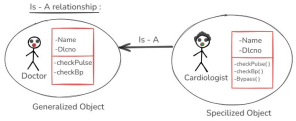
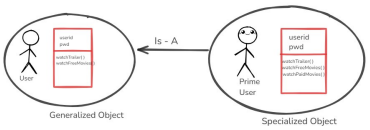
Is - A relationship :




Generalized Object

Specilized Object

-->cardioloist is a Doctor
-->Specialized Object will have all the properties of Generalized Object and one extra Behaviour.
-->cardiologist is a Specialized Doctor




Generalized Object

Specialized Object

1)Vehicle and car

car Is a Vehicle

Specialized

Generalized

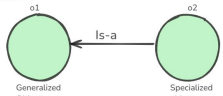
2)Student and Person

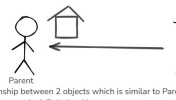
Student Is a Person

Specialized

Generalized

Define :



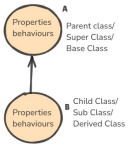




* The relationship between 2 objects which is similar to Parent and Child Child Is known as Is-A Relationship

* Where we can call the Child Object as Specialized Object and the Parent Object as Generalized Object

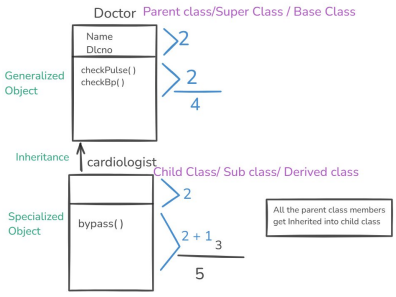
* We can achieve the Is-A Relationship by the concept in java called as Inheritance

**INHERITANCE :**


The Process of Acquiring all the members present in One class into another class is called as Inheritance
-->The class which is giving the properties is called as Parent Class
-->the class which is receiving properties is called as Child Class

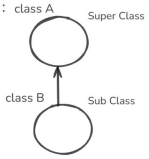
EX :




We can achieve Inheritance In java by Using 2 keywords

1)extends → 2 classes / 2 interfaces
2)implements → interface and class

Ex : class A




```
class  A {

}
class B extends A{

}
```

Syntax :

```
class SubClassName extends ParentClassName
{
}
```

Understanding the Inheritance with static and non-static members :

static members :


```
class A
{
static int age=10;
}
class B extends A{
public static void main(String[] args){
sopln(age);
}
}
```

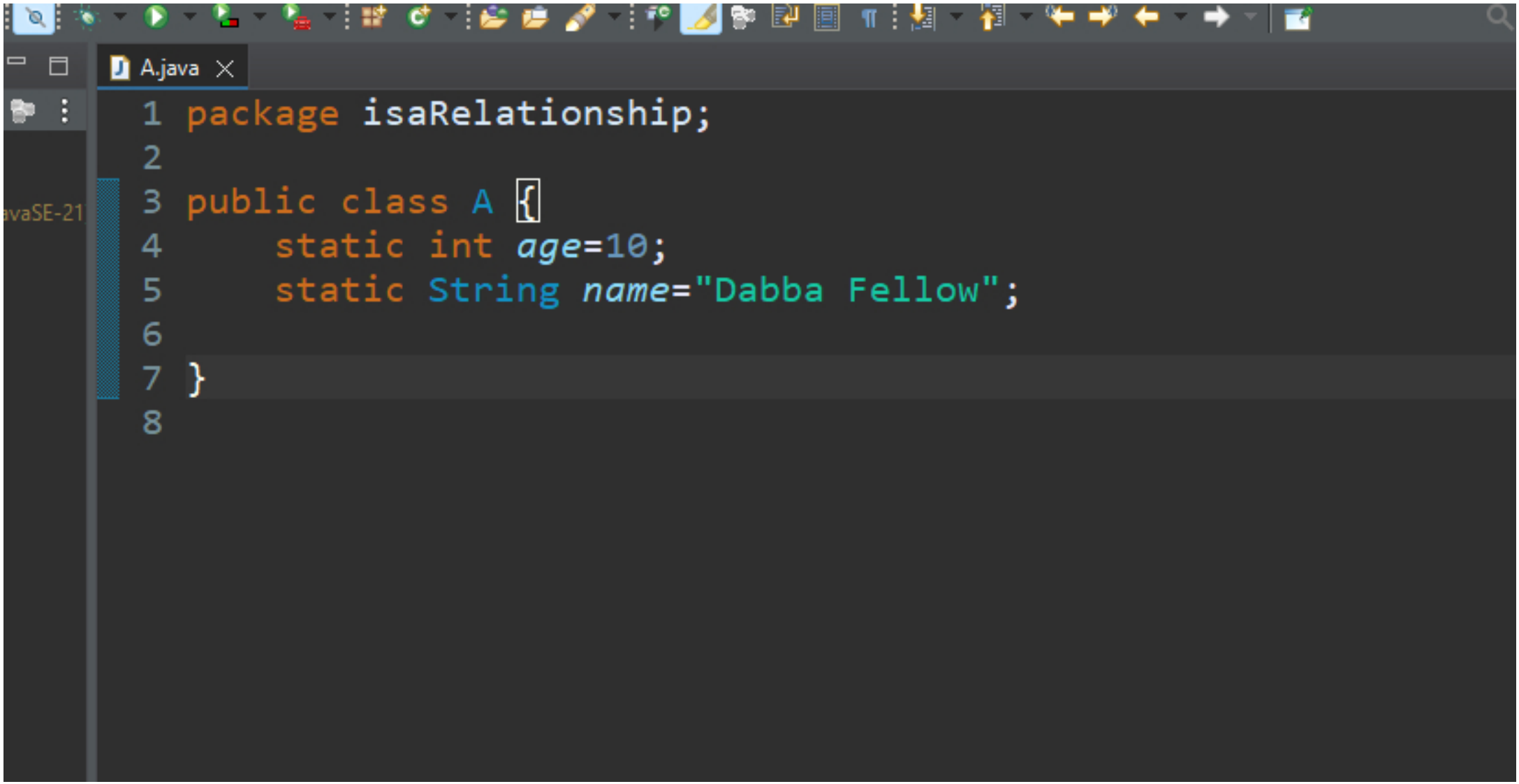
```
Ex : class parent{
    static void demo( ){
    Sopln("Hello from parent");
    }
}

    class Child extends Parent {
    public static void main(String[] args){
      demo();
    }
    }
```

All the static members we declare in parent class are acquired into child class
1)static members are Inherited
2) in subclass(Child class) static context we can use static members of Super and Sub Class directly without using classname as reference

non-static members :

```
    Ex : class Person {
            static String name="Pikachu";
              int age ;
           public void walk( ) {
             SopIn("i am walking");
    }
    }

    class Student extends Person {
    public static void main(String[] args){
      SopIn(name);
    SopIn(age);

    }
    }
```

Inside the non-static context of the subclass , we can always use non-static members of the parent class directly
--> inside the non-static context of subclass, we can use the static members and non-static membrs of the parent class directly


A.java
1 package isaRelationship;
2
3 public class A {
4 static int age=10;
5 static String name="Dabba Fellow";
6
7 }
8

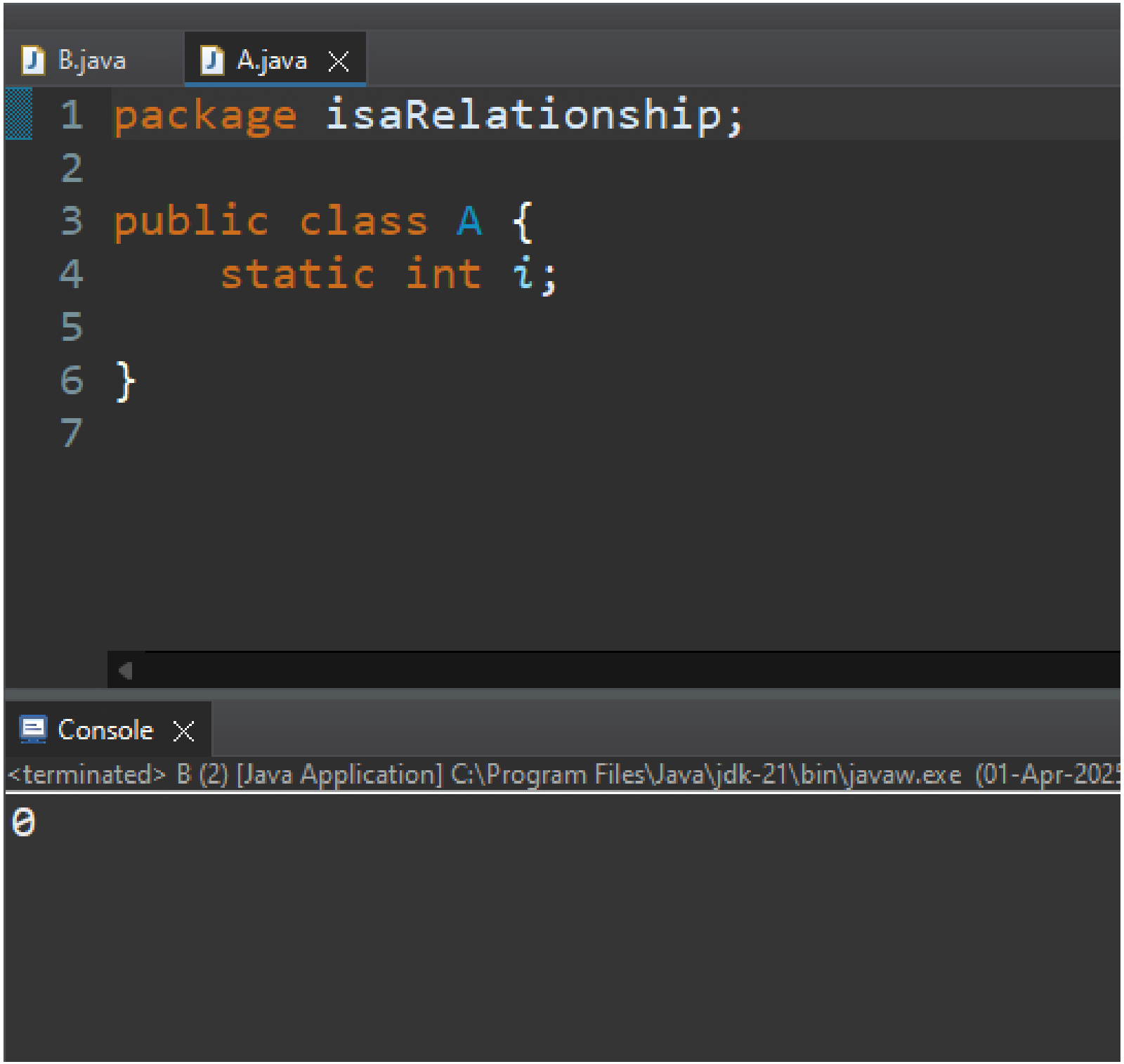
A.java B.java

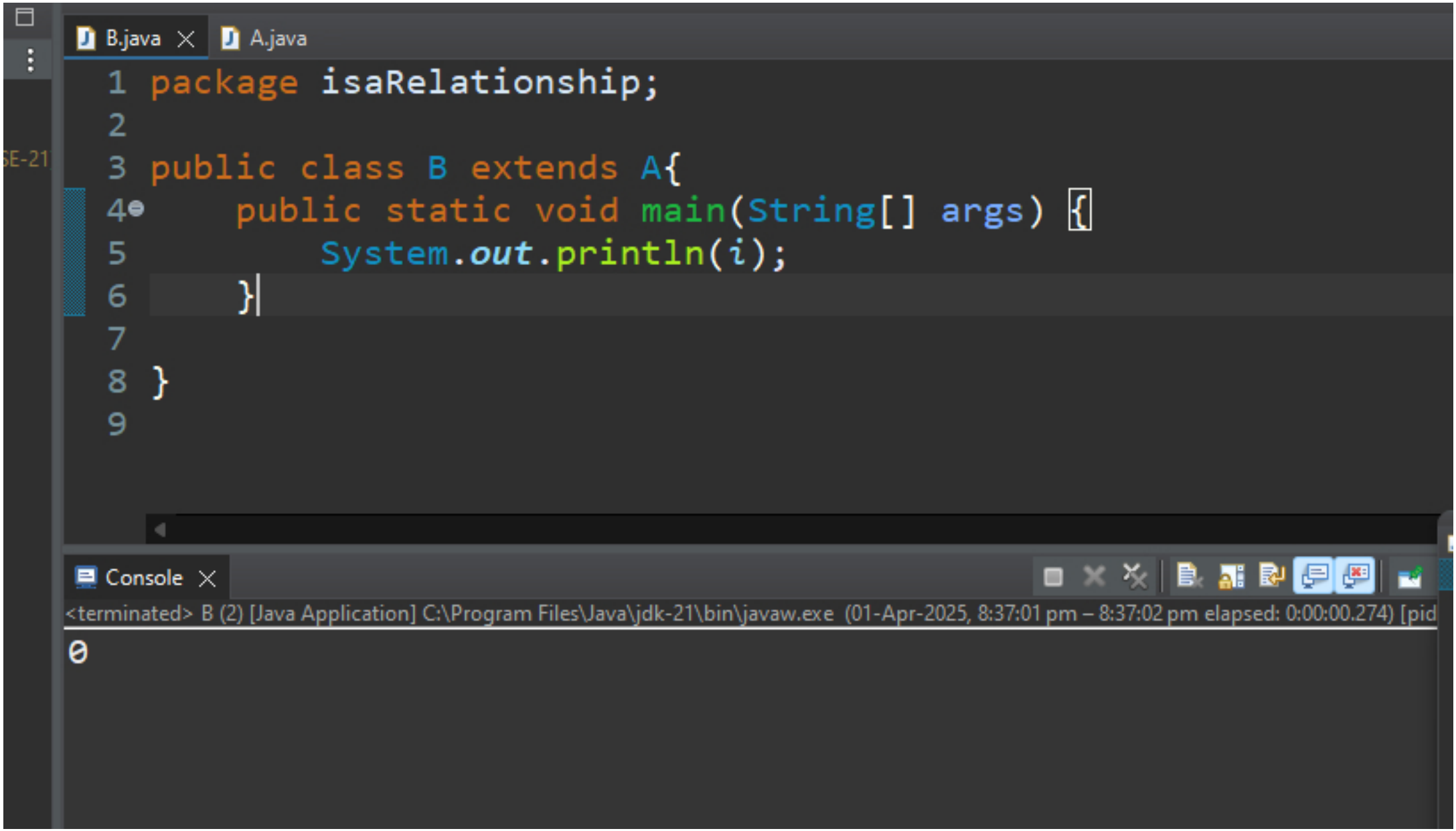
```java
package isaRelationship;

public class B extends A{
    public static void main(String[] args) {
        System.out.println(age);
        System.out.println(name);
    }

}

