

OOPS

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) By Durga Sir On 28-02-2018  
Press Esc to exit full screen 6 7 8 9 0  
1 OOPs:  
2 -----  
3 data hiding  
4 abstraction  
5 encapsulation  
6 inheritance  
7 has-a relation  
8 composition  
9 aggregation  
10 polymorphism  
11 overloading  
12 overriding  
13 method hiding  
14 coupling  
15 cohesion  
16 ...
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) By Durga Sir On 28-02-2018  
23  
24 Data Hiding:  
25 -----  
26 outside person should not get our data directly.  
27 Our internal data should not go out directly I  
28  
29 private modifier  
--  
1 public class Account  
2 {  
3     private double balance;  
4  
5     getBalance()  
6     {  
7         ...checkings...  
8     }  
9 }
```

Process of binding data and corresponding methods into single unit is called encapsulation

Variables are states (values it takes) methods are behaviours

```
52 Data  
53  
54 Behaviour  
55  
56  
57 Student  
58 name  
59 rollno  
60 marks  
61 read()  
62 write()  
63 sleep()  
64 eat()
```

Diagram illustrating encapsulation:

- Variables (name, rollno, marks) are circled in red and connected by arrows to the word "State".
- Methods (read(), write(), sleep(), eat()) are circled in red and connected by arrows to the word "Methods".

Questions table:

X	Question	Asker	Rec'd
1	enclosing data members and variables inside a class	Shruti Sahu	10:13 PM
2	s	Pooja Chavan	10:13 PM
3	s	Amit Kumar	10:13 PM

```

6/
68 Data Hiding and Abstraction...Encapsulation I
69
70 Encapsulation=Data Hiding+Abstraction
71

```

Data hiding is here. Where is abstraction. Here the calculations parts , all complex calculations are hidden from the user and we expose only the methods which are needed , this is called abstraction

```

1 public class Account
2 {
3     private double balance;
4
5     getBalance()
6     {
7         ...checkings...
8         return balance
9     }
10    setBalance(balance)
11    {
12        ...checkings...
13        update balance
14    }
15 }

```

About 1,52,00,000 results (0.37 seconds)

[In English](https://www.w3schools.com/java/java_abstract.asp)

Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either abstract classes or interfaces (which you will learn more about in the next chapter).

W3Schools https://www.w3schools.com/java/java_abstract.asp

Java Abstraction - W3Schools

People also ask

What is data abstraction?

Data abstraction is the reduction of a particular body of data to a simplified representation of the whole. Abstraction, in general, is the process of removing characteristics from something to reduce it to a set of essential elements.

tctarget.com <https://www.techtarget.com/whatis/definition/data-abstraction>

What is data abstraction in OOP? - TechTarget

Search for: What is data abstraction?

What is data abstraction with example?

For example, when using a cell phone, you can figure out how to answer incoming calls and respond to text messages. Thanks to data abstraction, you can't tell how the phone itself transmits signals. The purpose of data abstraction is to expose only the essential elements of a device. 19-Aug-2021

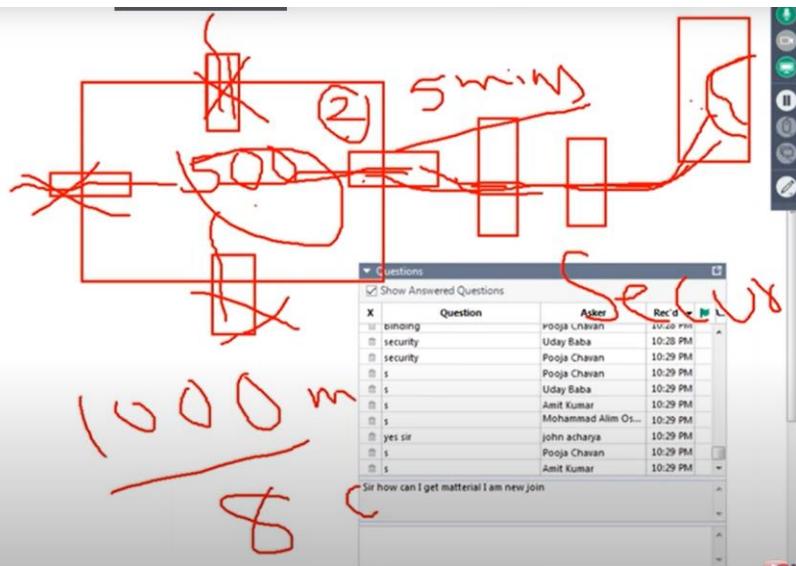
Indeed.com <https://www.indeed.com/career-development/data-abstraction>

6 Examples of Data Abstraction (With Definition and Benefits) |

Security – should compromise performance

Limit access. Getter setter authorization authentication etc. slows execution

```
1 public class Account
2 {
3     private double balance;
4
5     getBalance()
6     {
7         ...checkings...
8         return balance
9     }
10    setBalance(balance)
11    {
12        ...checkings...
13        update balance
14    }
15 }
16
17 Security
18 }
```



```
29 Data Hiding
30 Abstraction
31 Encapsulation
```

```
34 IS-A Relation:
35 -----
36 1. Inheritance
37 2. extends
38 3. Code Reusability
39
```

```
1 class P
2 {
3     //10 methods
4
5     class C extends P
6     {
7         //5 methods
8     }
9
10 }
```

C c = new C();
c.parent class and child class methods

Reusability

```

1 class P
2 {
3     public void m1()
4     {
5         System.out.println("Parent Method");
6     }
7 }
8 class C extends P
9 {
10    public void m2()
11    {
12        System.out.println("Child Method");
13    }
14 }
15 class Test
16 {
17    public static void main(String[] args)
18    {
19        C c= new C();
20        c.m1();
21        c.m2();
22    }
23 }
24
25

```

For Help, press F1 In 21 col 16 25 00 PC ANSI

```

6 Inheritance(IS-A Relationship)
7
8
9 for code reusability and to extends existing functionality with some extra functionality
10
11 1. What ever methods present in parent class are by default available to the child class.Hence on the child reference we can call the methods present in both parent class and child class
12

23     P p = new P();
24     p.m1(); I
25     //p.m2();
26
27 D:\durgaclasses>javac Test.java
28 Test.java:25: error: cannot find symbol
29             p.m2();
30                 ^
31               symbol:   method m2()
32               location: variable p of type P
33 1 error
34

```

The below is right parent ref child object. But why



```

22
23     P p = new P();
24     p.m1();
25     p.m2();*/
26
27     P p = new C(); I
28     p.m1();
29     p.m2();

17 compiler and jvm==>
18 compiler will always check syntax
19 jvm will always runs the code I
20
21
22 compiler will always consider reference type but not runtime object
23 but jvm will always consider runtime object but not reference type

```

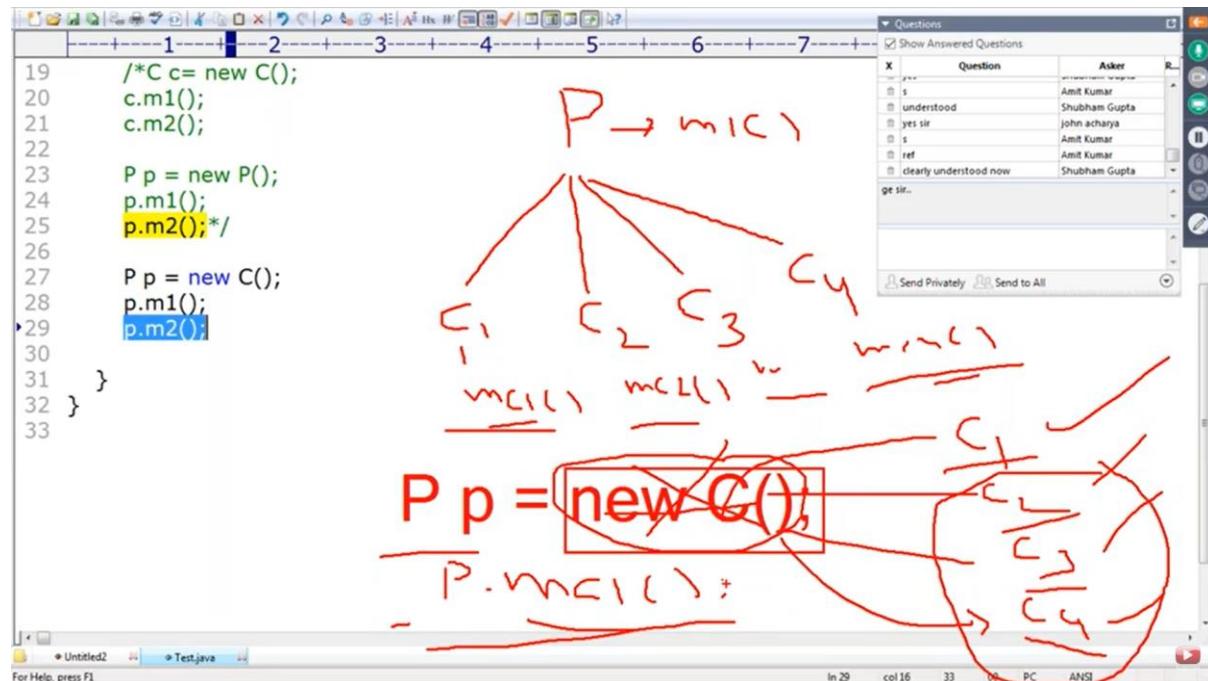
Compiler never checks below

