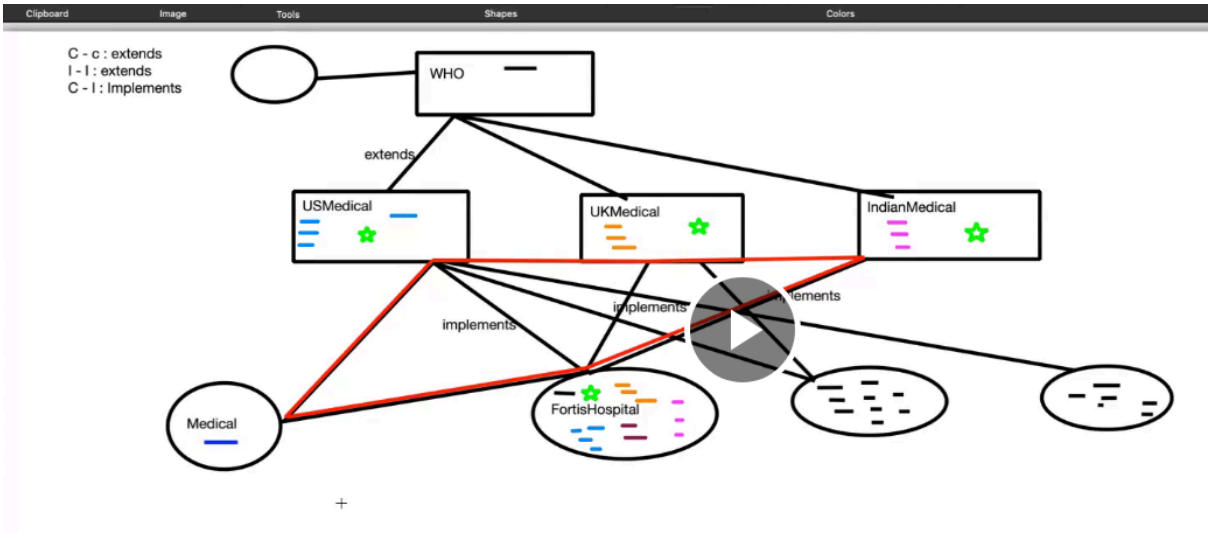
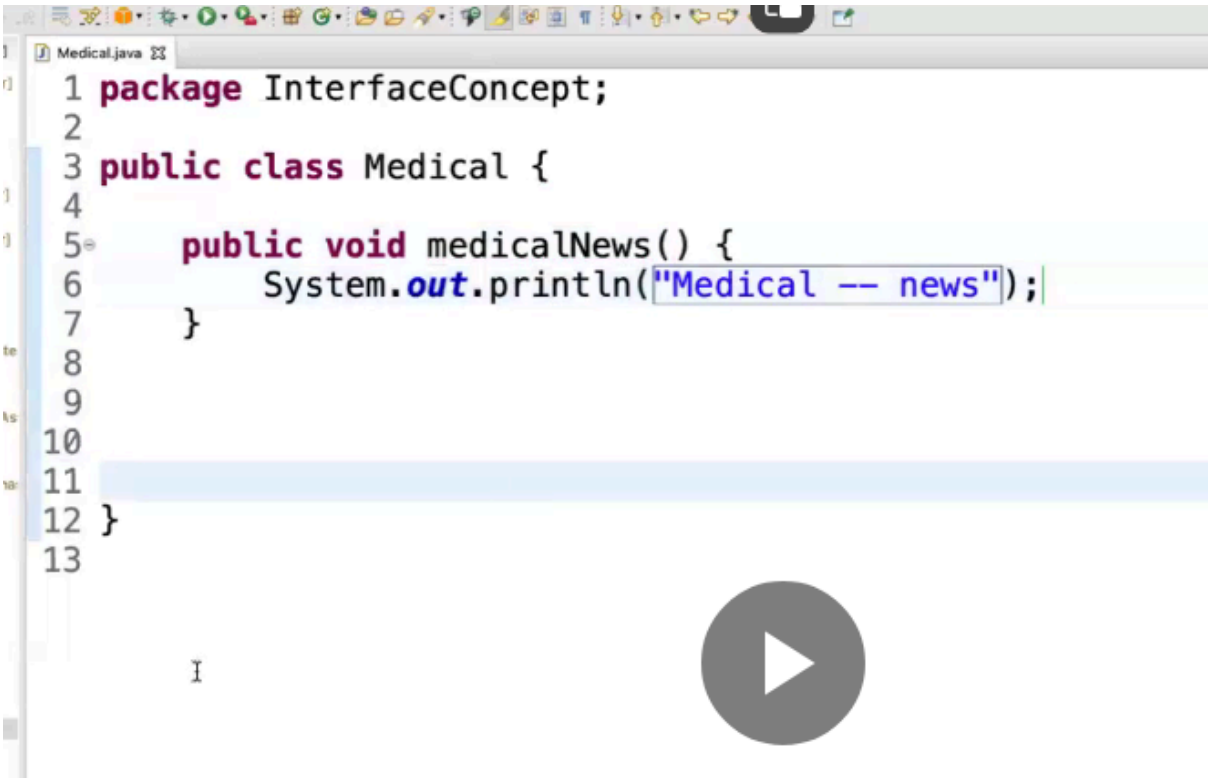
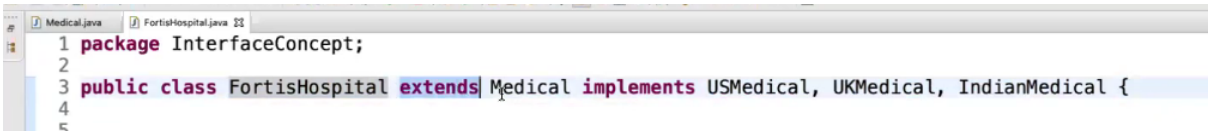
Multiple plus multi level inheritance-



Medical-

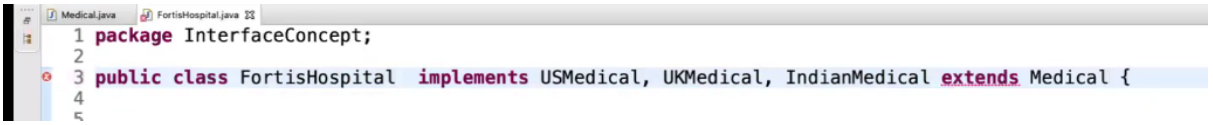


Fortis-



Change sequence-

Error. – //Syntax error on token "extends", permits expected



Test-

```
35          fh.medicalNews();
```

Medical news.

Medical-

One more method.

```
 9      public void medicalRD() {
10          System.out.println("medical R&D");
11      }
12
```

Override in fortis-

```
111     @Override
112     public void medicalRD() {
113         System.out.println("FH R&D");
114     }
115
```

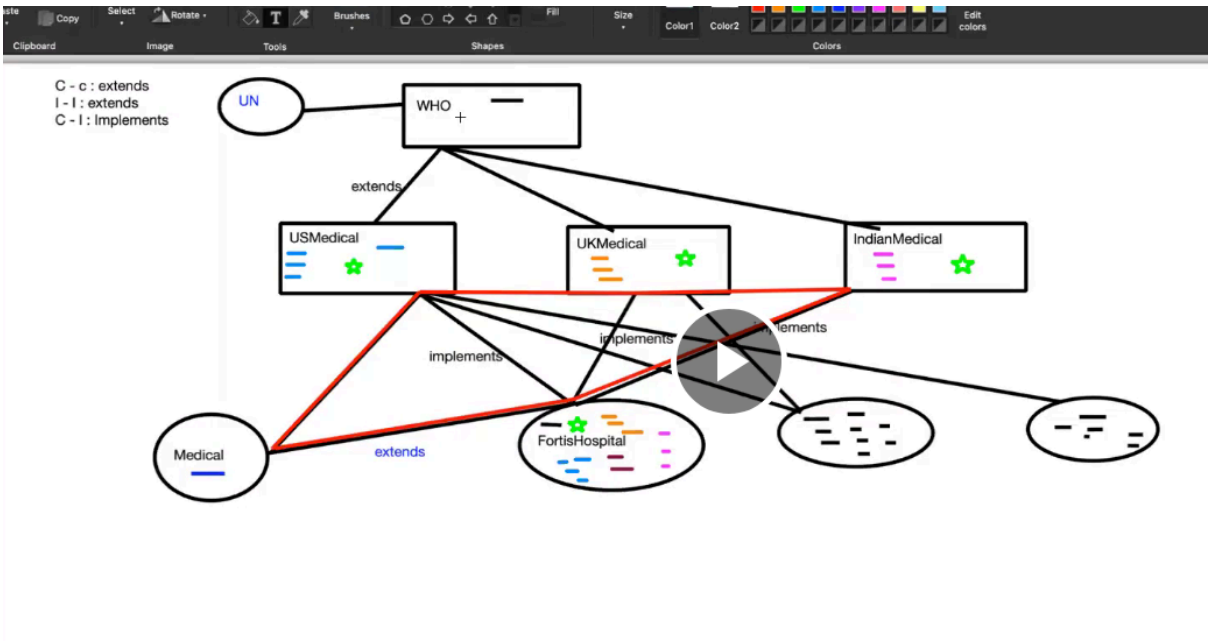
Test-

```
36          fh.medicalRD();
```

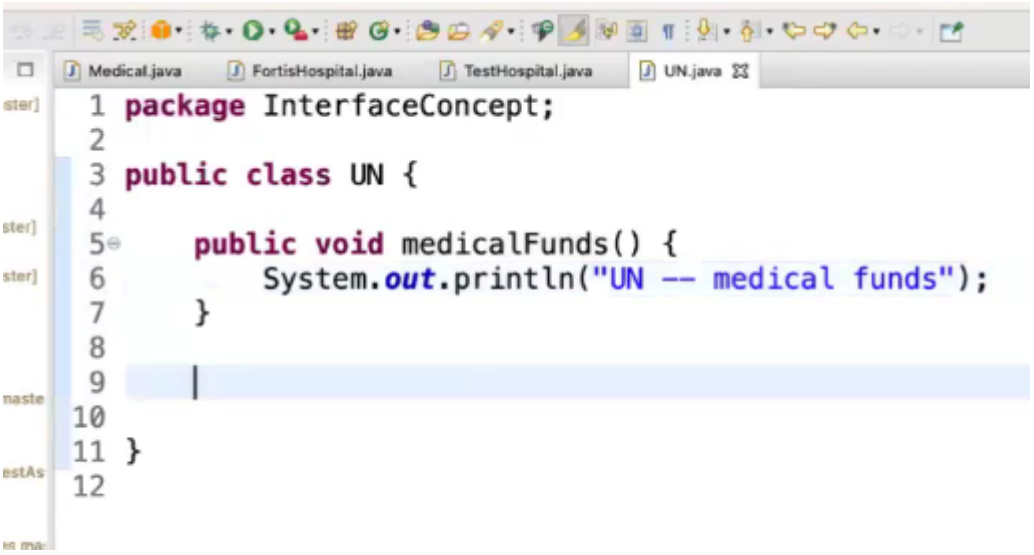
Fortis hospital medicalrd is called.

Interface parent is class-

Is it possible?



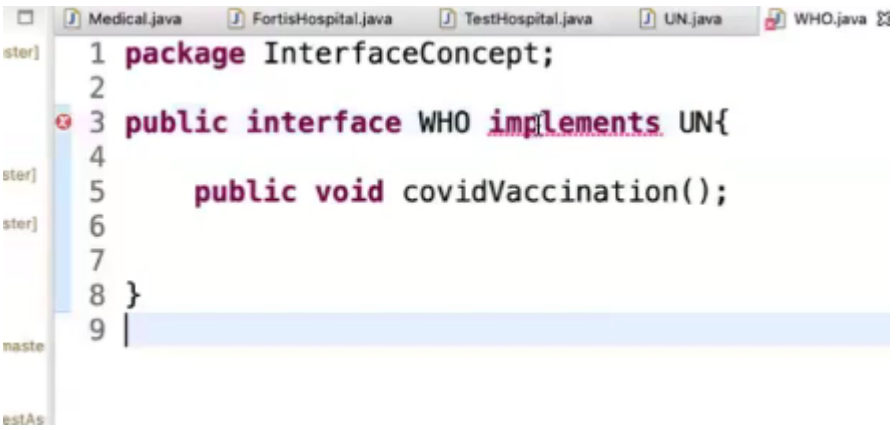
Un-



Who-

Cant implement class.

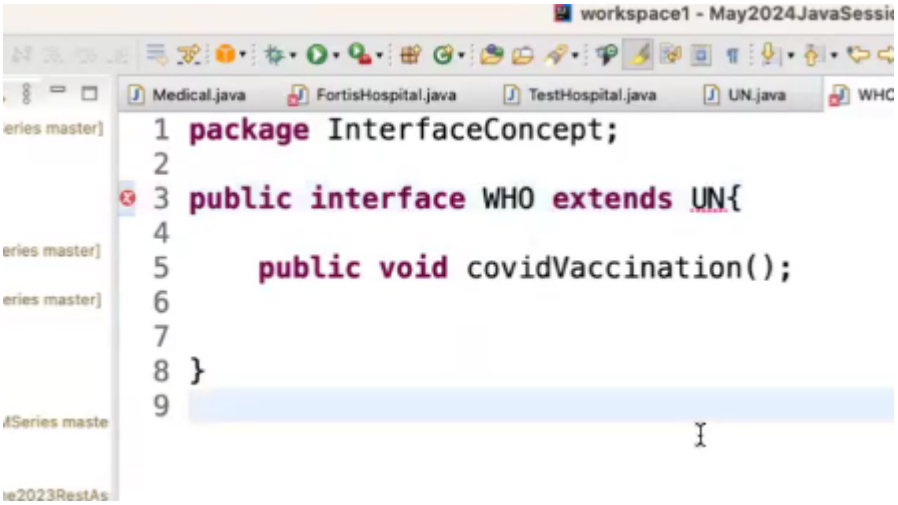
error - Syntax error on token "implements", permits expected



Extends-

Error - //The type un4 cannot be a superinterface of who4; a superinterface must be an interface

Cant do



Note-

Interface cannot have parent which is a normal class.

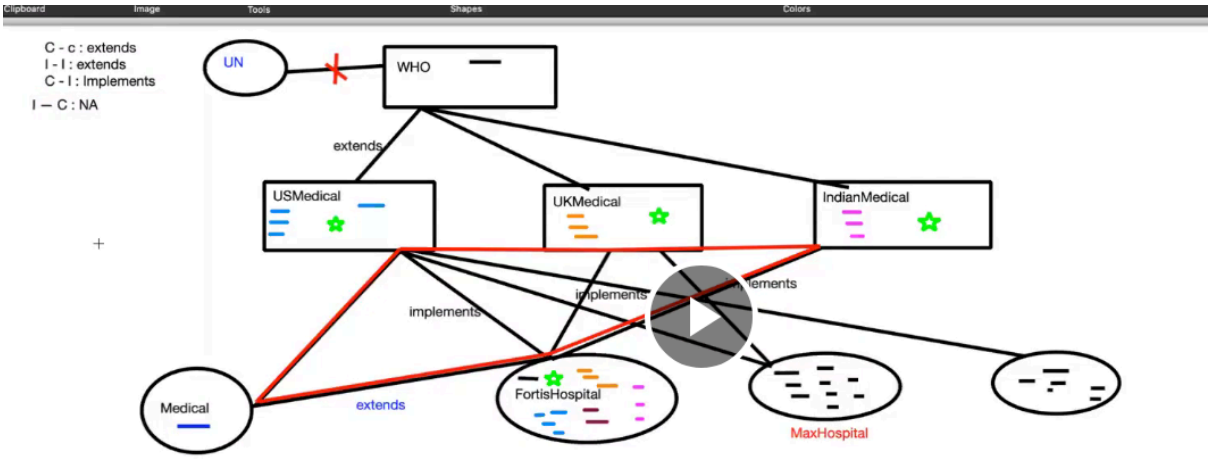
Interface can have another interface as parent.

Composition-

Has a relationship.

Sibling added.

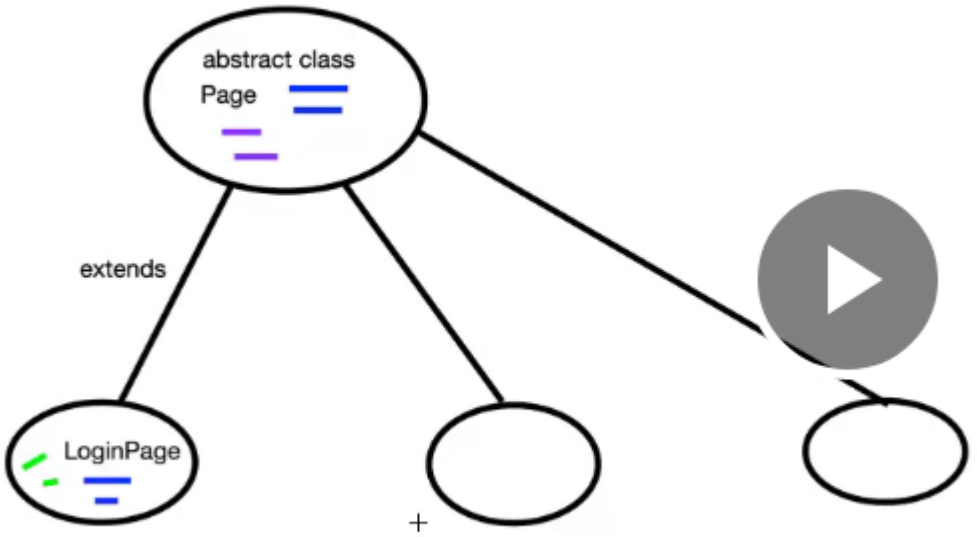
Max and fortis.




```
6   //interface vars are static and final in nature by default
7
8   //only method declaration: no method body
9   //only method prototype
10  //abstract methods: a method without body(only for non static methods)
11  //can not create the object of interface
12  //can not create the const.. of the Interface
13  //static method with body
14
15
16
```

As no object can be created of interface so no constructor.