```

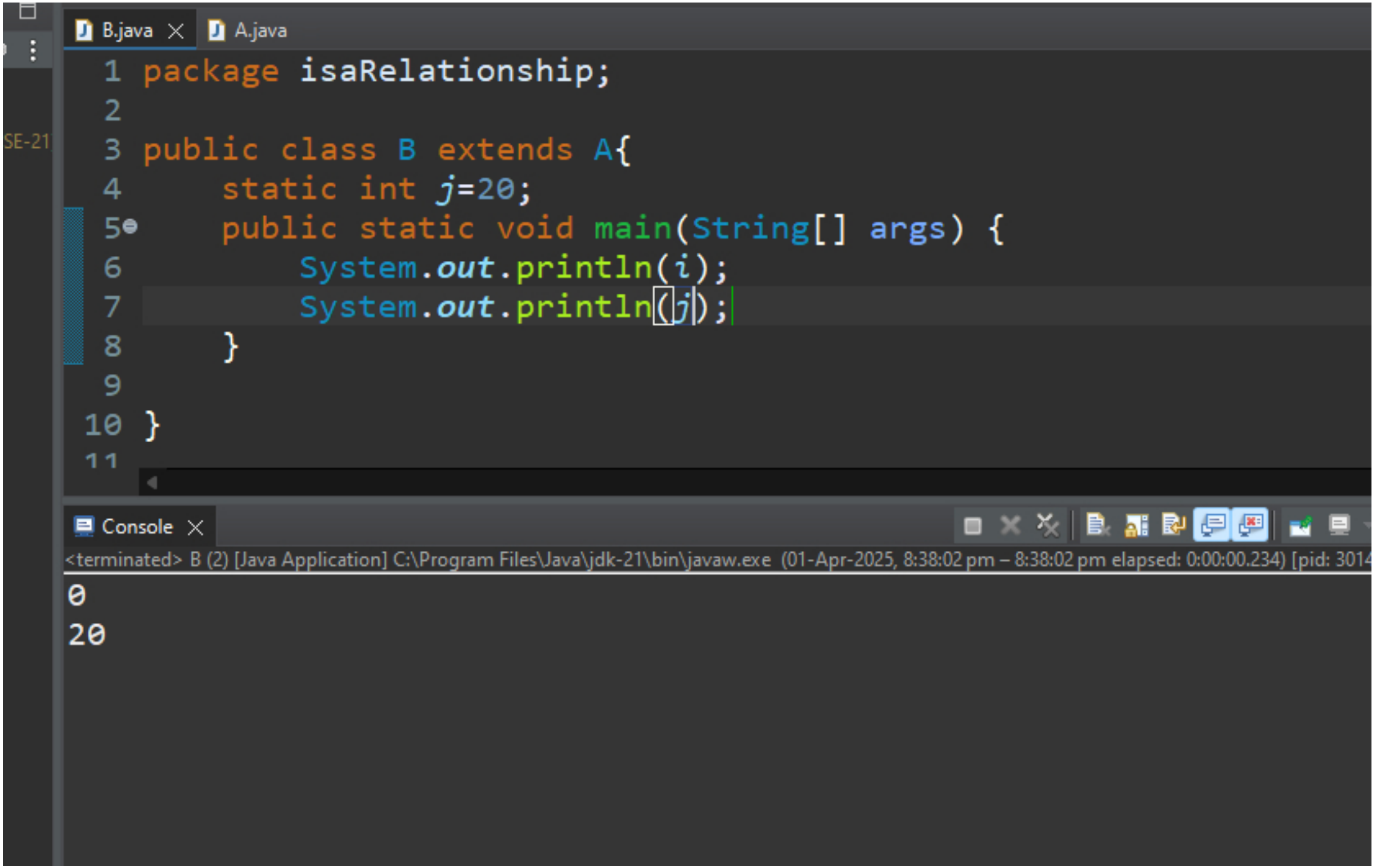
Console

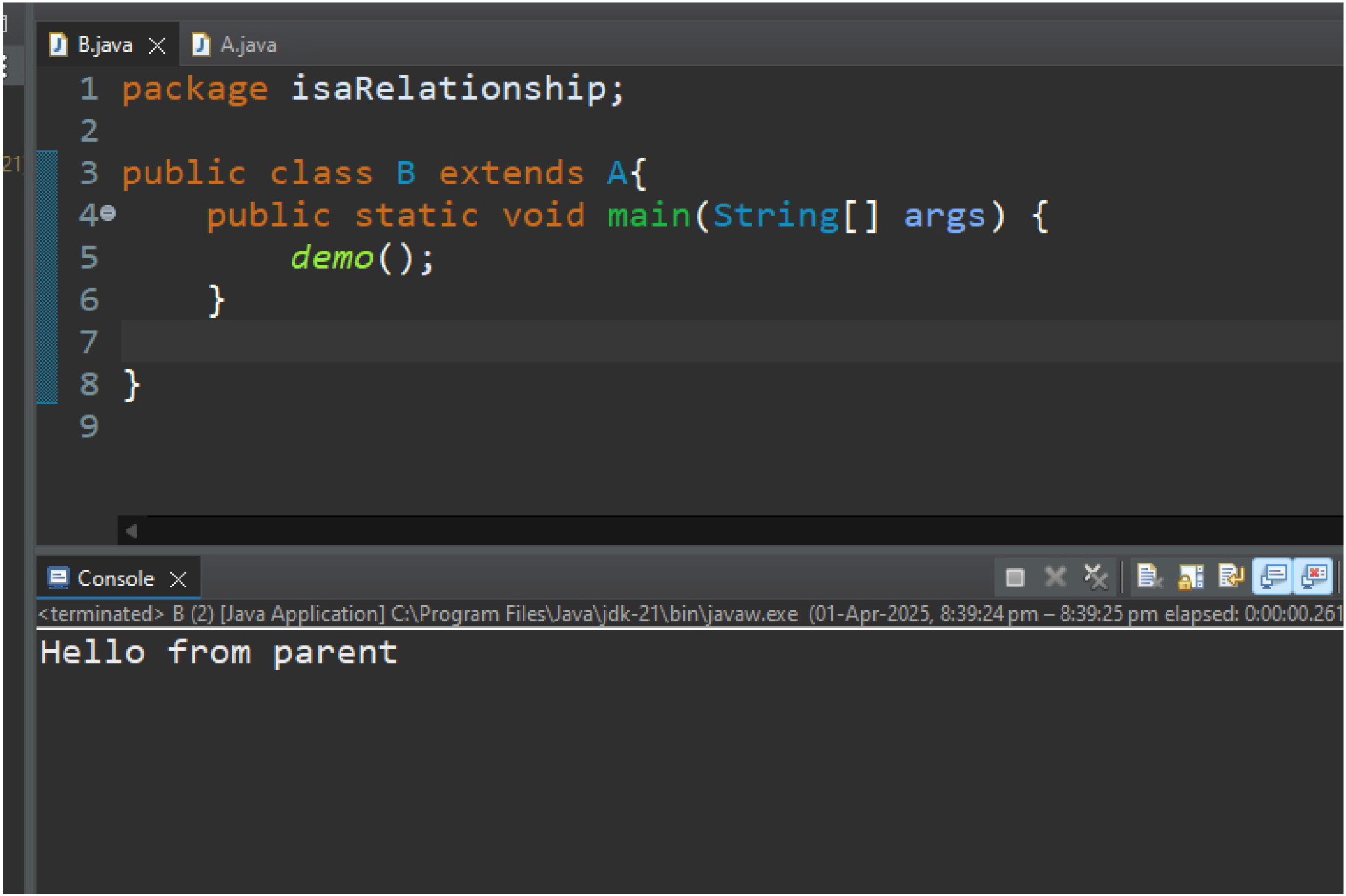
<terminated> B (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-2025, 8:35:57 pm – 8:35:57 pm elapsed: 0:00:00.108) [pid: 28156

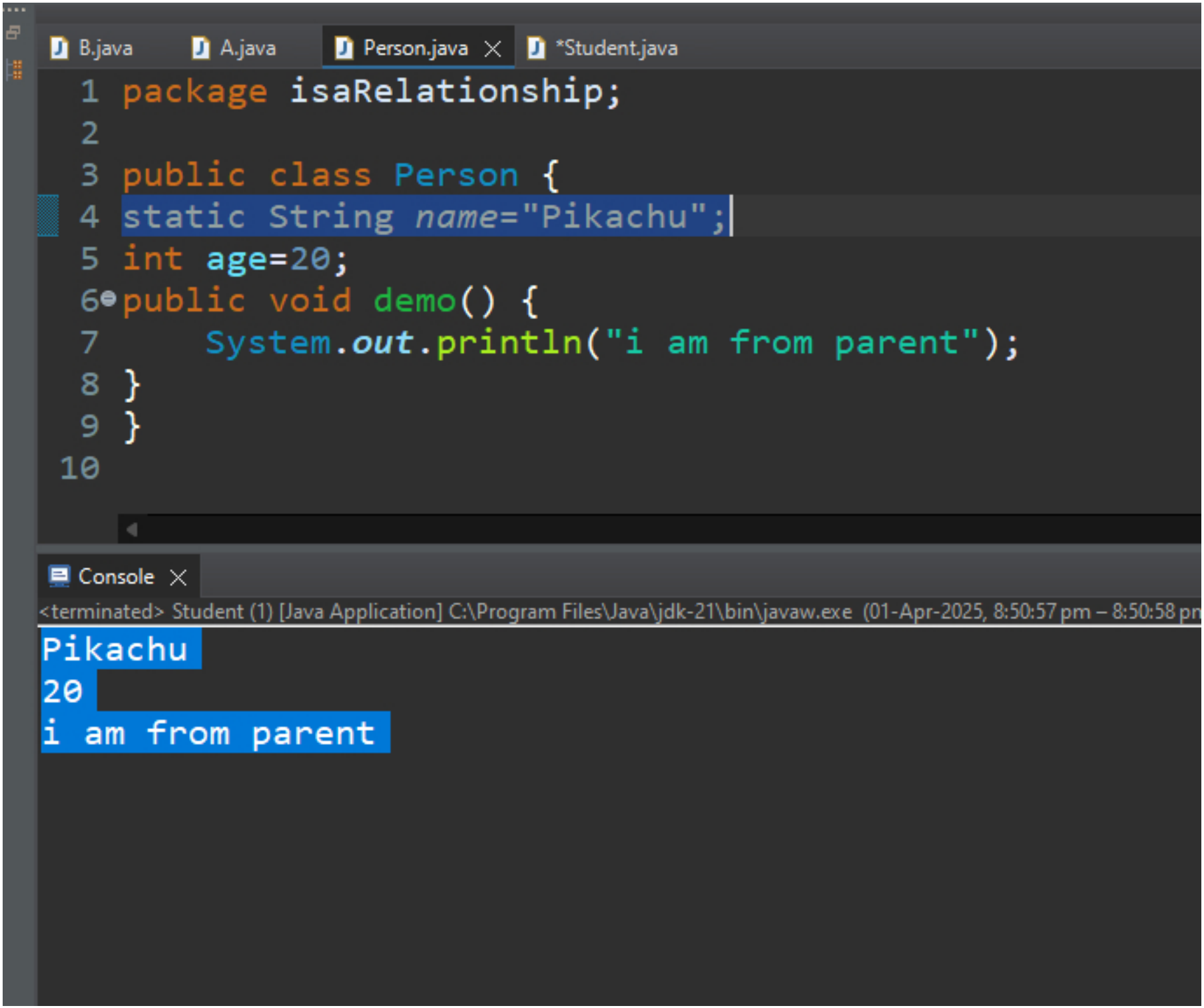
```
10
Dabba Fellow
```

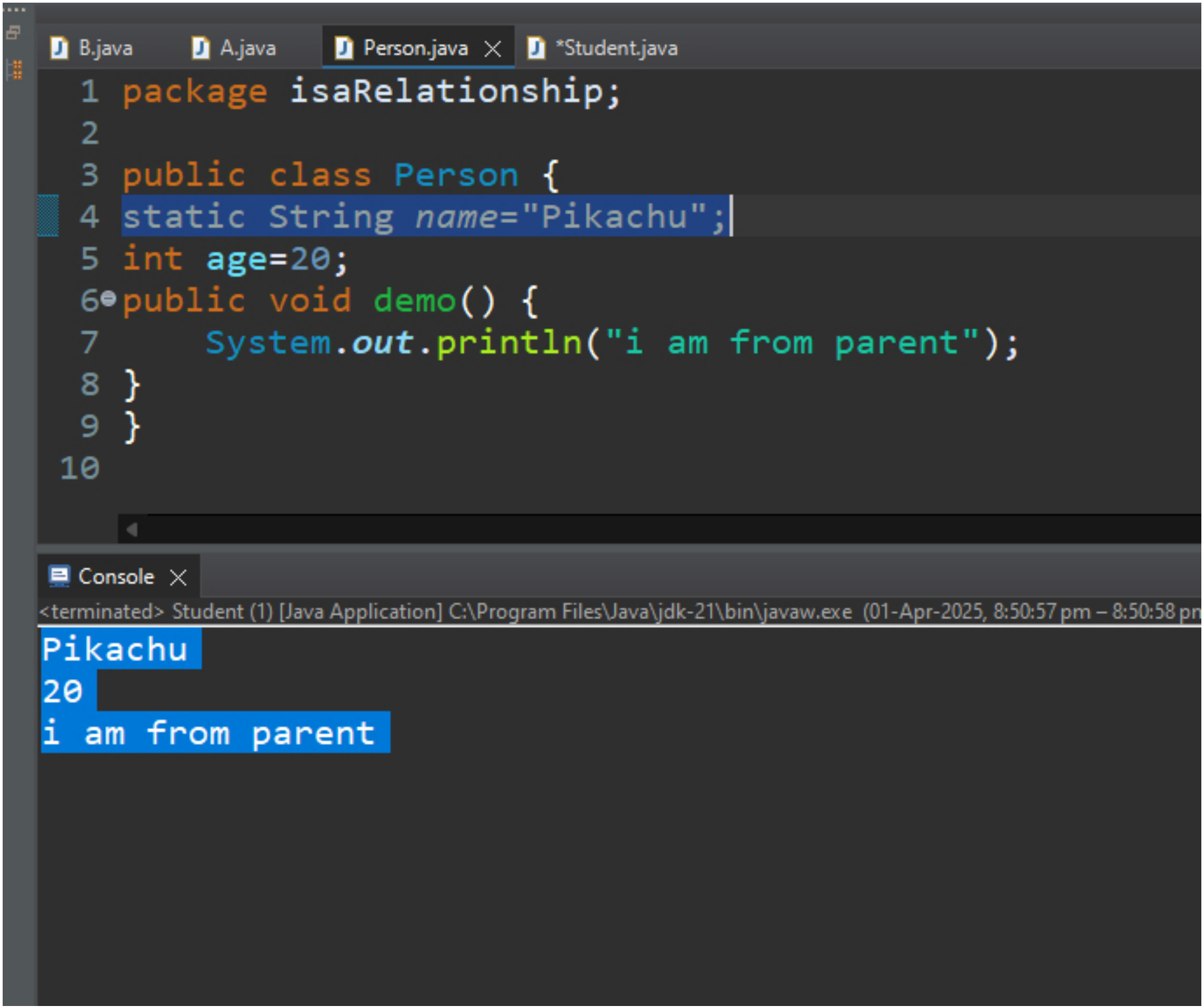

B.java
A.java
1 package isaRelationship;
2
3 public class A {
4 static int i;
5
6 }
7
Console
<terminated> B (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-202

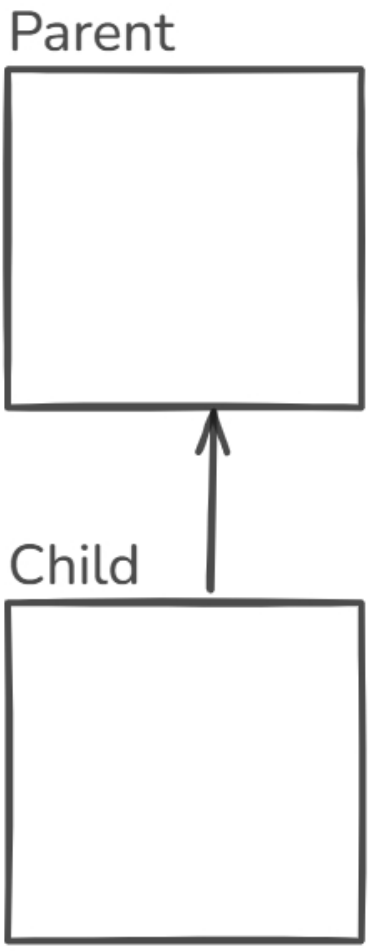

B.java
A.java
1 package isaRelationship;
2
3 public class B extends A{
4 public static void main(String[] args) {
5 System.out.println(i);
6 }
7
8 }
9
Console
<terminated> B (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-2025, 8:37:01 pm – 8:37:02 pm elapsed: 0:00:00.274) [pid
0


B.java
A.java
1 package isaRelationship;
2
3 public class B extends A{
4 static int j=20;
5 public static void main(String[] args) {
6 System.out.println(i);
7 System.out.println(j);
8 }
9
10 }
11
Console
<terminated> B (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-2025, 8:38:02 pm – 8:38:02 pm elapsed: 0:00:00.234) [pid: 3014
0
20


B.java
A.java
1 package isaRelationship;
2
3 public class B extends A{
4 public static void main(String[] args) {
5 demo();
6 }
7
8 }
9
Console
<terminated> B (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-2025, 8:39:24 pm – 8:39:25 pm elapsed: 0:00:00.261
Hello from parent


B.java
A.java
Person.java
*Student.java
1 package isaRelationship;
2
3 public class Person {
4 static String name="Pikachu";
5 int age=20;
6 public void demo() {
7 System.out.println("i am from parent");
8 }
9 }
10
Console
<terminated> Student (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-2025, 8:50:57 pm – 8:50:58 pm
Pikachu
20
i am from parent


B.java
A.java
Person.java
*Student.java
1 package isaRelationship;
2
3 public class Person {
4 static String name="Pikachu";
5 int age=20;
6 public void demo() {
7 System.out.println("i am from parent");
8 }
9 }
10
Console
<terminated> Student (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (01-Apr-2025, 8:50:57 pm – 8:50:58 pm
Pikachu
20
i am from parent




1)Sub Class always have
Inherited Members + Declared Memebers

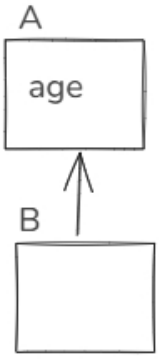
Total no.of members

2)In static context of Sub Class
we can use the static members of subclass
and super class directly

3)In non-static context of Sub Class
we can use static and non-static members
of both parent and child class directly

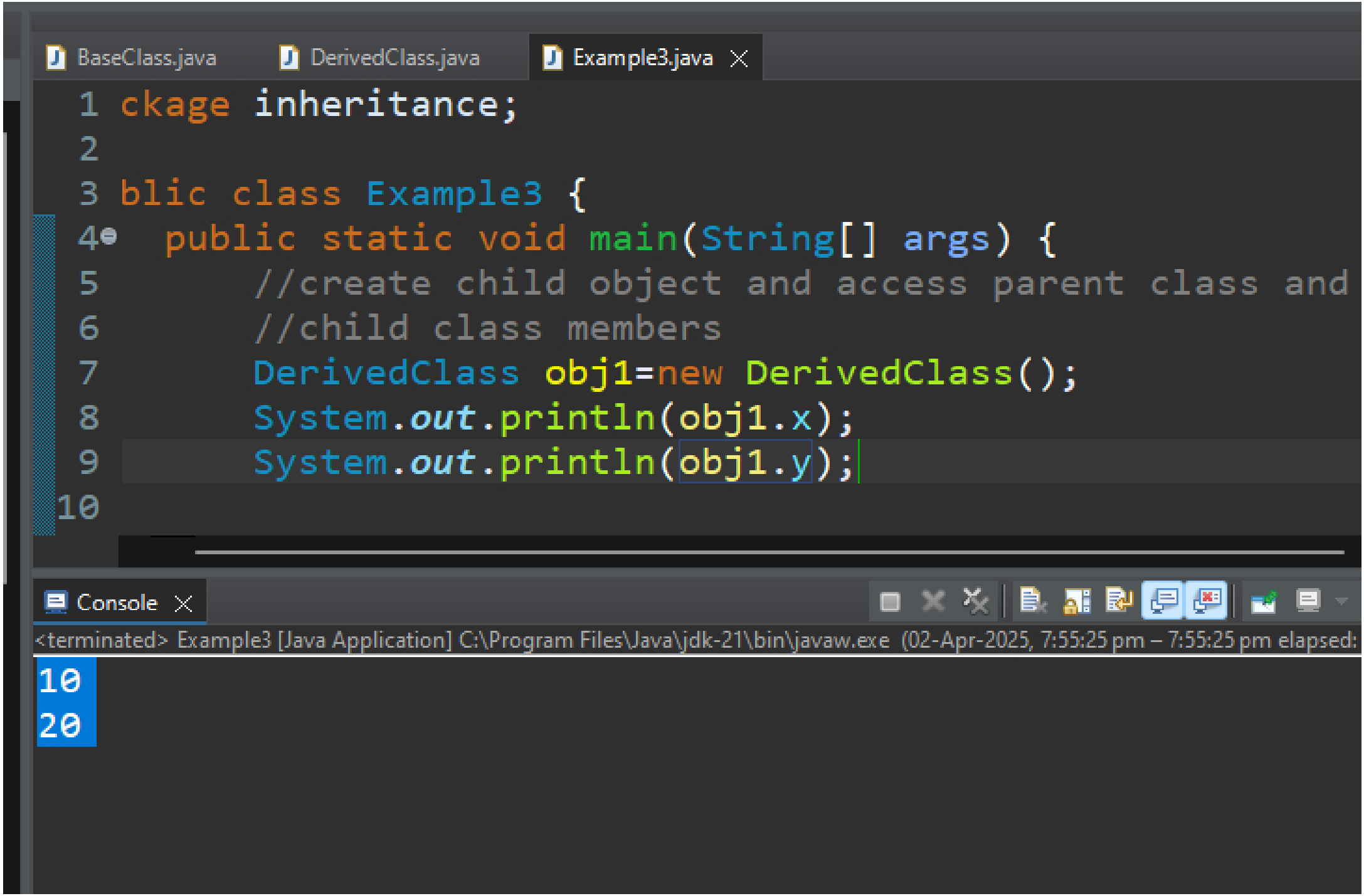
| | | Inherited | not-Inherited |
|---|---|---|---|
| class | static members | Variables ✓<br>methods ✓<br>Intializers — | ✗ |
| | non-static memebers | Variables ✓<br>methods ✓<br>Intializers ✗<br>Constructor ✗ | — |

