

OOPS

data hiding
abstraction
encapsulation
inheritance
has-a relation
composition
aggregation
polymorphism
overloading
overriding
method hiding
coupling
cohesion

```
{ Data Hiding:  
  -----  
  outside person should not get our data directly.  
  Our internal data should not go out directly  
}  
private modifier  
  
Security  
data members(variables)==>private  
}
```

Abstraction:

```
-----  
10000  
Help Desk(May I Help You!!!!)  
-----  
java or .net or c or c++  
interfaces,GUI Screens...  
I  
1. Security  
2. Enhancement...
```

Data Hiding
Abstraction
Encapsulation
Inheritance(IS-A Relationship)

for code reusability and to extends existing functionality with some extra functionality

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.
2. Whatever the methods present in child class are not available to the Parent . Hence on the parent reference we cannot call child specific methods
- 3.

```
/*C c= new C();  
c.m1();  
c.m2();
```

```
P p = new P();  
p.m1();  
p.m2();*/
```

```
P p = new C();  
p.m1();  
p.m2();
```

```
}
```

polymorphism



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

```
{  
/*C c= new C();  
c.m1();  
c.m2();
```

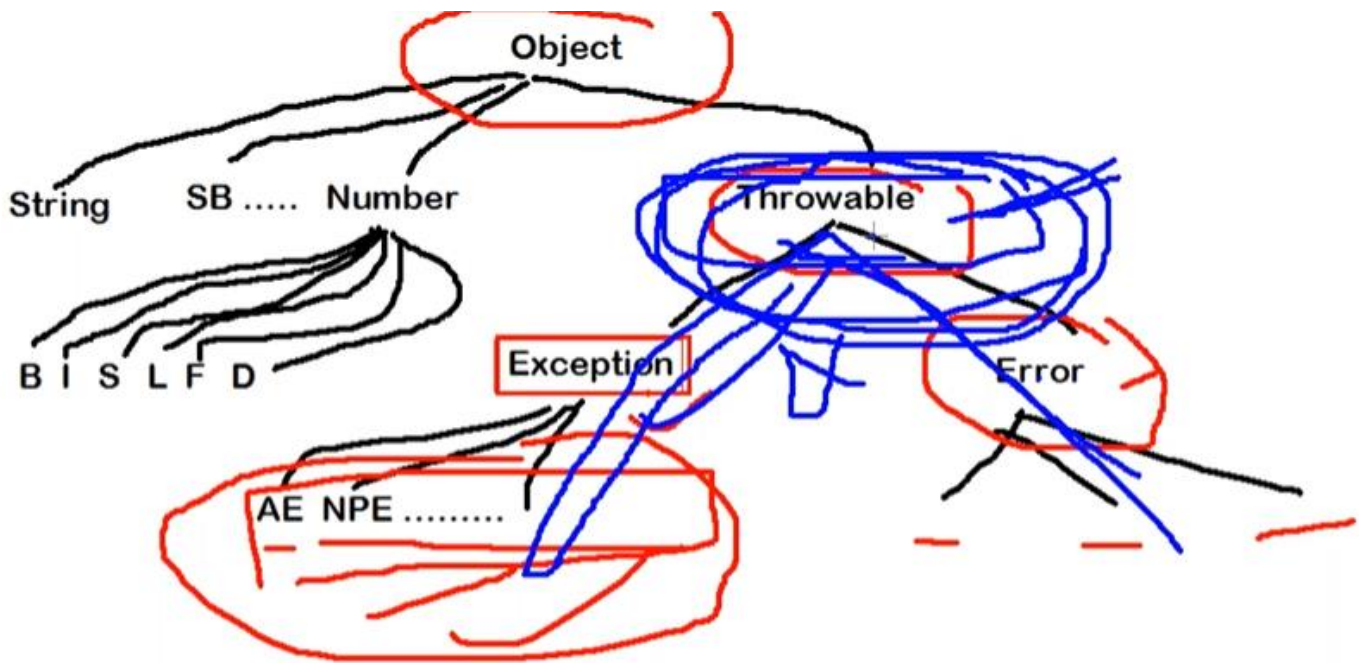
```
P p = new P();  
p.m1();  
p.m2();
```

```
P p = new C();  
p.m1();  
p.m2();*/
```

```
C c = new P();
```

```
}  
}
```





