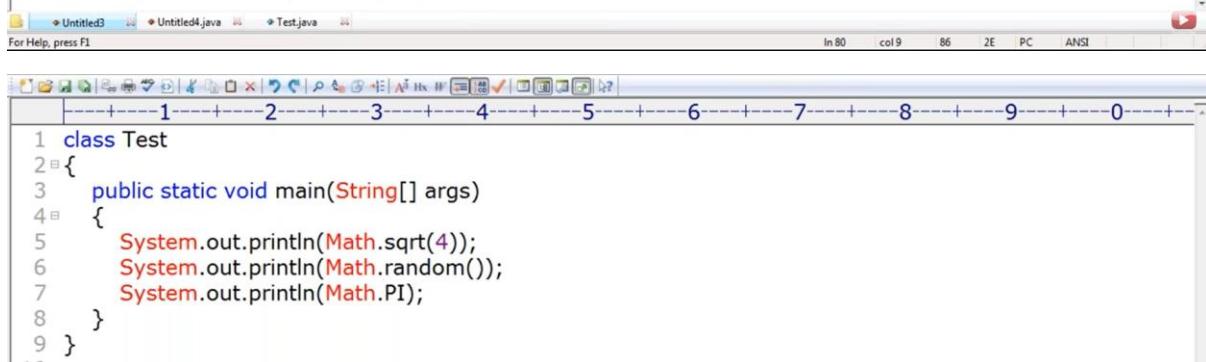


```
64
65
66
67 #include<stdio.h>      import java.io.*
68
69
70 if it refers at that particular time ..let us assume of 100 imports are there then it has to look up 100 times
in package right.wont it cause any delay?
71
72
```

When a classes are loaded , JVM will not load it again. If its diff classes then they will load everything for each classes/



```
70 if it refers at that particular time ..let us assume of 100 imports are there then it has to look up 100 times
in package right.wont it cause any delay?
71
72
73 static import:
74 -----
75 by using class name
76 Sop(Math.sqrt(4));
77 Sop(Math.sqrt(4));
78 Sop(Math.sqrt(4));
79 Sop(Math.sqrt(4));
80 Sop(Math.sqrt(4));
81 Sop(Math.sqrt(4));
82 Sop(Math.sqrt(4));
83 Sop(Math.PI);
84
85
86
```



```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         System.out.println(Math.sqrt(4));
6         System.out.println(Math.random());
7         System.out.println(Math.PI);
8     }
9 }
```

```
1 import static java.lang.Math.sqrt;
2 class Test
3 {
4     public static void main(String[] args)
5     {
6         System.out.println(sqrt(4));
7         System.out.println(Math.random());
8         System.out.println(Math.PI);
9     }
10}
```

```
1 //import static java.lang.Math.sqrt;
2 import static java.lang.Math.*;
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         System.out.println(sqrt(9));
8         System.out.println(random());
9         System.out.println(PI);
10    }
11}
```

Out is a static variable inside System class. And println is a method of out

```
1 OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 14-02-2018
2 class Test
3 {
4     public static void main(String[] args)
5     {
6         System.out.println("Hello Dont get shock");
7     }
8 }
```

```
1 OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 14-02-2018
2 import static java.lang.System.out;
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         out.println("Hello Dont get shock");
8         out.println("It is valid");
9     }
10}
```



Package:

```
5 -----
4 package statement
5 -----
6 It is an encapsulation mechanism to group related things into a single component.
7
8 classes and interfaces ==>java.sql package
9 file i/o operations==>java.io package
10
11 1. Modularity of application
12 2. Naming conflicts
13
14 Date
15 Date
16 java.sql.Date
```

```
13 3. We can achieve Security for our components
14
```

The screenshot shows a Java code editor with the following code:

```
11 1. Modularity of application
12 2. Naming conflicts , unique identification
13 3. We can achieve Security for our components
14
15
16 package pack1;
17 class A
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

To the right of the code, there is a hand-drawn diagram. It features a blue square labeled 'A' inside a red circle, which is itself inside a larger blue circle. Below the blue circle, the text 'pack1' is written in blue. A small red stick figure stands to the left of the circles, pointing towards them.

On the far right of the interface, there is a 'Questions' panel showing a list of questions and their askers:

X	Question	Asker
2		Srinivas Reddy
2		Shubham Gupta
two		Deepankaj Yadav
2		Sai chand Kilaru
SQL_util		Srinivas Reddy
fully qualified name		Shubham Gupta
ambiguity		Shubham Gupta
s		Shubham Gupta
ys sir		Deepankaj Yadav
i		Amit Kumar
more developer can work		Deepankaj Yadav
simultaneously on functionality		