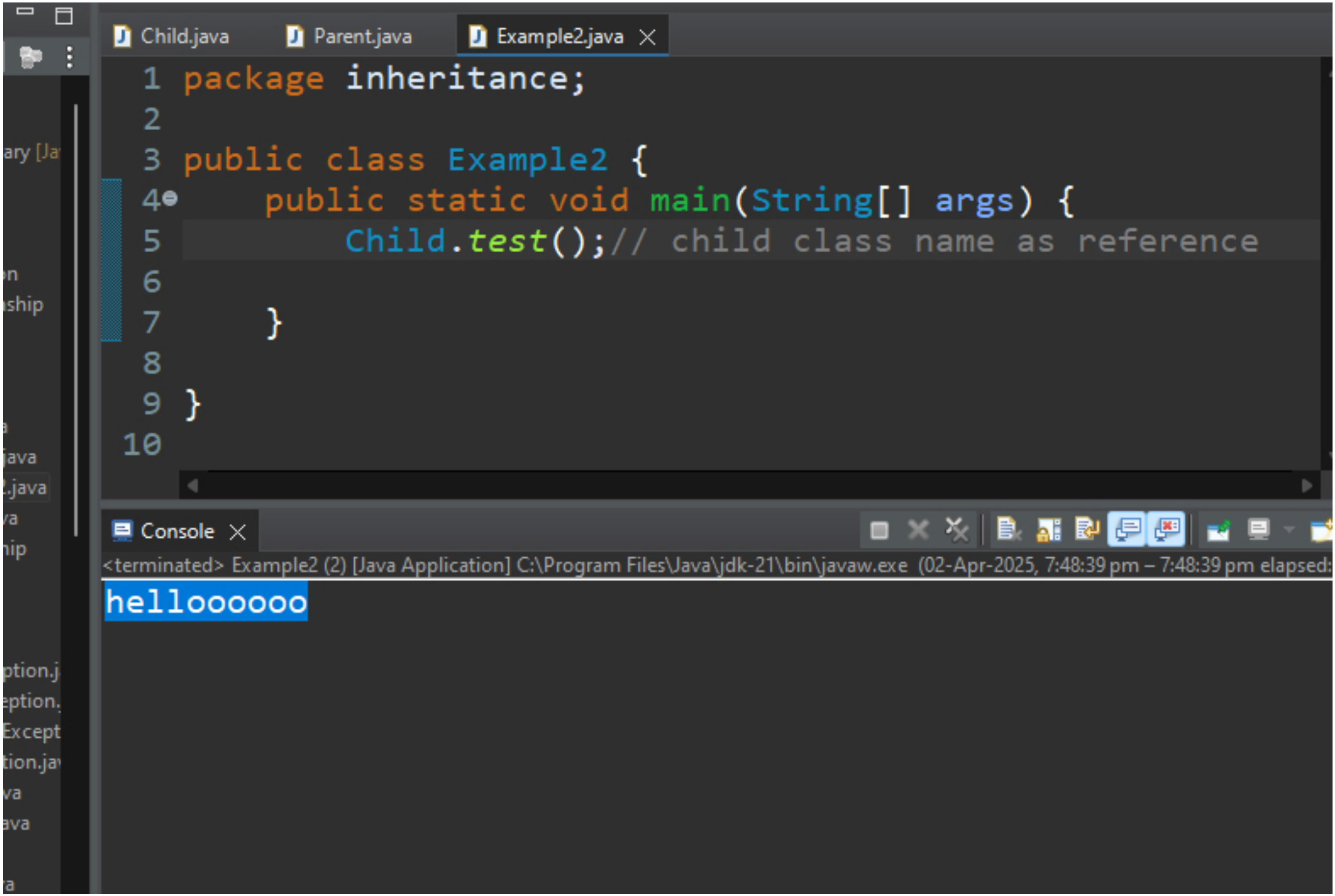
Understanding to access members using parent and child class :

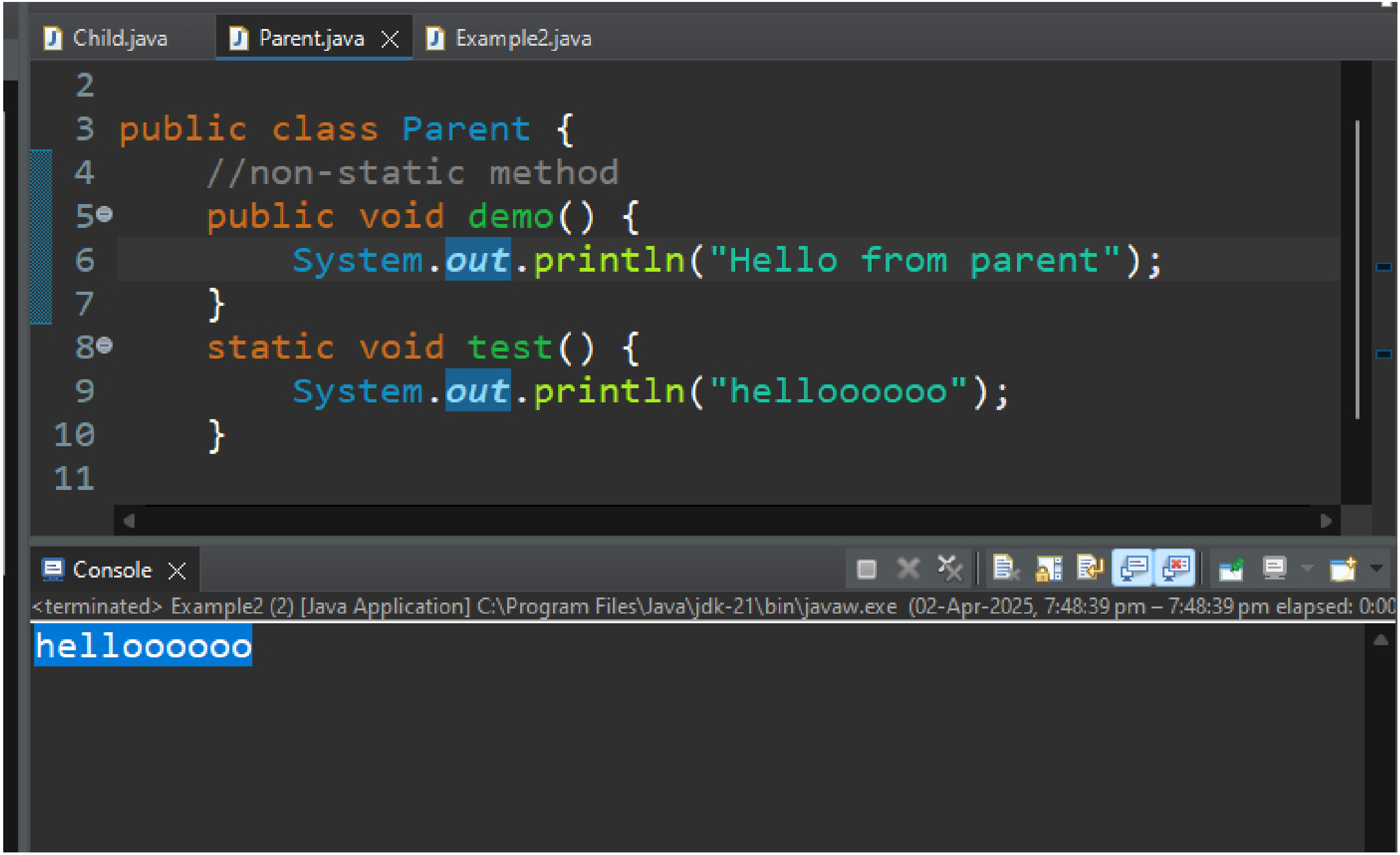


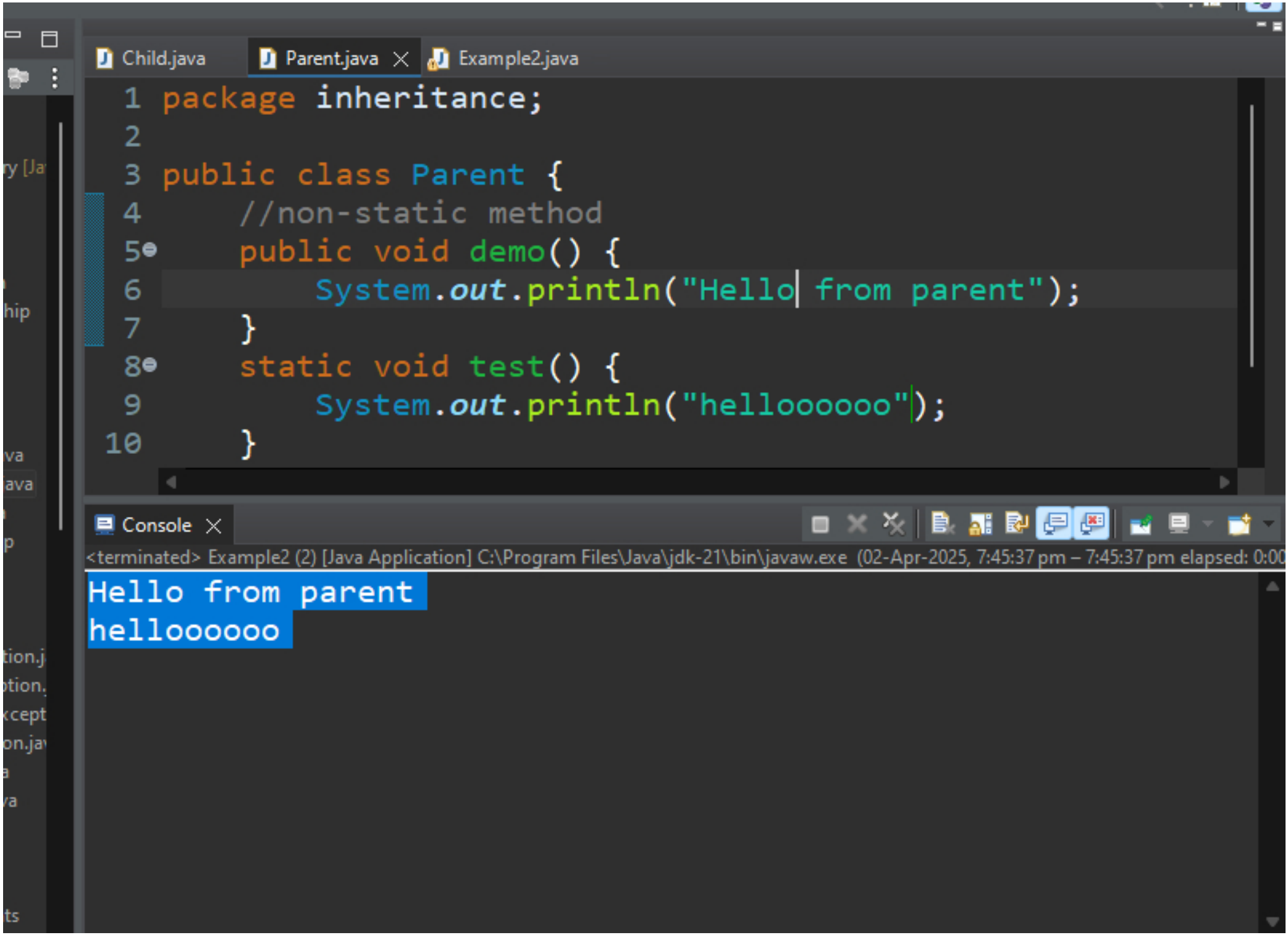

```
class A
{
int age;
}
class B extends A{

}
```

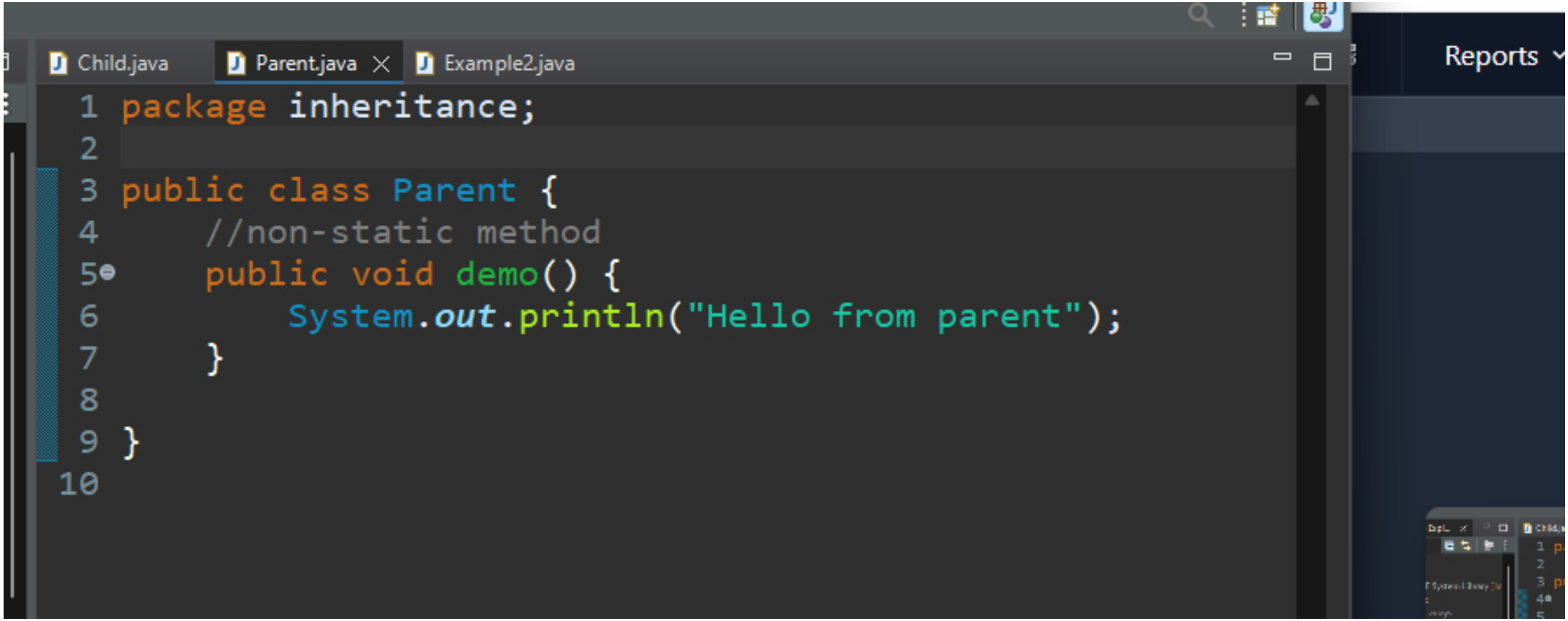
```
class Example {
public static void main(String[] args){
      A obj1=new A( );
      Sopln(obj1.age);        > parent class
       B obj2=new B( );
        Sopln(obj2.age);      > child class
}
}
```

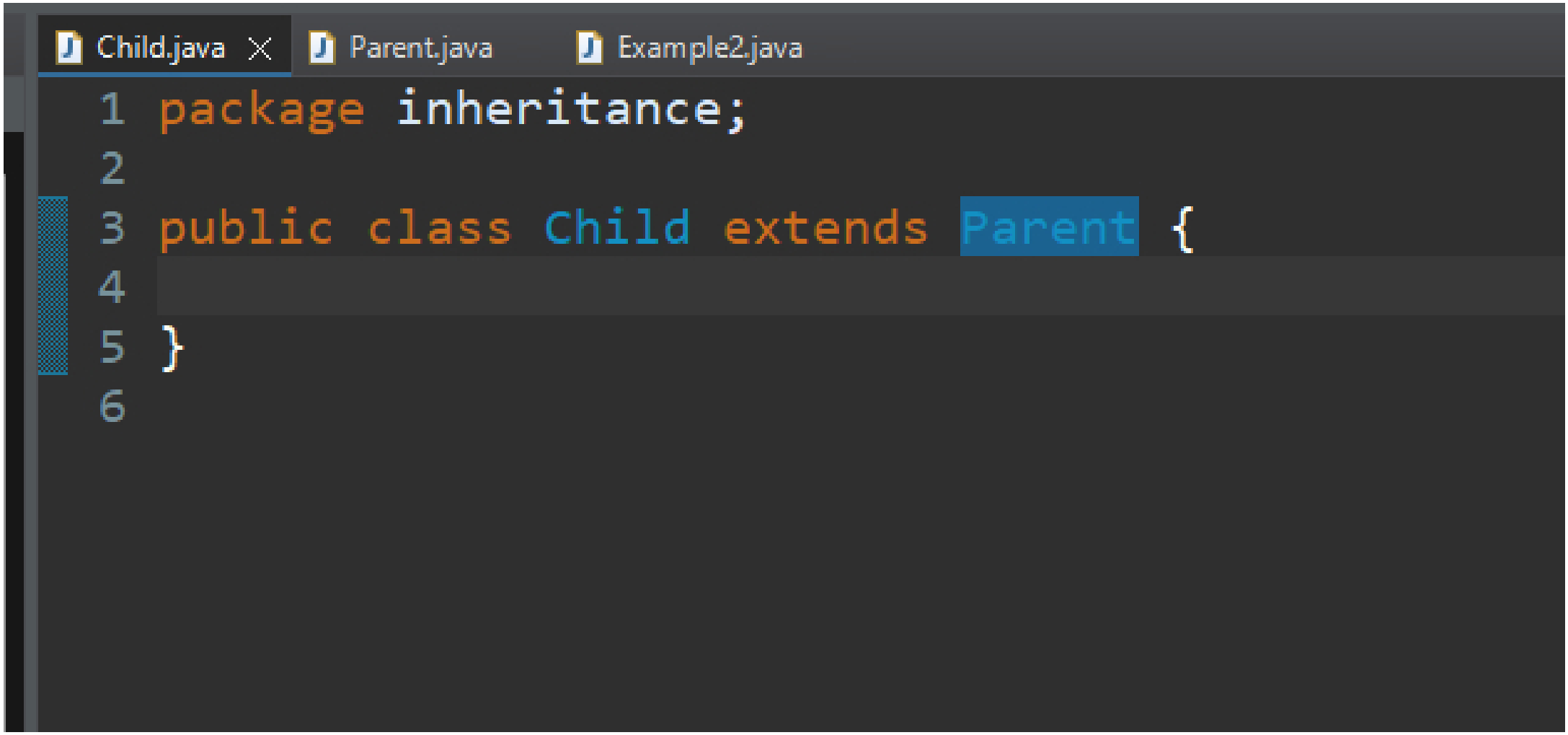

BaseClass.java
DerivedClass.java
Example3.java
1 ckage inheritance;
2
3 blic class Example3 {
4 public static void main(String[] args) {
5 //create child object and access parent class and
6 //child class members
7 DerivedClass obj1=new DerivedClass();
8 System.out.println(obj1.x);
9 System.out.println(obj1.y);
10
Console
<terminated> Example3 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (02-Apr-2025, 7:55:25 pm – 7:55:25 pm elapsed:
10
20