Abstract versus interface-



In interface not mandatory to write abstract word for methods-

us medical interface-

```
14
15
16
17      public abstract void physioServices();
18
```

paste page5, login5, testfor5-

page5.java | login5.java | testfor5.java

```java
package com.day20;

public abstract class page5 {

    //class with abstract and non abstract methods.

    // Abstract methods
    public abstract void title();       // without return
    public abstract String url();       // with return

    // Non-abstract method without return
    public void loading() {
        System.out.println("Page is loading...");
    }

    // Non-abstract method with return
    public boolean unloading() {
        System.out.println("Page is unloading...");
        return true;
    }

    //create static method without return.
    public static void test() {
        System.out.println("test static without return method");
    }

    //create static method with return.
    public static String getstatic(String name) {
        System.out.println("static method with return" + name);
        return "tiger";
    }

    //create final method without return.
    public final void getfinal(String name) {
        System.out.println("final method without return" + name);
    }

    //create final method with return.
    public final String getfinal1(String name) {
        System.out.println("final method with return" + name);
        return "lion";
    }

    //create static final method without return.
    public static final void getstaticfinal(String name) {
        System.out.println("static final method without return" + name);
    }
```

```
46          System.out.println("static final method without return" + name);
47      }
48
49      //create static final method with return.
50      public static final String getstaticfinal1(String name) {
51          System.out.println("static final method with return" + name);
52          return "tiger";
53      }
54
55      //create default method in abstract with return.
56      //Default methods are allowed only in interfaces.
57  //  default String getdefault(String name) {
58  //      System.out.println("default method with return" + name);
59  //      return "tiger";
60  //  }
61  //
62  //  //create default method in abstract without return.
63  //  Default methods are allowed only in interfaces.
64  //  default void getdefault1(String name) {
65  //      System.out.println("default method without return" + name);
66  //  }
67  }
```

```
66  //  }
67  }
68
```

login5.java | testfor5.java

```
1  package com.day20;
2
3  public class login5 extends page5 {
4
5      @Override
6      public void title() {
7          System.out.println("child class login title overridden method");
8      }
9
10     @Override
11     public String url() {
12         System.out.println("child class url overridden method");
13         return "gorilla";
14     }
15
16     @Override
17      public void loading() {
18         System.out.println("child Page is loading...");
19     }
20
21     //override not with static.
22     //The method getstatic(String) of type login5 must override
23     //or implement a supertype method
24 //  @Override
25 //  public static String getstatic(String name) {
26 //      System.out.println("static method with return" + name);
27 //      return "tiger";
28 //  }
29
```

```
//      System.out.println("static method with return" + name);
//      return "tiger";
//  }

    public static String getstatic(String name) {
    System.out.println("static method with return from child" + name);
    return "chimpaanzee";
}

//  @Override
//  //cannot use override with final.
//  //Cannot override the final method from page5
//  public final void getfinal(String name) {
//      System.out.println("final method without return" + name);
//  }

    //cannot use without annotation also.
    //Cannot override the final method from page5
//  public final void getfinal(String name) {
//      System.out.println("final method without return" + name);
//  }
```

```
//      System.out.println("final method without return" + name);
//  }

//  @Override
//  //cannot use override.
//  //The method getstaticfinal1(String) of type login5 must override
//  //or implement a supertype method
//  public static final String getstaticfinal1(String name) {
//      System.out.println("static final method with return" + name);
//      return "tiger";
//  }

    //without override also final not allowed.
    //Cannot override the final method from page5
//  public static final String getstaticfinal1(String name) {
//      System.out.println("static final method with return" + name);
//      return "tiger";
//  }

}
```

testfor5.java

```java
package com.day20;

public class testfor5 {

    public static void main(String[] args) {

        login5 l1 =new login5();
        l1.title();
        String s1=l1.url();
        System.out.println(s1);
        l1.loading();
    String s2    =l1.getstatic("karan");
    //warning for static.
    //The static method getstatic(String) from the type
    //login5 should be accessed in a static way
    System.out.println(s2);
    String s3=login5.getstatic("australia");
    System.out.println(s3);
    l1.getfinal("georgia");
    String s4=l1.getfinal1("cambodia");
    System.out.println(s4);
    l1.loading();
    boolean b1=l1.unloading();
    System.out.println(b1);
    l1.getstaticfinal("gimper");
    //warning for static.
    //The static method getstaticfinal(String) from the type page5
    //should be accessed in a static way
    //The static method getstaticfinal(String) from the type page5
    //should be accessed in a static way
    login5.getstaticfinal("usa");
    String s5=l1.getstaticfinal1("japan");
    System.out.println(s5);
    //static warning.
    //The static method getstaticfinal1(String) from
    //the type page5 should be accessed in a static way
    String s6=login5.getstaticfinal1("canada");
    System.out.println(s6);
    l1.test();

    //cannot create object of abstract.
    //Cannot instantiate the type page6
//  page6 p1=new page6();

    //can access only page methods not the child classes.
    System.out.println("---------------");
    page6 p1=new login6();
    p1.loading();
    boolean b2=p1.unloading();
    System.out.println(b2);
```

```
47      boolean b2=p1.unloading();
48      System.out.println(b2);
49
50
51
52      }
53
54  }
55
56  //child class login title overridden method
57  //child class url overridden method
58  //gorilla
59  //child Page is loading...
60  //static method with return from childkaran
61  //chimpaanzee
62  //static method with return from childaustralia
63  //chimpaanzee
64  //final method without returngeorgia
65  //final method with returncambodia
66  //lion
67  //child Page is loading...
68  //Page is unloading...
```

```
67  //child Page is loading...
68  //Page is unloading...
69  //true
70  //static final method without returngimper
71  //static final method without returnusa
72  //static final method with returnjapan
73  //tiger
74  //static final method with returncanada
75  //tiger
76  //test static without return method
77  //---------------
78  //Page is loading...
79  //child class override unloading
80  //true
81
82
```

paste page6, login6, testfor6-

testfor6.java | page6.java | login6.java

```java
package com.day20;

public abstract class page6 {

    //abstract class need not have abstract methods mandatorily.

    //create non abstract method with return.
    public boolean unloading() {
        System.out.println("Page is unloading...");
        return true;
    }

    //create non abstract method without return.
    public void loading() {
        System.out.println("Page is loading...");
    }

    //The abstract method getdetails in type page6 can only set
    //a visibility modifier, one of public or protected
    //cannot have abstract and final method together.
    public abstract final void getdetails(String name);

}
```

testfor6.java | login6.java

```java
package com.day20;

public class login6 extends page6 {

    @Override
    public boolean unloading() {
        System.out.println("child class override unloading");
        return true;
    }

    //create one method with return.
    public String url() {
        System.out.println("child class url overridden method");
        return "gorilla";
    }

    //create one method without return.
    public void title() {
        System.out.println("child class login title overridden method");
    }

}
```

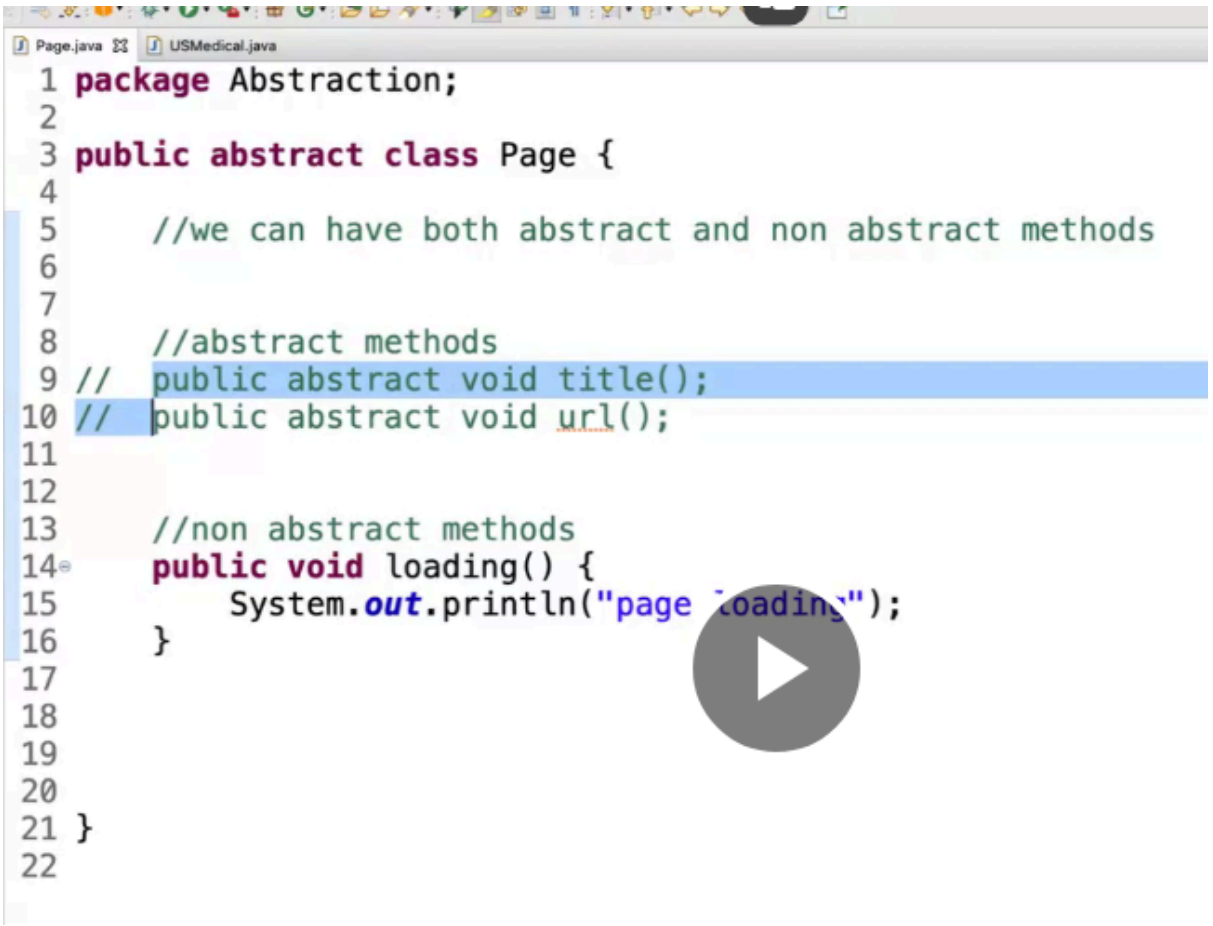
testfor6.java

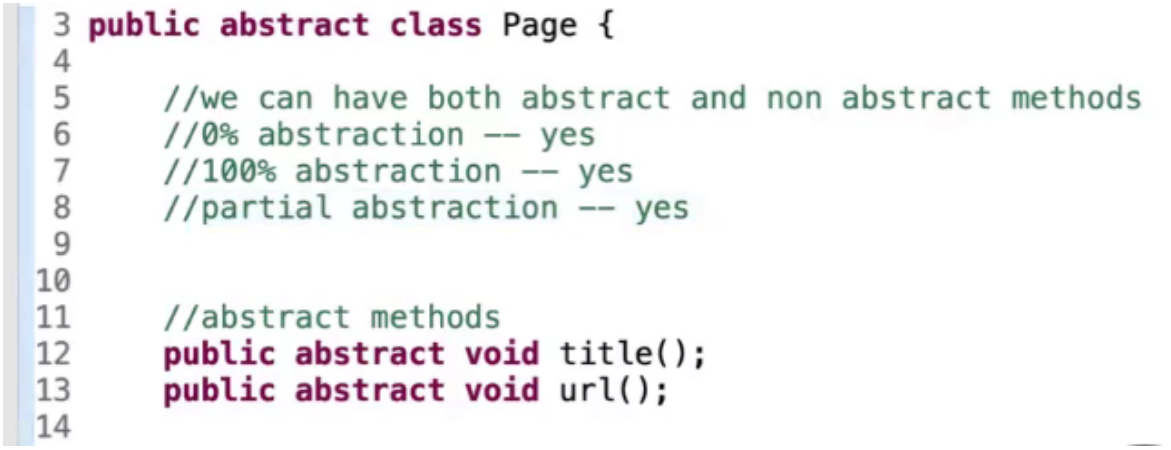
```java
package com.day20;

public class testfor6 {

    public static void main(String[] args) {

        login6 l1=new login6();
        l1.loading();
        l1.title();
        boolean b1=l1.unloading();
        System.out.println(b1);
        String s1=l1.url();
        System.out.println(s1);
    }

}

//Page is loading...
//child class login title overridden method
//child class override unloading
//true
//child class url overridden method
//gorilla
```

Abstract class-

Can have abstract and non abstract methods.

Not mandatory to write abstract methods. it means if the class is declared as abstract, but there are no abstract methods, it wont throw error.

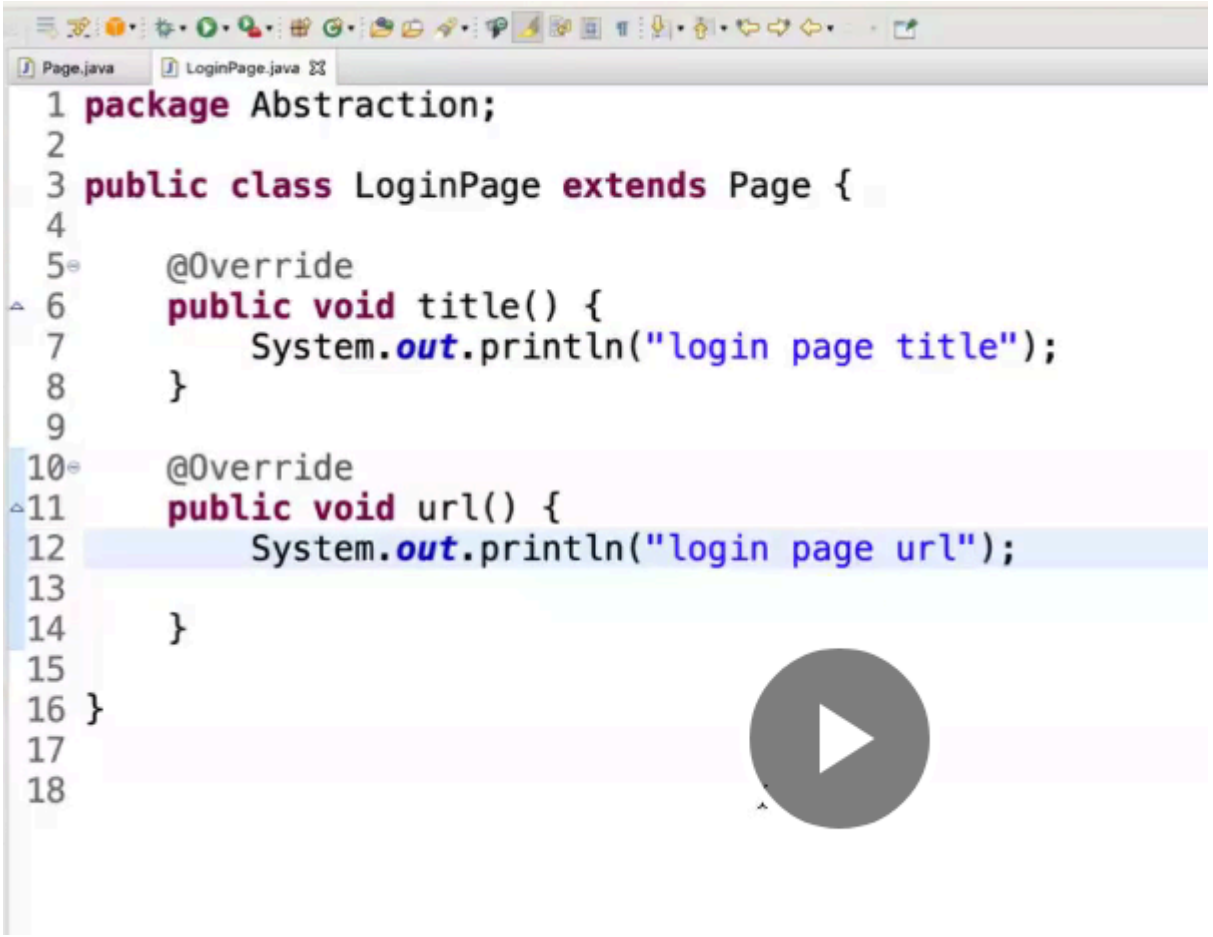




Try to create static methods in abstract class – hw. - allowed/

Try to create default methods in abstract class – hw. - not allowed. –Default methods are allowed only in interfaces.