```
22 package pack1;
23
24
25 unique identification : internet domain
26
27
28 com.icicibank.loan.housingloan.Account
29
```

```

1 package com.durgasoft.ocja;
2 public class Test
3 {
4     public static void main(String[] args)
5     {
6         System.out.println("package demo");
7     }
8 }
9

```

I am expecting the .class files to be generated in the package but its getting created in the current working directory.

```

D:\durgaclasses>javac Test.java
D:\durgaclasses>dir Test.class
Volume in drive D is New Volume
Volume Serial Number is 9430-0171

Directory of D:\durgaclasses

02/15/2018 10:01 PM      436 Test.class
   1 File(s)        436 bytes
   0 Dir(s) 278,959,550,464 bytes free

```

```

32
33 javac -d . Test.java
34

```

-d → destination

```

26
27
28 com.icicibank.loan.housingloan.Account
29
30
31 javac Test.java==>cwd
32
33 javac -d . Test.java==>
34
35
36

```

com
| -durgasoft
| -ocja
| -Test.class

X	Question	Asker	R...
1	It's outside directory now	Srinivas Reddy	
2	class not found	john acharya	
3	no package generated	Shubham Gupta	
4	class not found	Sai chand Kilar	
5	cant acces class	Deepankaj Yadav	
6	no such class	Srinivas Reddy	
7	error	Amit Kumar	
8	package problem	Surendra Babu Divi	
9	fully qualified name	SANDEEP REDDY	
10	no package generated in thi	Shubham Gupta	
11	directory	Amit Kumar	
12	cwd	Shubham Gupta	

```

D:\durgaclasses>javac -d . Test.java

```

```
0 Select C:\Windows\system32\cmd.exe
D:\durgaclasses>cd com
D:\durgaclasses\com>cd durgasoft
D:\durgaclasses\com\durgasoft>cd ocja
D:\durgaclasses\com\durgasoft\ocja>dir
 Volume in drive D is New Volume
 Volume Serial Number is 9430-0171

 Directory of D:\durgaclasses\com\durgasoft\ocja

02/15/2018  10:06 PM    <DIR>        .
02/15/2018  10:06 PM    <DIR>        ..
02/15/2018  10:06 PM       436 Test.class
               1 File(s)      436 bytes
               2 Dir(s)  278,959,550,464 bytes free
```

If the package structure is already there it will overwrite the class files else it will create a new folder structure

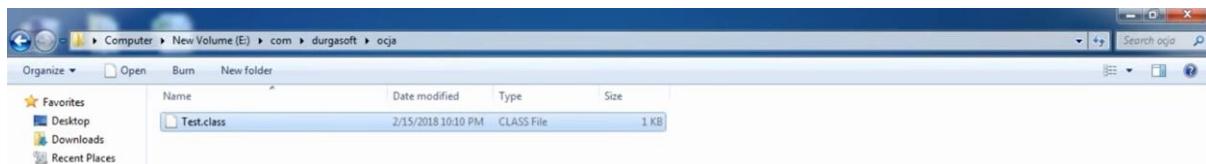
```
Directory of D:\durgaclasses\com\durgasoft\ocja

02/15/2018  10:06 PM    <DIR>        .
02/15/2018  10:06 PM    <DIR>        ..
02/15/2018  10:08 PM       436 Test.class
               1 File(s)      436 bytes
               2 Dir(s)  278,959,550,464 bytes free

D:\durgaclasses\com\durgasoft\ocja>
```

```
32
33 javac -d E: Test.java==>
~^

D:\durgaclasses>javac -d E: Test.java
```



```
41
42 javac Test.java==>CWD
43 javac -d E: Test.java==>
```

Whenever you are using package structure at run time u need to specify fully qualified package name

