

18_11_2022, interface, classmate - Notepad

File Edit View

Difference b/w extends vs implements

extends :: One class can extends only class at a time

eg:: class One{

class Two extends One{

interface IOne{

void m1();

}

interface ITwo{

void m2();

}

class CommonImpl implements IOne, ITwo{

public void m1(){

}

22°C

Polluted air

Q Search

Windows (CTRL)

100%

Windows (CTRL)

18:48

18-11-2022

18_11_2022, interface, classmate - Notepad

File Edit View

Difference b/w extends vs implements

extends :: One class can extends only class at a time

eg:: class One{

class Two extends One{

interface IOne{

void m1();

}

interface ITwo{

void m2();

}

class CommonImpl implements IOne, ITwo{

public void m1(){

}

22°C

Polluted air

Q Search

Windows (CTRL)

100%

Windows (CTRL)

18:48

18-11-2022



Nishu Manjunath

1

case2:

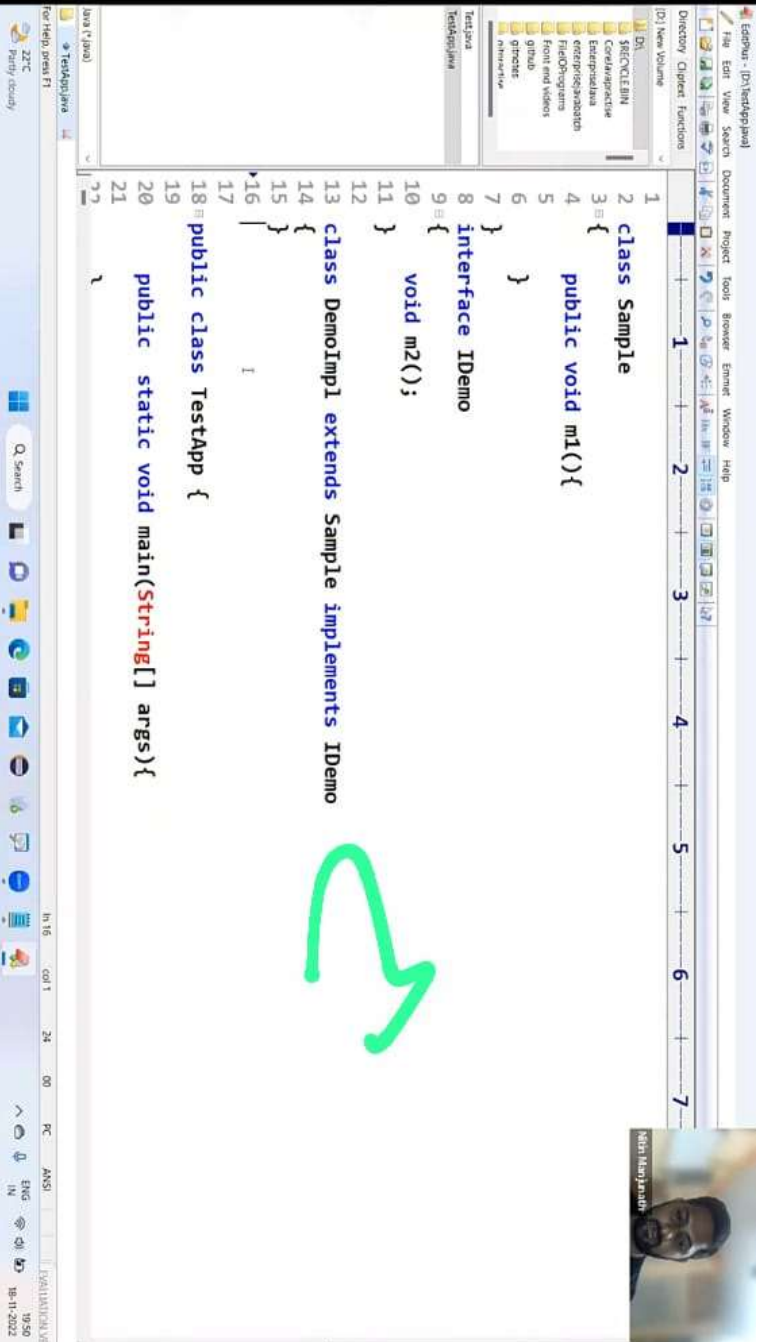
class can extend as well as can implement an interface

1. reusability => extends
2. implementation => interface

1

2





case2:

A class can extend a class and can implements any no of interfaces simultaneously.

eg: interface One{

public void methodOne();

}

class Two{

public void methodTwo(){}

}

class Three extends Two,implements One{

@Override

public void methodOne(){

....

}

@Override

public void methodTwo(){

..