Login page-



Abstract methods are mandatory to implement.

Abstract class-

If you want to override-

Login page

```
15
16      @Override
17      public void loading() {
18          System.out.println("Logi page loading in 5 secs");
19      }
20
21
```

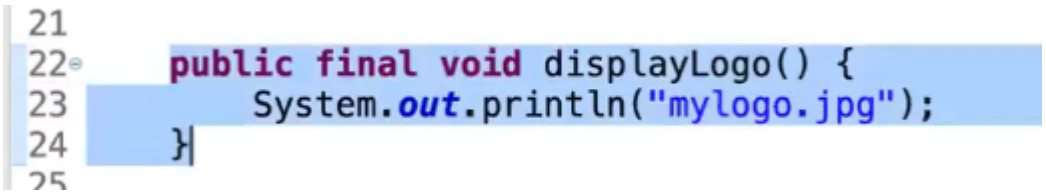
Final methods cannot be overridden-

abstract class-

```
15
16      //non abstract methods
17      public final void loading() {
18          System.out.println("page loading in 20 secs");
19      }
20
```

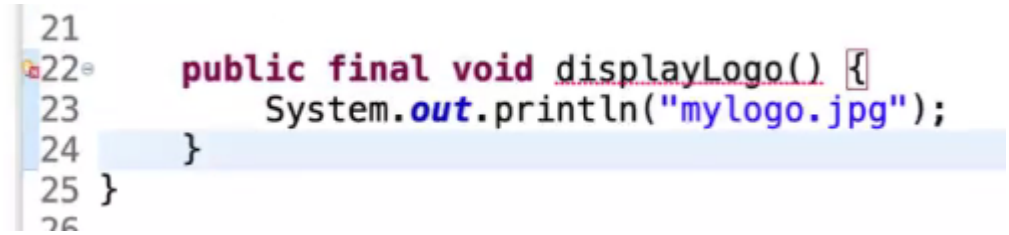
Abstract class-

Final method



```
21
22      public final void displayLogo() {
23          System.out.println("mylogo.jpg");
24      }
25
```

Login page-



```
21
22      public final void displayLogo() {
23          System.out.println("mylogo.jpg");
24      }
25  }
26
```

Error when overriding final - //Cannot override the final method from page5

Login-

Individual method

```
20
21      //individual
22      public void doLogin() {
23          System.out.println("login to app");
24      }
```

Test-

Page.java LoginPage.java TestPage.java

```
1  package Abstraction;
2
3  public class TestPage {
4
5      public static void main(String[] args) {
6
7          LoginPage lp = new LoginPage();
8          lp.title();
9          lp.url();
10         lp.loading();
11         lp.doLogin();
12
13
14
15     }
16
17 }
```

```
<terminated> TestPage (1) [Java Application] /Users/naveenautomationlabs/.p2/pool
login page title
login page url
Login page loading in 5 secs
login to app
```

```
4
5     //we can have both abstract and non abstract methods
6     //0% abstraction -- yes
7     //100% abstraction -- yes
8     //partial abstraction -- yes
9
10    //can not create the object of abstract class
11
```

```
13
14        //can not create the object of abstract class
15    //  Page p = new Page();
16
```

Top cast-

Cannot access the individual method of child.

```
17        //top casting:
18        //child class object can be referred by abstract parent class ref variable
19        Page pg = new LoginPage();
20        pg.title();
21        pg.url();
22        pg.loading();
23
24
25    }
```

```
login page title
login page url
Login page loading in 5 secs
```

Downcast-

Not allowed

paste page7, login7, testfor7-

page7.java | login7.java | testfor7.java

```java
package com.day20;

public abstract class page7 {

    //create one abstract method with return.
    public abstract boolean unloading();

    //create one abstract method without return.
    public abstract void loading();

    //create normal method with return.
    public String url() {
        System.out.println("parent class url method");
        return "gorilla";
    }

    //create normal method without return.
    public void title() {
        System.out.println("parent class login title  method");
    }

}
```

login7.java | testfor7.java

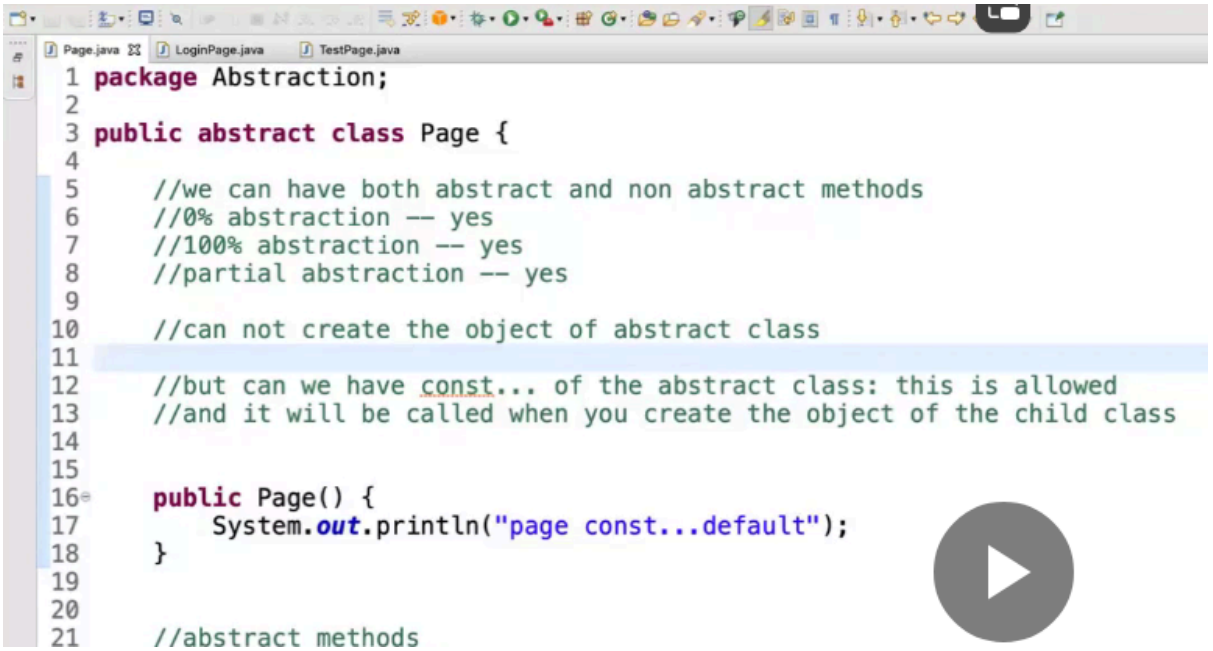
```java
package com.day20;

public class login7 extends page7 {

    @Override
    public boolean unloading() {
        System.out.println("child class override unloading");
        return false;
    }

    @Override
    public void loading() {
        System.out.println("child class override loading");
    }


}
```

testfor7.java

```java
package com.day20;

public class testfor7 {

    public static void main(String[] args) {

        login7 l1=new login7();

        l1.loading();
        l1.title();
        boolean b1=l1.unloading();
        System.out.println(b1);
        String s1=l1.url();
        System.out.println(s1);

        //downcast not allowed.
        //Cannot instantiate the type page7
        page7 p1=(login7)new page7();

    }

}
```

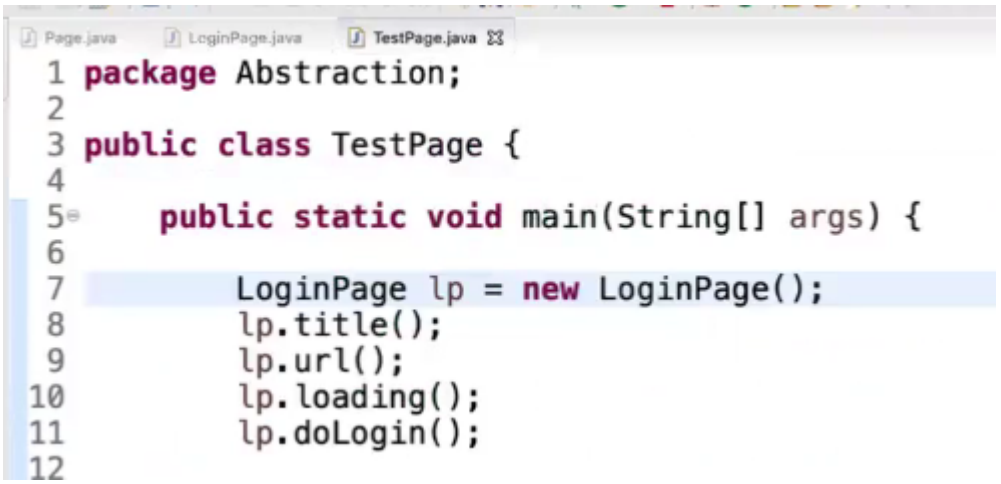
```java
}


//child class override loading
//parent class login title  method
//child class override unloading
//false
//parent class url method
//gorilla
```

Constructor allowed in abstract class-

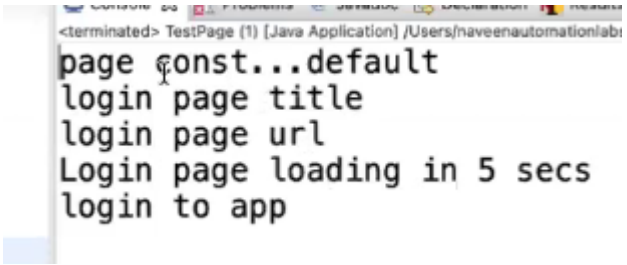


```
Page.java  LoginPage.java  TestPage.java
 1 package Abstraction;
 2
 3 public abstract class Page {
 4
 5     //we can have both abstract and non abstract methods
 6     //0% abstraction -- yes
 7     //100% abstraction -- yes
 8     //partial abstraction -- yes
 9
10     //can not create the object of abstract class
11
12     //but can we have const... of the abstract class: this is allowed
13     //and it will be called when you create the object of the child class
14
15
16     public Page() {
17         System.out.println("page const...default");
18     }
19
20
21     //abstract methods
```

test page-



```
Page.java  LoginPage.java  TestPage.java
 1 package Abstraction;
 2
 3 public class TestPage {
 4
 5     public static void main(String[] args) {
 6
 7         LoginPage lp = new LoginPage();
 8         lp.title();
 9         lp.url();
10         lp.loading();
11         lp.doLogin();
12
```



```
<terminated> TestPage (1) [Java Application] /Users/naveenautomationlabs
page const...default
login page title
login page url
Login page loading in 5 secs
login to app
```

Login page-

Same constructor as abstract class.

```java
package Abstraction;

public class LoginPage extends Page {


    public LoginPage() {
        System.out.println("LoginPage const...default");
    }


```

Test-

```java
package Abstraction;

public class TestPage {

    public static void main(String[] args) {

        LoginPage lp = new LoginPage();
        lp.title();
        lp.url();
        lp.loading();
        lp.doLogin();


```

```
page const...default
LoginPage const...default
login page title
login page url
Login page loading in 5 secs
login to app
```

First parent class constructor called then child class constructor called.

This sequence predefined in java for any inherited class.

paste page9, login9, testfor9-

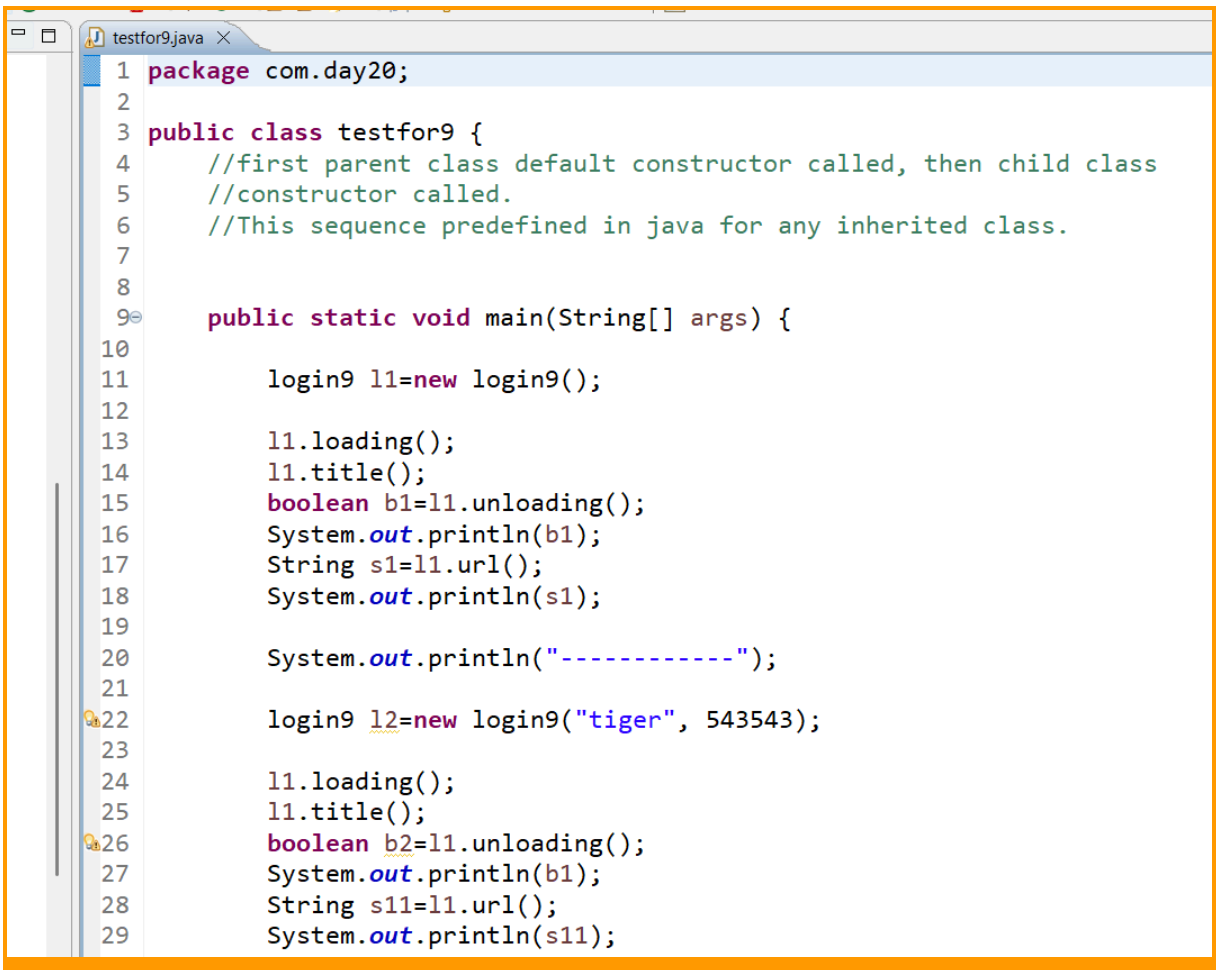
login9.java | page9.java | testfor9.java

```java
package com.day20;

public abstract class page9 {

    String name;
    int id;

    //constructor allowed in abstract class.
    public page9() {
        System.out.println("default constructor");
    }

    public page9(String name, int id) {
        this.name=name;
        this.id=id;
    }


    //create one abstract method with return.
    public abstract boolean unloading();

    //create one abstract method without return.
    public abstract void loading();

    //create normal method with return.
    public String url() {
        System.out.println("parent class url method");
        return "gorilla";
    }
        return "gorilla";
    }

    //create normal method without return.
    public void title() {
        System.out.println("parent class login title  method");
    }

}
```

login9.java testfor9.java

```java
package com.day20;

public class login9 extends page9 {

    //name and id taken from parent itself.

    //constructor in child class.
        public login9() {
            System.out.println("child class default constructor");
        }

        public login9(String name, int id) {
            this.name=name;
            this.id=id;
            System.out.println("child class parameter constructor.");
            System.out.println(this.name);
            System.out.println(this.id);
        }

    @Override
    public boolean unloading() {
        System.out.println("child class override unloading");
        return false;
    }

    @Override
    public void loading() {
        System.out.println("child class override loading");
    }
```

```java
        System.out.println("child class override loading");
    }


}