Child.java
Parent.java
Example2.java
1 package inheritance;
2
3 public class Example2 {
4 public static void main(String[] args) {
5 Child.test();// child class name as reference
6
7 }
8
9 }
10
Console
<terminated> Example2 (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (02-Apr-2025, 7:48:39 pm – 7:48:39 pm elapsed:
hellooooooo

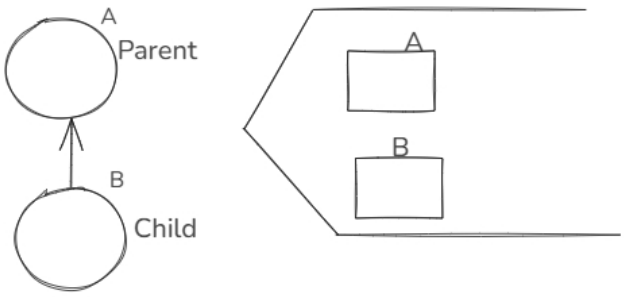

Child.java
Parent.java
Example2.java
2
3 public class Parent {
4 //non-static method
5 public void demo() {
6 System.out.println("Hello from parent");
7 }
8 static void test() {
9 System.out.println("hellooooooo");
10 }
11
Console
<terminated> Example2 (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (02-Apr-2025, 7:48:39 pm – 7:48:39 pm elapsed: 0:00
hellooooooo


Child.java
Parent.java
Example2.java
1 package inheritance;
2
3 public class Parent {
4 //non-static method
5 public void demo() {
6 System.out.println("Hello from parent");
7 }
8 static void test() {
9 System.out.println("hellooooooo");
10 }
Console
<terminated> Example2 (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (02-Apr-2025, 7:45:37 pm – 7:45:37 pm elapsed: 0:00
Hello from parent
hellooooooo

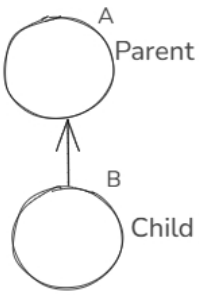

Child.java
Parent.java
Example2.java
Reports
1 package inheritance;
2
3 public class Parent {
4 //non-static method
5 public void demo() {
6 System.out.println("Hello from parent");
7 }
8
9 }
10


Child.java
Parent.java
Example2.java
1 package inheritance;
2
3 public class Child extends Parent {
4
5 }
6

Loading process with Inheritance :




-->First Parent class to be Loaded, then Child class
Is Loaded
-->if we are using members of A , B is not required
-->if we are using ,members of B , A is must required

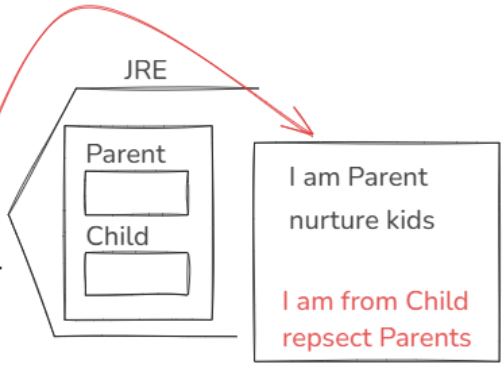



--> if we want to access members of
class B first it loads the parent class
then child class gets loaded

```
class Parent{
static {
Sopln("I am from Parent");
}
static void nurture( ){
Sopln("nurture kids");
}
}
```

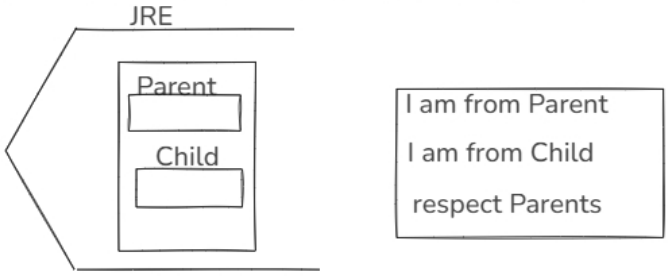
Case 1 :

```
class Case1{
p s v m(S[] args){
Parent.nurture( );
Child.respect( );
}
}
```




```
class child extends Parent{
static{
Sopln("I am from Child");
}
static void respect(){
Sopln("respect parents");
}
}
```

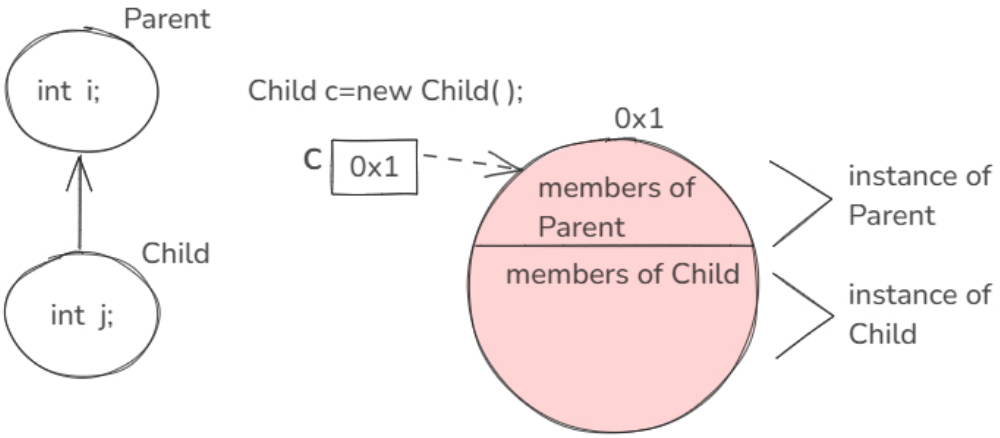
```
class Case2{
p s v m(S[] args){
Child.respect( );
}
}
```



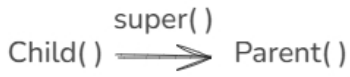

When a Child class will be Loaded

- if parent class exist in the memory → only Child class is Loaded
- if parent class doesn't exist in the memory → Parent class is loaded first, then Child class is Loaded

Loading Process of an Object with respect to Inheritance :




* Parent class non-static members are Loaded by Constructor
-->create Child( ) class Constructor , then it will be created and call the super class members also.




-->This super( ) statement will be added in all the Constrcutors which doesnot have this( ) statements
-->super( ) is used to call the constructor of Parent Class

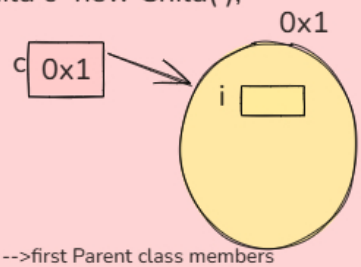
```
class Parent{
int i;
Parent(){
Sopln("From Parent");
}
}
```

```
class Parent{
int i;
Parent(){
super( );
Sopln("From Parent");
}
}
```

```
class Child extends Parent{
int j;
Child( ){
Sopln("From Child");
}
}
```

```
class Child extends Parent{
int j;
Child( ){
super( );
Sopln("From Child");
}
}
```

Child c=new Child( );




-->first Parent class members get Loaded
-->Child class memebrs get Loaded
-->with the help of child reference we can use non-static members of parent and child

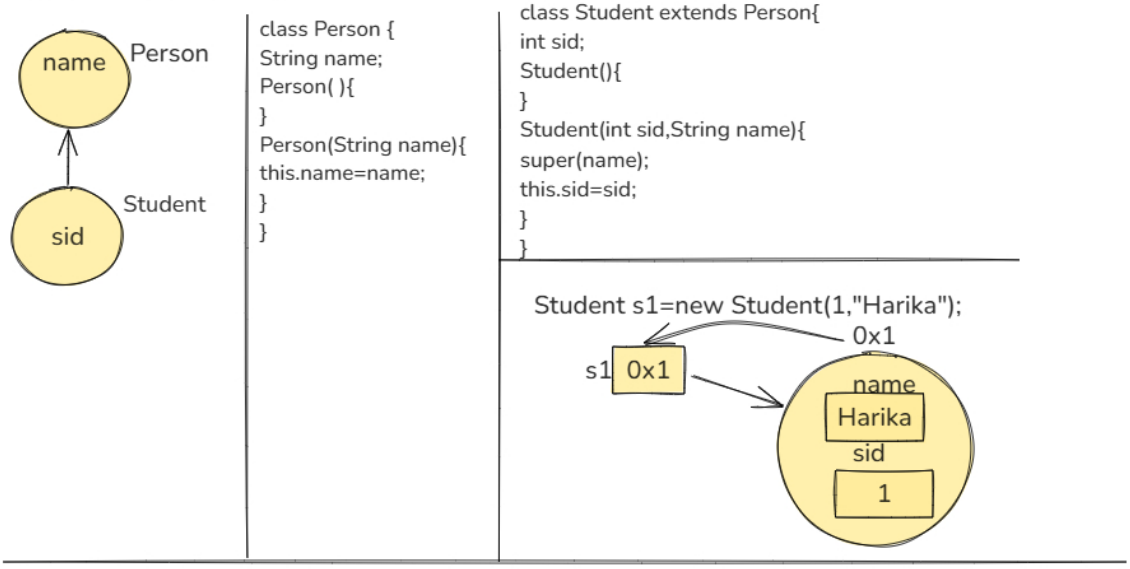
-->Whenever we create an object of subclass, the subclass Constructor will call the superclass constructor to load the non-static members of the superclass Inside the subclass object

super( ) :

Purpose of super( ):

1)it is used to load non-static members of Parent class into the Child class Object

2)it is used to initialize the data members of ParentClass via SubClass Constructor




Characteristics of super( ) Statement :

1)super( ) can be written only inside the Constructor Body

```
Public class Example{
Example( ){
super( );
}
void demo( ){
super( );
}
```

2)super( ) Statement should be the first Instruction inside the Constructor Body

```
public class Demo{
Demo( ){
Sopln("Hai");
super( );
}
```

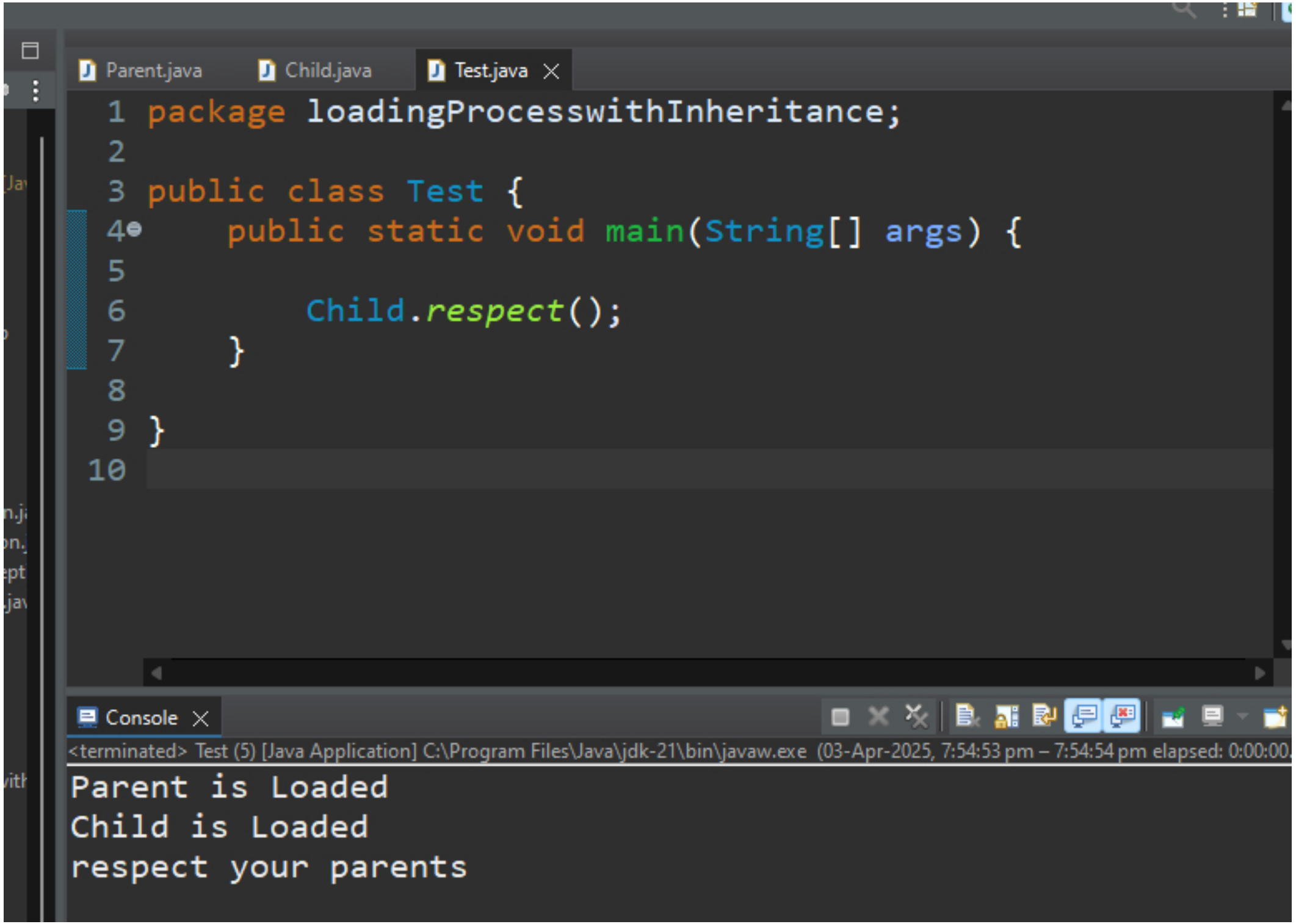
Constructor call must be the 1st Instruction inside the Constructor body

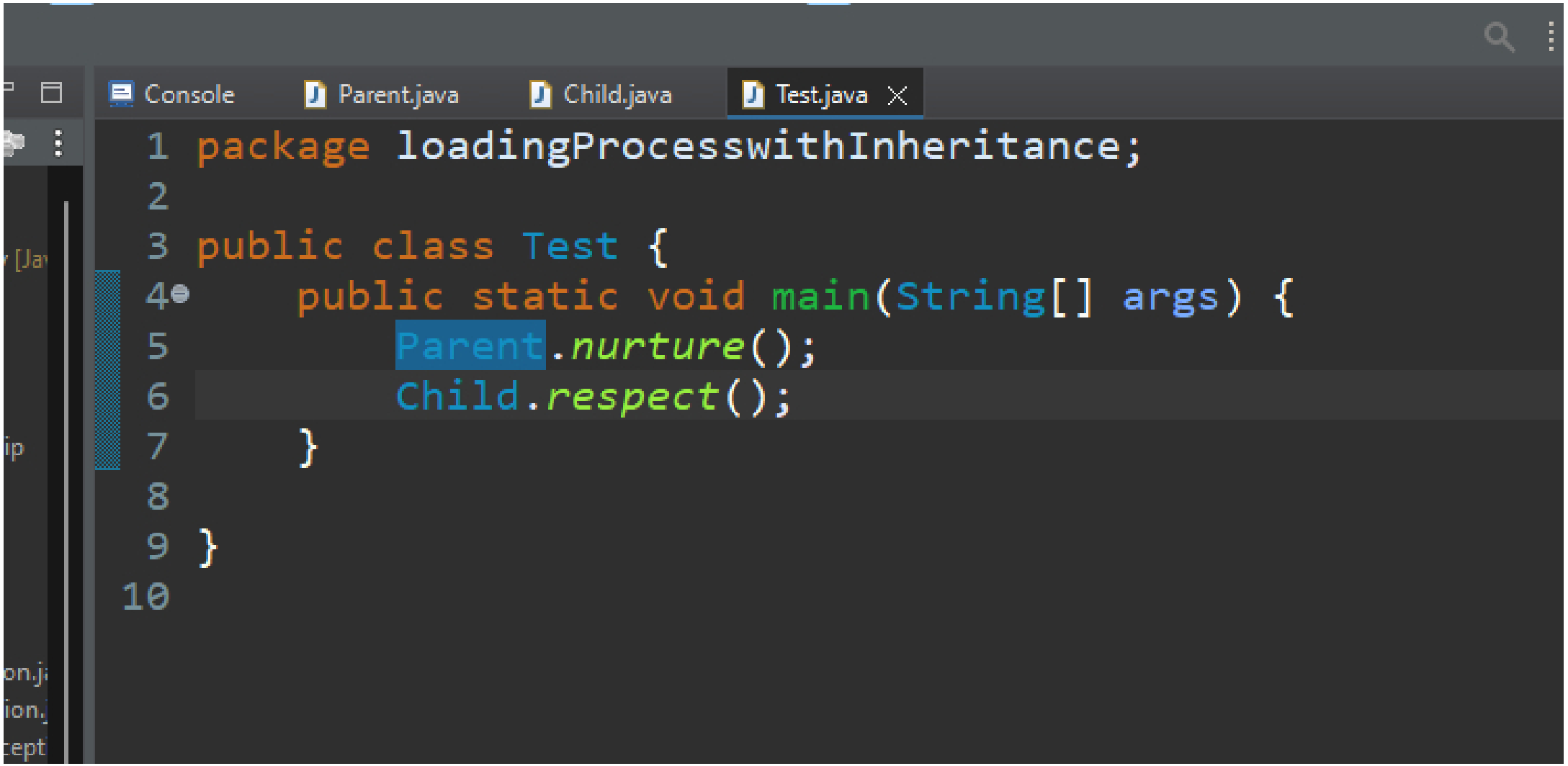
3) we cannot use super( ) , if this( ) is used