}

}

7



Editor - (D:\testapp\java)

File Edit View Search Document Project Tools Browser Emmet Window Help

Directory: C:\Users\Nikhil\Documents\testapp\java

1 interface IDemo1
2 {
3 void m1();
4 }
5
6 interface IDemo2
7 {
8 void m2();
9 }
10
11 interface IDemo3 extends IDemo1, IDemo2
12 {
13 void m3();
14 }
15
16 public class TestApp {
17
18 public static void main(String[] args){
19
20 }
21 }
22

TestApp.java

TestApp

TestApp.java

1
22 69 PC ANSI
19:56
18-11-2022

Nikhil Rajan

Answer the following?

Which of the following is true?

- a. A class can extend any no of class at a time.
- b. An interface can extend only one interface at a time.
- c. A class can implement only one interface at a time.
- d. A class can extend a class and can implement an interface but not both simultaneously.
- e. An interface can implements any no of interfaces at a time.
- f. None of the above

answer: f

✓



f. None of the above

answer: f

Consider the expression X extends Y which of the possibility of X and Y expression is true?

1. Both x and y should be classes.
2. Both x and y should be interfaces.
3. Both x and y can be classes or can be interfaces.
4. No restriction.

Equation : X extends Y ==> true

class extends class ==> true

interface extends interface ==> true;

Answer: 3



interface extends interface => true

Answer: 3

Q>

Predict X,Y,Z

- a. X extends Y,Z?
- b. X extends Y implements Z?
- c. X implements Y,Z?
- d. X implements Y extends Z?

a. X extends Y,Z?

X => interface

Y => interface

Z => interface

b. X extends Y implements Z?

3



- a. X extends Y,Z?
- b. X extends Y implements Z?
- c. X implements Y,Z?
- d. X implements Y extends Z?

a. X extends Y,Z?
X => interface
Y => interface
Z => interface

b. X extends Y implements Z?
X => class
Y => class
Z => interface

c. X implements Y,Z?
X => class
Y => interface
Z => interface



d. X implements Y extends Z
combination is illegal

Interface Methods

=====

Every method present inside the interface is public.

Every method present inside the interface is abstract.

How many declaration are valid?

a. void methodOne();

b. public void methodOne();

c. abstract void methodOne();

d. public abstract void methodOne();

answer: All are valid

21



Every method present inside the interface is abstract.

How many declaration are valid?

- a. void methodOne();
 - b. public void methodOne();
 - c. abstract void methodOne();
 - d. public abstract void methodOne();
- answer: All are valid

public => To make the method available for every implementation class.

abstract => Implementation class is responsible for providing the implementation.

eg: jdbc api (java.sql.*)

implementation should be given by

- a. mysql
- b. oracle
- c. postgresql



- a. mysql
- b. oracle
- c. postgresql

how many access modifiers in java?
method in interface

- a. public and abstract

- a. public b. private c. protected
- d. static e. synchronized f. strictfp
- g. final h. abstract i. native

applicable to only variables
j. transient k. volatile

2

I



- a. mysql
- b. oracle
- c. postgresql

Since the methods present inside the interface is

=> public, abstract we can't use the modifiers like static, private, protected, strictfp, synchronized, native, final.

static => associated with implementation
abstract => can't have implementation

1



IS

Interface variables

=====

=> Inside the interface we can define variables.

=> Inside the interface variables define requirement level constants.

=> Every variable present inside the interface is by default public static final.

eg:: interface ISample{

int x=10;

}

public :: To make it available for implementation class Object.

static :: To access it without using implementation class Name.

final :: Implementation class can access the value without any modification.

1

2



static :: To access it without using implementation class Name.

final :: Implementation class can access the value without any modification.

Variable declaration inside interface

- a. int x=10;
- b. public int x=10;
- c. static int x=10;
- d. final int x=10;
- e. public static int x=10;
- f. public final int x=10;
- g. static final int x=10;
- h. public static final int x=10;

Answer: All are legal

5





File Edit View
16-11-2022, interface, classmate - Notepad
=> interface variables can be accessed from implementation class, but cannot modify if we try to modify it would result in compile time error.

```
eg:: interface Remote{  
    int VOLUME = 100;  
}  
class Lg implements Remote{  
    public static void main(String... args){  
        VOLUME=0;//CE:: cannot assign a value to final variable VOLUME  
        System.out.println("value of volume is :: "+VOLUME);  
    }  
}  
eg:: interface Remote{  
    int VOLUME = 100;  
}  
class Lg implements Remote{  
    public static void main(String... args){  
        int VOLUME=0;  
        System.out.println("value of volume is :: "+VOLUME);  
    }  
}
```