```
D:\durgaclasses>java Test
Error: Could not find or load main class Test
D:\durgaclasses>java com.durgasoft.ocja.Test
package demo!!!
D:\durgaclasses>
```

Compile time error. Atmost 1 package for any java program

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 15-02-2018
1 package pack1;
2 package pack2;
3 public class Test
4 {
5     public static void main(String[] args)
6     {
7         System.out.println("package demo!!!");
8     }
9 } I
10
11 javac -d . Test.java
12
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 15-02-2018
Error: Could not find or load main class Test

D:\durgaclasses>java com.durgasoft.ocja.Test
package demo!!!

D:\durgaclasses>javac -d E: Test.java
Test.java:2: error: class, interface, or enum expected
package pack2;
^
1 error
```

In any program, first non comment line should be Package Statement

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 15-02-2018
1 import java.util.*;
2 package pack1;
3 public class Test
4 {
5     public static void main(String[] args)
6     {
7         System.out.println("package demo!!!");
8     }
9 }

D:\durgaclasses>javac -d E: Test.java
Test.java:2: error: class, interface, or enum expected
package pack1; ^ Mark
1 error
```

- 51 1. In any java program atmost one pkg statement is allowed
52 2. the pkg statement should be first non-comment statement

```
55  
56 package statement  
57 import statements  
58 class or interface or enum declarations  
59  
60  
61  
62  
63
```

Questions

Java source file structure. A java program can contain 0 or any no of class , interface or enum

```
55  
56  
57 package statement// atmost one  
58 import statements// any number  
59 class or interface or enum declarations// any number  
60
```

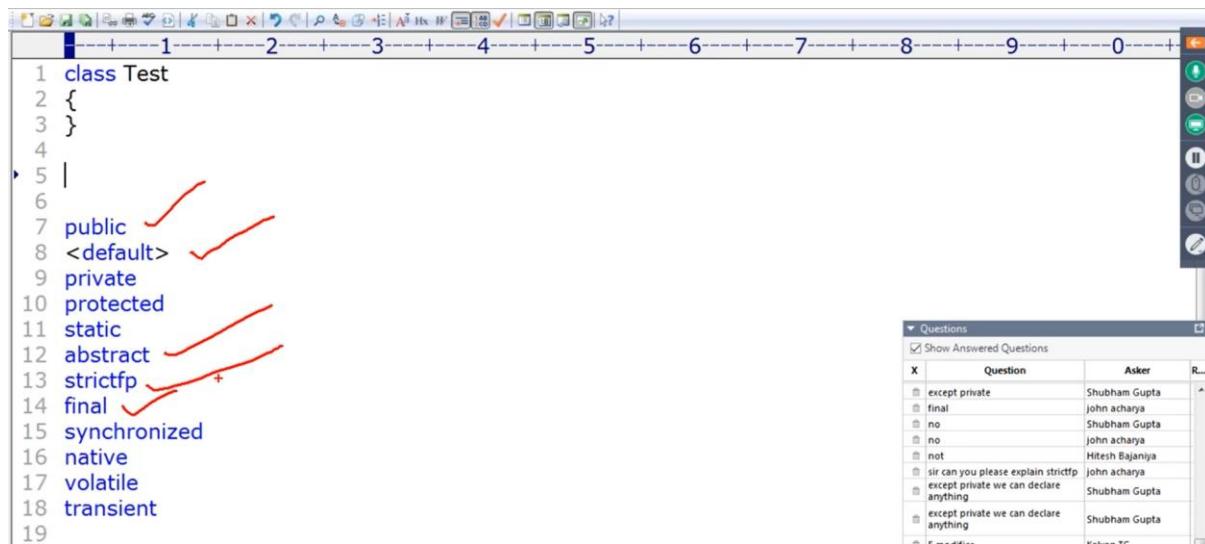
Modifiers