```
26  
27     P p = new C();  
28     p.m1();  
29     p.m2();  
30
```



Here compiler will not accept p.m2 because

P can have any no of child class and they would have their own imp of m2(). Compiler does not know what object p holds only at run time JVM knows and hence compiler does not accepts this .



Here `o.length()` throws error , because o is of object type. Parent reference can use any parent specific methods. Parent methods are available for any child class. But child methods are not available for any Parent reference. Hence u cant call `o.length()` which is specific to child class String whereas `o.hashCode()` specific to parent can be used

```
28  
29 Object o= m1();  
30 o.length();  
31 o.hashCode();  
32  
33  
34  
35 m1()  
36 {  
37 return String|Customer|Student|Account  
38 }
```

Parent ref can be used to hold child objects whereas child ref cannot parent objects. Below is wrong

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

```
30
31     C c = new P(); // Error: Cannot assign child type to parent reference
32
33 {
34     public static void main(String[] args)
35     {
36         /*C c = new C();
37         c.m1();
38         c.m2();*/
39
40         P p = new P(); // Error: Cannot assign child type to parent reference
41         p.m1();
42         p.m2();
43
44         P p = new C(); // Error: Cannot assign child type to parent reference
45         p.m1();
46         p.m2();*/
47
48         C c = new P(); // Error: Cannot assign child type to parent reference
49     }
50 }
```

Annotations in red highlight several parts of the code:

- A red box surrounds the line `C c = new P();` with a red X drawn over it.
- Red X's are drawn over the commented-out code blocks starting with `/*C c = new C();`, `P p = new P();`, and `P p = new C();`.
- A large red circle highlights the line `C c = new P();` with a red X drawn over it.

In the top right corner, there is a "Questions" panel showing a list of user interactions:

X	Question	Asker
invalid		Shubham Gupta
noo		Govindu Rayapur
yes		Shruti sahu
invalid		Uday Baba
s		Amit Kumar
yes		Shubham Gupta

At the bottom, there is a status bar showing the file name `Test.java`, line number `27:24`, column `col 24`, and other system information.

The screenshot shows the CodeWarrior IDE interface with the following code in the `testOnly.java` file:

```
1 package ExamplePrograms;
2
3 class testOnly {
4     public void m1() {
5     }
6 }
7
8 class test2 extends testOnly {
9     public void m2() {
10 }
11 }
12
13 public static void main(String args[]) {
14     testOnly p = new test2();
15     p.m1();
16     p.m2();
17 }
```

A tooltip window is open at the bottom of the code editor, showing the following message:

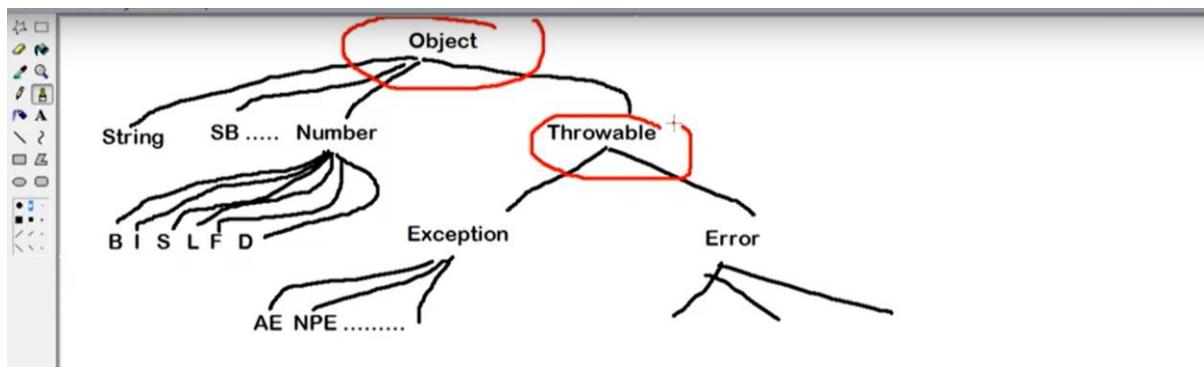
Cannot resolve method m2 in 'testOnly'
Cast qualifier to 'ExamplePrograms\$test2' Alt+Shift+Enter More actions... Alt+Enter
No candidates found for method call p.m2().
CodeWars

The IDE interface includes a project tree on the left and various toolbars and status bars at the bottom.

The screenshot shows a presentation slide with the following text:

- 31 common methods which are applicable for any child in parent class
- 32 The child specific methods in child class
- 33
- 34
- 35 Total Java API is developed based on inheritance
- 36

The most common methods which are applicable for all objects are available in Object class. The most common methods applicable for exceptions and errors are available in throwable



Why class cannot extend multiple class at the same time. Ambiguity. Multiple inheritance is not possible in Java

```
37 Multiple inheritance:  
38 -----  
39 Having more than one parent class  
40  
41  
42 P1 → m1()  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55
```

P1 → m1()
P2 → m1()
C.m1();

X	Question	Asker
in c++ it is allowed	Amit Kumar	
yes	Shubham Gupta	
yes	john acharya	
yes	Amit Kumar	
clear	Govindu Rayapur	
yes	Shruti sahu	

OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 01-03-2018

Multiple inheritance:
Having more than one parent class

```

Object
class A extends B
{
}

```

multi level inheritance

In 46 col 1 65 00 PC ANSI

interface A extends B,C

```

49
50 interface A extends B,C
51 {
52 }
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

In 50 col 24 67 00 PC ANSI

One implementation is enough. Ambiguity is not there in interface

```
1 interface Left
2 {
3     default void m1()
4     {
5         System.out.println("Left Default method");
6     }
7 }
8 interface Right
9 {
10    default void m1()
11    {
12        System.out.println("Right Default method");
13    }
14 }
15 class Test implements Left,Right
16 {
17     public static void main(String[] args)
18     {
19     }
20 }
21 }
22 }
```

You can avoid this by overriding the method m1 in child class like below

```
15 class Test implements Left,Right
16 {
17     public void m1()
18     {
19     }
20     public static void main(String[] args)
21     {
22 }
```



```
15 class Test implements Left,Right
16 {
17     public void m1()
18     {
19         Left.super.m1();
20         Right.super.m1();
21     }
22     public static void main(String[] args)
23     {
24         Test t = new Test();
25         t.m1();
26     }
27 }
```