18



```
18.11.2022,interface,Classmate - Notepad
File Edit View
class Lg implements Remote{
    public static void main(String... args){
        VOLUME=0;//CE:: cannot assign a value to final variable VOLUME
        System.out.println("value of volume is ::"+VOLUME);
    }
}
```

```
eg:: interface Remote{
    int VOLUME = 100;
}
class Lg implements Remote{
    public static void main(String... args){
        int VOLUME=0;//local variable
        System.out.println("value of volume is ::"+VOLUME);
    }
}
```

19

Editor - (D:\TestApp\java)

File Edit View Search Document Project Tools Browser Emmet Window Help

Directory: C:\Program Files\Java\jdk-11.0.10\bin

1 interface ISample

2 {

3 int a=10; //public static final

4 }

5 public class TestApp implements ISample{

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

7

8 int a = 20;

9 System.out.println(a); // 20

10 System.out.println(ISample.a); // 10

11 }

12 }

13 }

14 }

TestApp.java

TestApp.java

Java (JDK)

TestApp.java

For Help, press F1

Q Search

Taskbar

10 43 14 00 PC ANSI

2024 10-11-2022

20

Manjiv

Case 1::

If 2 interfaces contain a method with same signature and same return type in the implementation class only one method implementation is enough.

eg::

```
interface Left{ public void methodOne();}  
interface Right{public void methodOne();}  
class Test implements Left,Right{  
    @Override  
    public void methodOne(){  
        ...  
    }  
}
```

Case2:

If 2 interfaces contain a method with same name but different arguments in the implementation class we have to provide implementation for both methods and these methods acts as a Overload methods.

eg::

Ln 188, Col 6



```
}  
}
```

Case2:

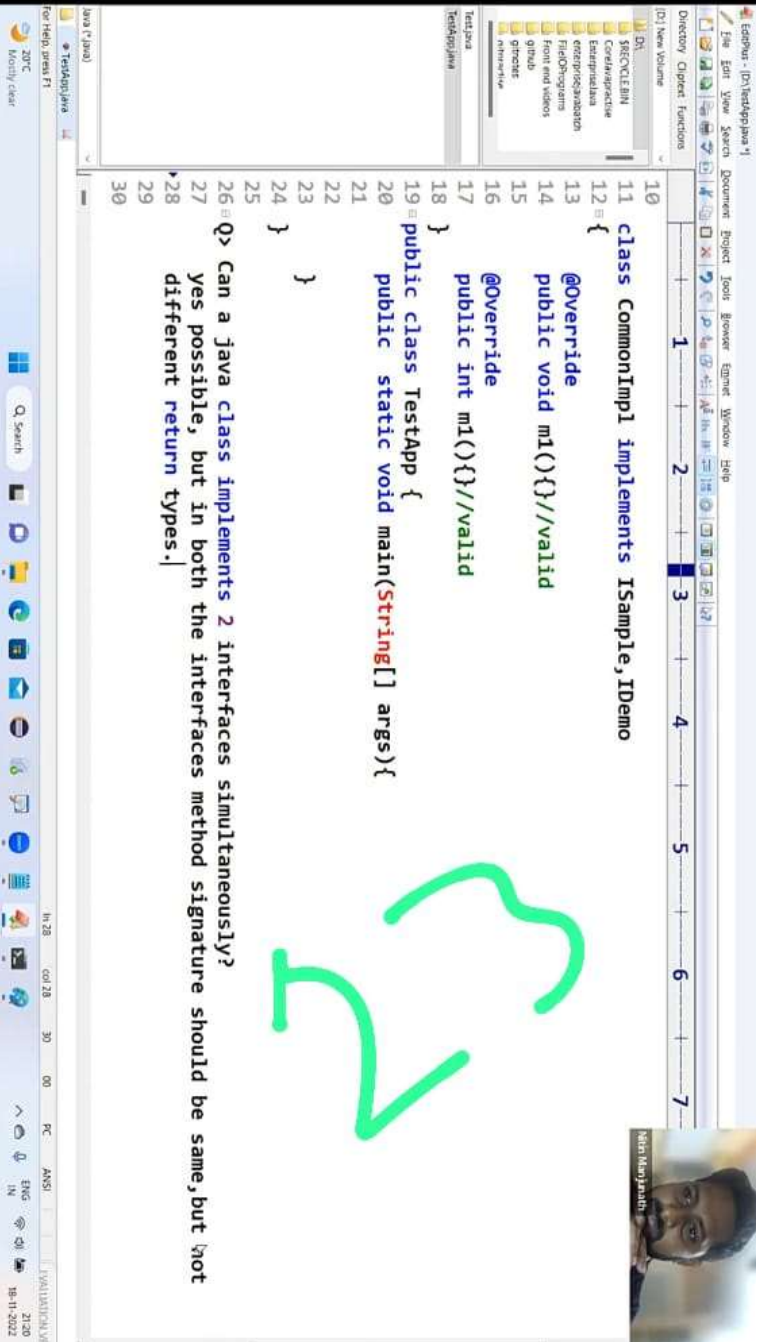
If 2 interfaces contain a method with same name but different arguments in the implementation class we have to provide implementation for both methods and these methods acts as a Overload methods.