```
1 class Test  
2 {  
3 }  
4  
5 public  
6 <default>  
7 private  
8 protected  
9 static  
10 abstract  
11 strictfp  
12 final  
13 synchronized  
14 native  
15 volatile  
16 transient  
17  
18  
19
```

Untitled1 Test.java

In 5 col1 19 70 PC ANSI

For a top level class only the below 5 modifiers can be used. Top level or outer class

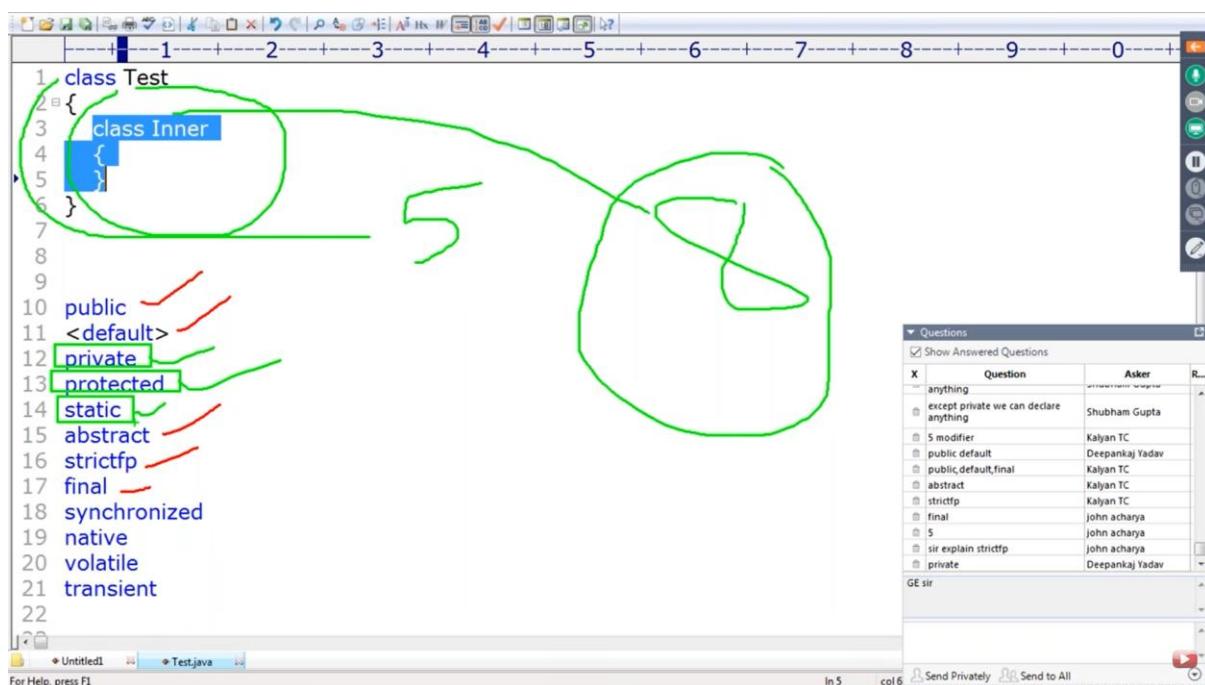


```
1 class Test
2 {
3 }
4
5
6
7 public
8 <default>
9 private
10 protected
11 static
12 abstract
13 strictfp
14 final
15 synchronized
16 native
17 volatile
18 transient
19
```

Questions pane:

X	Question	Asker	R...
1	except private	Shubham Gupta	
2	final	john acharya	
3	no	Shubham Gupta	
4	not	john acharya	
5	sir can you please explain strictfp	Hitesh Bajanija	
6	except private we can declare anything	john acharya	
7	except private we can declare anything	Shubham Gupta	
8	anything	Shubham Gupta	

For inner class 8 modifiers can be used

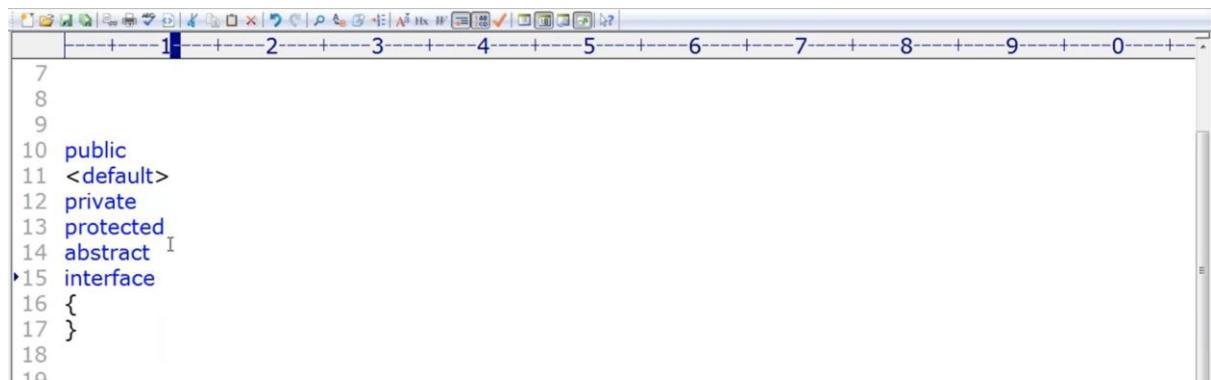