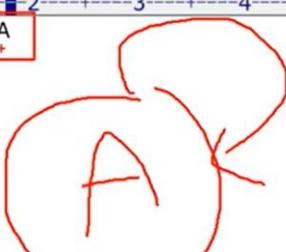
Interface static method is not available to implement class its available to all should be called with interface name. Interf.staticmethod()

```
1 interface Left
2 {
3     public static void m1()
4     {
5         System.out.println("Left Default method");
6     }
7 }
8 interface Right
9 {
10    default void m1()
11    {
12        System.out.println("Right Default method");
13    }
14 }
```

cyclic inheritance- Not allowed – Cyclic inheritance

```
1 class A extends B
2 {
3 }
4 class B extends A
5 {
6 }
```

```
1 class A extends A
2 {
3 }
4
```

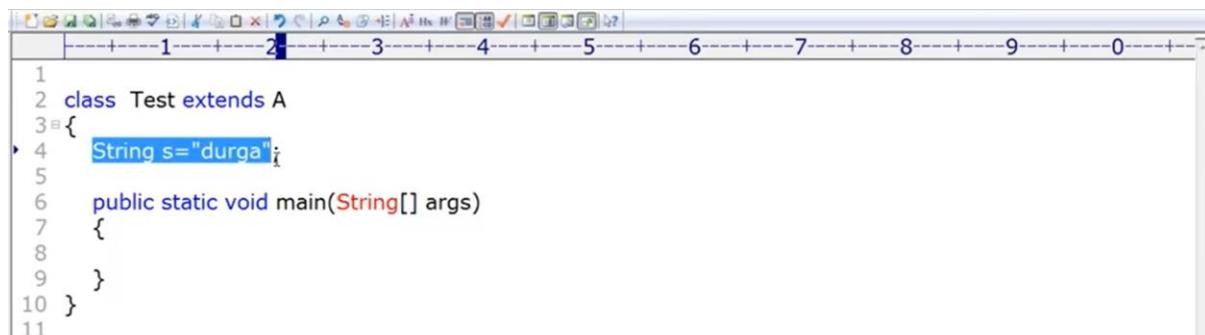


X	Question	Asker
1	cyclic not allowed	Ranjit Chavan
2	never seen this	Shubham Gupta
3	invalid	Govindu Rayapur
4	circular problem	Amit Kumar
5	cyclic	

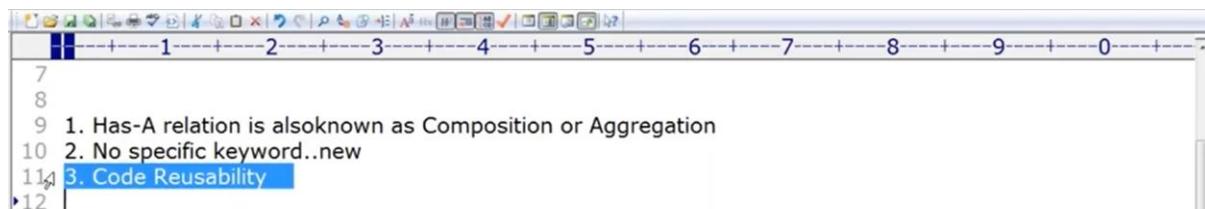
HAS A RELATION

If a class extends a class – Is A Relationship

If a class uses another class ref – line 4 Has a Relationship. Highly used relation



```
1
2 class Test extends A
3 {
4     String s="durga";
5
6     public static void main(String[] args)
7     {
8
9     }
10}
11
```



- 7
- 8
- 9 1. Has-A relation is also known as Composition or Aggregation
- 10 2. No specific keyword..new
- 11 3. Code Reusability
- 12

If you want to extend the functionality use Is A relation

If you want to use a functionality of a class use Has A relation. Use some other class

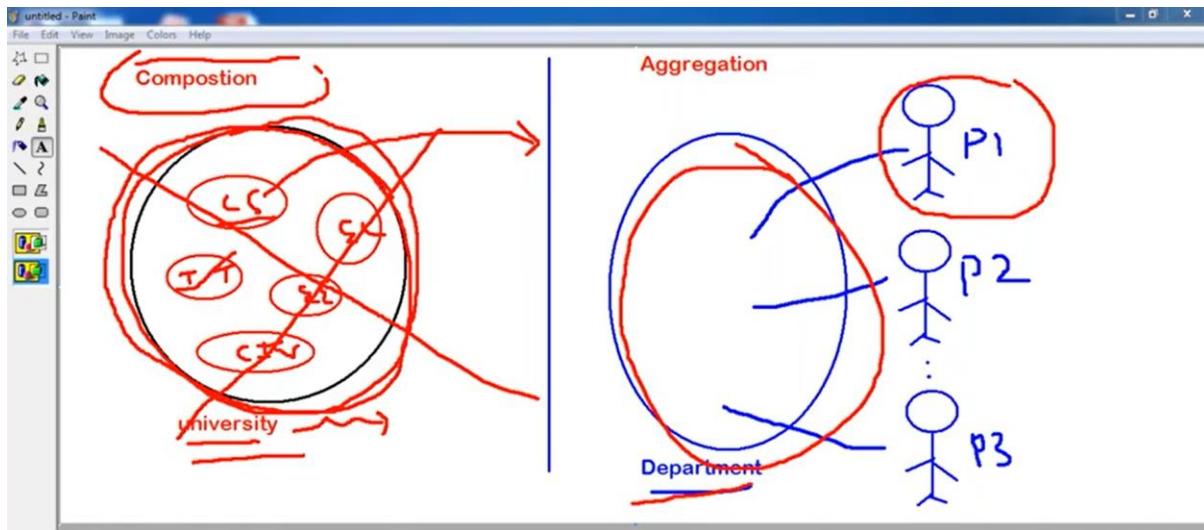
Has a relation

Has a Relation

Composition and Aggregation

If the contained objects are strongly associated with the container object it's called Composition.

If there is a week association then its called Aggregation



In composition , the contained objects are created inside the container object whereas in Aggregation the contained objects are outside the container object and accessed by reference

Both are has a , without name there could not be a student strong association , without college name there can be student. So there is a week association between student and cname

```
1 class Student
2 {
3     String name;
4     static String cname;
5 }
```

Composition is a strong type of "**has-a**" relationship because the containing object is its owner. So, objects are tightly coupled, which means if we delete the parent object, the child object will also get deleted with it.

```

Composition
256 final class Car {
    private final Engine engine;
    Car(EngineSpecs specs) {
        engine = new Engine(specs);
    }
    void move() {
        engine.work();
    }
}

```

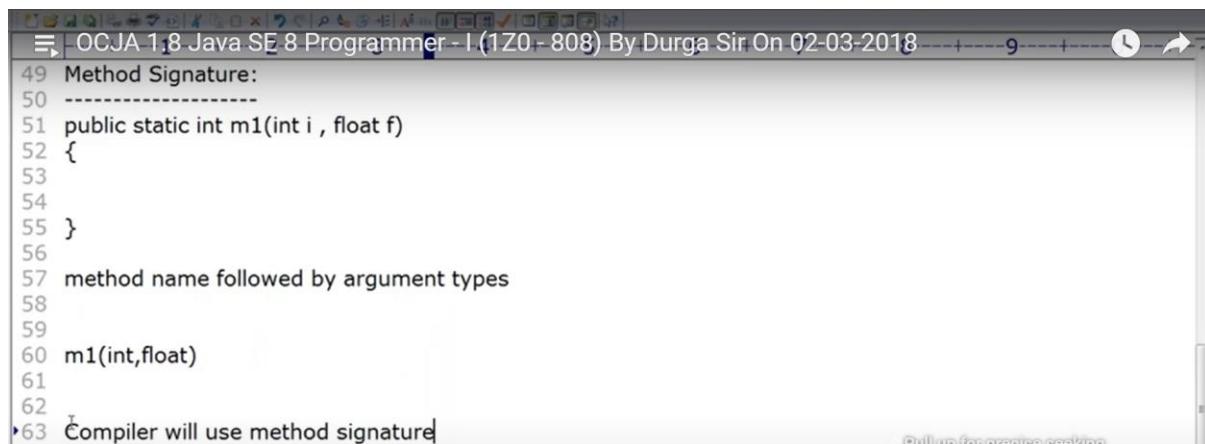
Aggregation

```

final class Car {
    private Engine engine;
    void setEngine(Engine engine) {
        this.engine = engine;
    }
    void move() {
        if (engine != null)
            engine.work();
    }
}

```

In the case of composition, the Engine is completely encapsulated by the Car. There is no way for the outside world to get a reference to the Engine. The Engine lives and dies with the car. With aggregation, the Car also performs its functions through an Engine, but the Engine is not always an internal part of the Car. Engines may be swapped, or even completely removed. Not only that, but the outside world can still have a reference to the Engine, and tinker with it regardless of whether it's in the Car.