eg::

```
interface Left{ public void methodOne();}  
interface Right{public void methodOne(int i);}  
class Test implements Left,Right{  
    @Override  
    public void methodOne(){  
        ...  
    }  
    @Override  
    public void methodOne(int i){  
        ...  
    }  
}
```

2





Editor - (D:\TestApp\java)

File Edit View Search Document Project Tools Browser Editor Window Help

Directory: C:\Program Files\Java\jdk-11.0.10\bin

1 interface ISample
2 {
3 int x = 8888;
4 }
5
6 interface IDemo
7 {
8 int x = 9999;
9 }
10
11 class TestApp implements ISample, IDemo
12 {
13 public static void main(String[] args){
14 System.out.println("ISample variable x is :: "+ISample.x);
15 System.out.println("IDemo variable x is :: "+IDemo.x);
16 }
17 }
18
19
20
21

TestApp.java

TestApp

TestApp.java

Java (.java)

TestApp.java

For Help, press F1

Q Search

ln 15 col 61 21 2E PC ANSI IN 27:29 18-11-2022

Neo Manjusha

```
}
```

Note:

Q> Can a java class implements 2 interfaces simultaneously?

Yes possible, except if two interfaces contains a method with same signature but different return types.

Variable naming conflicts::

Two variables can contain a variable with same name and there may be a chance variable naming conflicts but we can resolve variable naming conflicts by using interface names.

example 1:

```
interface Left{ int x=888;}  
interface Right{ int x=999;}  
public class Test implements Left,Right{  
    public static void main(String... args){  
        System.out.println(Left.x);  
        System.out.println(Right.x);  
    }  
}
```

57



```
D:\>javap java.lang.Runnable
Compiled from "Runnable.java"
public interface java.lang.Runnable {
    public abstract void run();
}
```

```
D:\>javap java.io.Serializable
Compiled from "Serializable.java"
public interface java.io.Serializable {
}
```

```
D:\>javap java.lang.Cloneable
Compiled from "Cloneable.java"
public interface java.lang.Cloneable {
}
```

```
D:\>
```

26



Note:

inside interface the methods are by default "public and abstract".

inside interface the variables are by default " public static and final".

We can also write an interface without any variable or abstract methods.

```
interface Serializable{
```

```
}
```

```
class SampleImpl implements Serializable{
```

```
}
```

```
interface Cloneable{
```

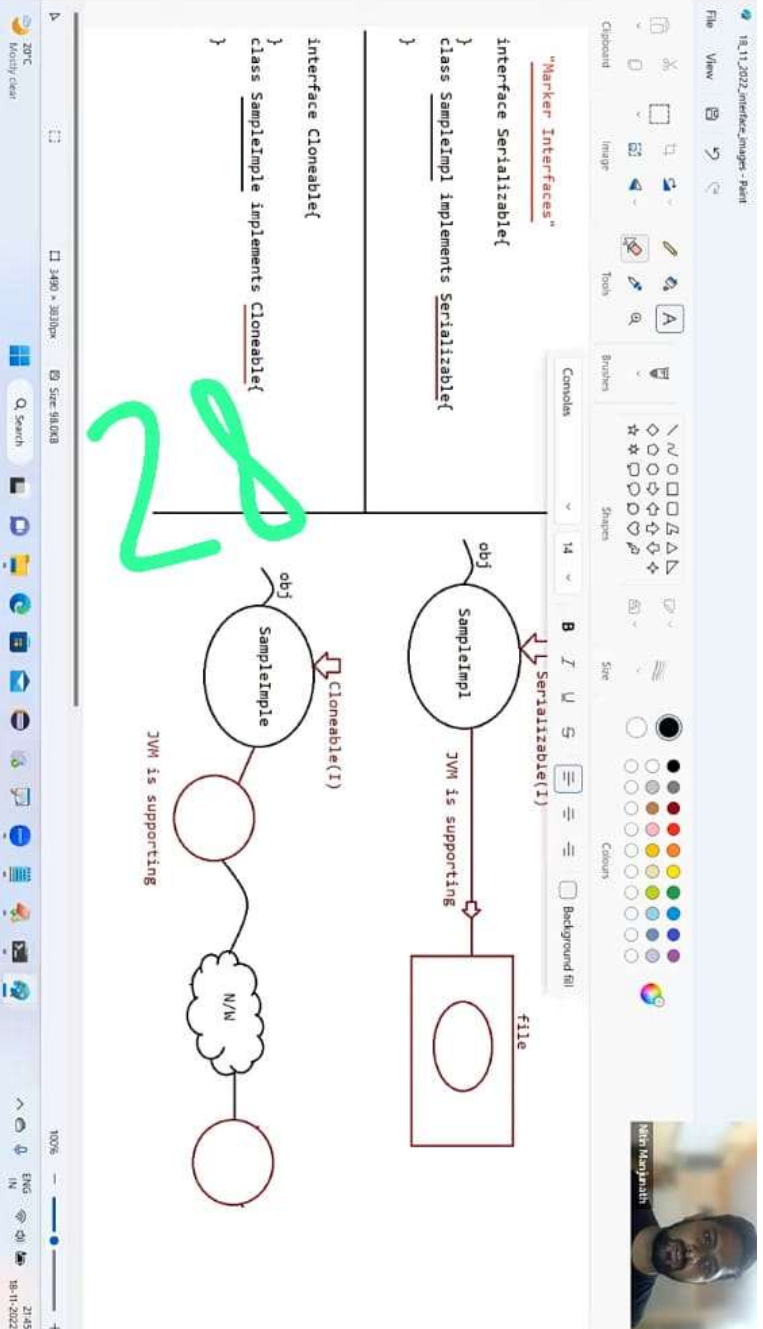
```
}
```

```
class SampleImpl implements Cloneable{
```

```
}
```