```


testfor9.java

```java
package com.day20;

public class testfor9 {
    //first parent class default constructor called, then child class
    //constructor called.
    //This sequence predefined in java for any inherited class.


    public static void main(String[] args) {

        login9 l1=new login9();

        l1.loading();
        l1.title();
        boolean b1=l1.unloading();
        System.out.println(b1);
        String s1=l1.url();
        System.out.println(s1);

        System.out.println("------------");

        login9 l2=new login9("tiger", 543543);

        l1.loading();
        l1.title();
        boolean b2=l1.unloading();
        System.out.println(b1);
        String s11=l1.url();
        System.out.println(s11);
```

```
28          String s11=l1.url();
29          System.out.println(s11);
30
31      }
32
33  }
34
35
36  //default constructor
37  //child class default constructor
38  //child class override loading
39  //parent class login title  method
40  //child class override unloading
41  //false
42  //parent class url method
43  //gorilla
44  //------------
45  //default constructor
46  //child class parameter constructor.
47  //tiger
48  //543543
49  //child class override loading
50  //parent class login title  method
51  //child class override unloading
52  //false
53  //parent class url method
54  //gorilla
```

```
53  //parent class url method
54  //gorilla
55
56
57
58
59
```

paste page10, login10, testfor10-

page10.java | login10.java | testfor10.java

```java
package com.day20;

public abstract class page10 {

    String name;
    int id;

    //only parameter constructor.
    public page10(String name, int id) {
        this.name=name;
        this.id=id;
        System.out.println(this.name);
        System.out.println(this.id);
        System.out.println("parent class param constructor.");
    }


    //create one abstract method with return.
    public abstract boolean unloading();

    //create one abstract method without return.
    public abstract void loading();

    //create normal method with return.
    public String url() {
        System.out.println("parent class url method");
        return "gorilla";
    }
```

```java
        return "gorilla";
    }

    //create normal method without return.
    public void title() {
        System.out.println("parent class login title  method");
    }

}
```

login10.java testfor10.java

```java
package com.day20;

public class login10 extends page10 {

    //name and id taken from parent itself.

    //constructor in child class.
    //Implicit super constructor page10() is undefined.
    //Must explicitly invoke another constructor
    //when parent doesnt have default constructor then child also cant have.

//      public login10() {
//          //if we add super it says there is no default page10 constructor.
//          super(); //The constructor page10() is undefined
//          System.out.println("child class default constructor");
//      }

    //have to call one of the constructor from parent to work.
    public login10() {
        super("karan", 3445);
        System.out.println("default constructor of child class.");
    }
```

```java
        System.out.println("default constructor of child class.");
    }

    //Implicit super constructor page10() is undefined.
    //Must explicitly invoke another constructor
//      public login10(String name, int id) {
//          this.name=name;
//          this.id=id;
//          System.out.println("child class parameter constructor.");
//          System.out.println(this.name);
//          System.out.println(this.id);
//      }

    //have to call one of the constructor from parent to work.
    public login10(String name, int id) {
        super(name, id);
        this.name=name;
        this.id=id;
        System.out.println("child class parameter constructor.");
        System.out.println(this.name);
        System.out.println(this.id);
    }
```

```
41          System.out.println(this.id);
42      }
43
44      @Override
45      public boolean unloading() {
46          System.out.println("child class override unloading");
47          return false;
48      }
49
50      @Override
51      public void loading() {
52          System.out.println("child class override loading");
53      }
54
55
56  }
57
```

testfor10.java

```
1  package com.day20;
2
3  public class testfor10 {
4      //first parent class default constructor called, then child class
5      //constructor called.
6      //This sequence predefined in java for any inherited class.
7
8
9      public static void main(String[] args) {
10
11         login10 l1=new login10();
12
13         l1.loading();
14         l1.title();
15         boolean b1=l1.unloading();
16         System.out.println(b1);
17         String s1=l1.url();
18         System.out.println(s1);
19
20         System.out.println("------------");
21
22         login10 l2=new login10("tiger", 543543);
23
24         l1.loading();
25         l1.title();
26         boolean b2=l1.unloading();
27         System.out.println(b1);
28         String s11=l1.url();
29         System.out.println(s11);
```

```
27              System.out.println(b1);
28              String s11=l1.url();
29              System.out.println(s11);
30
31          }
32
33  }
34
35
36  //karan
37  //3445
38  //parent class param constructor.
39  //default constructor of child class.
40  //child class override loading
41  //parent class login title  method
42  //child class override unloading
43  //false
44  //parent class url method
45  //gorilla
46  //------------
47  //tiger
48  //543543
49  //parent class param constructor.
50  //child class parameter constructor.
51  //tiger
```

```
50 //child class parameter constructor.
51 //tiger
52 //543543
53 //child class override loading
54 //parent class login title  method
55 //child class override unloading
56 //false
57 //parent class url method
58 //gorilla
59
60
61
62
63
64
```

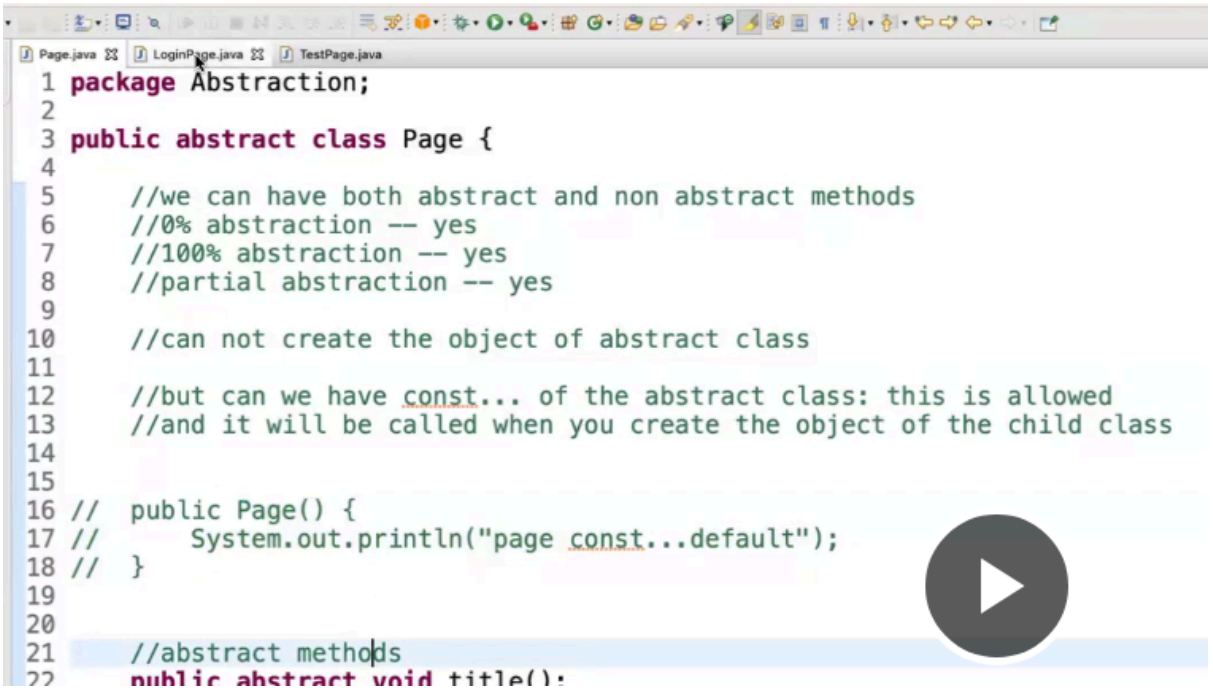
paste page11-

page11.java

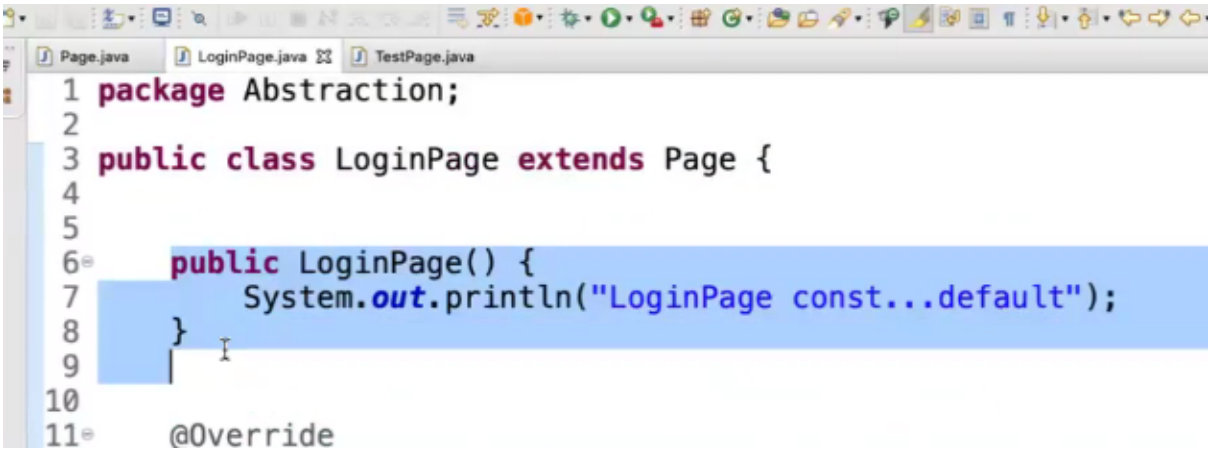
```java
package com.day20;

public abstract class page11 {

    //only parameter constructor.
    public page11(String name, int id) {
        this.name=name; //when variable not declared then error.
        //name cannot be resolved or is not a field
        this.id=id;
        //id cannot be resolved or is not a field
        System.out.println(this.name);
        System.out.println(this.id);
        System.out.println("parent class param constructor.");
    }


    //create one abstract method with return.
    public abstract boolean unloading();

    //create one abstract method without return.
    public abstract void loading();

    //create normal method with return.
    public String url() {
        System.out.println("parent class url method");
        return "gorilla";
    }
```

```java
        return "gorilla";
    }

    //create normal method without return.
    public void title() {
        System.out.println("parent class login title  method");
    }

}
```

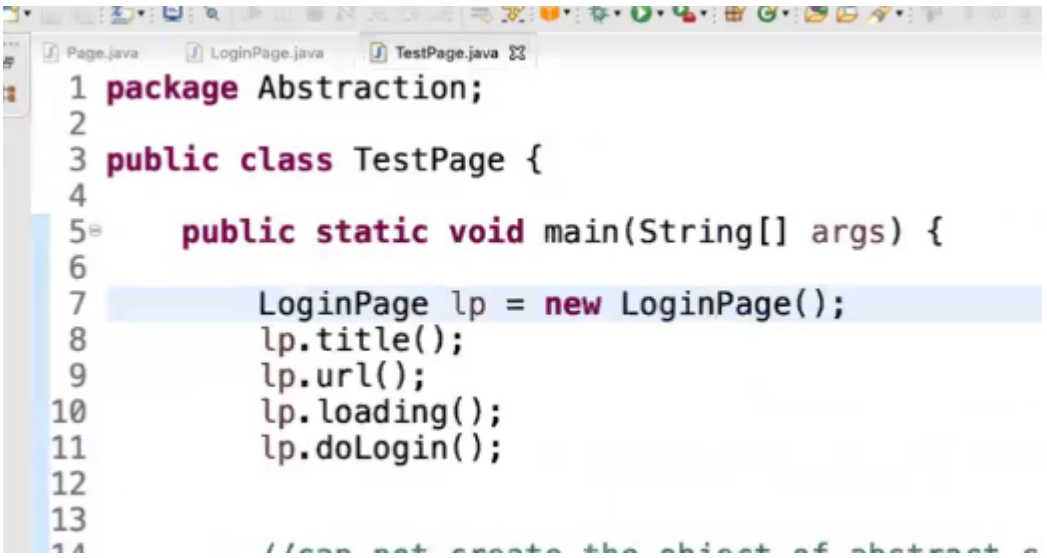
Parent class no constructor-

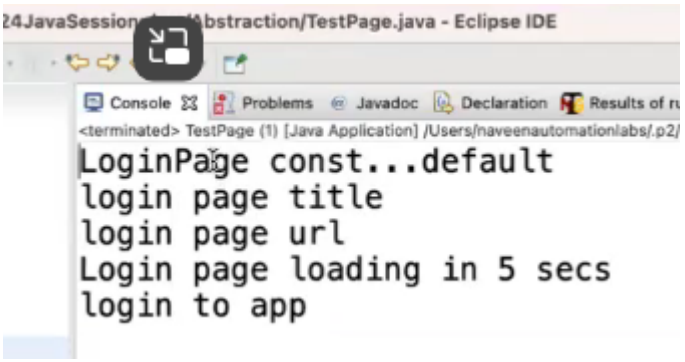


Login page with constructor-



Test-





paste page12, login12, testfor12-

page12.java

```java
package com.day20;

public abstract class page12 {

    String name;
    int id;


    //create one abstract method with return.
    public abstract boolean unloading();

    //create one abstract method without return.
    public abstract void loading();

    //create normal method with return.
    public String url() {
        System.out.println("parent class url method");
        return "gorilla";
    }

    //create normal method without return.
    public void title() {
        System.out.println("parent class login title  method");
    }

}

```

login12.java

```java
package com.day20;

public class login12 extends page12 {

    //name and id taken from parent itself.

    public login12() {
        this.name=name;
        this.id=id;
        System.out.println(this.name);
        System.out.println(this.id);
        System.out.println("default child constructor");
    }

    public login12(String name, int id) {
        this.name=name;
        this.id=id;
        System.out.println(this.name);
        System.out.println(this.id);
        System.out.println("parameter child constructor");
    }

    @Override
    public boolean unloading() {
        System.out.println("child class override unloading");
        return false;
    }
```

```java
        return false;
    }

    @Override
    public void loading() {
        System.out.println("child class override loading");
    }


}
```

login12.java testfor12.java

```java
package com.day20;

public class testfor12 {
    //first parent class default constructor called, then child class
    //constructor called.
    //This sequence predefined in java for any inherited class.

    //here parent class doesnt have any constructor so no issues.
    //everything called from child.


    public static void main(String[] args) {

        login12 l1=new login12();

        l1.loading();
        l1.title();
        boolean b1=l1.unloading();
        System.out.println(b1);
        String s1=l1.url();
        System.out.println(s1);

        System.out.println("------------");

        login12 l2=new login12("tiger", 543543);
```

```
24
25          login12 l2=new login12("tiger", 543543);
26
27          l1.loading();
28          l1.title();
29          boolean b2=l1.unloading();
30          System.out.println(b1);
31          String s11=l1.url();
32          System.out.println(s11);
33
34      }
35
36  }
```

```
35
36 }
37
38
39 //null
40 //0
41 //default child constructor
42 //child class override loading
43 //parent class login title  method
44 //child class override unloading
45 //false
46 //parent class url method
47 //gorilla
48 //------------
49 //tiger
50 //543543
51 //parameter child constructor
52 //child class override loading
53 //parent class login title  method
54 //child class override unloading
55 //false
56 //parent class url method
57 //gorilla
58
59
60
```

```
59
60
61
62
63
64
```

paste page13, login13, testfor13-

testfor13.java | page13.java | login13.java

```java
package com.day20;

public abstract class page13 {

    String name;
    int id;


    //create one abstract method with return.
    public abstract boolean unloading();

    //create one abstract method without return.
    public abstract void loading();

    //create normal method with return.
    public String url() {
        System.out.println("parent class url method");
        return "gorilla";
    }

    //create normal method without return.
    public void title() {
        System.out.println("parent class login title  method");
    }

}

```

testfor13.java | login13.java

```java
package com.day20;

public class login13 extends page12 {

    //name and id taken from parent itself.

    public login13() {
        //add super here.
        super(); // no constructor to call hence super has no effect.
        this.name=name;
        this.id=id;
        System.out.println(this.name);
        System.out.println(this.id);
        System.out.println("default child constructor");
    }

    public login13(String name, int id) {
//      super(name, id);//The constructor page12(String, int) is undefined
        //since there is no parent constructor.
        this.name=name;
        this.id=id;
        System.out.println(this.name);
        System.out.println(this.id);
        System.out.println("parameter child constructor");
    }

```

```
24          System.out.println("parameter child constructor");
25      }
26
27      @Override
28      public boolean unloading() {
29          System.out.println("child class override unloading");
30          return false;
31      }
32
33      @Override
34      public void loading() {
35          System.out.println("child class override loading");
36      }
37
38
39  }
40
```

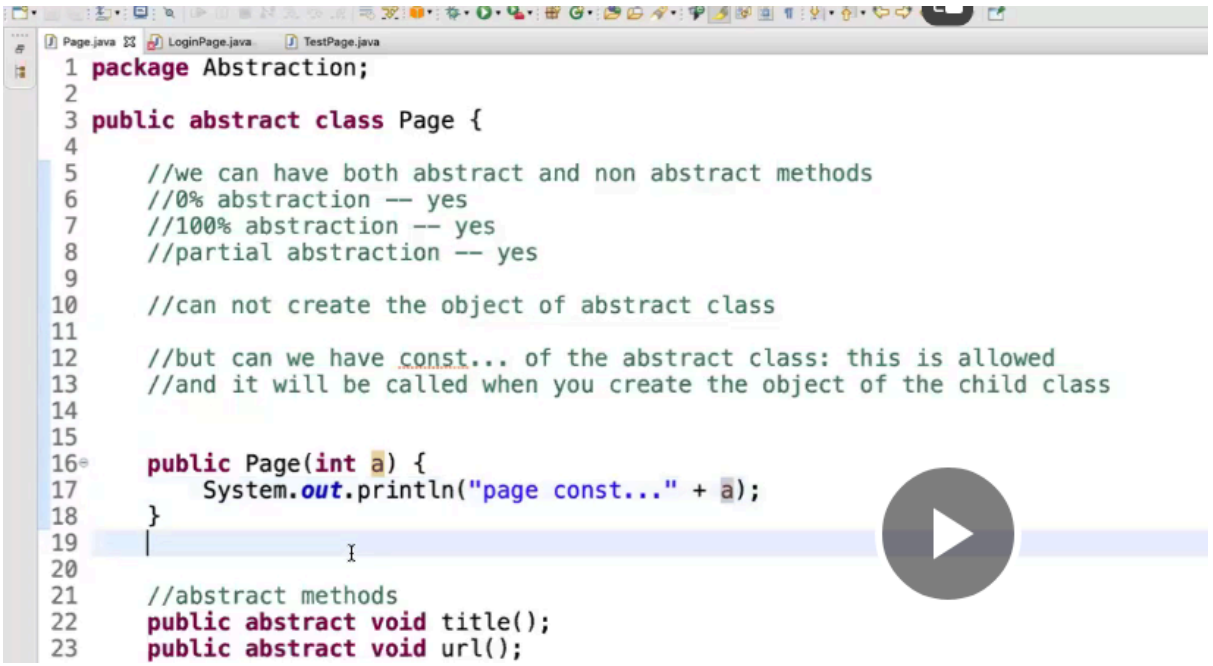
testfor13.java

```
1  package com.day20;
2
3  public class testfor13 {
4      //first parent class default constructor called, then child class
5      //constructor called.
6      //This sequence predefined in java for any inherited class.
7
8      //here parent class doesnt have any constructor so no issues.
9      //everything called from child.
10
11
12     public static void main(String[] args) {
13
14         login13 l1=new login13();
15
16         l1.loading();
17         l1.title();
18         boolean b1=l1.unloading();
19         System.out.println(b1);
20         String s1=l1.url();
21         System.out.println(s1);
22
23         System.out.println("------------");
24
25         login13 l2=new login13("tiger", 543543);
```

```
24
25          login13 l2=new login13("tiger", 543543);
26
27          l1.loading();
28          l1.title();
29          boolean b2=l1.unloading();
30          System.out.println(b1);
31          String s11=l1.url();
32          System.out.println(s11);
33
34      }
35
36  }
37
38
39  //null
40  //0
41  //default child constructor
42  //child class override loading
43  //parent class login title  method
44  //child class override unloading
45  //false
46  //parent class url method
47  //gorilla
```