The screenshot shows a Java code editor window with the title bar "OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) By Durga Sir On 02-03-2018". The code in the editor is:

```

49 Method Signature:
50 -----
51 public static int m1(int i , float f)
52 {
53
54
55 }
56
57 method name followed by argument types
58
59
60 m1(int,float)
61
62
63 Compiler will use method signature

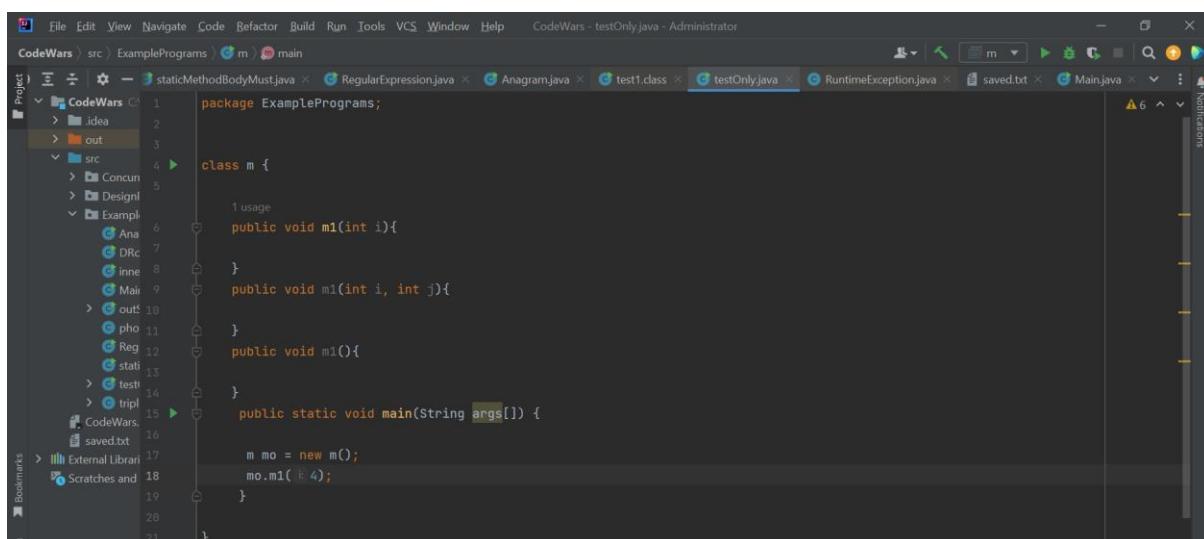
```

The status bar at the bottom right says "Pull up for more code scrolling".

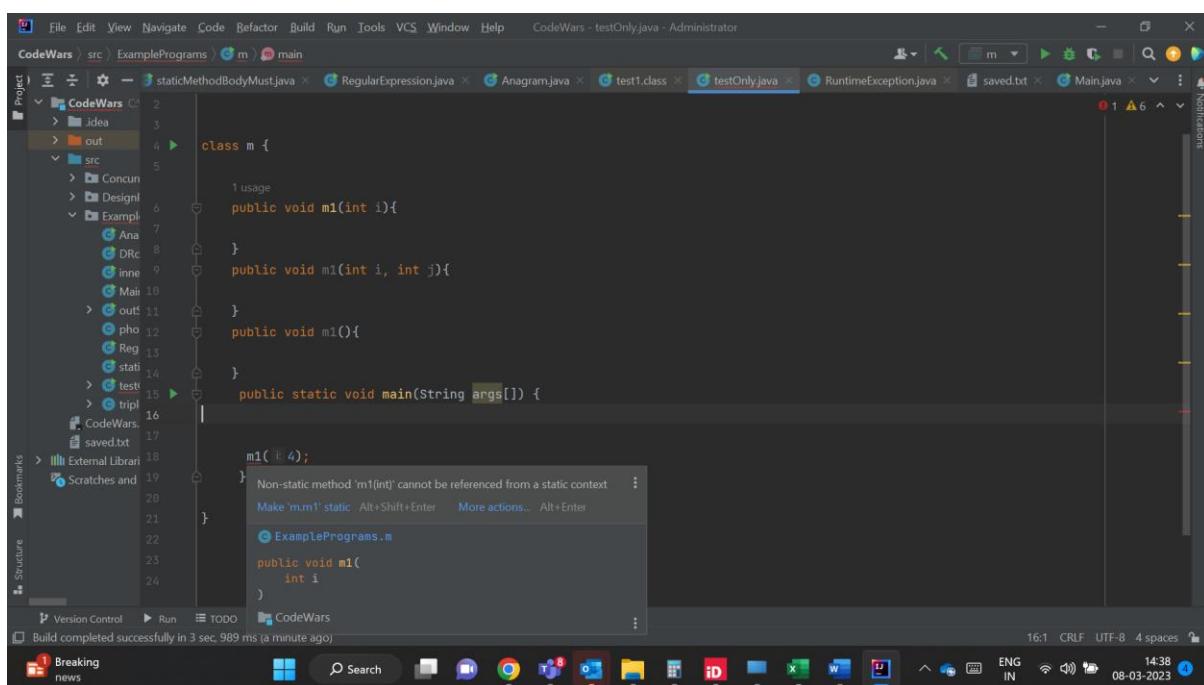
Look below the methods are not static but we are able to access non static method from main method by object creation



```
1 class Test
2 {
3     public void m1(int i){}
4     public void m1(int i,double d){}
5     public void m1(String s){}
6     public static void main(String[] args)
7     {
8         Test t = new Test();
9         t.m1(10);
10    }
11 }
```



```
1 package ExamplePrograms;
2
3 class m {
4     public void m1(int i){
5     }
6     public void m1(int i, int j){
7     }
8     public void m1(){
9     }
10    public static void main(String args[]){
11        m mo = new m();
12        mo.m1(4);
13    }
14 }
```



```
1 package ExamplePrograms;
2
3 class m {
4     public void m1(int i){
5     }
6     public void m1(int i, int j){
7     }
8     public void m1(){
9     }
10    public static void main(String args[]){
11        m1(4);
12    }
13 }
```

Compiler will always check method signature when solving methods

```
D:\durgaclasses>javac Test.java
D:\durgaclasses>javac Test.java
Test.java:9: error: no suitable method found for
        t.m1(10,"durga",10.5);
                           ^
method Test.m1(int) is not applicable
    (actual and formal argument lists differ in length)
method Test.m1(int,double) is not applicable
    (actual and formal argument lists differ in length)
method Test.m1(String) is not applicable
    (actual and formal argument lists differ in length)
1 error
```

Two methods with same signature and same return type is not possible

```
1 class Test
2 {
3     public void m1(int i)
4     {
5     }
6     public int m1(int j)
7     {
8         return 10;
9     }
10 }
```