23





example1

By implementing Serializable interface we can send that object across the network and we can save state of an object into the file.

example2

By implementing SingleThreadModel interface servlet can process only one client request at a time so that we can get "Thread Safety".

example3

By implementing Cloneable Interface our object is in a position to provide exactly duplicate cloned object.

Without having any methods in marker interface how objects will get ability?

Ans.JVM is responsible to provide required ability.

Why JVM is providing the required ability to Marker Interfaces?

Ans. To reduce the complexity of the programming.

Can we create our own marker interface?

Ln:286, Col:4



```
}
```

MarkerInterface

=====

=> If an interface does not contain any methods and by implementing that interface if our Object will get some ability such type of interface are called "Marker Interface"/"Tag Interface"/"Ability Interface".

=> example

Serializable, Cloneable, SingleThreadModel.

30

example1

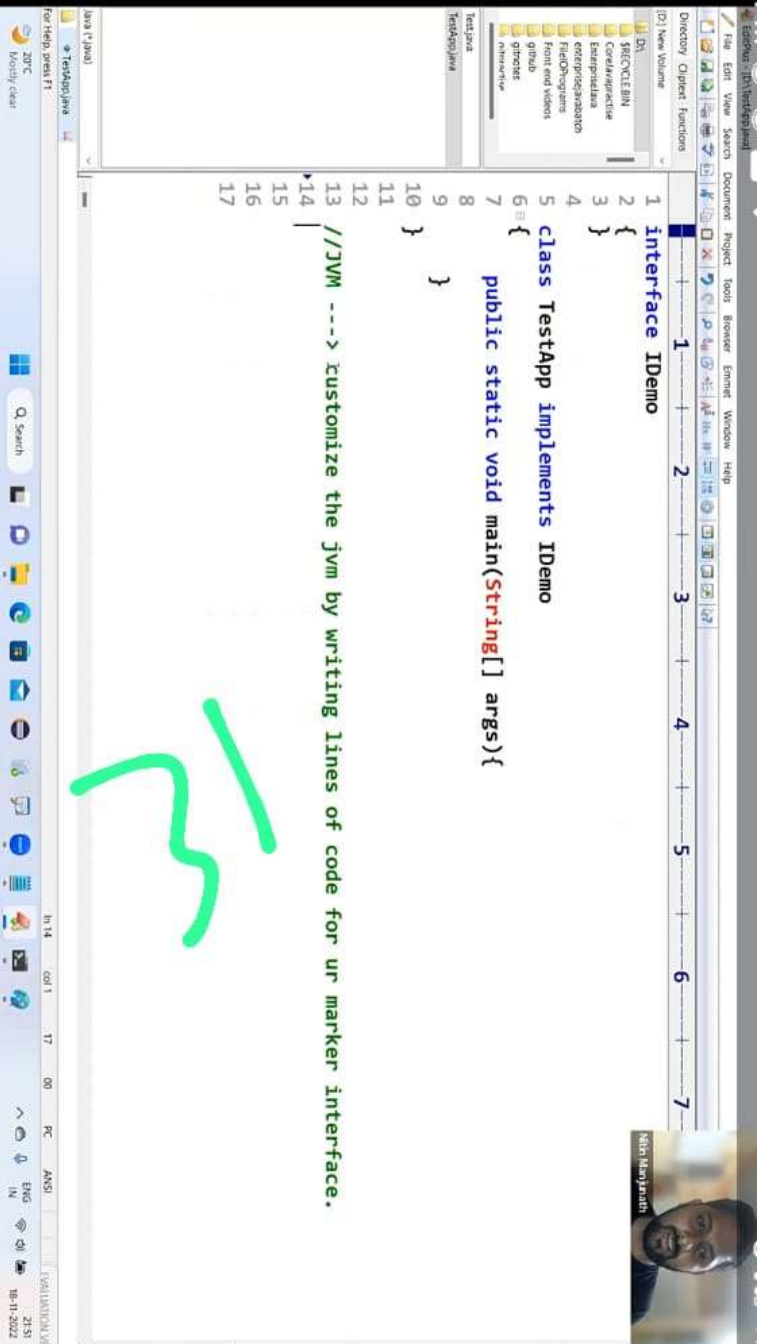
By implementing Serializable interface we can send that object across the network and we can save state of an object into the