```
1 class Test
2 {
3     class Inner
4     {
5     }
6 }
7
8
9
10 public
11 <default>
12 private
13 protected
14 static
15 abstract
16 strictfp
17 final
18 synchronized
19 native
20 volatile
21 transient
22
```

Questions pane:

X	Question	Asker	R...
1	anything		
2	except private we can declare anything	Shubham Gupta	
3	5 modifier	Kalyan TC	
4	public default	Deepankaj Yadav	
5	public,default,final	Kalyan TC	
6	abstract	Kalyan TC	
7	strictfp	Kalyan TC	
8	final	john acharya	
9	5	john acharya	
10	sir explain strictfp	john acharya	
11	private	Deepankaj Yadav	

Exam -



A screenshot of a Java code editor window. The title bar includes standard icons for file operations like Open, Save, and Print. The main area shows the following Java code:

```
7  
8  
9  
10 public  
11 <default>  
12 private  
13 protected  
14 abstract I  
15 interface  
16 {  
17 }  
18
```

The code is numbered from 1 to 18. Line 15 starts with a right-angle bracket 'I' indicating an inheritance relationship. The code is currently incomplete, ending with a closing brace '}' on line 17.

Access Modifier

```
1 package pack1;
2 public class A
3 {
4     public void m1()
5     {
6         System.out.println("A class method");
7     }
8 }
9
```

When u import a class from another package it should be Public Then only you can use.

```
1 package pack2;
2 import pack1.A;
3 public class Test
4 {
5     public static void main(String[] args)
6     {
7         A a = new A();
8         a.m1();
9     }
10 }
```

```
14
15
16 1. public :
17 -----
18 2. default:
19 -----
20
21 default access ==>package level access
22 |
```

```
D:\durgaclasses>javac -d . A.java
D:\durgaclasses>javac -d . Test.java
Test.java:2: error: A is not public in pack1; cannot be accessed from outside package
import pack1.A;
               ^
Test.java:7: error: cannot find symbol
      A a = new A();
             ^
symbol:   class A
location: class Test
Test.java:7: error: cannot find symbol
      A a = new A();
             ^
symbol:   class A
location: class Test
3 errors
```

```
1 class P
2 {
3     public void property()
4     {
5         System.out.println("Power+ca$h+Land+Gold");
6     }
7     public final void marry()
8     {
9         System.out.println("Appalamma");
10    }
11 }
12 class C extends P
13 {
14     public void marry()
15     {
16         System.out.println("3Sha|4Me|9Tara");
17     }
18 }
```

```
1 final class P
2 {
3
4 }
5 class C extends P
6 {
7
8 }
```

```
symbol:   class A
location: class Test
3 errors

D:\durgaclasses>javac P.java
P.java:14: error: marry() in C cannot override marry() in P
      public void marry()
                  ^
      overridden method is final
1 error

D:\durgaclasses>javac P.java

D:\durgaclasses>javac P.java
P.java:5: error: cannot inherit from final P
class C extends P
      ^
1 error

D:\durgaclasses>
```

```

19 -----
20 3. final:
21 -----
22 applicable for classes,methods and variables
23
24 final methods are cannot be overridden in child class
25

```

```

26 final class
27
28 3. abstract:
29 -----
30 methods and classes but not for variables
31 -----
32

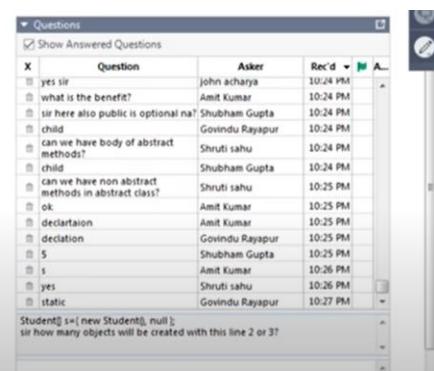
```

Final synchro static all works around implementation.. wheras abstract does not

```

19
20
21
22 abstract method:
23
24
25
26
27

```



```

29 abstract class:
30 -----
31 partially implemented class I
32
33
34 It is not possible to create object for abstract class..

```

Cant create an object

```

1 abstract class Test
2 {
3     public static void main(String[] args)
4     {
5         Test t = new Test();
6     }
7 }
8

```