**Test t = new Test();
t.m1(10);**

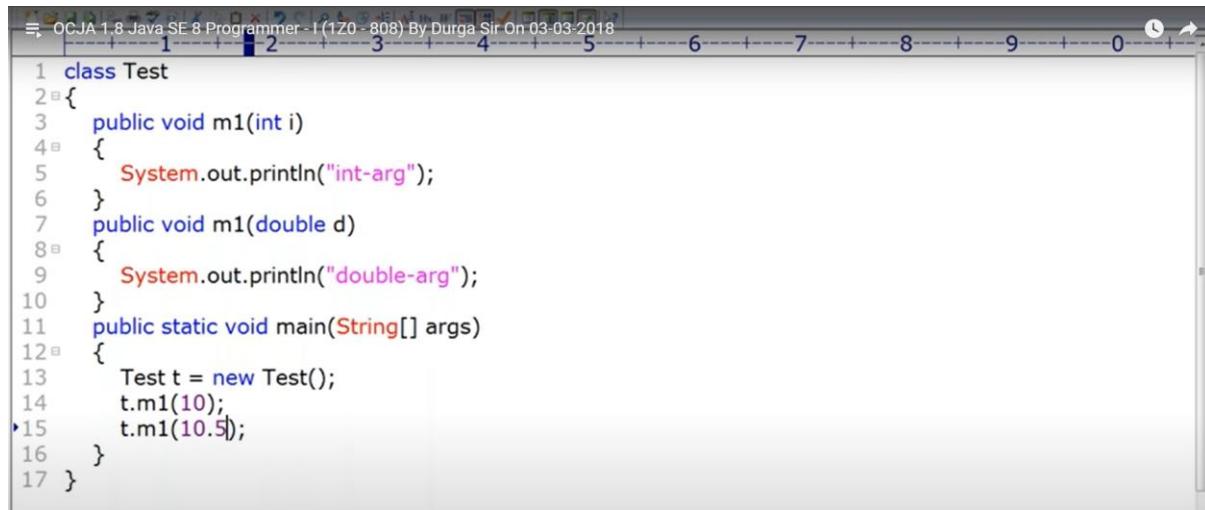
```
OCJA1.8 Java SE 8 Programmer I (1Z0-808) By Durga Sir On 02-03-2018
1 error

D:\durgaclasses>javac Test.java
Test.java:6: error: method m1(int) is already defined in class Test
    public int m1(int j)
                           ^
```

Overloading

Method with diff arg types , different sequence and no of args. Even if they are present in child class if the sig are diff its called overloading

```
11
12 Java:
13 abs(int)
14 abs(long)
15 abs(float)
16 abs(double)
17
18
19 m1(int i,float f)
20 m1(float f,int i)
```



A screenshot of a Java code editor showing a class named Test. The code defines two methods: m1(int i) which prints "int-arg" and m1(double d) which prints "double-arg". In the main() method, two calls are made: t.m1(10); and t.m1(10.5);. The code editor highlights the first call in blue and the second in red, demonstrating how the compiler resolves the method based on the reference type of the variable t.

```
1 class Test
2 {
3     public void m1(int i)
4     {
5         System.out.println("int-arg");
6     }
7     public void m1(double d)
8     {
9         System.out.println("double-arg");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1(10);
15        t.m1(10.5);
16    }
17 }
```

In Overloading the method resolution taken care by compiler, which method to be executed, based **on ref type**. Hence its resolved during compile time its called compile time polymorphism.

Ref type – Test T. Early binding since its resolved compile time

```
1 class Test
2 {
3     public void m1(int i)
4     {
5         System.out.println("int-arg");
6     }
7     public void m1(double d)
8     {
9         System.out.println("double-arg");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1(10);
15        t.m1(10.5);
16    }
17 }
```

In overloading automatic promotion is taken care. If exact match is not found promote to next level.
If int is not available then call long

Byte->Short-> int->Long

Char-> int

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

```
1 class Test
2 {
3     public void m1(int i)
4     {
5         System.out.println("int-arg");
6     }
7     public void m1(long l)
8     {
9         System.out.println("long-arg");
10    }
11    public void m1(float i)
12    {
13        System.out.println("float-arg");
14    }
15    public static void main(String[] args)
16    {
17        Test t = new Test();
18        t.m1(10);
19        t.m1(10L);
20        t.m1('a');
21
22    }
}
```

The code defines a class `Test` with three methods: `m1` taking an `int`, `m1` taking a `long`, and `m1` taking a `float`. The `main` method calls each of these methods with the arguments `10`, `10L`, and `'a'` respectively. The output of the program is "int-arg", "long-arg", and "float-arg" respectively, demonstrating that the `int` argument is promoted to a `long` before being passed to the method.

Double to float not possible below compile time error

The screenshot shows a Java code editor with the same code as the previous screenshot, but with a syntax error highlighted at the line `t.m1(10.5);`. The number `10.5` is underlined with a blue line, indicating a compilation error. This error occurs because Java does not allow implicit conversion from `double` to `float`.

```
4 {
5     System.out.println("int-arg");
6 }
7 public void m1(long l)
8 {
9     System.out.println("long-arg");
10 }
11 public void m1(float i)
12 {
13     System.out.println("float-arg");
14 }
15 public static void main(String[] args)
16 {
17     Test t = new Test();
18     t.m1(10);
19     t.m1(10L);
20     t.m1('a');
21     t.m1(10.5);
22
23 }
24 }
```

Below is correct

A screenshot of a Java code editor window. The title bar reads "OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) By Durga Sir On 03-03-2018". The code editor displays the following Java code:

```
1 class Test
2 {
3     public void m1(String s)
4     {
5         System.out.println("String version");
6     }
7     public void m1(Object s)
8     {
9         System.out.println("Object version");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1("durga");
15        t.m1(new Object());
16    }
17 }
18 }
```

The code defines a class named `Test` with two methods: `m1` for `String` and `m1` for `Object`. The `main` method creates an instance of `Test` and calls both `m1` methods with different arguments. The code editor interface includes a toolbar at the top, a status bar at the bottom, and tabs for multiple files.

Null is valid object for String and object, but in the case low level or child argument -> String is going to get pref. More specific argument will get the highest priority. If no spec arg then high level arg

A screenshot of a Java code editor window. The title bar reads "OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) By Durga Sir On 03-03-2018". The code editor displays the same Java code as the previous screenshot, but with a difference in the `main` method:

```
1 class Test
2 {
3     public void m1(String s)
4     {
5         System.out.println("String version");
6     }
7     public void m1(Object s)
8     {
9         System.out.println("Object version");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1("durga");
15        t.m1(new Object());
16        t.m1(null);           I
17    }
18 }
19 }
```

In this version, the `main` method includes a call to `t.m1(null)`. The code editor highlights the argument `null` in blue, indicating it is being passed to the `m1` method that expects a `String`. This demonstrates that the more specific argument (null) takes precedence over the higher-level argument type (`Object`). The code editor interface is identical to the first screenshot.

Exact match gets high priority

```
1 class Test
2 {
3     public void m1(String s)
4     {
5         System.out.println("String version");
6     }
7     public void m1(StringBuffer sb)
8     {
9         System.out.println("StringBuffer version");
10 }
11 public static void main(String[] args)
12 {
13     Test t = new Test();
14     t.m1("durga");
15 }
```

Questions

- String one
- Error
- CE
- CE :Same Level ambiguity
- String
- Stringbuffer
- string
- because stringbuffer is child of string
- String version
- difference between string and stringbuffer?
- String Version
- Iol sorry

Asker R...

SHUBHAM GUPTA RAMESWARA REDDY
RANJIT CHAVAN SHRUTI SAHU
RICH COHEN HARSH KHANDELWAL
AMIT KUMAR RICH COHEN
GOVINDU RAYAPUR AMIT KUMAR
RAMESWARA REDDY RICH COHEN

why this priority though? Can you give an example why it's helpful to give chance to String over object

```
1 class Test
2 {
3     public void m1(String s)
4     {
5         System.out.println("String version");
6     }
7     public void m1(StringBuffer sb)
8     {
9         System.out.println("StringBuffer version");
10 }
11 public static void main(String[] args)
12 {
13     Test t = new Test();
14     t.m1("durga");
15     t.m1(new StringBuffer("durga"));
16 }
17 }
```

Compile time error, since both are specific

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

```
1 class Test
2 {
3     public void m1(String s)
4     {
5         System.out.println("String version");
6     }
7     public void m1(StringBuffer sb)
8     {
9         System.out.println("StringBuffer version");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1("durga");
15        t.m1(new StringBuffer("durga"));
16        t.m1(null);
17    }
18 }
```

A red squiggly underline is under the call to `t.m1(null);`, indicating a compilation error. The status bar at the bottom of the IDE window shows the message "Method overloading error".