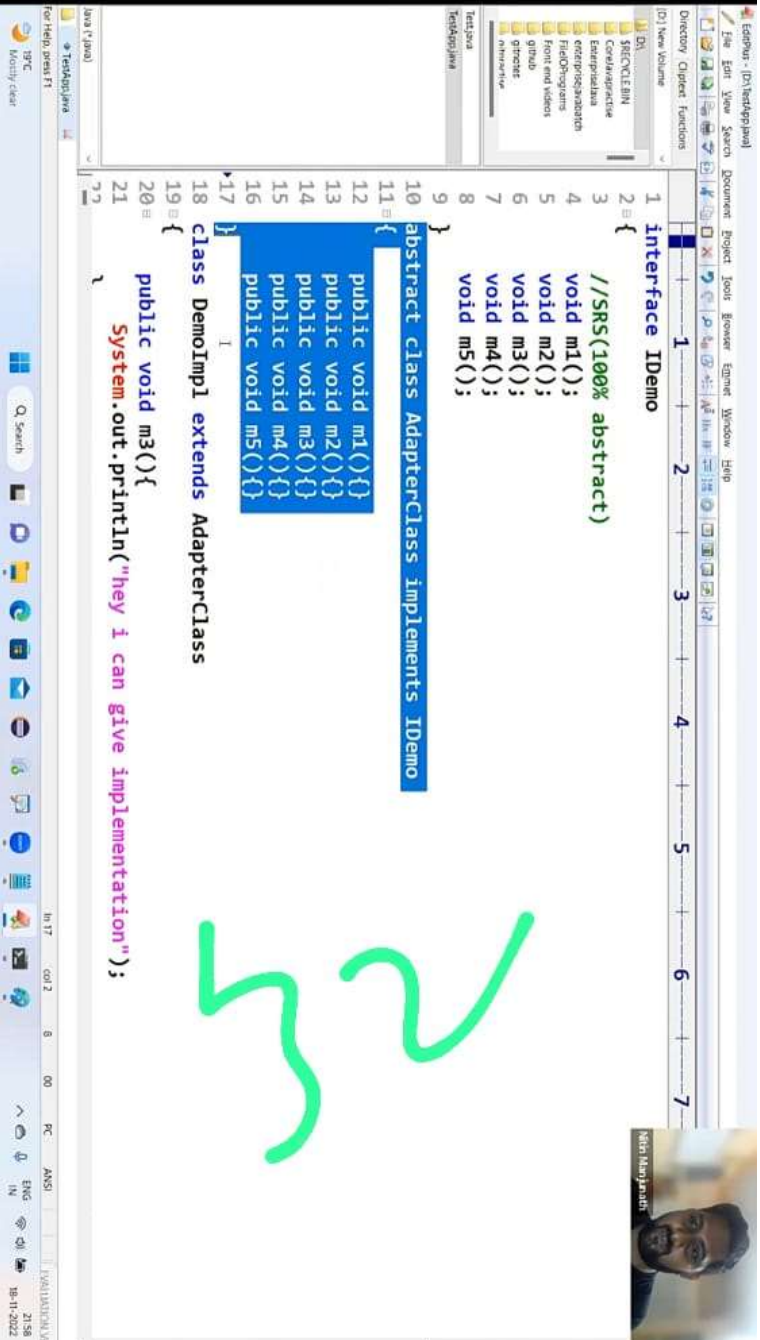
example2

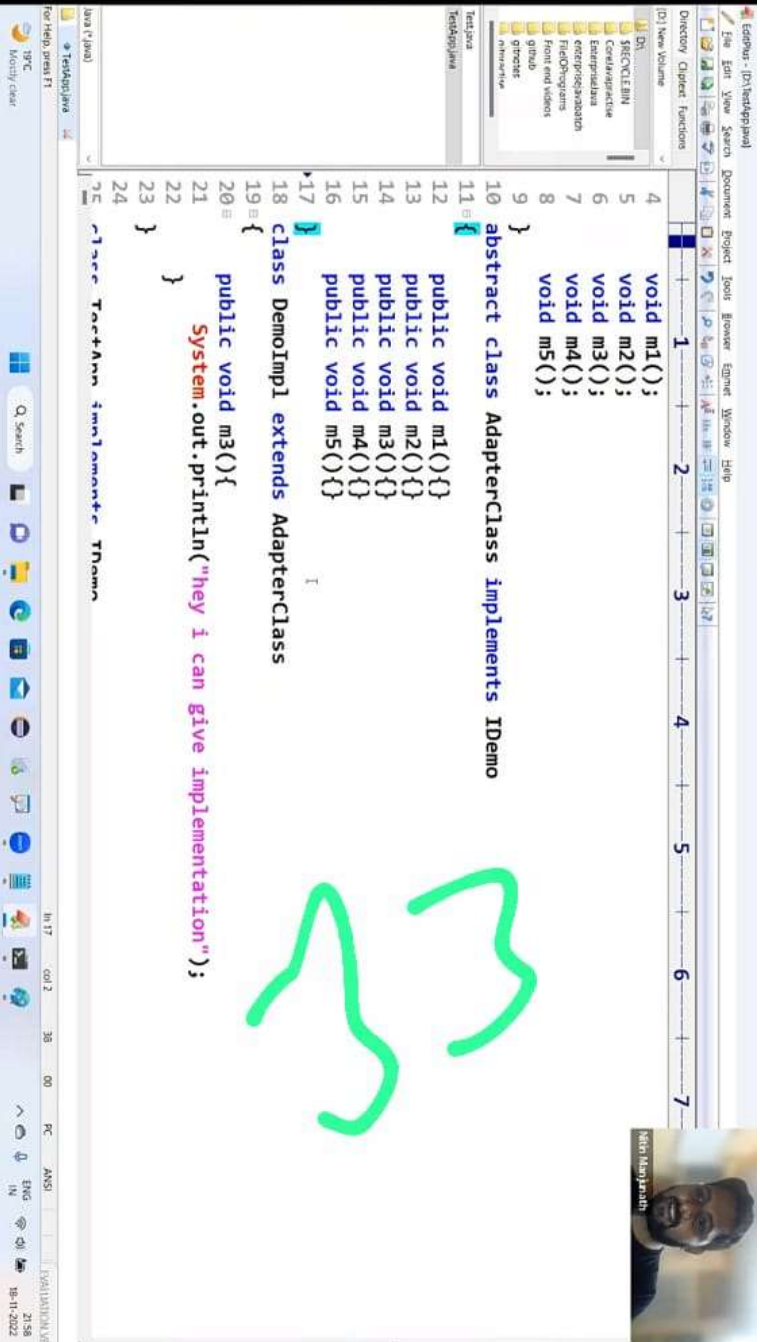
By implementing SingleThreadModel interface servlet can process only one client request at a time so that we can get "Three

examples

By implementing Cloneable interface our object is in a position to provide exactly duplicate cloned object.







codebar - IDemoApp.java

File Edit View Search Document Project Tools Browser Emmet Window Help

Directory: C:\pset1\functions

16 public void m5(){}

17 }

18 class DemoImpl extends AdapterClass

19 {

20 public void m3(){

21 system.out.println("hey i can give implementation");

22 }

23 }

24 }

25 class TestApp implements IDemo

26 {

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

28 IDemo demo = new DemoImpl();

29 demo.m3();

30 }

31 }

32 }

33 }

34 }

35 }

36 }

37 }

1 2 3 4 5 6 7

TestApp
TestApp.java

Java (*.java)


19°C
Borcity, India

Q Search

14:29 100% 2 00 PC ANSI

code 19 2 00 PC ANSI

19-11-2022



54

Can we create our own marker interface?

Yes, it is possible but we need to customize JVM. (hard for beginner)

=====

Adapter class (It is a design pattern allowed to solve the problem of direct implementation of interface)

=====

It is a simple java class that implements an interface only with empty implementation for every method.

If we implement an interface compulsorily we should give the body for all the methods whether it is required or not. This approach increases the length of the code and reduces readability.

eg:: interface X{

void m1();

void m2();

void m3();

void m4();

void m5();

}

class Test implements X{

public void m3(){

}

Ln 304, Col 2

18°C
Borjny, near

Q Search

Windows taskbar icons: File Explorer, Edge, Chrome, VS Code, etc.