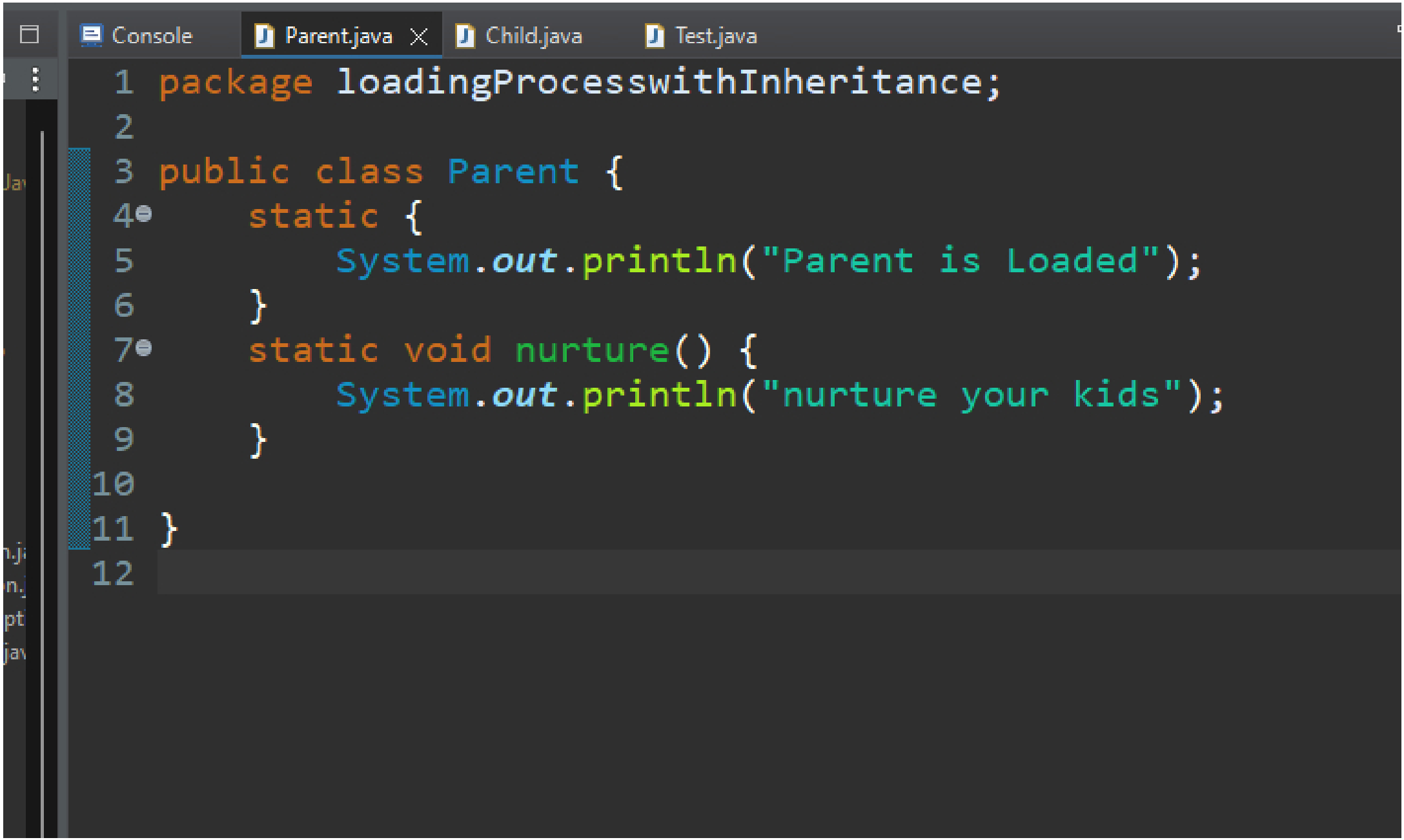
```
public class Test{
Test( ){
}
Test( int i){
this( );
super( );
}
```

super( );
this( );

4) we can pass the arguments in super call statement, it calls the Parameterized Constructor of super class

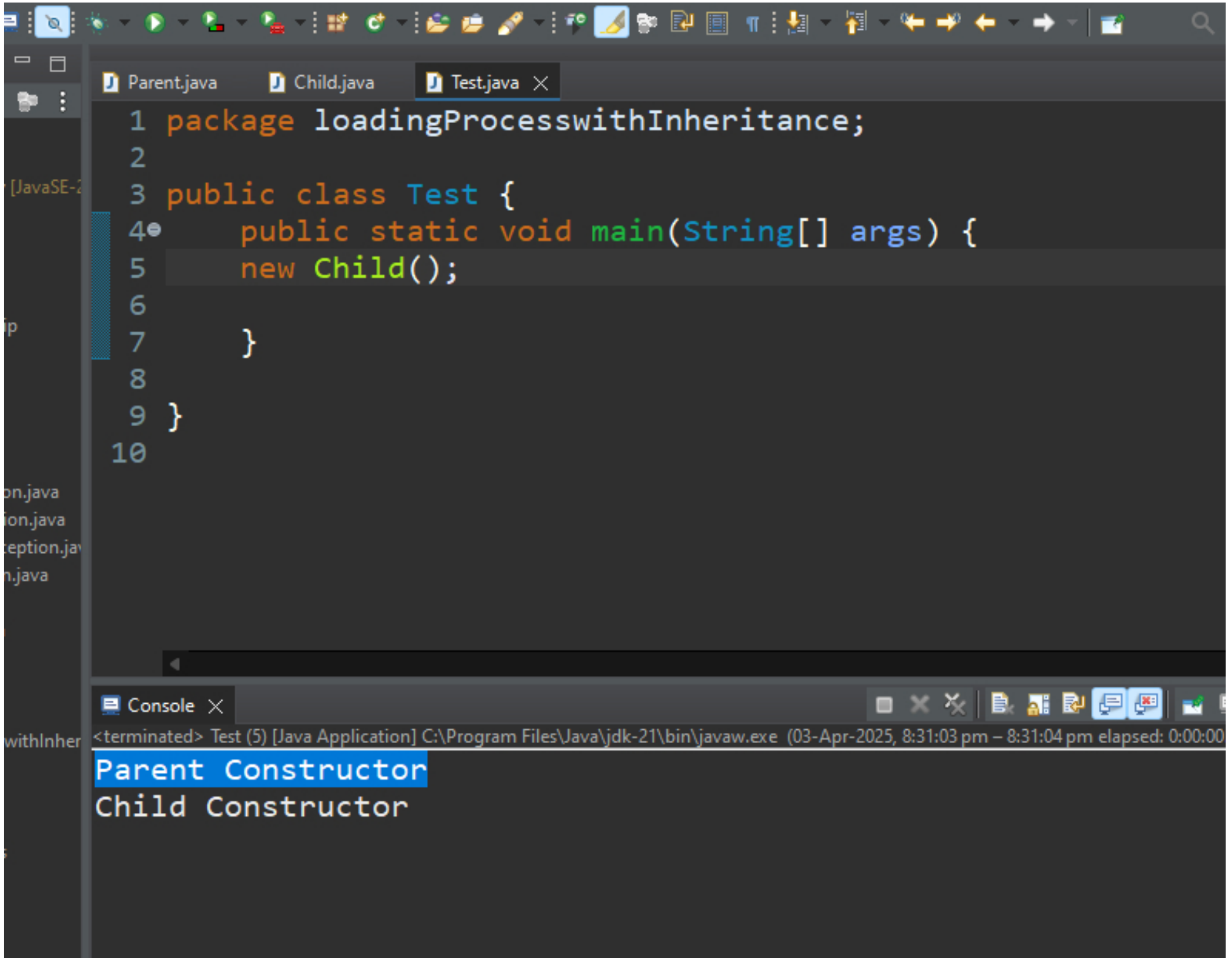

Parent.java
Child.java
Test.java
1 package loadingProcesswithInheritance;
2
3 public class Test {
4 public static void main(String[] args) {
5
6 Child.respect();
7 }
8
9 }
10
Console
<terminated> Test (5) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Apr-2025, 7:54:53 pm – 7:54:54 pm elapsed: 0:00:00.
Parent is Loaded
Child is Loaded
respect your parents

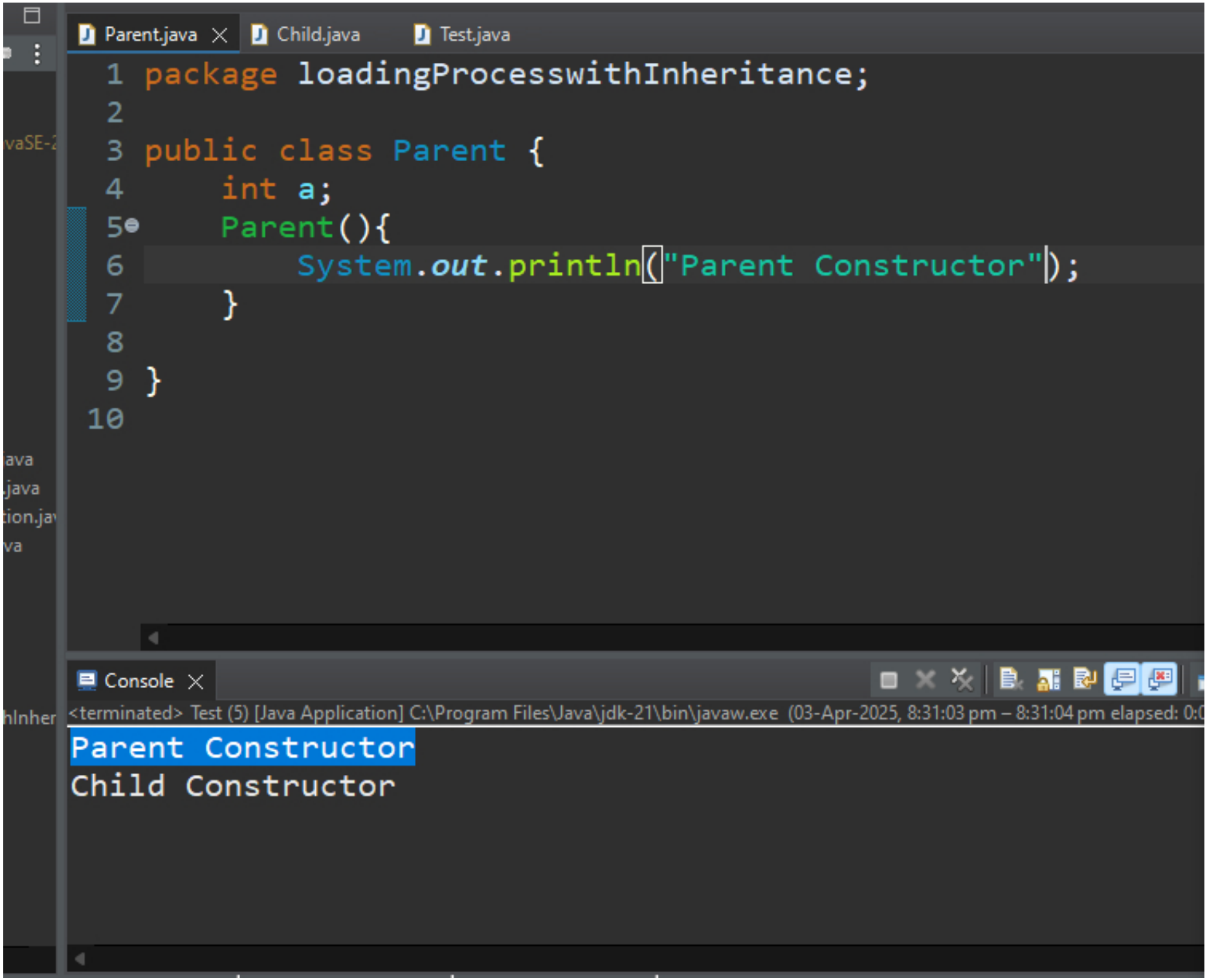

Console
Parent.java
Child.java
Test.java
1 package loadingProcesswithInheritance;
2
3 public class Test {
4 public static void main(String[] args) {
5 Parent.nurture();
6 Child.respect();
7 }
8
9 }
10

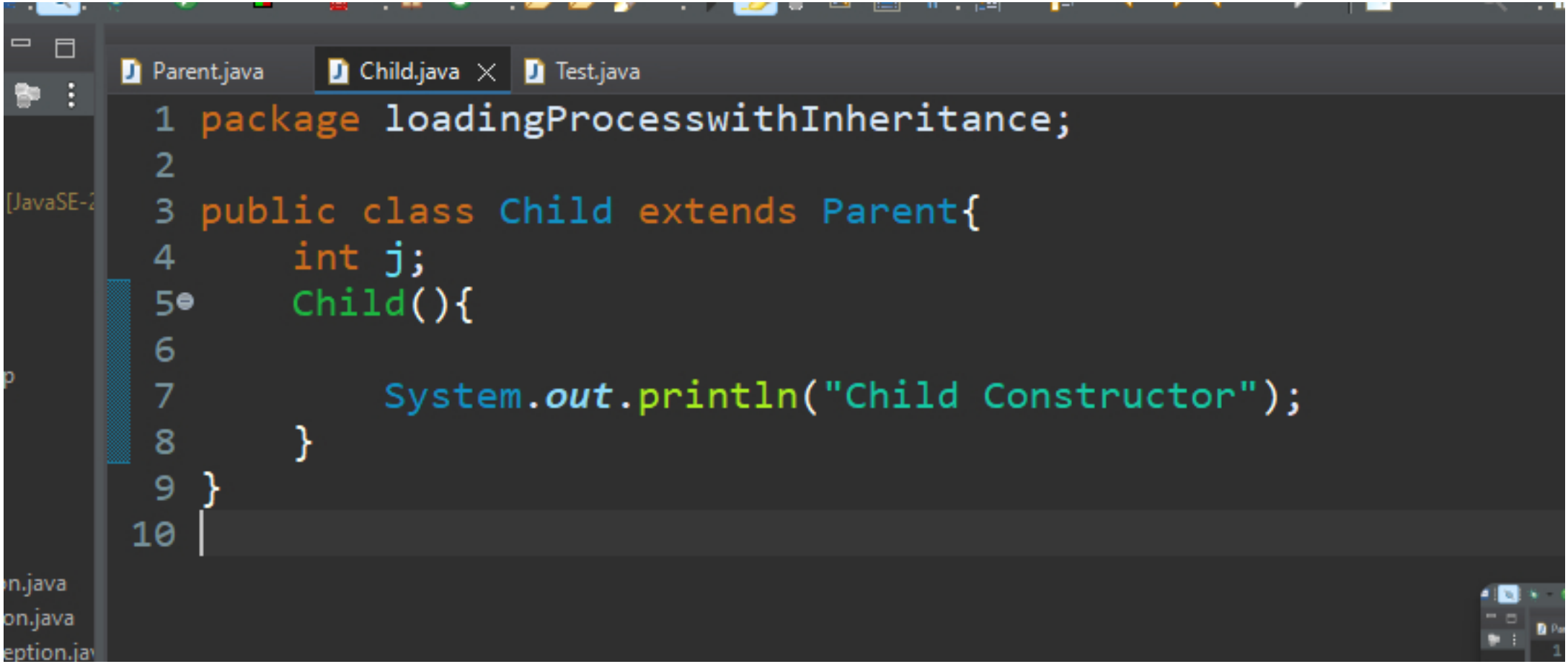

Console
Parent.java
Child.java
Test.java
1 package loadingProcesswithInheritance;
2
3 public class Parent {
4 static {
5 System.out.println("Parent is Loaded");
6 }
7 static void nurture() {
8 System.out.println("nurture your kids");
9 }
10
11 }
12

Console | Parent.java | Child.java | Test.java

```java
package loadingProcesswithInheritance;

public class Child extends Parent{
    static {
        System.out.println("Child is Loaded");
    }
    static void respect() {
        System.out.println("respect your parents");
    }

}

```


Parent.java
Child.java
Test.java
1 package loadingProcesswithInheritance;
2
3 public class Test {
4 public static void main(String[] args) {
5 new Child();
6
7 }
8
9 }
10
Console
<terminated> Test (5) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Apr-2025, 8:31:03 pm – 8:31:04 pm elapsed: 0:00:00
Parent Constructor
Child Constructor


Parent.java
Child.java
Test.java
1 package loadingProcesswithInheritance;
2
3 public class Parent {
4 int a;
5 Parent(){
6 System.out.println("Parent Constructor");
7 }
8
9 }
10
Console
<terminated> Test (5) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Apr-2025, 8:31:03 pm – 8:31:04 pm elapsed: 0:0
Parent Constructor
Child Constructor


Parent.java
Child.java
Test.java
1 package loadingProcesswithInheritance;
2
3 public class Child extends Parent{
4 int j;
5 Child(){
6
7 System.out.println("Child Constructor");
8 }
9 }
10

super keyword :

```
class Color{
String name="Blue";
}
class Black extends Color{
String name="Black";
 void demo(){
Sopln(name);
Sopln(super.name);
}
}
```

```
class Driver{
p s v m (s[] args){
Black b1=new Black();
b1.demo( ); //Black
}
}
```

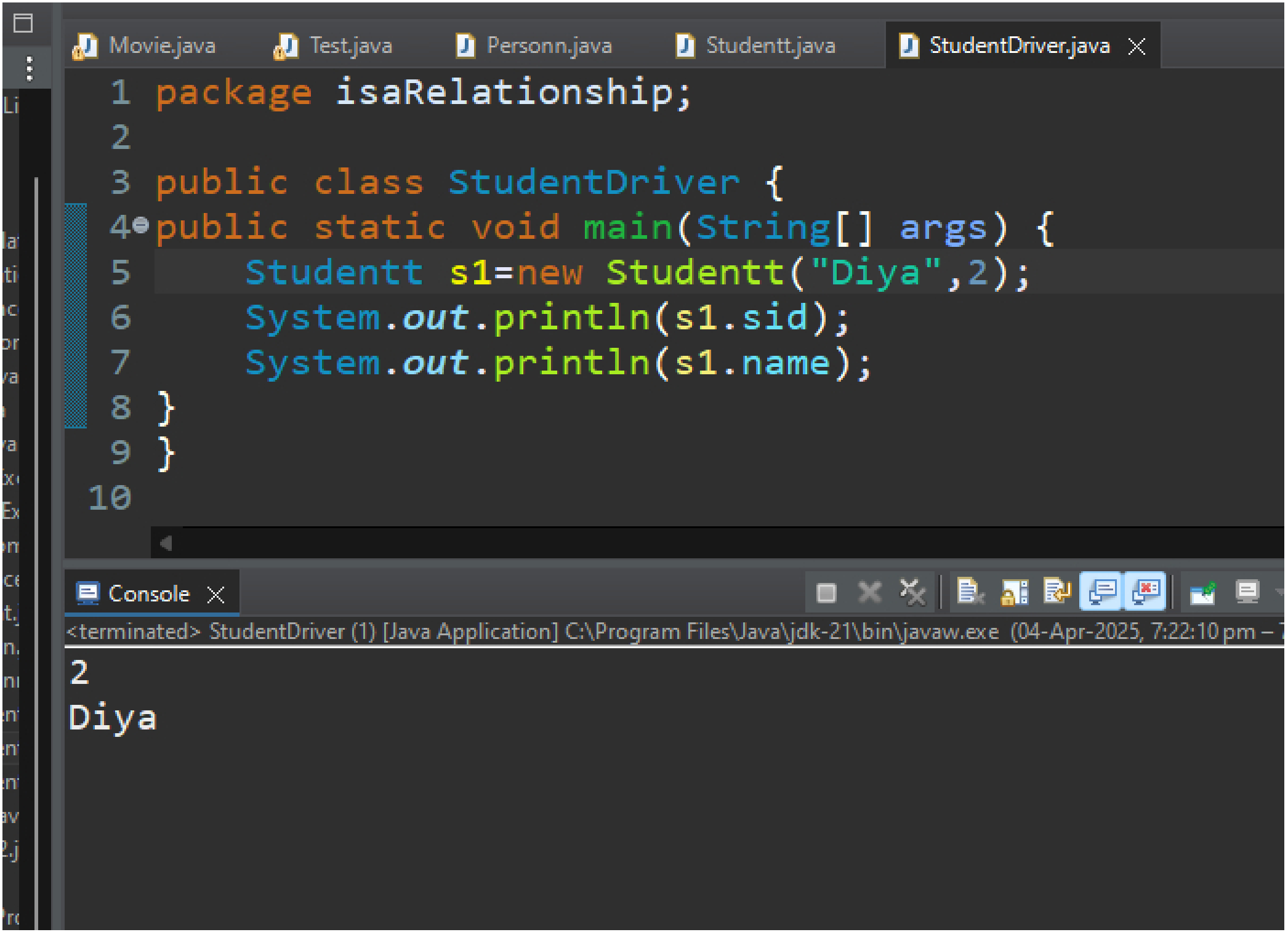
-->super : it is a keyword which is used to access static and non-static members of the parent class from the child class non-static area(Context);