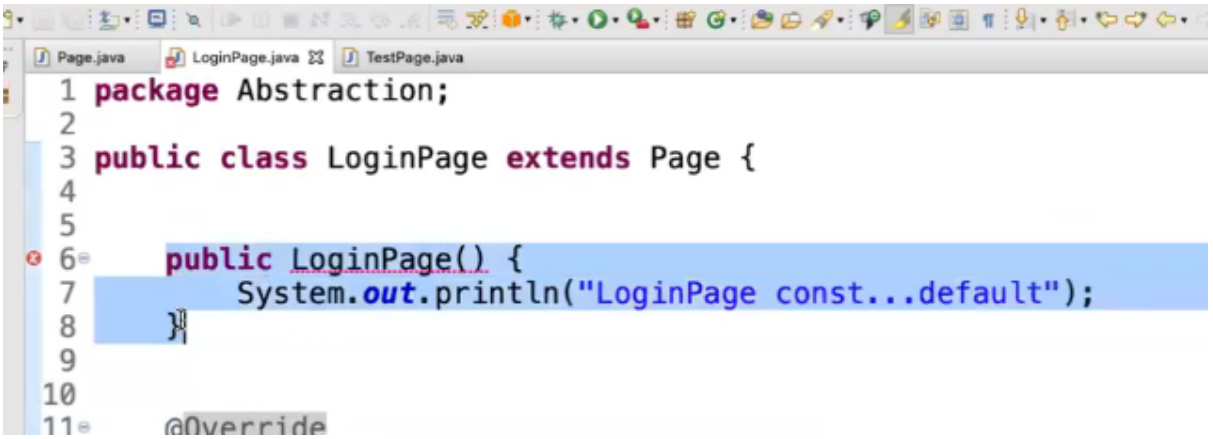
```
46 //parent class url method
47 //gorilla
48 //------------
49 //tiger
50 //543543
51 //parameter child constructor
52 //child class override loading
53 //parent class login title  method
54 //child class override unloading
55 //false
56 //parent class url method
57 //gorilla
58
59
60
61
62
63
64
65
```

Parent has parameterised constructor-



Login page gives error-/Implicit super constructor page10() is undefined.

//Must explicitly invoke another constructor



Even with parameter in child class constructor same error-/Implicit super constructor page10() is undefined.

//Must explicitly invoke another constructor

```java
package Abstraction;

public class LoginPage extends Page {


    public LoginPage() {
        System.out.println("LoginPage const...default");
    }

    public LoginPage(int a) {
        System.out.println("LoginPage const..." + a);
    }

```

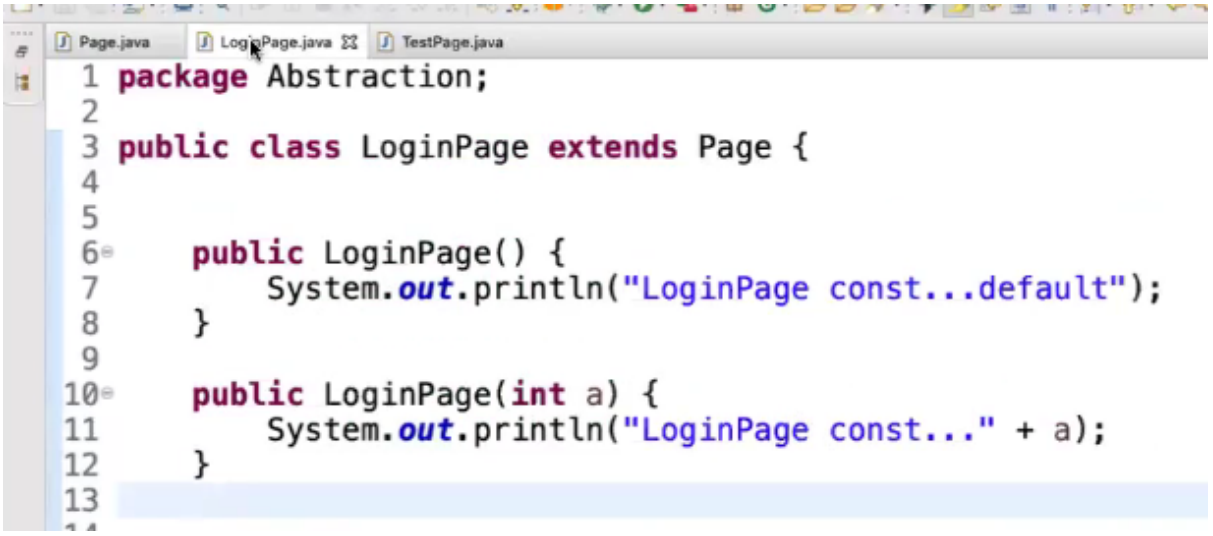
It is mandatory now to write default constructor in parent if we wrote parameterised constructor.

Add default and parameterised in parent-

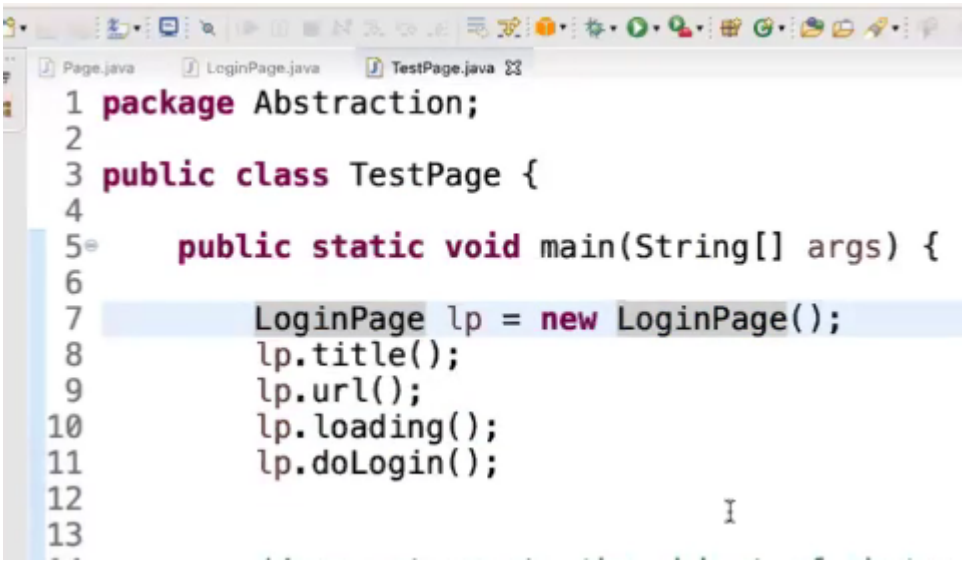
```java
package Abstraction;

public abstract class Page {

    //we can have both abstract and non abstract methods
    //0% abstraction -- yes
    //100% abstraction -- yes
    //partial abstraction -- yes

    //can not create the object of abstract class

    //but can we have const... of the abstract class: this is allowed
    //and it will be called when you create the object of the child class


    public Page() {
        System.out.println("page const..default");
    }

    public Page(int a) {
        System.out.println("page const..." + a);
    }

```

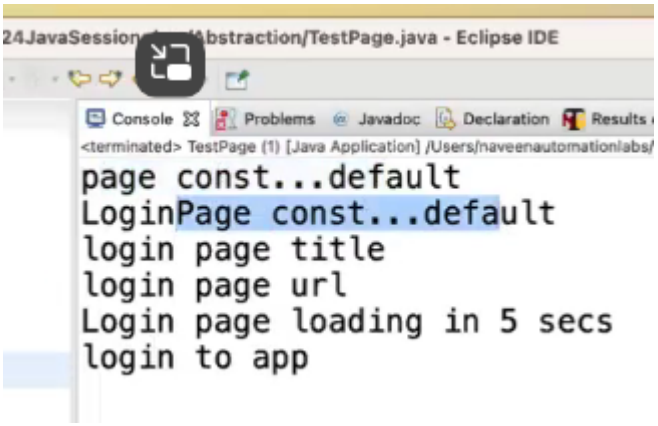
```java

    //this is mandatory if you have any parameterize const.. in parent class
    public Page() {
        System.out.println("page const...default");
    }

    public Page(int a) {
```

Login page-

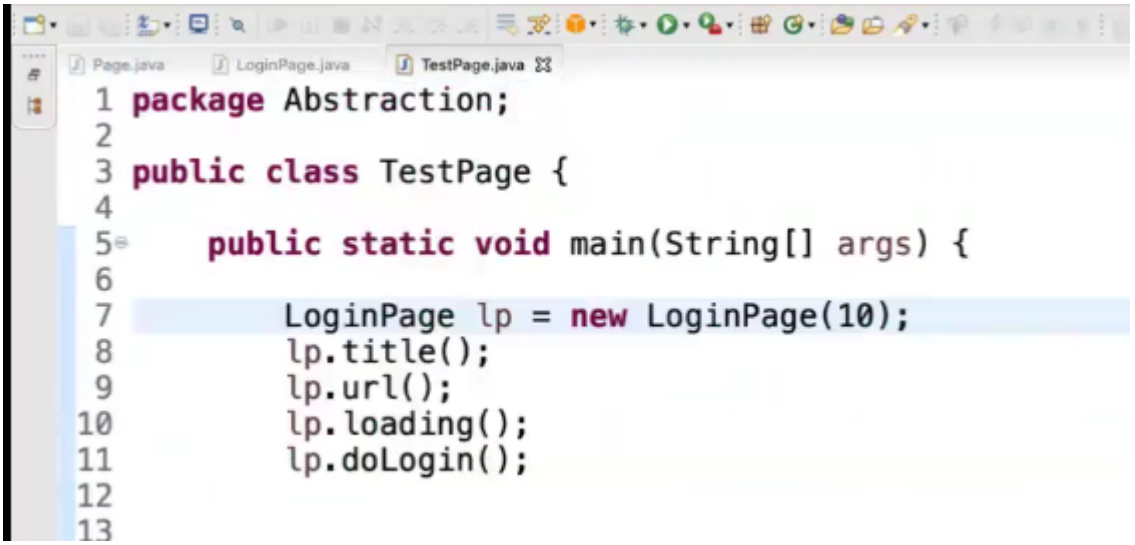


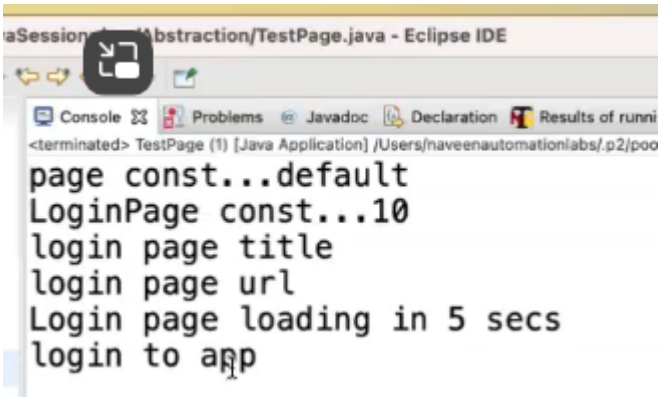
Test-





Pass param-





Default from parent called, parameterised one from child called.

Always remember parent class default constructor will be called everytime.

Default is called-

sequence

```java
package Abstraction;

public class TestPage {

    public static void main(String[] args) {

        LoginPage lp = new LoginPage();//default parent class ---> child class default
        lp.title();
        lp.url();
        lp.loading();
        lp.doLogin();

```

```
page const...default
LoginPage const...default
login page title
login page url
Login page loading in 5 secs
login to app
```

Param constructor- sequence-

```java
package Abstraction;

public class TestPage {

    public static void main(String[] args) {

        //LoginPage lp = new LoginPage();//default parent class ---> child class default
        LoginPage lp1 = new LoginPage(10);//default parent class ---> child class param const..
//      lp.title();
//      lp.url();
//      lp.loading();
//      lp.doLogin();

```

```
page const...default
LoginPage const...10
```

Page-

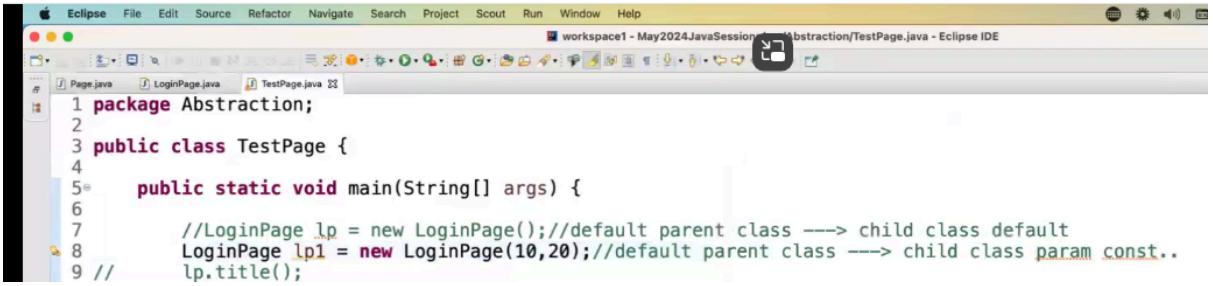
```
23
24    public Page(int a, int b) {
25        System.out.println("page const..." + a+b);
26    }
27
```

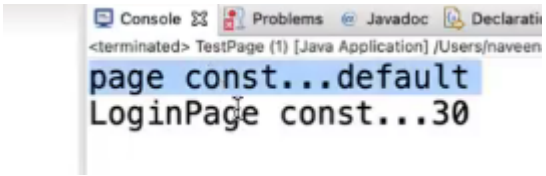
Login page-

Same in child.