Multiple inheritance with interface

```

interface Left
{
    default void m1()
    {
        System.out.println("Left Default method");
    }
}

interface Right
{
    default void m1()
    {
        System.out.println("Right Default method");
    }
}

class Test implements Left, Right
{
    public void m1()
    {
        Left.super.m1();
        Right.super.m1();
    }

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

```

interface Left
{
    public static void m1()
    {
        System.out.println("Left Default method");
    }
}

interface Right
{
    public static void m1()
    {
        System.out.println("Right Default method");
    }
}

class Test implements Left, Right
{
    public static void main(String[] args)
    {
        Left.m1();
    }
}
  
```

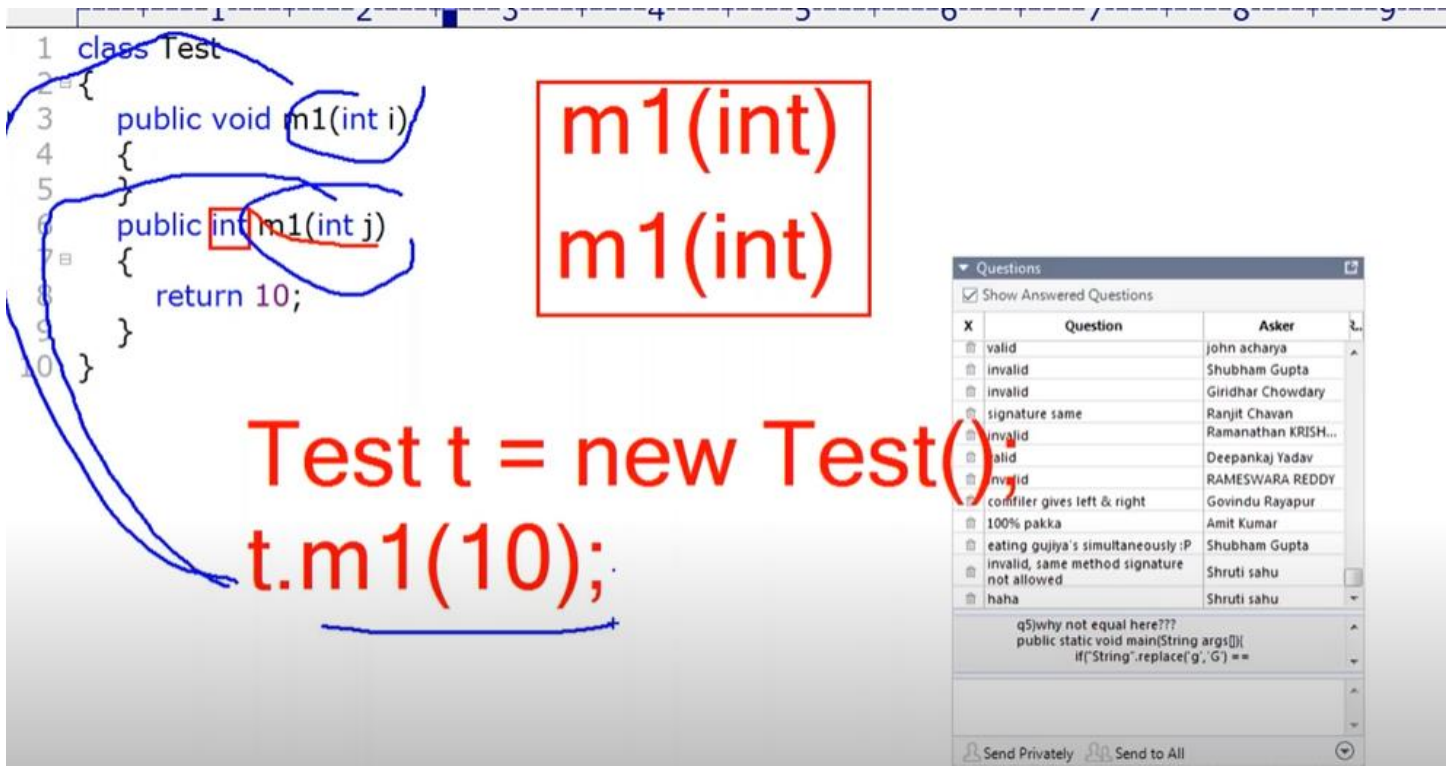
Method Overloading and overriding

Overloading

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

m1(int)
m1(int)

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



X	Question	Asker
<input checked="" type="checkbox"/>	valid	john acharya
<input checked="" type="checkbox"/>	invalid	Shubham Gupta
<input checked="" type="checkbox"/>	invalid	Giridhar Chowdary
<input checked="" type="checkbox"/>	signature same	Ranjit Chavan
<input checked="" type="checkbox"/>	invalid	Ramanathan KRISH...
<input checked="" type="checkbox"/>	valid	Deepankaj Yadav
<input checked="" type="checkbox"/>	invalid	RAMESWARA REDDY
<input checked="" type="checkbox"/>	compiler gives left & right	Govindu Rayapur
<input checked="" type="checkbox"/>	100% pakka	Amit Kumar
<input checked="" type="checkbox"/>	eating gujiya's simultaneously :P	Shubham Gupta
<input checked="" type="checkbox"/>	invalid, same method signature not allowed	Shruti sahu
<input checked="" type="checkbox"/>	haha	Shruti sahu

q5)why not equal here???
public static void main(String args[]){
 if("String".replace('g','G'))==

Send Privately Send to All

m1(double d)

In C Language:

abs(int)
labs(long)
fabs(float)

Java:

abs(int)
abs(long)
abs(float)
abs(double)

AUTO INCREMENT