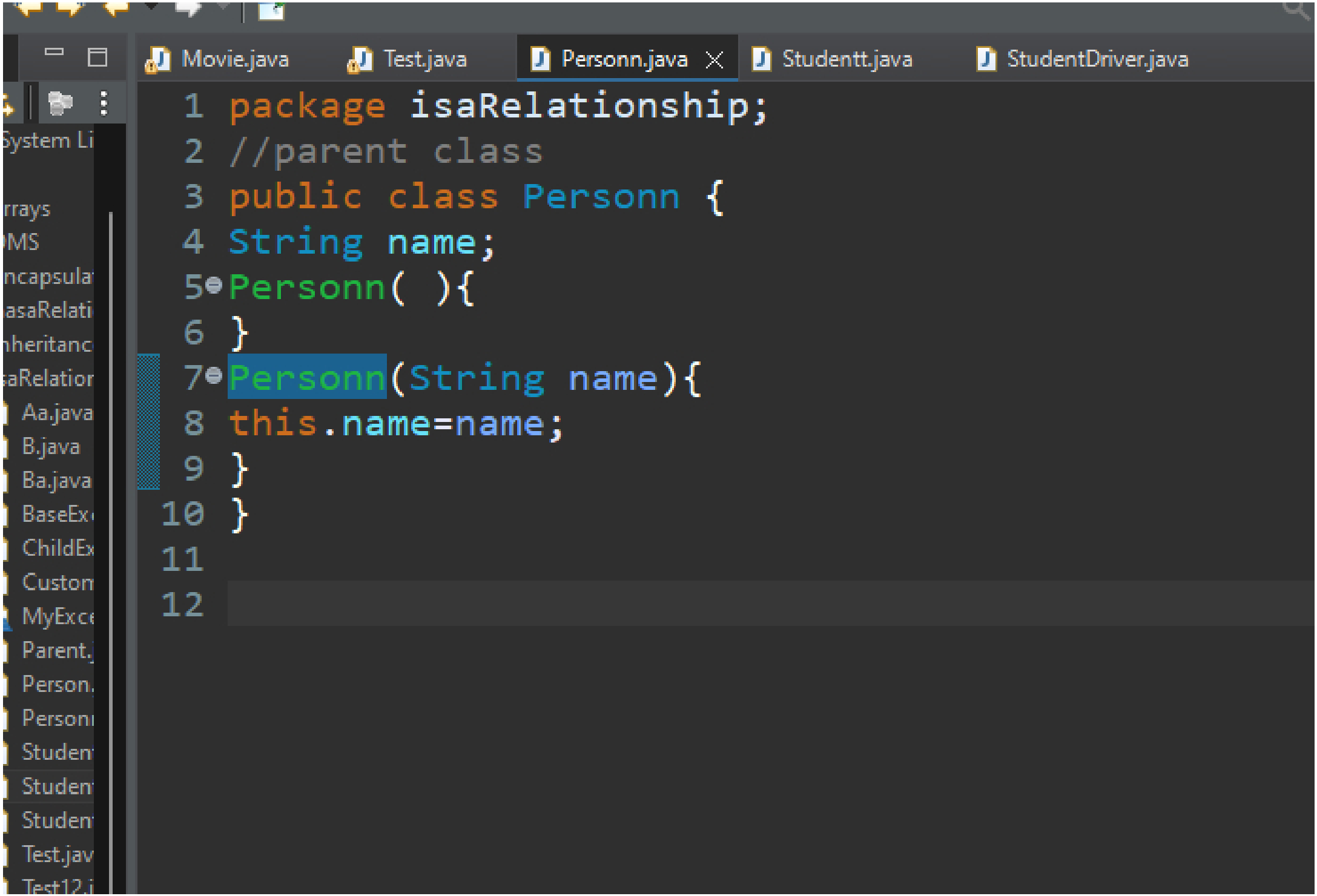
Variable Shadowing :

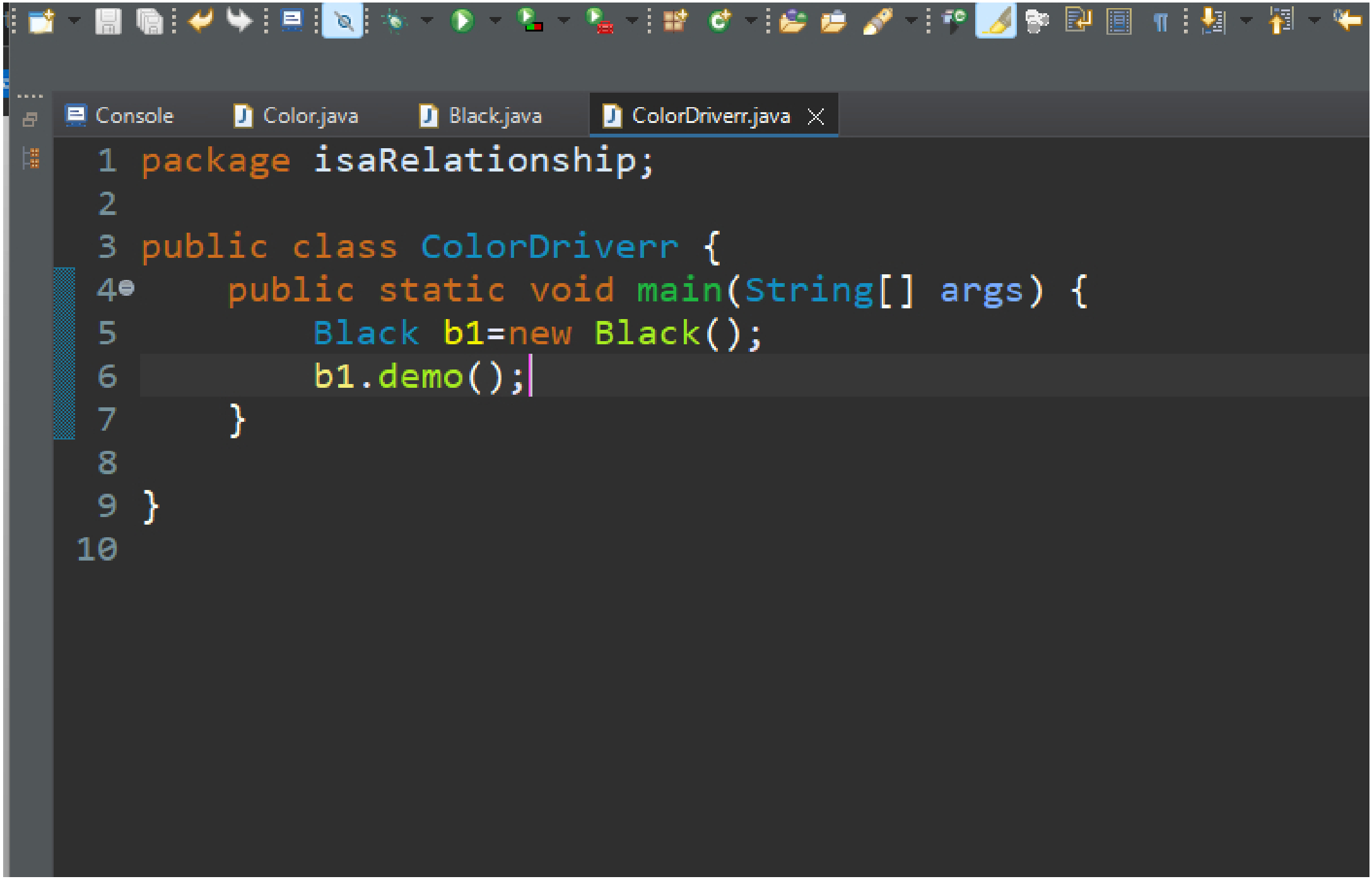
When the super class and sub class are having variables with same name is known as variable shadowing

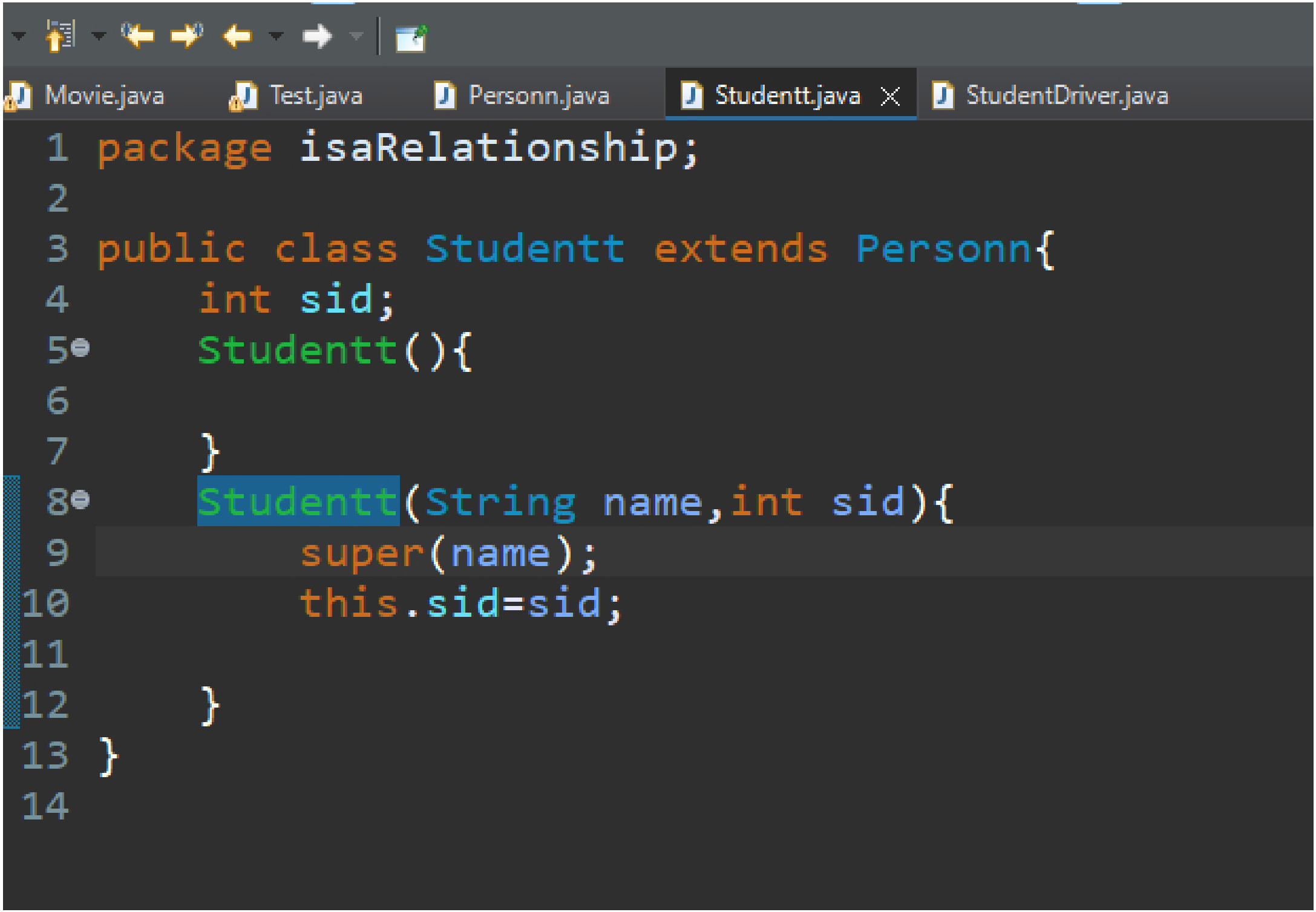
| super( ) | super |
|---|---|
| 1.It is used to cal / invoke the constructor of parent class from sub class<br>2.it can be used only inside Constructor body<br>3.it must be the first statement of the constructor | 1.it is used to access variables and methods of parent class from sub class<br>2.it can be used inside any non-static context<br>3. itcan be used anywhere inside the non-static context |

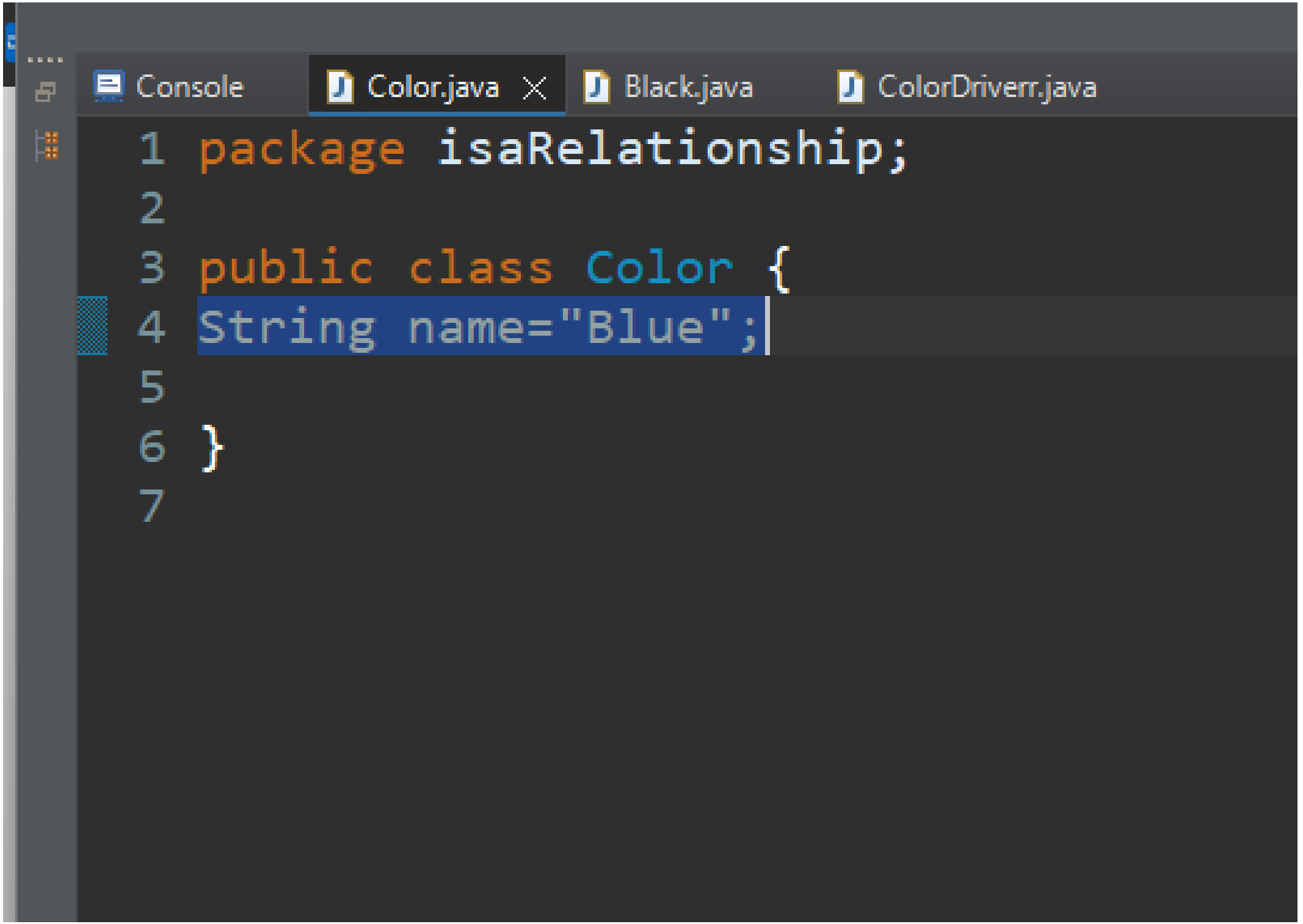
| super( ) | this( ) |
|---|---|
| 1.it is used to cal/invoke the parent class Constructor<br>2. Can be added in all the Constructors of a class<br>3.compiler will implicitly add no-arg super( ) to the constrcutors which doesn't this( ) | 1.it is used to cal/invoke the Constructor of same class<br>2.Can be added only n-1 Constructors of a class<br>3.compiler doesn't add this( ) implicitly |