The screenshot shows a terminal window with the following output:

```
Select C:\Windows\system32\cmd.exe
String version
StringBuffer version

D:\durgaclasses>javac Test.java
Test.java:16: error: reference to m1 is ambiguous
          t.m1(null);
                  ^
both method m1(String) in Test and method m1(StringBuffer) in Test match
1 error

D:\durgaclasses>
```

Int to float promotion is possible, but why are u seeing int first see second method go from right to left and compare args. There is not left to right or right to left

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

```
1 class Test
2 {
3     public void m1(int i, float f)
4     {
5         System.out.println("int-float version");
6     }
7     public void m1(float f, int i)
8     {
9         System.out.println("float-int version");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1(10,10.5f);
15        t.m1(10.5f,10);
16        t.m1(10,10);
17    }
18 }
```

Annotations with red arrows point to the parameters of the first two `m1` method definitions, highlighting the order of arguments. A yellow box highlights the call `t.m1(10,10.5f);`. A red box highlights the call `t.m1(10,10);`.

To the right of the code editor, there is a sidebar titled "Questions" containing a list of user questions and answers. Some questions are marked with a red "X".

Question	Asker	Replies
int-float	Amit Kumar	1
Ambiguity	RAMESWARA REDDY	1
CE	Ranjit Chavan	1
error	Rich Cohen	1
int will promote to float	Shruti sahu	1
haaaaa	Ranjit Chavan	1
float*	Shruti sahu	1
int-float 100%	Shubham Gupta	1
yes	Shruti sahu	1
int-float	Shubham Gupta	1
int-float version	Shruti sahu	1
why int-float?	Rich Cohen	1
ERROR	Ranjit Chavan	1
int-float	Amit Kumar	1

```
Select C:\Windows\system32\cmd.exe
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 03-03-2018
int-float version
float-int version

D:\durgaclasses>javac Test.java
Test.java:16: error: reference to m1 is ambiguous
        t.m1(10,10);
               ^
    both method m1(int,float) in Test and method m1(float,int) in Test match
1 error
```

CE, cant promote from float to int.

```
Select C:\Windows\system32\cmd.exe
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 03-03-2018
1 class Test
2 {
3     public void m1(int i,float f)
4     {
5         System.out.println("int-float version");
6     }
7     public void m1(float f, int i)
8     {
9         System.out.println("float-int version");
10    }
11    public static void main(String[] args)
12    {
13        Test t = new Test();
14        t.m1(10,10.5f);
15        t.m1(10.5f,10);
16        //t.m1(10,10);
17        t.m1(10.5f,10.5f);
18    }
19 }
```

```
Select C:\Windows\system32\cmd.exe
D:\durgaclasses>javac Test.java
Test.java:16: error: reference to m1 is ambiguous
        t.m1(10,10);
               ^
    both method m1(int,float) in Test and method m1(float,int) in Test match
1 error

D:\durgaclasses>javac Test.java
Test.java:17: error: no suitable method found for m1(float,float)
        t.m1(10.5f,10.5f);
               ^
method Test.m1(int,float) is not applicable
    (argument mismatch; possible lossy conversion from float to int)
method Test.m1(float,int) is not applicable
    (argument mismatch; possible lossy conversion from float to int)
1 error
```

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 03-03-2018  
1 class Animal  
2 {  
3 }
```

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 03-03-2018  
4 class Monkey extends Animal  
5 {  
6 }  
7 class Test  
8 {  
9     public void m1(Animal a)  
10    {  
11        System.out.println("Animal version");  
12    }  
13    public void m1(Monkey mi)  
14    {  
15        System.out.println("Monkey version");  
16    }  
17    public static void main(String[] args)  
18    {  
19        Test t = new Test();  
20  
21        Animal a = new Animal();  
22        t.m1(a);  
23  
24        Monkey m = new Monkey();  
25        t.m1(m);  
26    }  
27 }  
28  
29  
30
```

Compiler never checks about run time object it only checks the ref object. Here it checks m1(Animal)

It never checks run time object during overloading

```
27 Animal a1= new Monkey();  
28 t.m1(a1);  
29  
30
```

In Method overloading method resolution is taken care by compiler – Overriding by JVM

Overloading we consider only ref type – Overriding we consider Run time object

```
both methods having same name but different argument types  
4 Compiler based on reference type  
5 Static polymorphism or compile time polymorphism or early binding  
6
```

Overriding

Compiler job is to choose parent.

```
10 public void marry()
11 {
12     System.out.println("Child");
13 }
14 }
15 class Test
16 {
17     public static void main(String[] args)
18     {
19         P p = new P();
20         p.marry();
21
22         C c = new C();
23         c.marry();
24
25         P p1 = new C(); // Red box
26         p1.marry(); // Red circle
27     }
28
29 }
```

For Help, press F1 * Untitled1 * Test.java In 26 col 17 30 28 PC ANSI

Questions

X	Question	Asker
mary()	mary()	Amit Kumar
enter	enter	Shubham Gupta
child	child	Rich Cohen
marry()	marry()	Shubham Gupta
Parent	Parent	Govindu Rayapur
parent	parent	Amit Kumar
hahaha	hahaha	Shruti Sahu
booz child object is created so child method will be called	booz child object is created so child method will be called	Amit Kumar
somy	somy	Shubham Gupta
parent	parent	Amit Kumar
child method	child method	Amit Kumar
run time overidding	run time overidding	Amit Kumar
child	child	Ranjit Chavan
late binding	late binding	Amit Kumar
parent	parent	Srujanreddy Gavar...
as child object so child	as child object so child	Rich Cohen

If no how lectures are pending

Send Privately Send to All

At run time it sees what object type it is, then sees if the method is there or not and choose child

```
10 public void marry()
11 {
12     System.out.println("Child");
13 }
14 }
15 class Test
16 {
17     public static void main(String[] args)
18     {
19         P p = new P();
20         p.marry();
21
22         C c = new C();
23         c.marry();
24
25         P p1 = new C(); // Red box
26         p1.marry(); // Red circle
27     }
28
29 }
```

For Help, press F1 * Untitled1 * Test.java In 26 col 17 30 28 PC ANSI

Questions

X	Question	Asker
mary()	mary()	Amit Kumar
enter	enter	Shubham Gupta
child	child	Rich Cohen
marry()	marry()	Shubham Gupta
Parent	Parent	Govindu Rayapur
parent	parent	Amit Kumar
hahaha	hahaha	Shruti Sahu
booz child object is created so child method will be called	booz child object is created so child method will be called	Amit Kumar
somy	somy	Shubham Gupta
parent	parent	Amit Kumar
child method	child method	Amit Kumar
run time overidding	run time overidding	Amit Kumar
child	child	Ranjit Chavan
late binding	late binding	Amit Kumar
parent	parent	Srujanreddy Gavar...
as child object so child	as child object so child	Rich Cohen

If no how lectures are pending

Send Privately Send to All

If marry method is not available in Parent class then we are going to get compile time error.