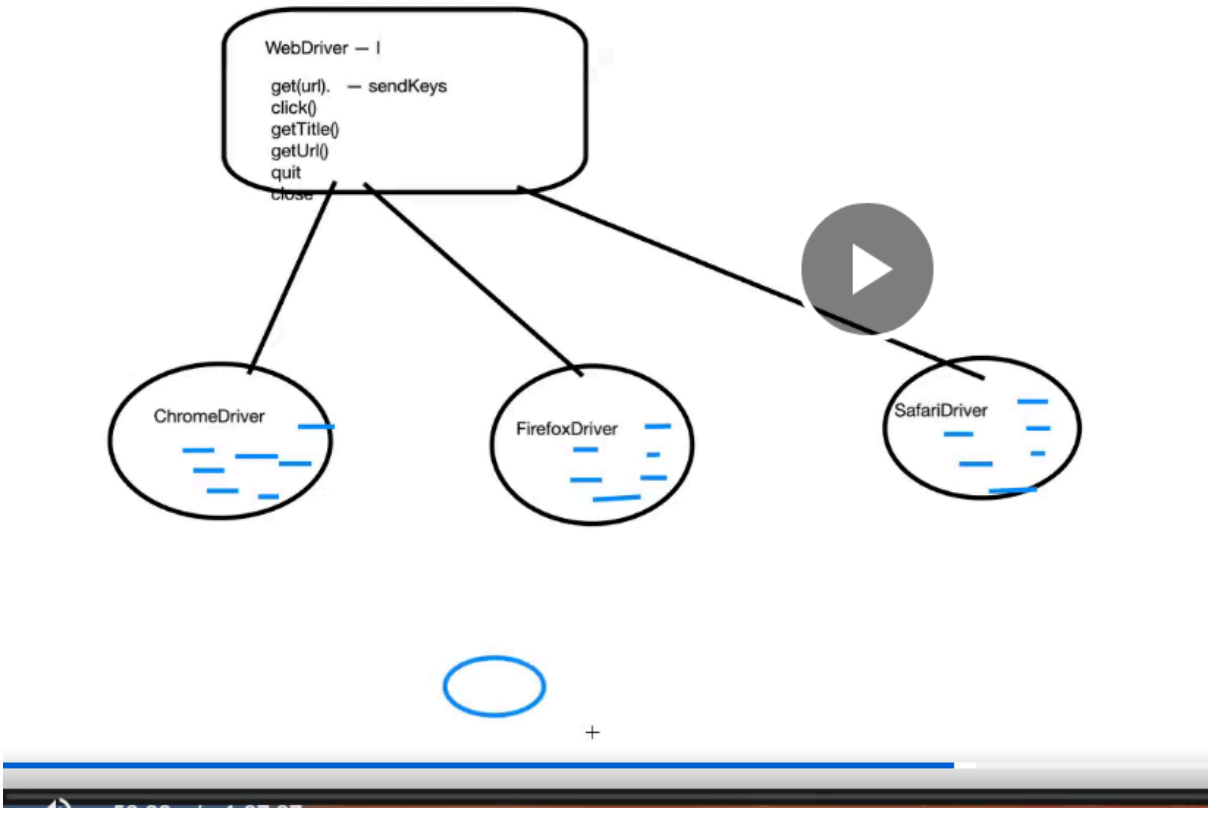
```
12    }
13
14    public LoginPage(int a, int b) {
15        System.out.println("LoginPage const..." + a+b);
16    }
17
18
19    @Override
```

Test-

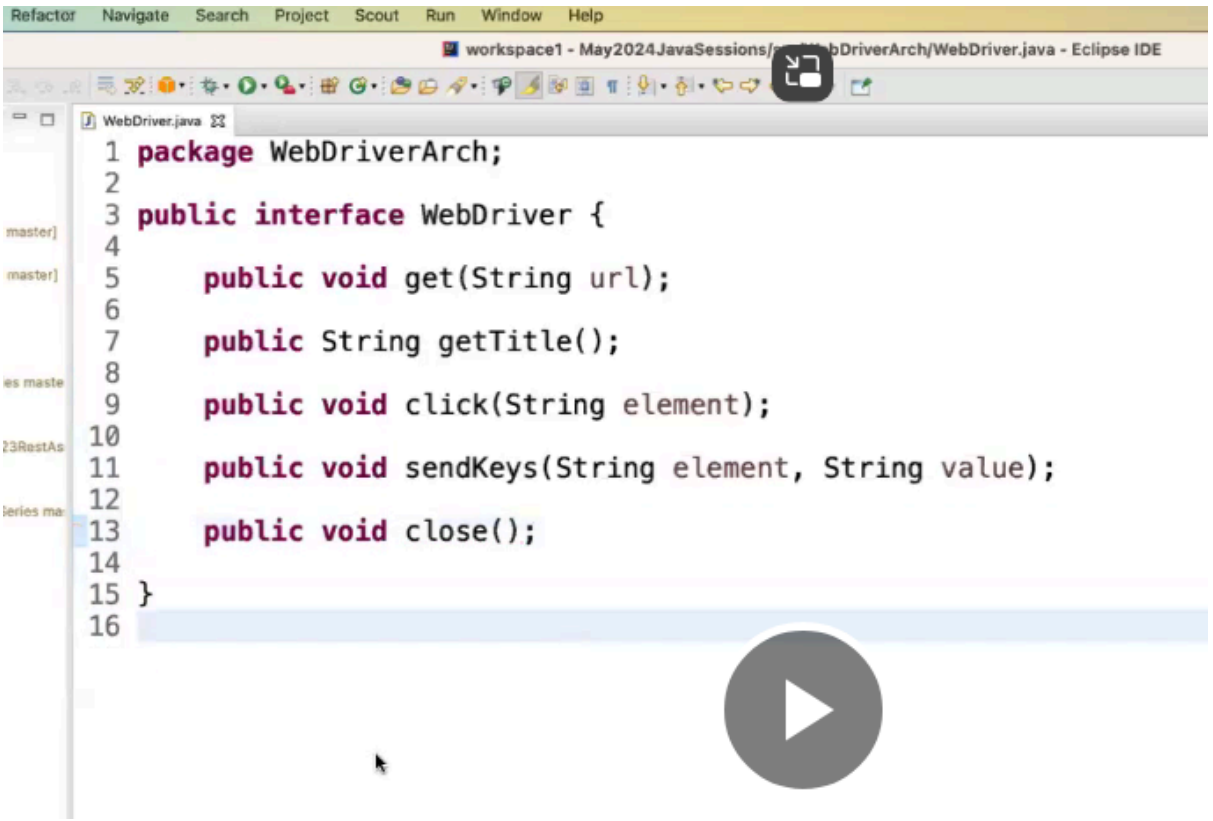




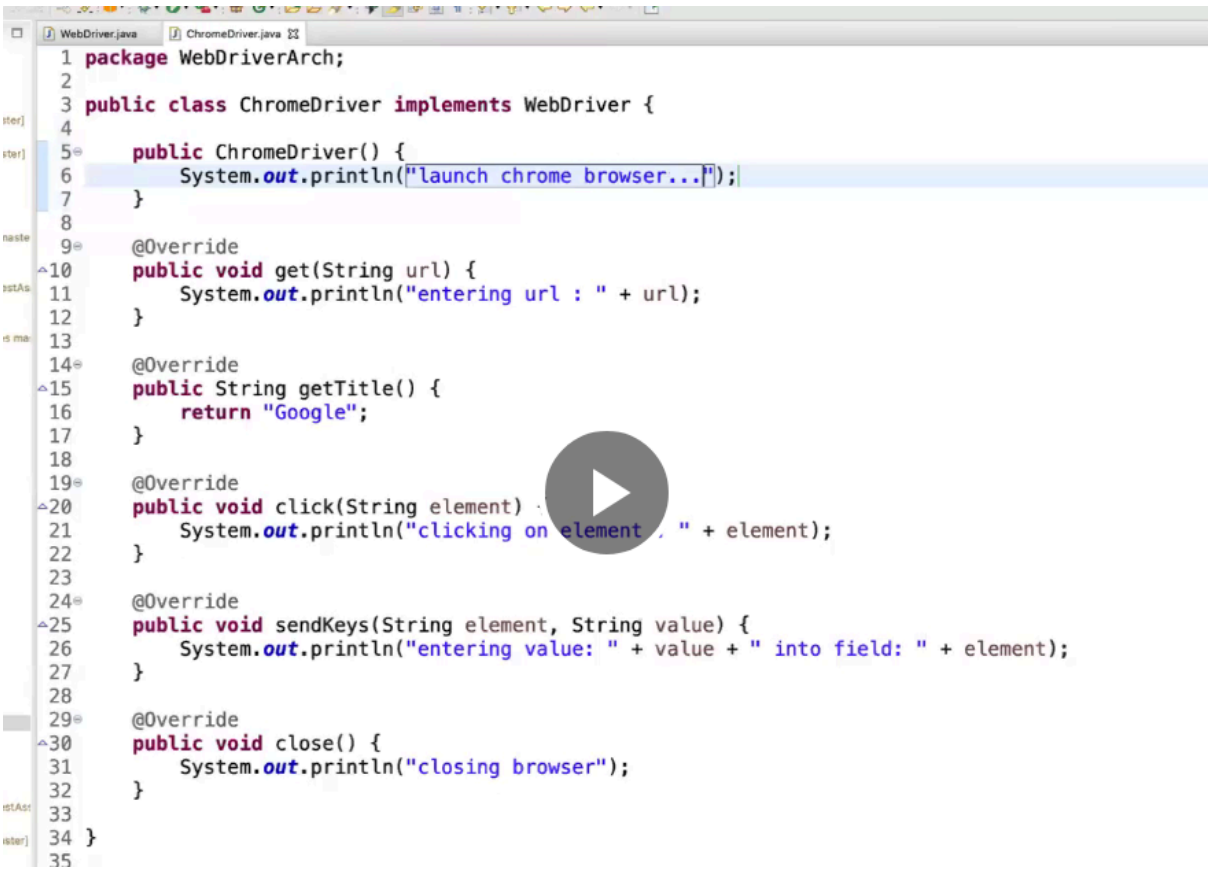
We will create this-


WebDriver — I
get(url). — sendKeys
click()
getTitle()
getUrl()
quit
ChromeDriver
FirefoxDriver
SafariDriver

Webdriver-



Chrome driver-

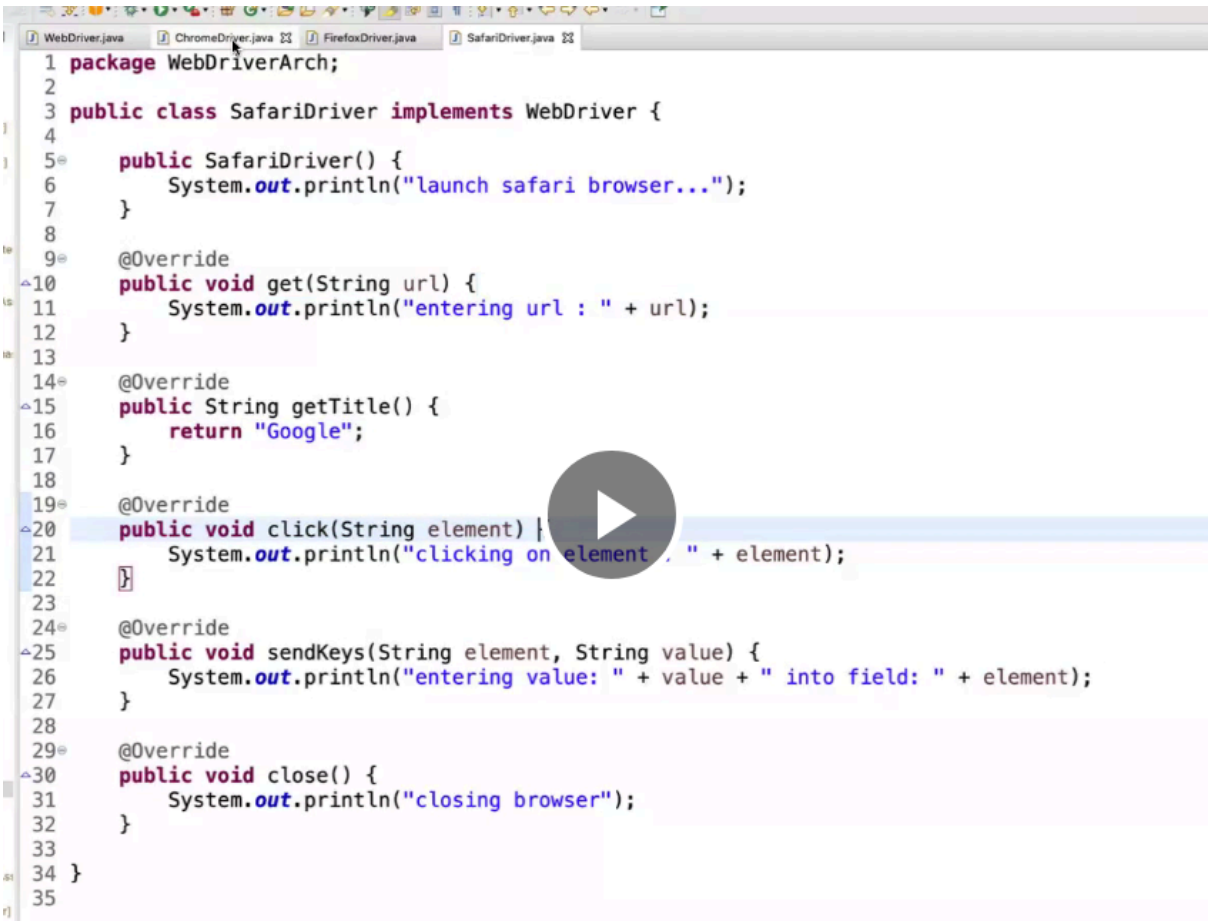

WebDriver.java
ChromeDriver.java
1 package WebDriverArch;
2
3 public class ChromeDriver implements WebDriver {
4
5 public ChromeDriver() {
6 System.out.println("launch chrome browser...");
7 }
8
9 @Override
10 public void get(String url) {
11 System.out.println("entering url : " + url);
12 }
13
14 @Override
15 public String getTitle() {
16 return "Google";
17 }
18
19 @Override
20 public void click(String element)
21 System.out.println("clicking on element " + element);
22 }
23
24 @Override
25 public void sendKeys(String element, String value) {
26 System.out.println("entering value: " + value + " into field: " + element);
27 }
28
29 @Override
30 public void close() {
31 System.out.println("closing browser");
32 }
33
34 }
35

ff-

WebDriver.java ChromeDriver.java FirefoxDriver.java

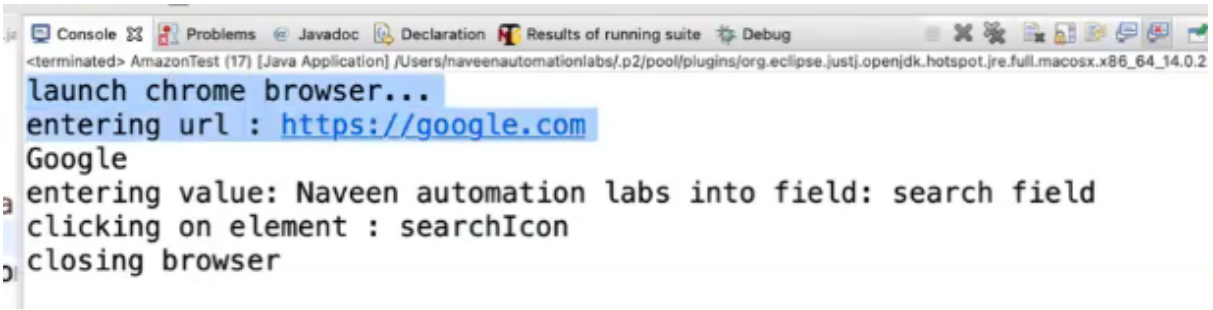
```java
package WebDriverArch;

public class FirefoxDriver implements WebDriver {

    public FirefoxDriver() {
        System.out.println("launch firefox browser...");
    }

    @Override
    public void get(String url) {
        System.out.println("entering url : " + url);
    }

    @Override
    public String getTitle() {
        return "Google";
    }

    @Override
    public void click(String element) {
        System.out.println("clicking on element : " + element);
    }

    @Override
    public void sendKeys(String element, String value) {
        System.out.println("entering value: " + value + " into field: " + element);
    }

    @Override
    public void close() {
        System.out.println("closing browser");
    }

}
```

Safari-


WebDriver.java
ChromeDriver.java
FirefoxDriver.java
SafariDriver.java
1 package WebDriverArch;
2
3 public class SafariDriver implements WebDriver {
4
5 public SafariDriver() {
6 System.out.println("launch safari browser...");
7 }
8
9 @Override
10 public void get(String url) {
11 System.out.println("entering url : " + url);
12 }
13
14 @Override
15 public String getTitle() {
16 return "Google";
17 }
18
19 @Override
20 public void click(String element) {
21 System.out.println("clicking on element " + element);
22 }
23
24 @Override
25 public void sendKeys(String element, String value) {
26 System.out.println("entering value: " + value + " into field: " + element);
27 }
28
29 @Override
30 public void close() {
31 System.out.println("closing browser");
32 }
33
34 }
35

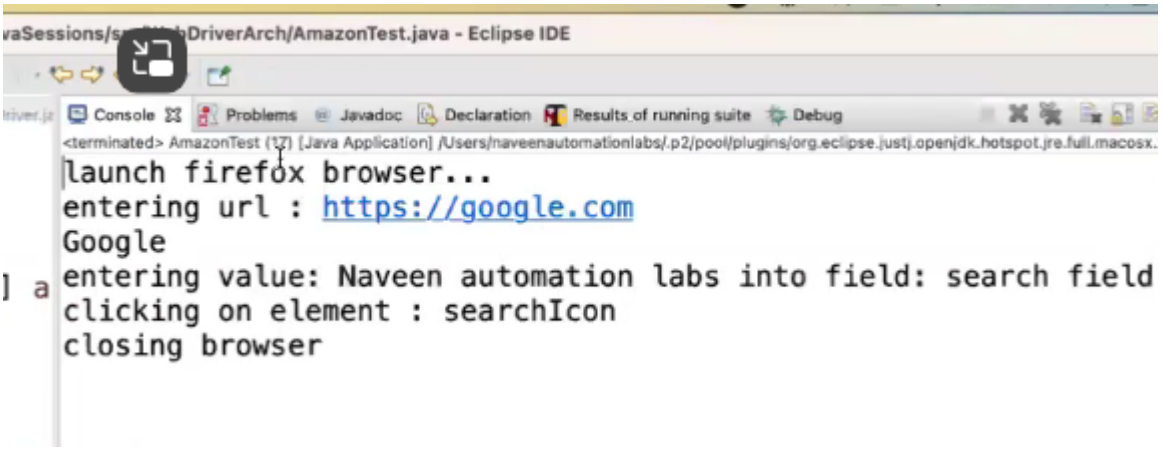
Test-


WebDriver.java
ChromeDriver.java
FirefoxDriver.java
SafariDriver.java
AmazonTest.java
1 package WebDriverArch;
2
3 public class AmazonTest {
4
5 public static void main(String[] args) {
6
7 ChromeDriver driver = new ChromeDriver();
8
9 driver.get("https://google.com");
10
11 String title = driver.getTitle();
12 System.out.println(title);
13
14 driver.sendKeys("search field", "Naveen automation labs");
15 driver.click("searchIcon");
16
17 driver.close();
18
19
20
21 }
22
23 }
24