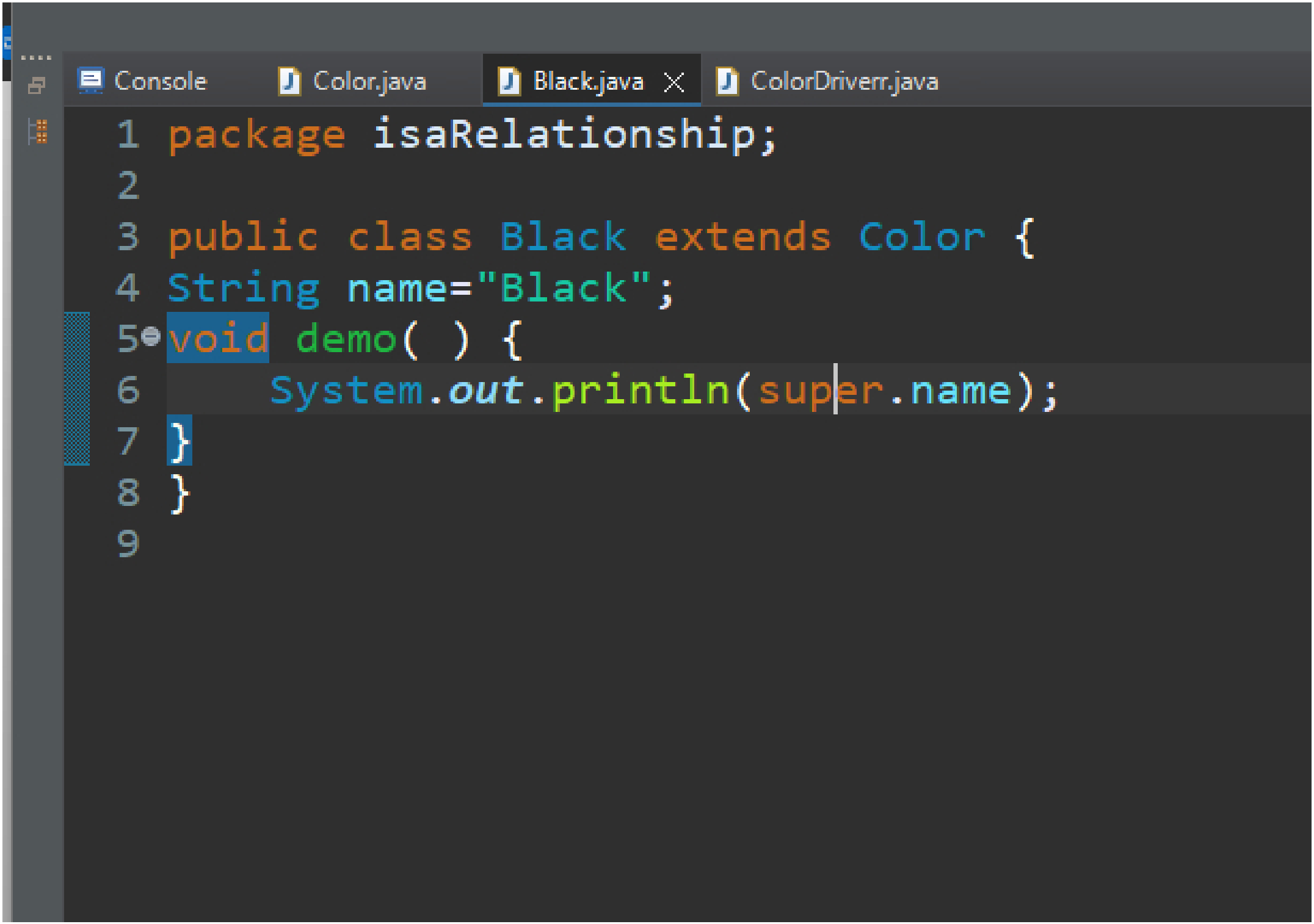

Movie.java
Test.java
Personn.java
Studentt.java
StudentDriver.java
1 package isaRelationship;
2
3 public class StudentDriver {
4 public static void main(String[] args) {
5 Studentt s1=new Studentt("Diya",2);
6 System.out.println(s1.sid);
7 System.out.println(s1.name);
8 }
9 }
10
Console
<terminated> StudentDriver (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (04-Apr-2025, 7:22:10 pm –
2
Diya


Movie.java
Test.java
Personn.java
Studentt.java
StudentDriver.java
1 package isaRelationship;
2 //parent class
3 public class Personn {
4 String name;
5 Personn( ){
6 }
7 Personn(String name){
8 this.name=name;
9 }
10 }
11
12


Console
Color.java
Black.java
ColorDriverr.java
1 package isaRelationship;
2
3 public class ColorDriverr {
4 public static void main(String[] args) {
5 Black b1=new Black();
6 b1.demo();
7 }
8
9 }
10

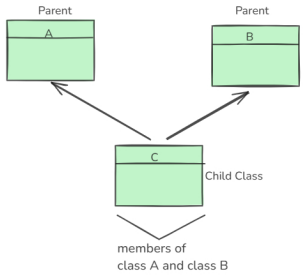

Movie.java
Test.java
Personn.java
Studentt.java
StudentDriver.java
1 package isaRelationship;
2
3 public class Studentt extends Personn{
4 int sid;
5 Studentt(){
6
7 }
8 Studentt(String name,int sid){
9 super(name);
10 this.sid=sid;
11
12 }
13 }
14


Console
Color.java
Black.java
ColorDriverr.java
1 package isaRelationship;
2
3 public class Color {
4 String name="Blue";
5
6 }
7

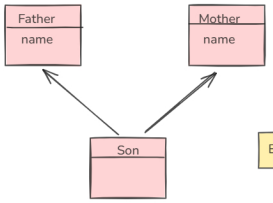

Console
Color.java
Black.java
ColorDriverr.java
1 package isaRelationship;
2
3 public class Black extends Color {
4 String name="Black";
5 void demo( ) {
6 System.out.println(super.name);
7 }
8 }
9

## 4.Multiple Inheritance

If a Child class is Having Multiple Parent (Super class) in the same level
Is known as Multiple Inheritance




Ex :




```
class Father{
}
class Mother{
}
class Child extends Father
                    or
                    Mother
```

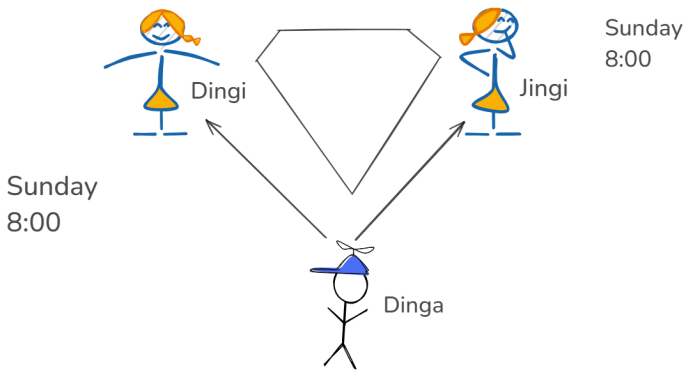
Extending both is restricted

Multiple Inheirtance has a Problem Known as "Diamond Problem"
Because of this Diamod Problem , we can't achieve Multiple Inheritance In Java Using classes

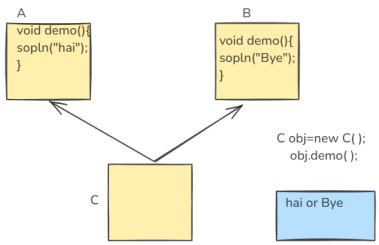
-->A class can Inherit only 1 Parent Class

-->We can achieve this Multiple Inheritance by Interface

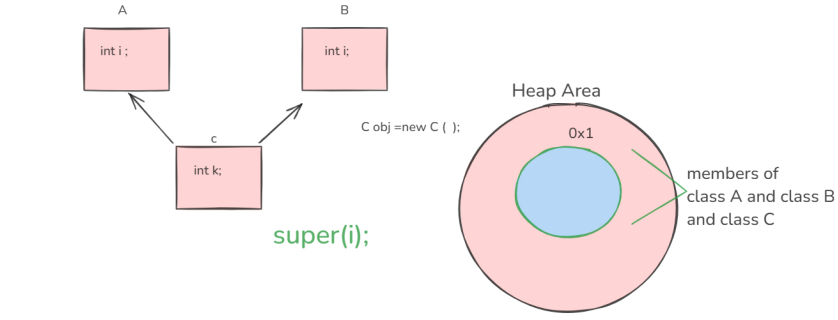
## DIAMOND PROBLEM :




Ex :

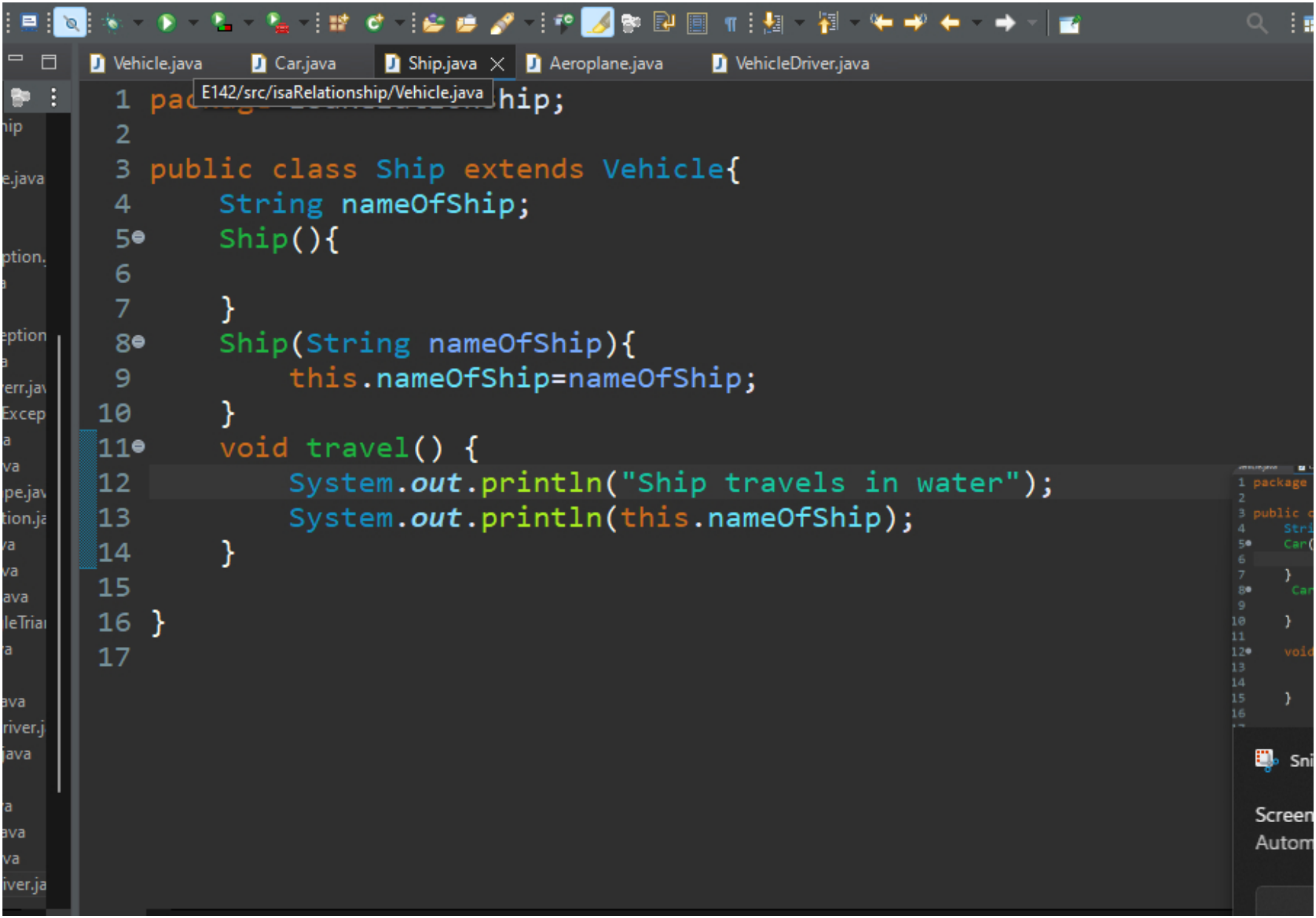



Ex :



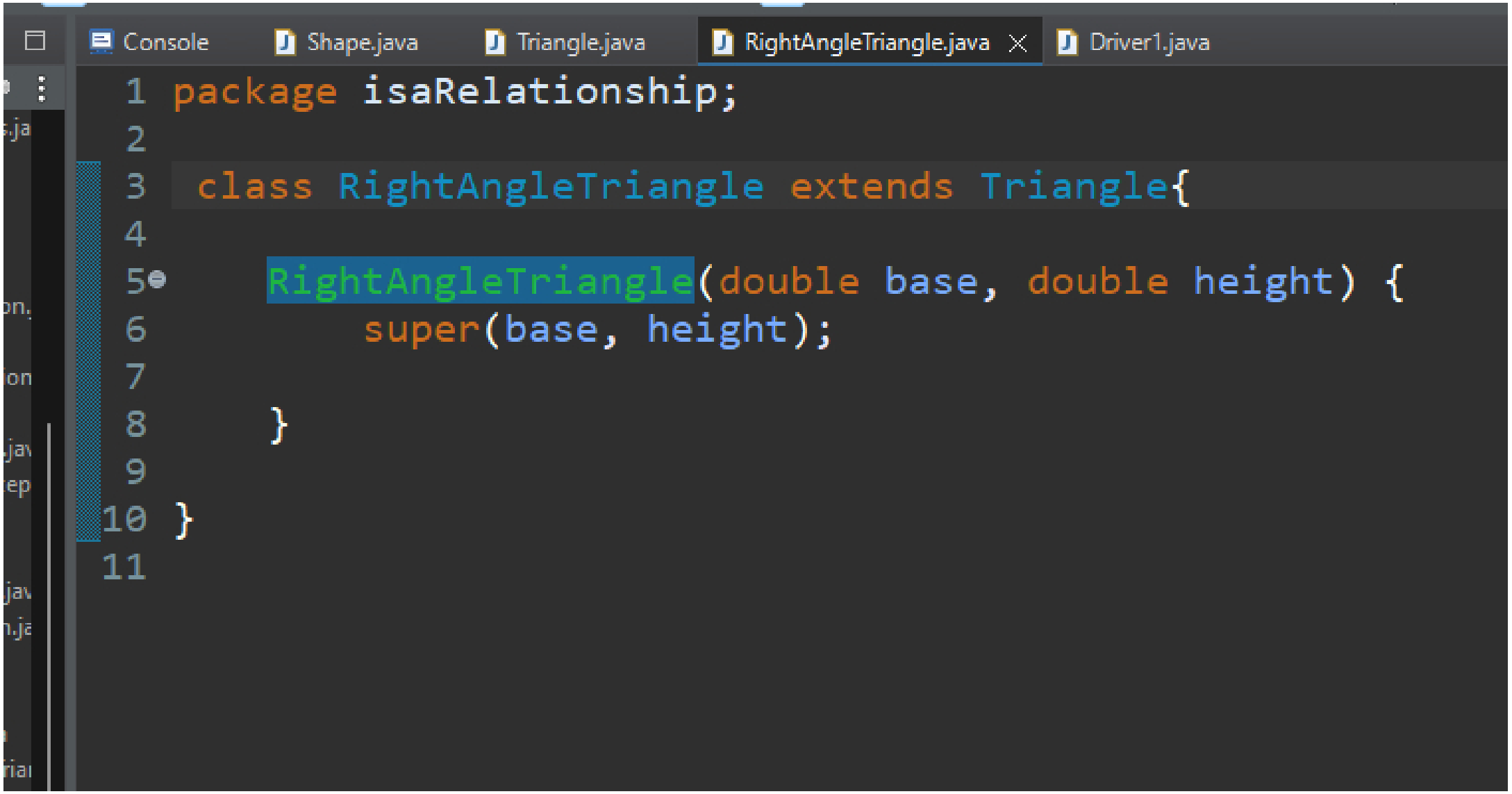

Diamond Problem or Constructor Chaining Problem :

Whenever we represent Multiple Inheirtance using classes , The compiler gets confused to perfrom the Compile Time Binding between super( ) statement of the sub class Constructor and the Constructors of the super classes present in the same level.
This Ambiguity arises for the Compiler is known as Diamond Problem or Constructor Chaining Problem


Vehicle.java
Car.java
Ship.java
Aeroplane.java
VehicleDriver.java
E142/src/isaRelationship/Vehicle.java
1 pa hip;
2
3 public class Ship extends Vehicle{
4 String nameOfShip;
5 Ship(){
6
7 }
8 Ship(String nameOfShip){
9 this.nameOfShip=nameOfShip;
10 }
11 void travel() {
12 System.out.println("Ship travels in water");
13 System.out.println(this.nameOfShip);
14 }
15
16 }
17

Console | Shape.java | Triangle.java | RightAngleTriangle.java | Driver1.java

```java
package isaRelationship;

public class Shape {
    double area;
    void computeArea() {
        System.out.println("it calculates Area");
    }

}

```


Console
Shape.java
Triangle.java
RightAngleTriangle.java
Driver1.java
1 package isaRelationship;
2
3 class RightAngleTriangle extends Triangle{
4
5 RightAngleTriangle(double base, double height) {
6 super(base, height);
7
8 }
9
10 }
11

```java
package isaRelationship;

public class Car extends Vehicle {
    String brandName;
    Car(){

    }
     Car(String brandName) {
        this.brandName = brandName;
    }

    void display() {
        System.out.println("Car has 4 wheels");
        System.out.println(this.brandName);
    }


}

```

```java
package isARelationship;

public class Vehicle {
    String type;
    void modeOfTransport() {
        System.out.println("Every vehicle has it's own way of Transport");
    }

}

```

Shape.java | Triangle.java | RightAngleTriangle.java | Driver1.java

```
2
3 public class Driver1 {
4     public static void main(String[] args) {
5         RightAngleTriangle t1=new RightAngleTriangle(3,4);
6         System.out.println(t1.area);
7         System.out.println(t1.base);
8         System.out.println(t1.height);
9         t1.computeArea();
10        t1.display();
11    }
12
```

Console

<terminated> Driver1 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (05-Apr-2025, 7:36:18 pm – 7:36:18 pm elapsed: 0:00:00.346) [pid: 8656]

```
0.0
3.0
4.0
it calculates Area
4.0
3.0
```

Vehicle.java | Car.java | Ship.java | Aeroplane.java | VehicleDriver.java

E142/src/isaRelationship/Aeroplane.java

```java
package isaRelationship;

public class VehicleDriver {
    public static void main(String[] args) {
        Car c1=new Car("Ferrai");
        c1.display();
//      System.out.println(c1.brandName);
//      c1.type="Car";
//      System.out.println(c1.type);
        Ship s1=new Ship("Titanic");
        s1.travel();
//      System.out.println(s1.nameOfShip);
//      s1.type="Ship";
//      System.out.println(s1.type);
        Aeroplane a1=new Aeroplane("indgo");
        a1.wing();
//      System.out.println(a1.brandName);
//      a1.type="Plane";
//      System.out.println(a1.type);

    }

}

```

Vehicle.java | Car.java | Ship.java | Aeroplane.java | VehicleDriver.java

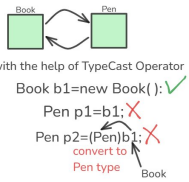
```java
package isaRelationship;

public class Aeroplane extends Vehicle {
    String brandName;
    Aeroplane(){

    }

        Aeroplane(String brandName) {
         this.brandName = brandName;
    }
    void wing() {
        System.out.println("i flies in the Skyyyyy");
        System.out.println(this.brandName);
    }

}
```

## Non-Primitive TypeCasting / Derived TypeCasting

The Process of Converting reference type from One-non-Primitive type into another non-Primitive type.