```

D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
D:\durgaclasses>javac Test.java
Test.java:5: error: Test is abstract; cannot be instantiated
        Test t = new Test();
                           ^
1 error

```

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 20-02-2018  
Press Esc to exit full screen  
1 class Test  
2 {  
3     public void m1(){}
4     public void m2(){}
5     public void m3(){}
6     public void m4(){}
7     public void m1000(){}
8 }
9  
I
```

YouTube IN

```
Search  
9 abstract:  
10 -----  
11 methods and classes but not for variables  
12  
13 abstract method:  
14 -----  
15 public abstract void m1();  
16 public abstract void m1(){}  
17  
18  
19  
20  
21
```

Adapter Class

```
1 m5();
2 ....
3 m1000();
4 }
5 abstract class AdapterX implements X
6 {
7     m1(){}
8     m2(){}
9     m3(){}
10    m4(){}
11    m1000(){}
12 }
13 class Test extends AdapterX
14 {
15     m3(){}
16     proper implementation
17 }
18 }
19 class Demo implements X
20 {
21     I
22     m4()
23 }
```

The screenshot shows a Java code editor window with the file 'Test.java' open. The code defines an abstract class 'AdapterX' that implements an interface 'X'. It contains five methods: m5(), m1000(), m1(), m2(), and m3(). A concrete class 'Test' extends 'AdapterX' and overrides the m3() method with a comment 'proper implementation'. Another class 'Demo' implements the 'X' interface and provides a method m4(). The code editor has a toolbar at the top, a status bar at the bottom, and tabs for 'Untitled1' and 'Test.java'.

Possible

```
30
31 what happen if we create instance of another class in abstract class?
32
33
34
35
36
37
```

```
1 abstract class Test
2 {
3     public static void main(String[] args)
4     {
5         String s = new String("durga"); I
6         System.out.println(s.length());
7     }
8 }
```

The screenshot shows a Java code editor window with the file 'Test.java' open. The code defines an abstract class 'Test' with a main() method. Inside the main() method, there is a line of code 'String s = new String("durga");' followed by a cursor 'I'. The code editor has a toolbar at the top, a status bar at the bottom, and tabs for 'Untitled1' and 'Test.java'.

A abstract method cannot be final. A class that contains abstact method cannot be final. Or an abstract class cannot be final

```
34+
35 abstract vs final:
36 -----
37 1. final method  abstract method
38 2. final class   abstract class
39
```

Abstract cannot hv body. CE

```
1 class P
2 {
3     public abstract void m1(); I
4 }
5
```

```
D:\durgaclasses>javac P.java
P.java:1: error: P is not abstract and does not override abstract method m1() in P
class P
^
P.java:3: error: abstract methods cannot have a body
    public abstract void m1(){}
               ^
```

Only abstract method can have only dec. Missing method body or declare abstact

```
1 class P
2 {
3     public void m1(); I
4 }
```

Abstract class only can have absrtact method

```
1 class P
2 {
3     public abstract void m1(); I
4 }
5
```

All methods in the abstact class should be implemented. If any abstact methods not implemented.
CE

```
1 abstract class P
2 {
3     public abstract void m1();
4     public abstract void m2();
5 }
6 class C extends P
7 {
8     public void m1(){}
9 }
10
```

Valid.

C is also defined as abstract so if does not hv to implement all methods, SubC is a class it has to implement m2

```
1 abstract class P
2 {
3     public abstract void m1();
4     public abstract void m2();
5 }
6 abstract class C extends P
7 {
8     public void m1(){}
9 }
10 class SubC extends C
11 {
12     public void m2(){}
13 }
14
```

3 is invalid and 4 th one is valid

```
39
40 3. final class can contain abstract methods
41 4. abstract class can contain final methods
42
```

Only an abstract class can contain abstract method

```
1 final class Test
2 {
3     abstract void m1();
4 }
```

```
D:\durgaclasses>javac P.java
D:\durgaclasses>javac P.java
P.java:10: error: SubC is not abstract and does not override abstract method m2() in P
class SubC extends C
^
1 error
```

Abstract class can contain final method. This is not a abstact method . And cannot be overriden

```
1 abstract class Test
2 {
3     final void m1(){}
4 }
```

StrictFloatingPoint

All floating point in the class should follow IEEE standard. FP implementation differs from platform to platform If u want the same results across platform use StrictFP. Can be applied at class level (both inner and outer class).