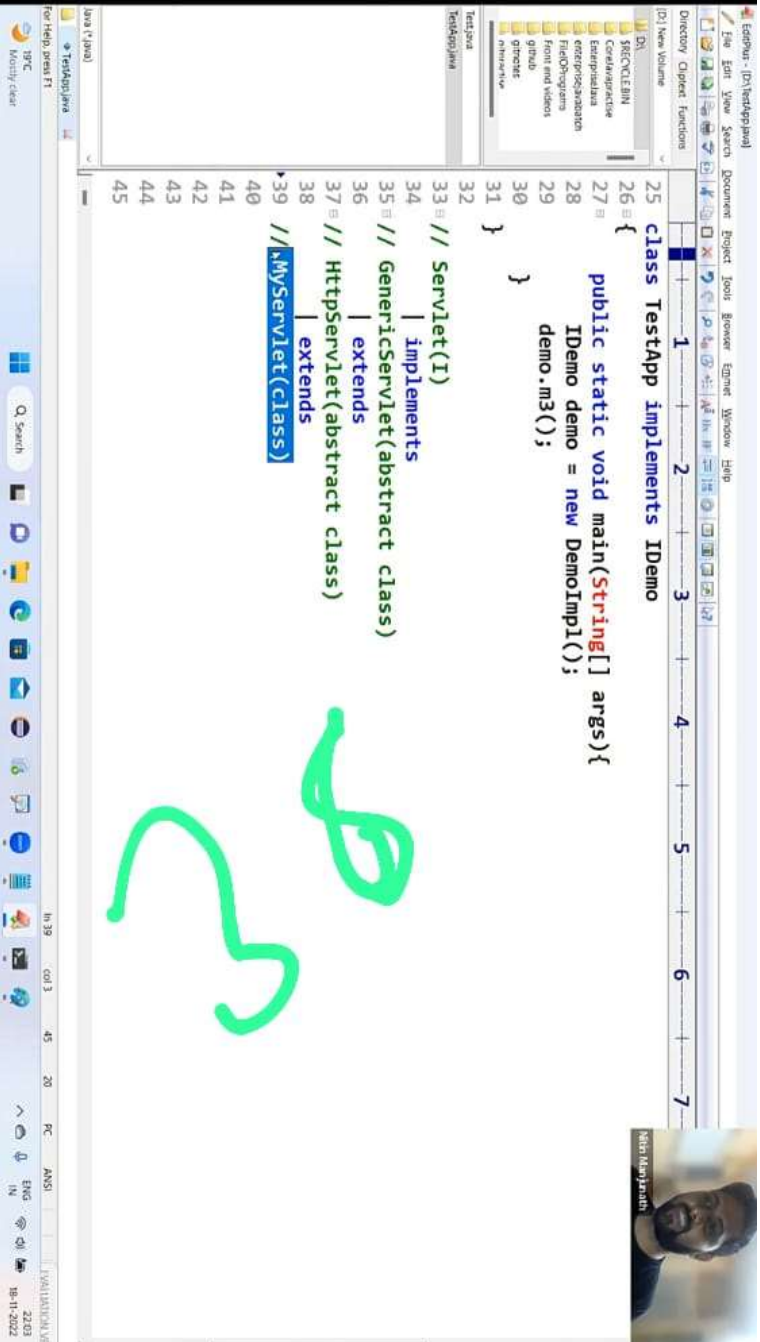
100%

Windows (CTRL)

UTF-8

ENG IN 21:59 18-11-2022





Q>

```
String s1 = "null"+null+1;  
System.out.println(s1);//nullnull1
```

Q>

```
String s1 = 1+null+"null";  
System.out.println(s1);//CE
```

Q>

```
String str = "sachin ramesh tendulkar";  
System.out.println(str.indexOf('a') + str.indexOf("dulkar"));//18
```

Q> String name = null;

```
System.out.println(name.toUpperCase());//NPE
```

3



```
Q>
String[] strings = {"Java", "JEE", "Hibernate", "Spring", "SpringBoot"};
String languages = String.join("_", strings);
System.out.println(languages);// Java_JEE_Hibernate_Spring_SpringBoot
```

```
Q>
String string = "JAVA";
StringBuffer sbuffer = new StringBuffer(string);
StringBuidler sbuilder = new StringBuidler(string);
System.out.println(string.equals(sbuffer));//false
System.out.println(string.equals(sBuidler));//false
System.out.println(sbuffer.equals(sBuidler));//false
```

38



Q>

```
public class Test{  
    Boolean b[] = new Boolean[2];  
    public static void main(String... args){  
        Test t = new Test();  
        System.out.println(t.b[0] + ":" + t.b[1]);  
    }  
}
```

- A. NullPointerException
- B. false:false
- C. true:true
- D. null:null
- E. RuntimeException other than NullPointerException

Answer: D

40