If marry method is not there

```
1 D:\durgaclasses>javac Test.java
1
1 D:\durgaclasses>javac Test.java
1 Test.java:23: error: cannot find symbol
1     p1.marry();
1         ^
1   symbol:   method marry()
1   location: variable p1 of type P
1
1 error
```

Compiler is gonna check at compile time if marry method is there are not in Parent Class if not it will throw CE

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 05-03-2018
Press Esc to exit full screen 6-----7-----8-----9-----0-----+
4 Compiler based on reference type
5 Static polymorphism or compile time polymorphism or early binding
6 -----
7 Overriding:
8 -----
9
10 Based on Runtime object
11 Runtime polymorphism
12 dynamic
13 Late Binding
```

```
18 overloading:
19 -----
20 compiler
21 based on reference type
22
23 overriding:
24 -----
25 jvm
26 based on runtime object
```

Order of signature is also very important

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 05-03-2018
Press Esc to exit full screen 1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----+
27
28
29 Rules for overriding:
30 -----
31 1. Method Names same
32 Argument Types must be same
33 Method Signatures
```

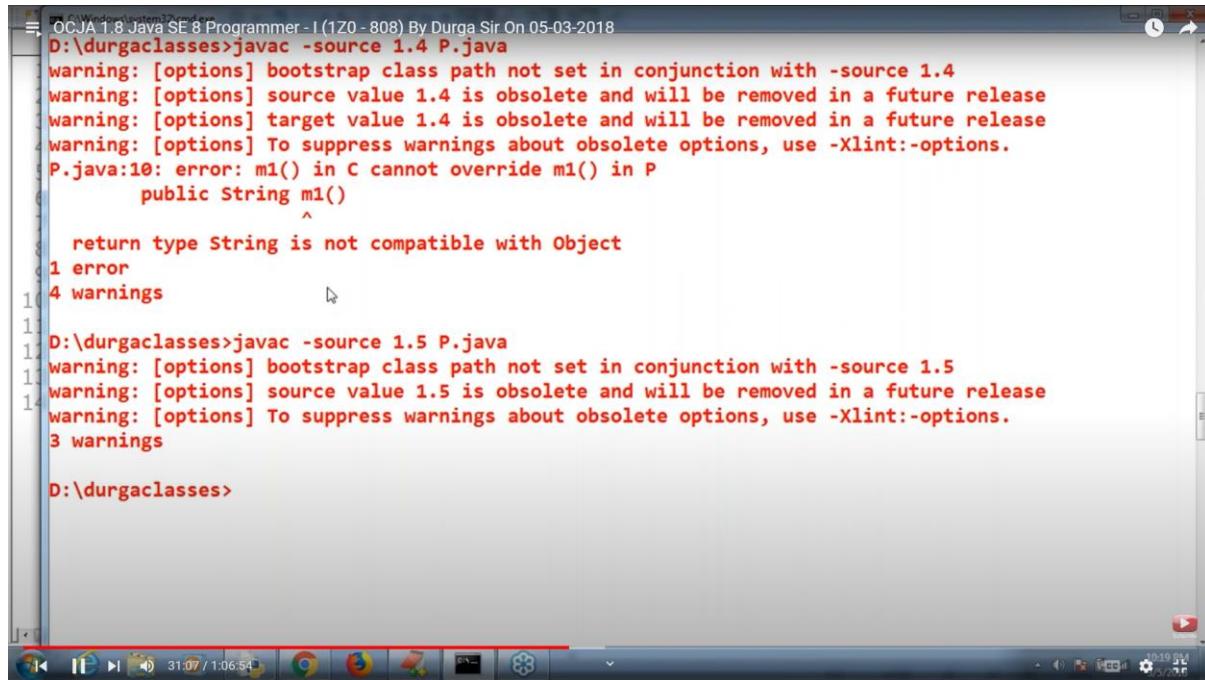
```
35 2.  
36 1.4 Version only  
37 1.5 version onwards  
38 co-varient return types
```

Child method return type can be diff from Parent method return type- This is called covariant return type



```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 05-03-2018  
1 class P  
2 {  
3     public Object m1()  
4     {  
5         return null;  
6     }  
7 }  
8 class C extends P  
9 {  
10    public String m1()  
11    {  
12    }  
13 }
```

1.5 and 1.4

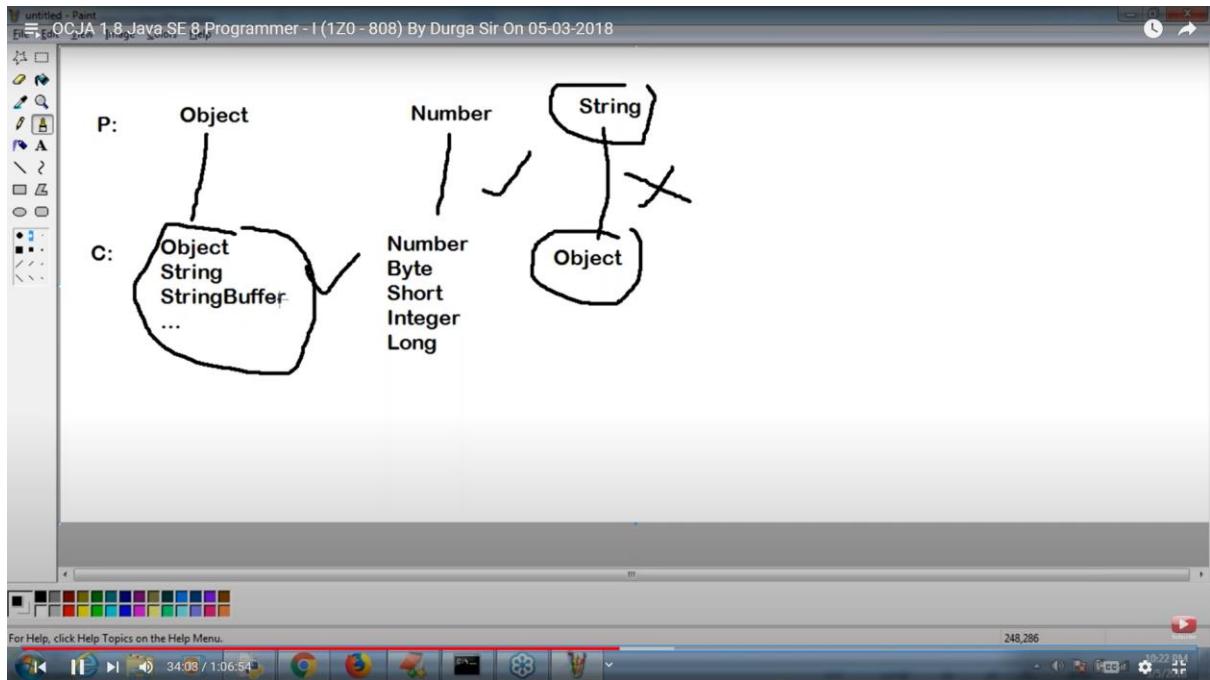


```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) By Durga Sir On 05-03-2018  
D:\durgaclasses>javac -source 1.4 P.java  
warning: [options] bootstrap class path not set in conjunction with -source 1.4  
warning: [options] source value 1.4 is obsolete and will be removed in a future release  
warning: [options] target value 1.4 is obsolete and will be removed in a future release  
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.  
P.java:10: error: m1() in C cannot override m1() in P  
        public String m1()  
                           ^  
          return type String is not compatible with Object  
1 error  
4 warnings  
D:\durgaclasses>javac -source 1.5 P.java  
warning: [options] bootstrap class path not set in conjunction with -source 1.5  
warning: [options] source value 1.5 is obsolete and will be removed in a future release  
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.  
3 warnings  
D:\durgaclasses>
```

Return type criteria.

Child class method return type can be of same type a parent class method return type or child of parent return types

But parent method return type cannot be a child of child method return type



The below is correct

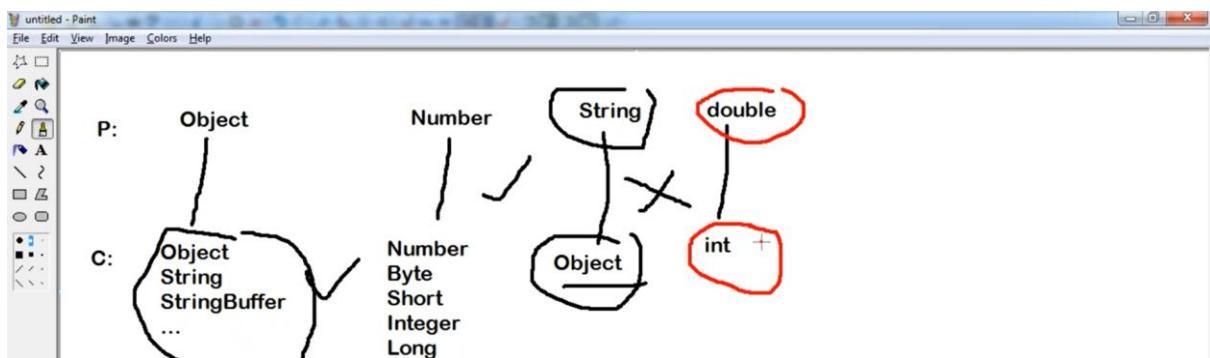
```
1 class P
2 {
3     public Object m1()
4     {
5         return null;
6     }
7 }
8 class C extends P
9 {
10    public String m1()
11    {
12        return null;
13    }
14 }
```

The below is an error

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 05-03-2018
1 class P
2 {
3     public String m1()
4     {
5         return null;
6     }
7 }
8 class C extends P
9 {
10    public Object m1()
11    {
12        return null;
13    }
14 }
```

```
D:\durgaclasses>javac P.java
P.java:10: error: m1() in C cannot override m1() in P
    public Object m1()
                           ^
    return type Object is not compatible with String
1 error
```

Covariant return type only matches for only for object types and not for primitive types



If parent is double child should also be double. Both should be int

Private methods are not available for child class then overriding is not applicable

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 05-03-2018
1 class P
2 {
3     private void m1()
4     {
5     }
6 }
7 class C extends P
8 {
9 }
10 }
```

Same private methods can be defined in child class, but this is not overriding..