```
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);
17    }
18 }
19 }
```

String

X	Question	Answer
<input type="checkbox"/>	yes	Amit K
<input type="checkbox"/>	yes	Shruti
<input type="checkbox"/>	yes	Shubh
<input type="checkbox"/>	understood now string is ana	Shubh
<input type="checkbox"/>	String bersion	Shruti
<input type="checkbox"/>	string	Rich C
<input type="checkbox"/>	version"	Shruti
<input type="checkbox"/>	String	Ranjit
<input type="checkbox"/>	string	Amit K
<input type="checkbox"/>	string	Harsh
<input type="checkbox"/>	String is attender	Shruti
<input type="checkbox"/>	string attender	Rich C
<input type="checkbox"/>	object--collector	Amit K
<input type="checkbox"/>	collector--high level	Shruti

if byte value is passed but we have short, int a will the promotion be?


```

{
    System.out.println("Animal version");
}
public void m1(Monkey m)
{
    System.out.println("Monkey version");
}
public static void main(String[] args)
{
    Test t = new Test();

    Animal a = new Animal();
    t.m1(a);

    Monkey m = new Monkey();
    t.m1(m);

    Animal a1 = new Monkey();
    t.m1(a1);
}

```

Compile time overloaded

Overriding

```

1 class P
2 {
3     public void property()
4     {
5         System.out.println("Ca$h+GOLD+Land+Power");
6     }
7     public 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("4Me|3Sha|9Tara");
17     }
18 }
19

```

→ Overridden method

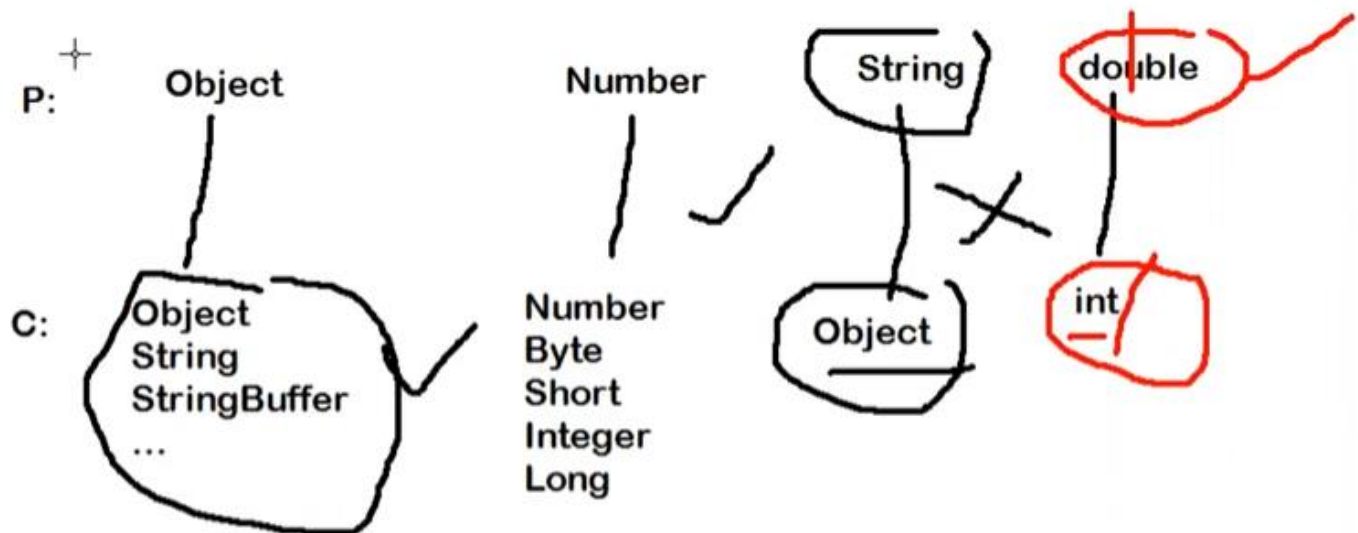
→ Overriding method

Questions

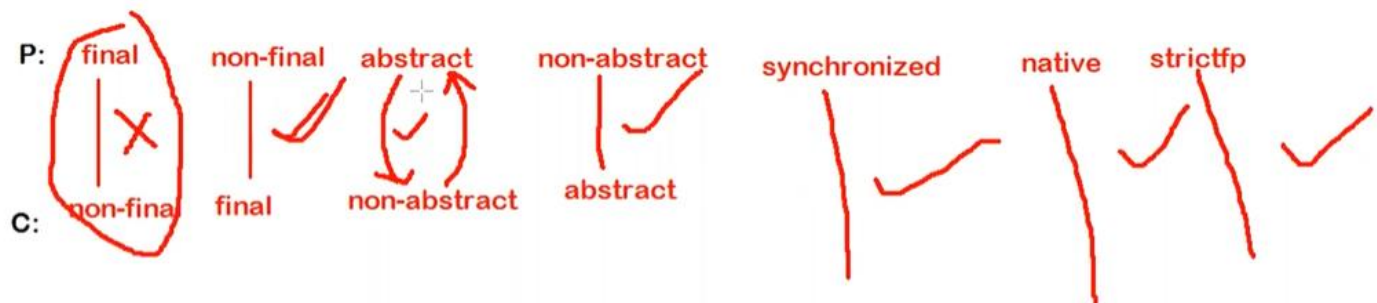
☒ Show Answered Questions

X	Question	Asker
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

If no how lectures are pending



Some modifiers for overriding



```

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

```

Red annotations: A red circle around the `public void m1()` method in class P. A red circle around the `public abstract void m1();` method in class C. A red circle around the `public abstract void m1();` method in class CC. A red circle around the `public abstract void m1();` method in class C and the `public abstract void m1();` method in class CC.

Questions	Answers
X	Show Answered
valid	Qu
possible	
but not overridi	
not possible	
sir what will be	
this??	
valid	
valid	
child use	
haha yes	
haha	
toy	
yes	
yes	
s	
ok	