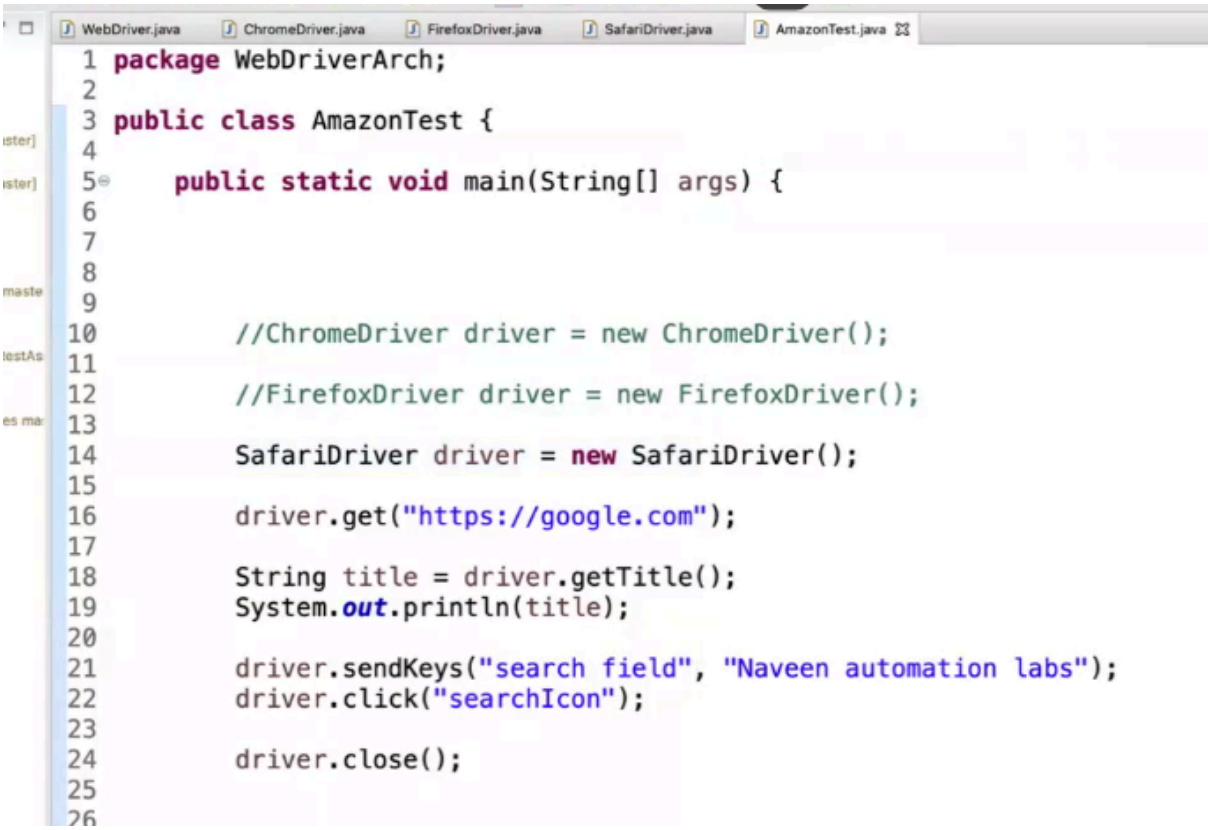

Console
Problems
Javadoc
Declaration
Results of running suite
Debug
<terminated> AmazonTest (17) [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_14.0.2
launch chrome browser...
entering url : https://google.com
Google
entering value: Naveen automation labs into field: search field
clicking on element : searchIcon
closing browser

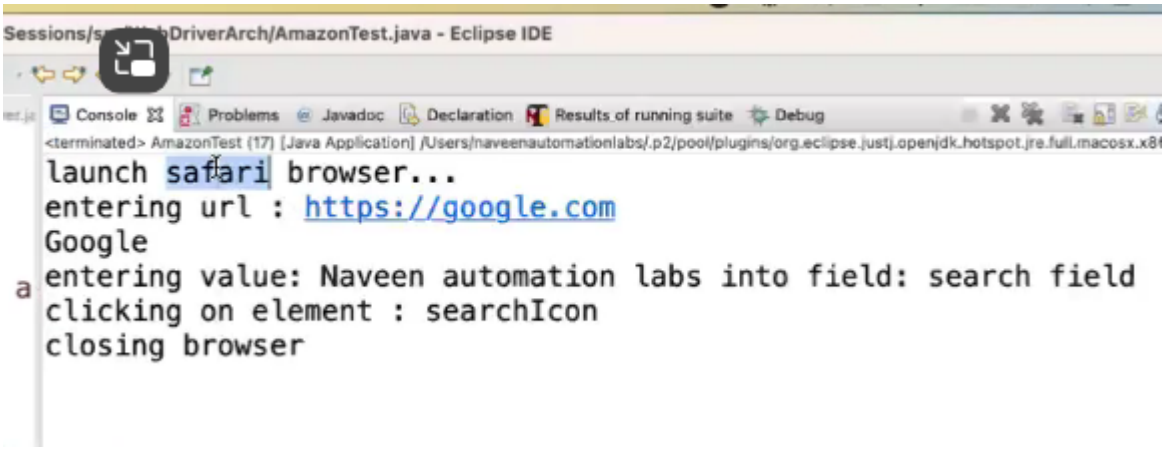
Ff-


11
12 FirefoxDriver driver = new FirefoxDriver();
13
14
15 driver.get("https://google.com");
16
17 String title = driver.getTitle();
18 System.out.println(title);
19
20 driver.sendKeys("search field", "Naveen automation labs");
21 driver.click("searchIcon");
22
23 driver.close();
24
25
26


DriverArch/AmazonTest.java - Eclipse IDE
Console
Problems
Javadoc
Declaration
Results of running suite
Debug
<terminated> AmazonTest (17) [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.
launch firefox browser...
entering url : https://google.com
Google
entering value: Naveen automation labs into field: search field
clicking on element : searchIcon
closing browser

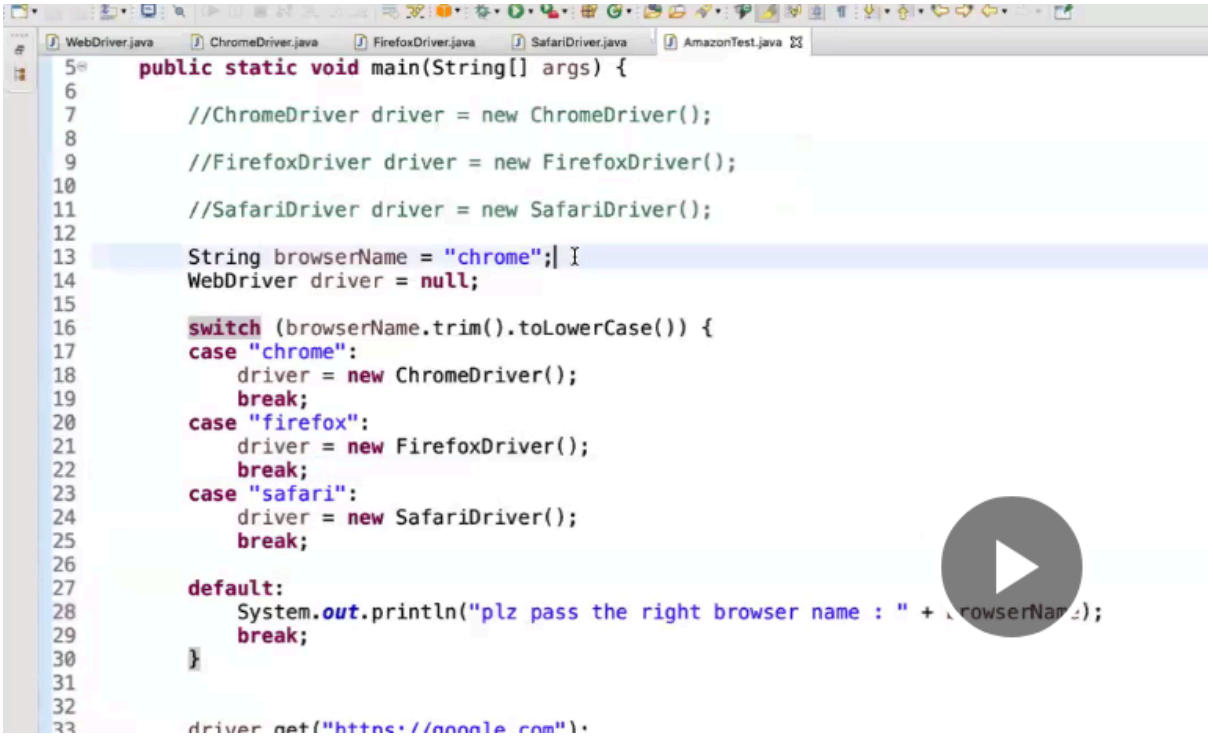
Safari-

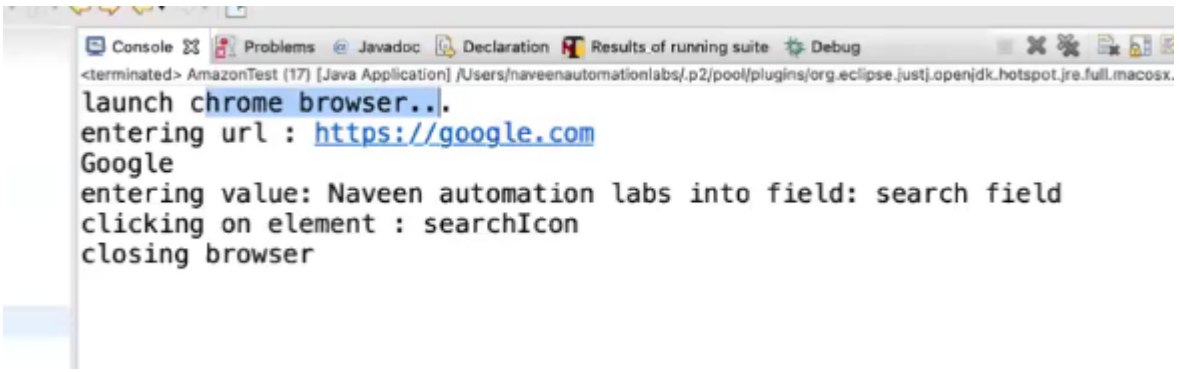

WebDriver.java
ChromeDriver.java
FirefoxDriver.java
SafariDriver.java
AmazonTest.java
package WebDriverArch;
public class AmazonTest {
public static void main(String[] args) {
//ChromeDriver driver = new ChromeDriver();
//FirefoxDriver driver = new FirefoxDriver();
SafariDriver driver = new SafariDriver();
driver.get("https://google.com");
String title = driver.getTitle();
System.out.println(title);
driver.sendKeys("search field", "Naveen automation labs");
driver.click("searchIcon");
driver.close();



Problem-

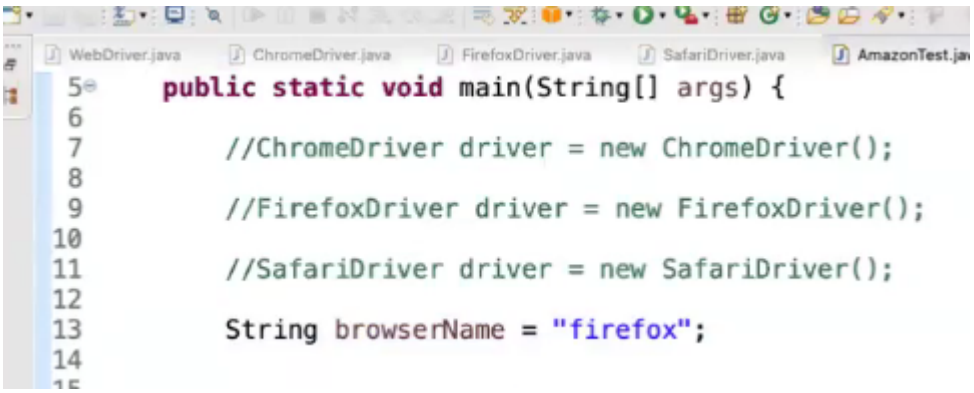
Every time we want to run on browser, we have to comment and uncomment. Top cast can solve the problem.