CE. P is private how can u call it from outside fool

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 05-03-2018
1 class P
2 {
3     private void m1()
4     {
5     }
6 }
7 class C extends P
8 {
9     private void m1()
10    {
11    }
12 }
13
14 P p = new C();
15 p.m1();
```

Overriding is not possible for final methods

```
OCJA 1.8 Java SE 8 Programmer - I (120 - 808) By Durga Sir On 05-03-2018
1 class P
2 {
3     public final void m1()
4     {
5     }
6 }
7 class C extends P
8 {
9     public void m1()
10    {
11    }
12 }
13
14
```

Next level overriding is not possible

```
1 class P
2 {
3     public void m1()
4     {
5     }
6 }
7 class C extends P
8 {
9     public final void m1()
10    {
11    }
12 }
```



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

Child class extending c should implement. Abstract to non abstract and Non Abstract to Abstract is possible.

Why?

If I don't want to use parent method but I don't know how to implement the method then we can use this way. Or we want child class to implement these methods

Parent method implementation is not available to next level child class and they should provide their own implementation

```
1 class P
2 {
3     public void m1()
4     {
5     }
6 }
7 class C extends P
8 {
9 }
10 class CC extends C
11 {
12 }
13
14
15
```

public abstract void m1();

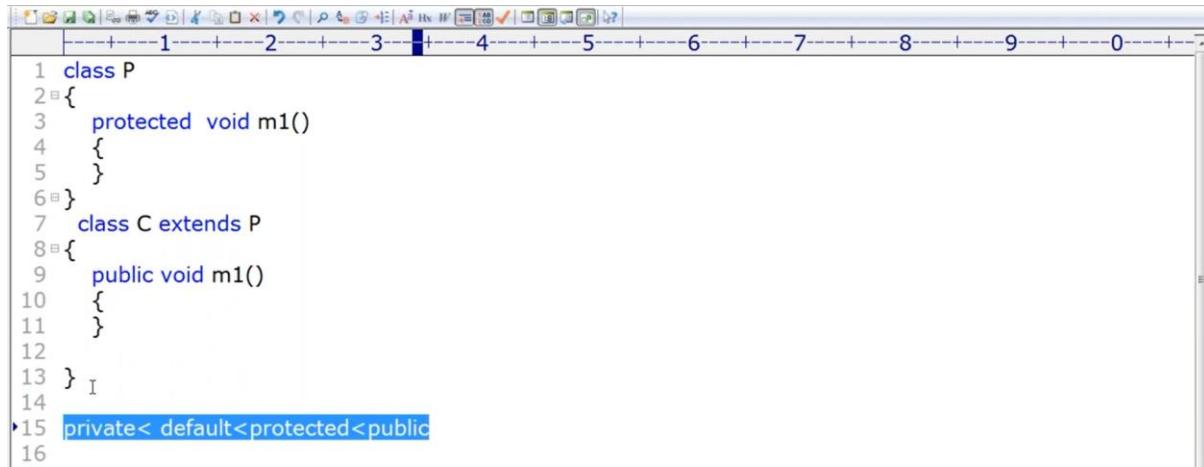
Questions

X	Question	Asker
valid	Shubham Gupta	
possible	Rich Cohen	
but not overriding	Rich Cohen	
not possible	Deepankaj Yadav	
sir what will be the need of doing this??	Shruti sahu	
valid	Shubham Gupta	
valid	RAMESWARA REDDY	
child use	Ranjit Chavan	
haha yes	Shruti sahu	
haha	Shruti sahu	
toy	Shubham Gupta	
yes	Shruti sahu	
yes	Shubham Gupta	
ok	Amit Kumar	
where we can implement such kind of condition?	Amit Kumar	
sir If child doesn't contains many method then what will happen in last case		

Send Privately Send to All

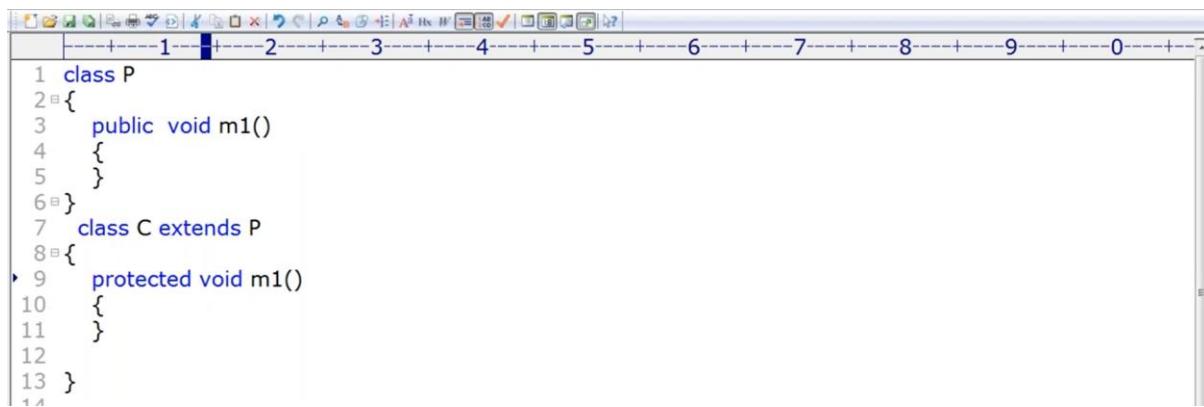
While overriding we can't reduce the scope of access modifiers, but you can increase the scope.

Protected to public is increasing the scope so it's acceptable

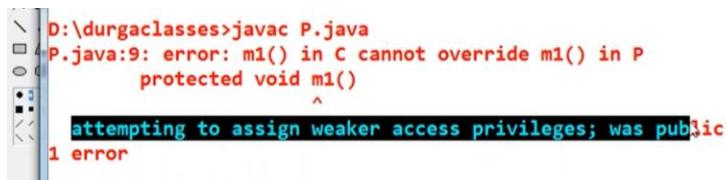


```
1 class P
2 {
3     protected void m1()
4     {
5     }
6 }
7 class C extends P
8 {
9     public void m1()
10    {
11    }
12 }
13 }
14
15 private< default<protected<public
16 }
```

Below is invalid



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



```
D:\durgaclasses>javac P.java
D:\durgaclasses\P.java:9: error: m1() in C cannot override m1() in P
        protected void m1()
                           ^
attempting to assign weaker access privileges; was public
1 error
```

Every method in interface is by default public the child class method is default type which is lower scope . public method overriding should be Public only, cant reduce the scope of modifier

```
1 interface P
2 {
3     void m1();
4 }
5 class C implements P
6 {
7     void m1()
8     {
9     }
10 }
```

PYSic

X	Question	Asker
v		Shubham Gupta
no		Shruti sahu
valid		Govindu Rayapur
s valid		Uday Baba
valid		Amit Kumar
✓		Hitesh Bajaniya

Same as above

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