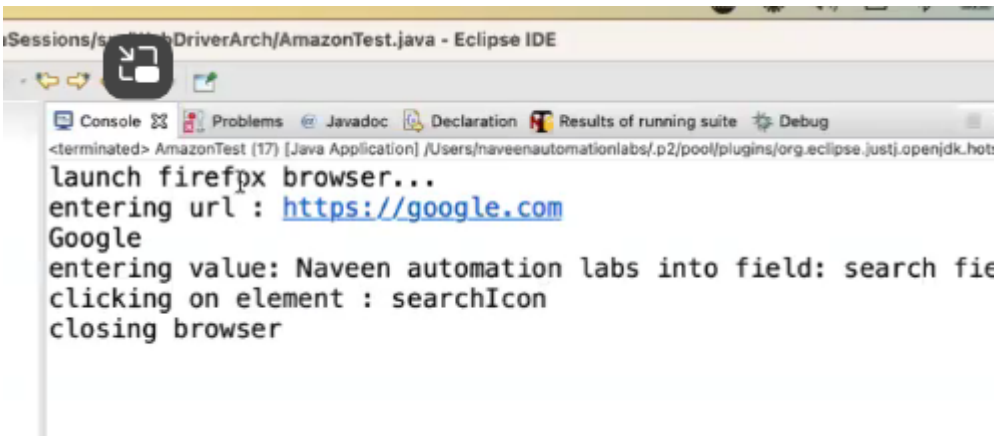




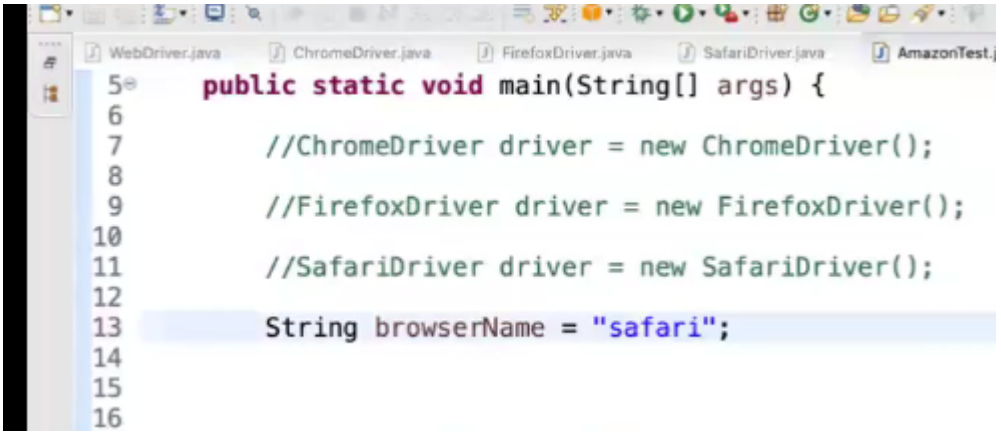
change browser name-

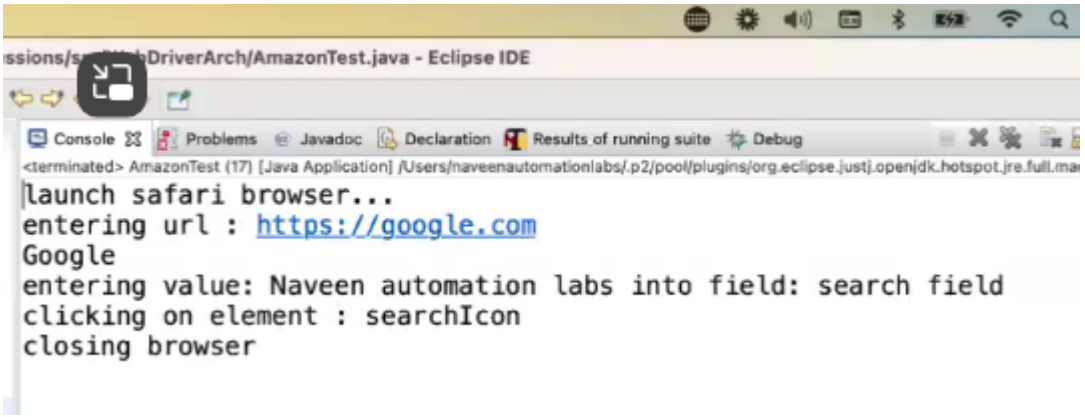
line 13-

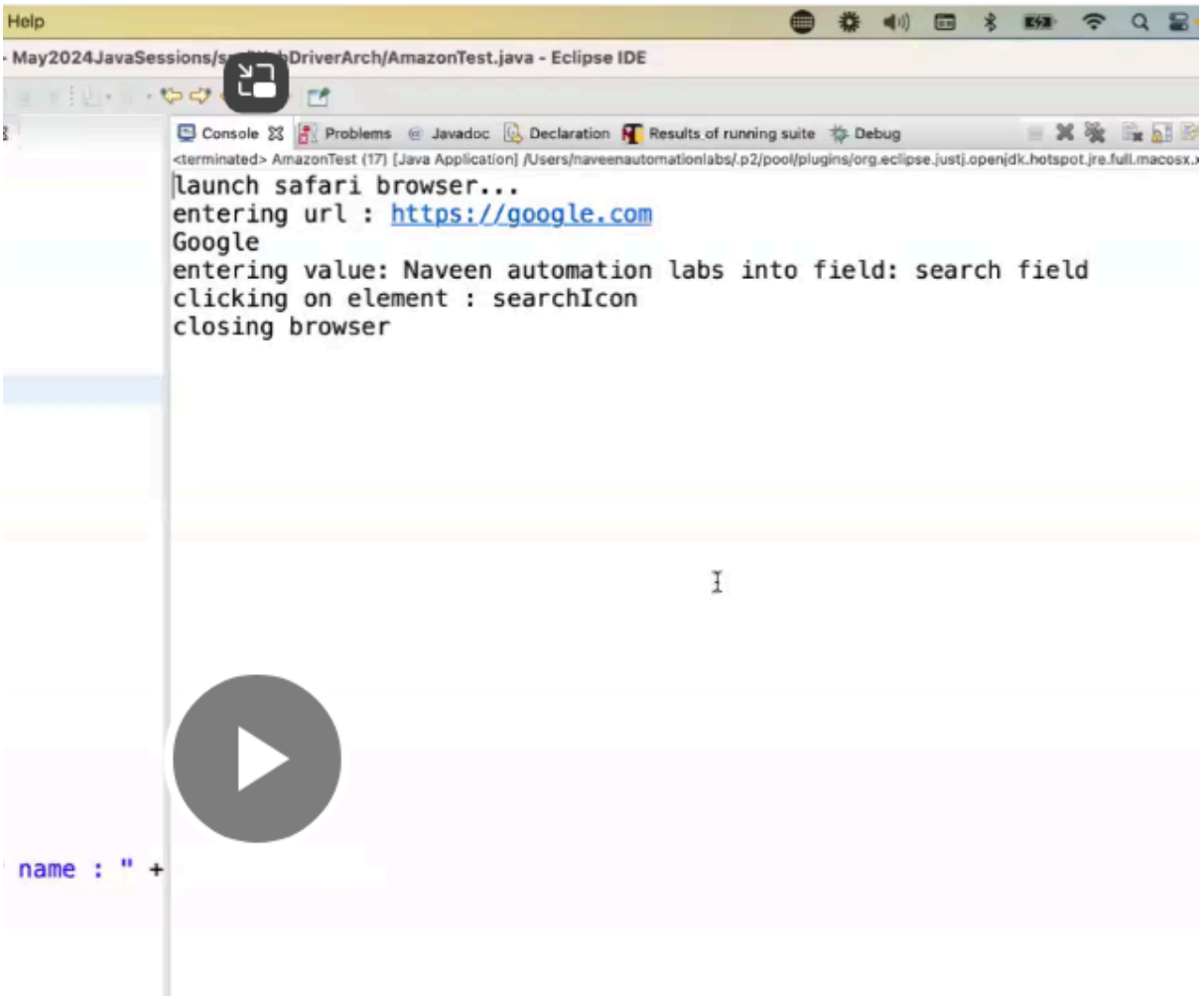




change line 13 browser name-


WebDriver.java
ChromeDriver.java
FirefoxDriver.java
SafariDriver.java
AmazonTest.j
public static void main(String[] args) {
//ChromeDriver driver = new ChromeDriver();
//FirefoxDriver driver = new FirefoxDriver();
//SafariDriver driver = new SafariDriver();
String browserName = "safari";

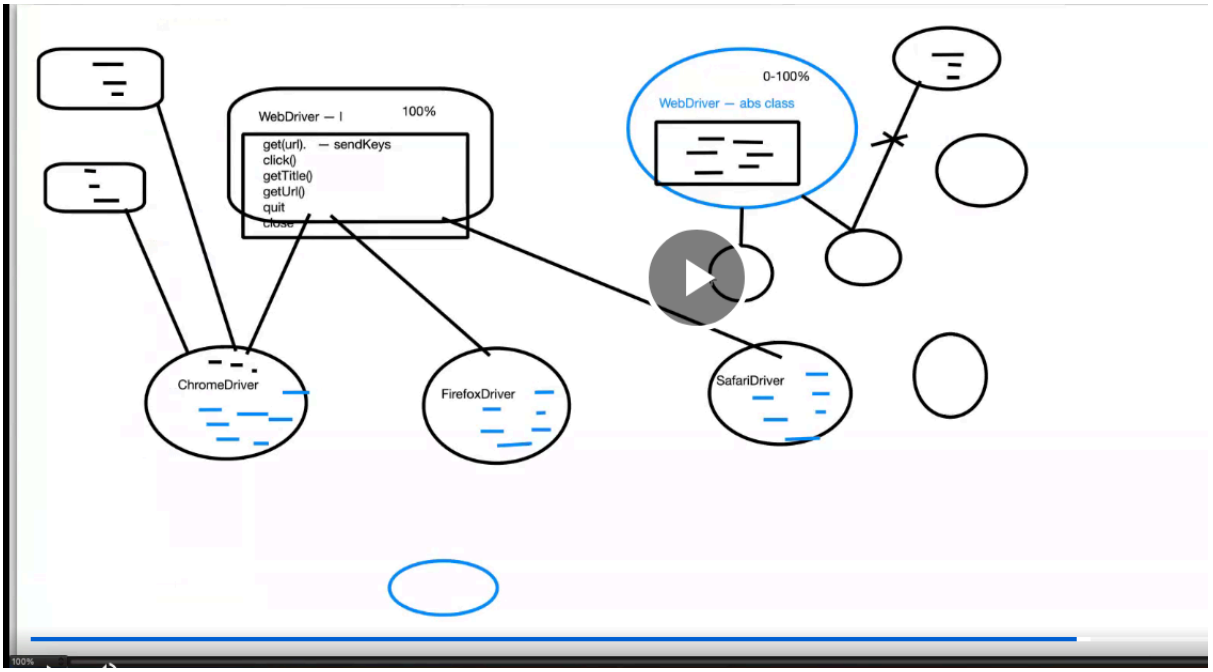

DriverArch/AmazonTest.java - Eclipse IDE
Console
Problems
Javadoc
Declaration
Results of running suite
Debug
launch safari browser...
entering url : https://google.com
Google
entering value: Naveen automation labs into field: search field
clicking on element : searchIcon
closing browser



example of real time top casting-

cross browser logic-

When to use interface and when to use abstract-



interface -

Multiple inheritance possible.

Methods have to be implemented by child class ensuring consistency.

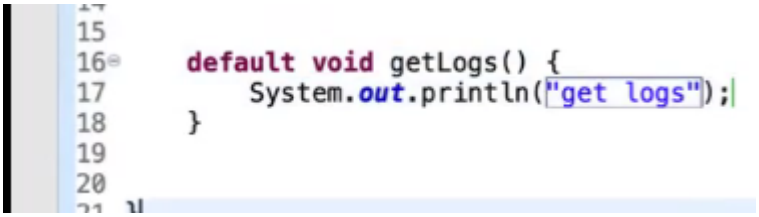
100 percent abstraction in interface but abstract (may or may not).

Note-

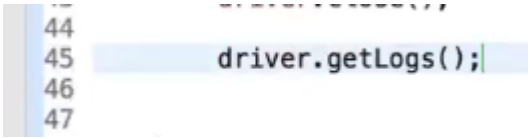
When local variable is declared, we need to initialise it before using, else when we use, we get error.

Example, webdriver driver=null;

Default method in interface with body-



Test class-



Get logs.

paste webdriver3, chromedriver3, ffdriver3, safaridriver3, testforwebdriver3-

webdriver3.java | chromedriver3.java | ffdriver3.java | safaridriver3.java | testforwebdriver3.java

```java
package com.day20;

public interface webdriver3 {

    public void get(String url);

    public String getTitle();

    public void click(String element);

    public void sendKeys(String element, String value);

    public void close();

    //default method in webdriver.
    default void getlogs() {
        System.out.println("get logs");
    }

    //create default method with parameter and return type.
    default String getlocations(String url) {
        System.out.println("get locations");
        return url;
    }
        return url;
    }

    //default method which is static.
    //Illegal combination of modifiers for the interface method getnames;
    //only one of abstract, default, or static permitted
//  default static String getnames(String name) {
//      System.out.println("get names");
//      return name;
//  }

}

```

chromedriver3.java | ffdriver3.java | safaridriver3.java | testforwebdriver3.java

```java
package com.day20;

public class chromedriver3 implements webdriver3 {

    //constructor.
    public chromedriver3() {
        System.out.println("default constructor of chrome driver class");
    }

    @Override
    public void get(String url) {
        System.out.println("Launching Chrome browser and navigating to: " + url);
    }

    @Override
    public String getTitle() {
        System.out.println("Getting the page title from Chrome browser...");
        return "Sample Chrome Page Title";
    }

    @Override
    public void click(String element) {
        System.out.println("Clicking on element: " + element + " in Chrome browser");
    }
```

```java
        System.out.println("Clicking on element: " + element + " in Chrome browser");
    }

    @Override
    public void sendKeys(String element, String value) {
        System.out.println("Typing '" + value + "' into element: " + element + " in Chrome browser");
    }

    @Override
    public void close() {
        System.out.println("Closing Chrome browser");
    }


}

```

ffdriver3.java | safaridriver3.java | testforwebdriver3.java

```java
package com.day20;

public class ffdriver3 implements webdriver3 {

    //constructor.
    public ffdriver3() {
        System.out.println("default constructor of ff driver class");
    }

    @Override
    public void get(String url) {
        System.out.println("Launching ff browser and navigating to: " + url);
    }

    @Override
    public String getTitle() {
        System.out.println("Getting the page title from ff browser...");
        return "Sample ff Page Title";
    }

    @Override
    public void click(String element) {
        System.out.println("Clicking on element: " + element + " in ff browser");
    }

    @Override
    public void sendKeys(String element, String value) {
        System.out.println("Typing '" + value + "' into element: " + element + " in ff browser");
    }
```

```
28          System.out.println("Typing '" + value + "' into element: " + element + " in ff browser");
29      }
30
31      @Override
32      public void close() {
33          System.out.println("Closing ff browser");
34      }
35
36
37  }
38
```

safaridriver3.java testforwebdriver3.java

```
1  package com.day20;
2
3  public class safaridriver3 implements webdriver3 {
4
5      //constructor.
6      public safaridriver3() {
7          System.out.println("default constructor of safari driver class");
8      }
9
10     @Override
11     public void get(String url) {
12         System.out.println("Launching safari browser and navigating to: " + url);
13     }
14
15     @Override
16     public String getTitle() {
17         System.out.println("Getting the page title from safari browser...");
18         return "Sample ff Page Title";
19     }
20
21     @Override
22     public void click(String element) {
23         System.out.println("Clicking on element: " + element + " in safari browser");
24     }
25
26     @Override
27     public void sendKeys(String element, String value) {
28         System.out.println("Typing '" + value + "' into element: " + element + " in safari browser");
29     }
30
```

```
28          System.out.println("Typing '" + value + "' into element: " + element + " in safari browser");
29      }
30
31      @Override
32      public void close() {
33          System.out.println("Closing safari browser");
34      }
35
36
37  }
38
```

testforwebdriver3.java

```
1  package com.day20;
2
3  public class testforwebdriver3 {
4
5      public static void main(String[] args) {
6
7          //top casting solves manual changes.
8          String browsername ="chrome";
9          webdriver3 driver=null;
10
11         switch(browsername.trim().toLowerCase()){
12         case "chrome":
13             driver=new chromedriver3();
14             break;
15         case "ff":
16             driver=new ffdriver3();
17             break;
18         case "safari":
19             driver=new safaridriver3();
20             break;
21         default:
22             System.out.println("invalid browser name " + browsername);
23             break;
24             }
```

```
23             break;
24             }
25
26         //can call default methods from interface directly.
27         driver.getlogs();
28         String s1=driver.getlocations("tatanka");
29         System.out.println(s1);
30
31     }
32
33 }
34
35 //default constructor of chrome driver class
36 //get logs
37 //get locations
38 //tatanka
39
40
41
42
43
44
45
```

Chrome driver-

Can be inherited also – default methods.

```
29      @Override
30      public void close() {
31          System.out.println("closing browser");
32          getLogs();
33      }
34
35  }
```

paste webdriver4, chromedriver4, ffdriver4, safaridriver4, testforwebdriver4-

testforwebdriver4.java | safaridriver4.java | ffdriver4.java | chromedriver4.java | webdriver4.java

```
1  package com.day20;
2
3  public interface webdriver4 {
4
5      public void get(String url);
6
7      public String getTitle();
8
9      public void click(String element);
10
11     public void sendKeys(String element, String value);
12
13     public void close();
14
15     //default method in webdriver.
16     default void getlogs() {
17         System.out.println("get logs");
18     }
19
20     //create default method with parameter and return type.
21     default String getlocations(String url) {
22         System.out.println("get locations");
23         return url;
24     }
25
```

```
23         return url;
24     }
25
26     //default method which is static.
27     //Illegal combination of modifiers for the interface method getnames;
28     //only one of abstract, default, or static permitted
29 //  default static String getnames(String name) {
30 //      System.out.println("get names");
31 //      return name;
32 //  }
33
34 }
35
```

testforwebdriver4.java | safaridriver4.java | ffdriver4.java | chromedriver4.java

```java
package com.day20;

public class chromedriver4 implements webdriver4 {

    //constructor.
    public chromedriver4() {
        System.out.println("default constructor of chrome driver class");
    }

    @Override
    public void get(String url) {
        System.out.println("Launching Chrome browser and navigating to: " + url);
    }

    @Override
    public String getTitle() {
        System.out.println("Getting the page title from Chrome browser...");
        return "Sample Chrome Page Title";
    }

    @Override
    public void click(String element) {
        System.out.println("Clicking on element: " + element + " in Chrome browser");
        getlogs();
    }

    @Override
    public void sendKeys(String element, String value) {
        System.out.println("Typing '" + value + "' into element: " + element + " in Chrome browser");
    }

    @Override
    public void close() {
        System.out.println("Closing Chrome browser");
        String s1=getlocations("kanada");
        System.out.println(s1);
    }


}
```

testforwebdriver4.java | safaridriver4.java | ffdriver4.java

```java
package com.day20;

public class ffdriver4 implements webdriver4 {

    //constructor.
    public ffdriver4() {
        System.out.println("default constructor of ff driver class");
    }

    @Override
    public void get(String url) {
        System.out.println("Launching ff browser and navigating to: " + url);
    }

    @Override
    public String getTitle() {
        System.out.println("Getting the page title from ff browser...");
        return "Sample ff Page Title";
    }

    @Override
    public void click(String element) {
        System.out.println("Clicking on element: " + element + " in ff browser");
    }
```

```
        System.out.println("Clicking on element: " + element + " in ff browser");
    }

    @Override
    public void sendKeys(String element, String value) {
        System.out.println("Typing '" + value + "' into element: " + element + " in ff browser");
    }

    @Override
    public void close() {
        System.out.println("Closing ff browser");
    }


}
```

testforwebdriver4.java | safaridriver4.java

```
package com.day20;

public class safaridriver4 implements webdriver4 {

    //constructor.
    public safaridriver4() {
        System.out.println("default constructor of safari driver class");
    }

    @Override
    public void get(String url) {
        System.out.println("Launching safari browser and navigating to: " + url);
    }

    @Override
    public String getTitle() {
        System.out.println("Getting the page title from safari browser...");
        return "Sample ff Page Title";
    }

    @Override
    public void click(String element) {
        System.out.println("Clicking on element: " + element + " in safari browser");
    }

    @Override
    public void sendKeys(String element, String value) {
        System.out.println("Typing '" + value + "' into element: " + element + " in safari browser");
    }

```

```
        System.out.println("Typing '" + value + "' into element: " + element + " in safari browser");
    }

    @Override
    public void close() {
        System.out.println("Closing safari browser");
    }


}
```

testforwebdriver4.java

```java
package com.day20;

public class testforwebdriver4 {

    public static void main(String[] args) {

        //top casting solves manual changes.
        String browsername ="chrome";
        webdriver4 driver=null;

        switch(browsername.trim().toLowerCase()){
        case "chrome":
            driver=new chromedriver4();
            break;
        case "ff":
            driver=new ffdriver4();
            break;
        case "safari":
            driver=new safaridriver4();
            break;
        default:
            System.out.println("invalid browser name " + browsername);
            break;
            }
```

```
            break;
            }

        driver.get("https://www.google.com/");
        String title=driver.getTitle();
        System.out.println(title);
        driver.sendKeys("search chrome", "chrome browser search is happening");
        driver.click("search chrome");
        driver.close();

    }

}

//default constructor of chrome driver class
//Launching Chrome browser and navigating to: https://www.google.com/
//Getting the page title from Chrome browser...
//Sample Chrome Page Title
//Typing 'chrome browser search is happening' into element: search chrome in Chrome browser
//Clicking on element: search chrome in Chrome browser
//get logs
//Closing Chrome browser
```

```
43 //get logs
44 //Closing Chrome browser
45 //get locations
46 //kanada
47
48
49
50
51
52
53
54
55
```