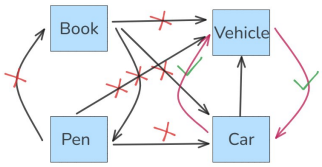
```
class Book {
}
class Pen{
}
```

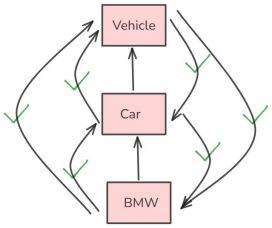
```
Book b;
Pen p;
b=new Book( );
p=new Pen( );
b=new Pen( );
```

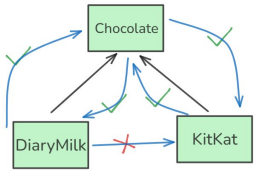


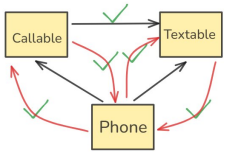

***
***

We Can Achieve non-Primitive TypeCasting only when there is Is-A-Relationship between them.



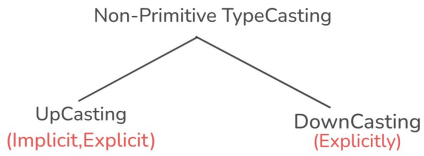



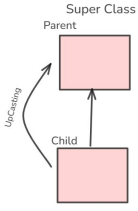

It is Possible to Convert the Reference of one parent type to another parent type only if they have same child

--> 2 Possible Cases
1) Is-A Relationship
2)Multiple Inheritance



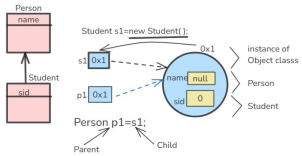

## UpCasting

The Process of Converting the reference of Sub Class type to Super Class type is known as UpCasting.




-->It can be done in 2 ways
1)Implicit
2)Explicit

Ex :

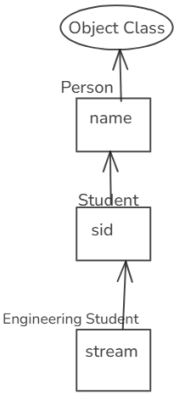



Child --> Parent (UpCasting)
Compiler implicits do UpCasting

-->s1 and p1 are pointing towards same object but both are not same

-->we can use all the members of Student in Person reference variable

UpCasted Scenario :



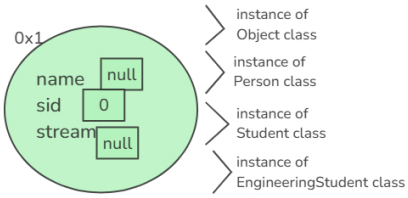

Student s1=new Student( );

Person p1=s1;

Person p2= new Student( ); → implicit

Person p3=(Person)new Student( );

new EngineeringStudent( );

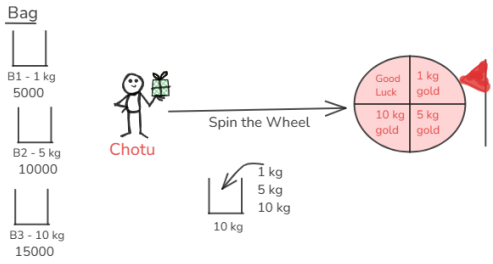


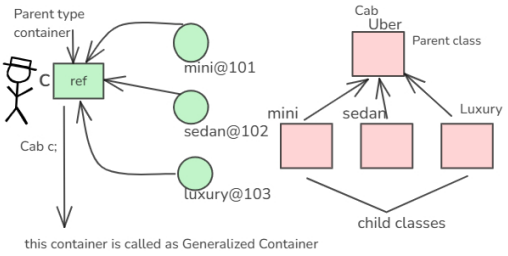

Purpose of UpCasting :

to achieve

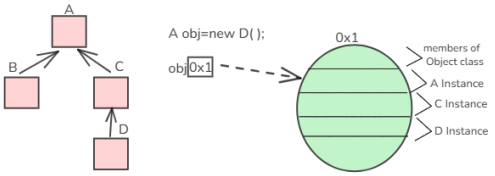
1.Generalization

2.Run Time Polymorphism

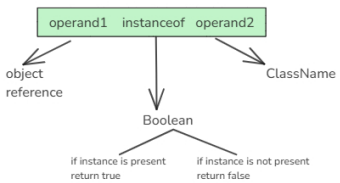



Instanceof Operator :



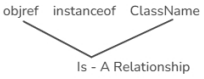

-->To check the Instance of specific class is Present inside the Object or not




```
class A {
}
class B extends A {

}
class C extends A {

}
class D extends C {
}
```

```
Object o =new D( );
S.o.pln( o instanceof Object);
S.o.pln( o instanceof A);
S.o.pln( o instanceof C);
S.o.pln( o instanceof D);
S.o.pln( o instanceof B); → false
```

Note :



Sedan.java | Test.java | Test2.java | Test3.java

```java
package nonprimitive;

public class Test3 {
    public static void displayReference(Object c) {
        System.out.println(c);
    }
    public static void main(String[] args) {
        displayReference(new Mini());
        displayReference(new Sedan());
        displayReference(new Luxury());
        displayReference(new Book());
    }

}

```

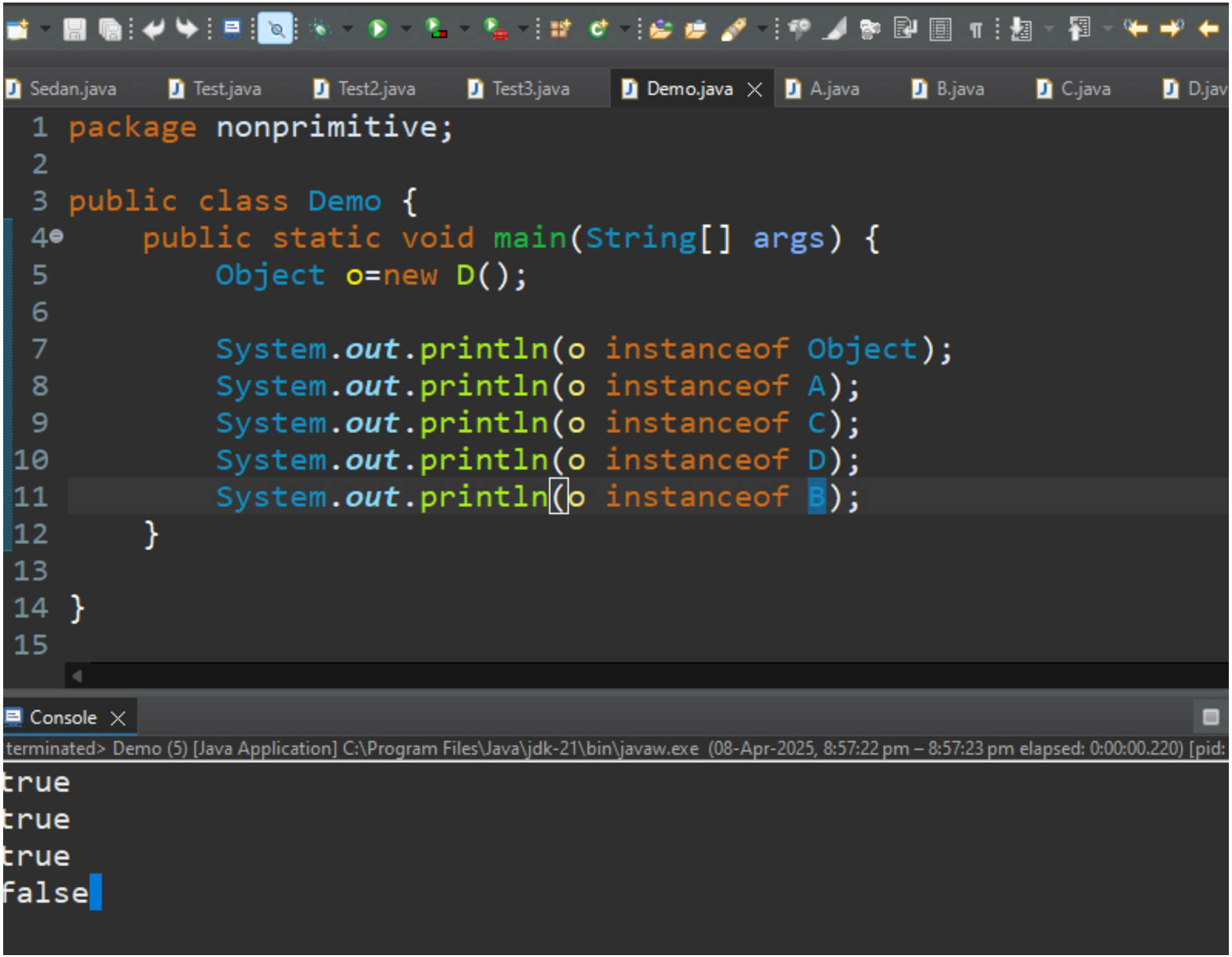
Console

terminated> Test3 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (08-Apr-2025, 8:31:58 pm – 8:31:58 pm elapsed: 0:00:00.276) [pid: 7140]

```
nonprimitive.Mini@2f92e0f4
nonprimitive.Sedan@5305068a
nonprimitive.Luxury@279f2327
nonprimitive.Book@54bedef2
```

Cab.java | Mini.java | Sedan.java | Luxury.java | Test.java | Te

```
2
3  import java.util.Scanner;
4
5  public class Test2 {
6      public static void main(String[] args) {
7          System.out.println("***Select a Cab***");
8          System.out.println("Press 1 for Mini");
9          System.out.println("Press 2 for Sedan");
10         System.out.println("Press 3 for Luxury");
11         System.out.println("Enter your Choice");
12         Scanner sc=new Scanner(System.in);
13         int choice=sc.nextInt();
14         Cab c;
15         switch(choice) {
16         case 1: {
17             c=new Mini();
18             System.out.println("you selected Mini");
19             System.out.println(c);
20
21         }
22         break;
23         case 2: {
24             c=new Sedan();
25             System.out.println("you selected Sedan");
26             System.out.println(c);
27         }
28         break;
29         case 3: {
30             c=new Luxury();
31             System.out.println("you selected Luxury");
32             System.out.println(c);
33         }
34         break;
35         default : System.out.println("invalid input");
36     }
37
38
39 }
```


Sedan.java
Test.java
Test2.java
Test3.java
Demo.java
A.java
B.java
C.java
D.jav
1 package nonprimitive;
2
3 public class Demo {
4 public static void main(String[] args) {
5 Object o=new D();
6
7 System.out.println(o instanceof Object);
8 System.out.println(o instanceof A);
9 System.out.println(o instanceof C);
10 System.out.println(o instanceof D);
11 System.out.println(o instanceof B);
12 }
13
14 }
15
Console
terminated> Demo (5) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (08-Apr-2025, 8:57:22 pm – 8:57:23 pm elapsed: 0:00:00.220) [pid:
true
true
true
false

Sedan.java | Test.java | Test2.java | Test3.java | Demo.java | A.java | B.java | C.java

```java
package nonprimitive;

public class Demo {
    public static void main(String[] args) {
        A a=new D();

        System.out.println(a instanceof Object);
        System.out.println(a instanceof A);
        System.out.println(a instanceof C);
        System.out.println(a instanceof D);
        System.out.println(a instanceof B);
    }

}

```

Console

erminated> Demo (5) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (08-Apr-2025, 9:00:17 pm – 9:00:18 pm elapsed: 0:00

```
rue
rue
rue
rue
alse
```

Luxury.java

```java
package nonprimitive;

public class Luxury extends Cab{

}

```

Sedan.java Test.java Test2.java

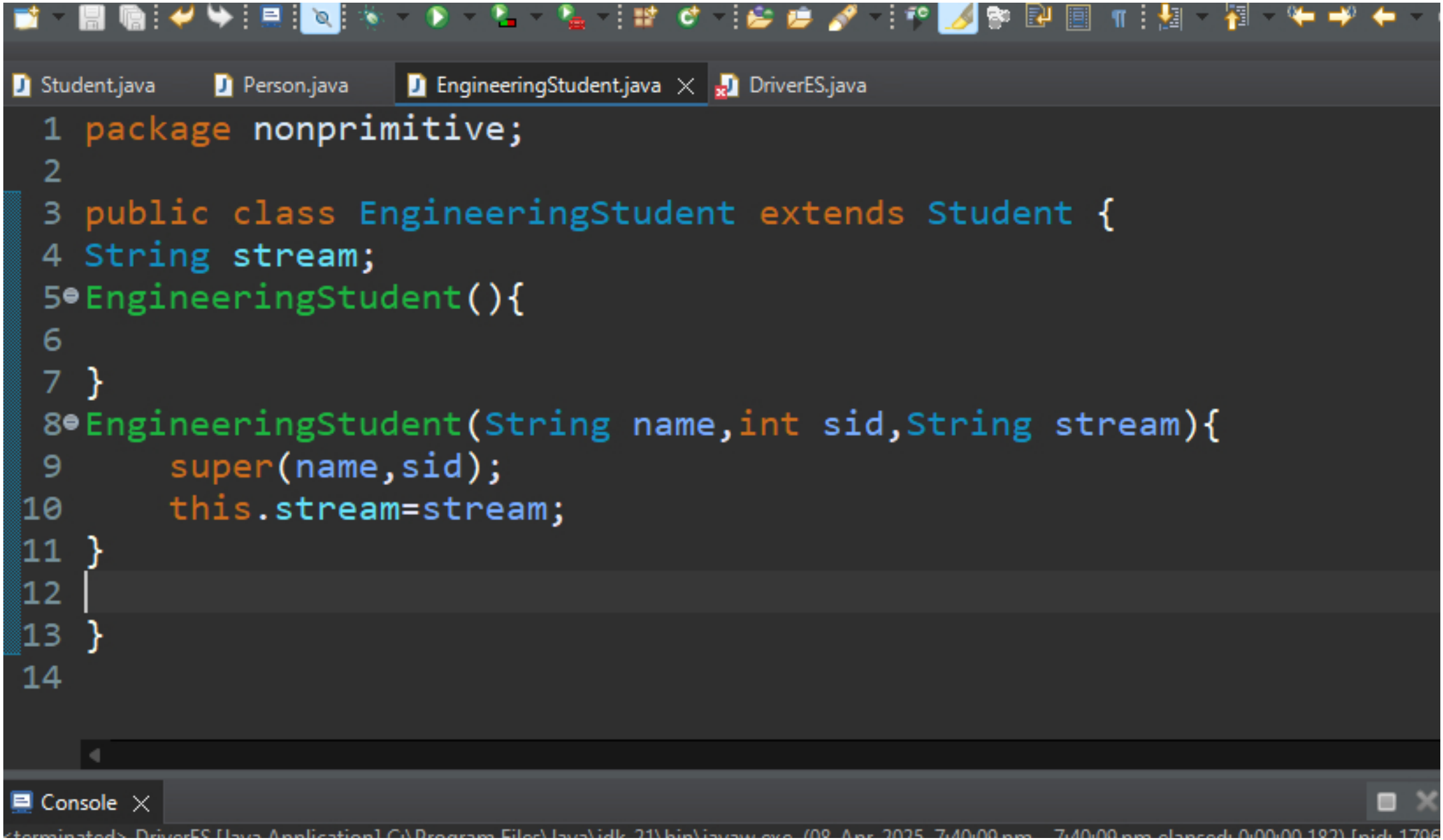
```java
package nonprimitive;

public class Sedan extends Cab {

}

```

Mini.java

```java
package nonprimitive;

public class Mini extends Cab{

}

```

Cab.java

```java
package nonprimitive;

public class Cab {

}

```


Student.java
Person.java
EngineeringStudent.java
DriverES.java
1 package nonprimitive;
2
3 public class EngineeringStudent extends Student {
4 String stream;
5 EngineeringStudent(){
6
7 }
8 EngineeringStudent(String name,int sid,String stream){
9 super(name,sid);
10 this.stream=stream;
11 }
12
13 }
14
Console

Student.java | Person.java | EngineeringStudent.java | DriverES.java

```java
package nonprimitive;

public class Person {
    String name;
    Person(){

    }
    Person(String name){
        this.name=name;
    }

}

```

Student.java | Person.java | EngineeringStudent.java | DriverES.java

```java
package nonprimitive;

public class DriverES {
    public static void main(String[] args) {
        EngineeringStudent s1=new EngineeringStudent();
        Student s2=new EngineeringStudent();
        Student s3=new EngineeringStudent();
        Object s4=new EngineeringStudent("Sanjay",1,"Mechanical");
        System.out.println(s4);
        //Book b1=new EngineeringStudent();//CTE
    }

}

```

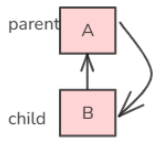
Console

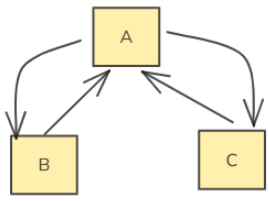
terminated> DriverES [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (08-Apr-2025, 7:39:03 pm – 7:39:03 pm elapsed: 0:00:00.326) [pid: 29460]

```
nonprimitive.EngineeringStudent@28a418fc
```

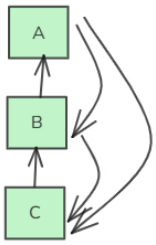
DownCasting :

The process of Converting parent type reference into child type reference is called as Downcasting



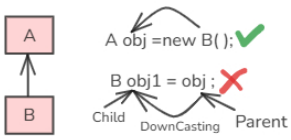




Hierarchical Inheritance




Multi-Level

-->The DownCasting is never done by the Compiler only it can be done explicitly by the Programmer.

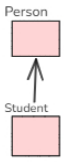



Syntax for DownCasting :

(Target Type) obj_ref

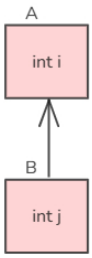
typecast operator

B obj1= (B)obj

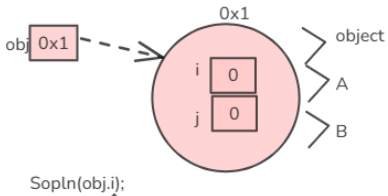



Person p=new Student( );//upcasting
Student s=p;//CTE
Student s=(Student)p;

Upcasted scenario :




A obj=new B( );




Sopln(obj.i);

CTS RTS

Sopln(obj.j);

CTS

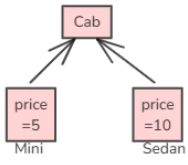
B obj2=(B)obj; → downacasting

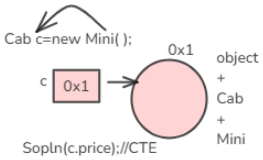
Sopln(obj2.j);

CTS RTS

ClassCastException :

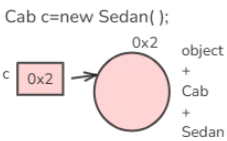
Whenever we perform the Downcasting without Performing Upcasting and trying to access the members with the help of subclass Reference variable then we get an Exception called as ClassCastException.




Sopln(c.price);//CTE

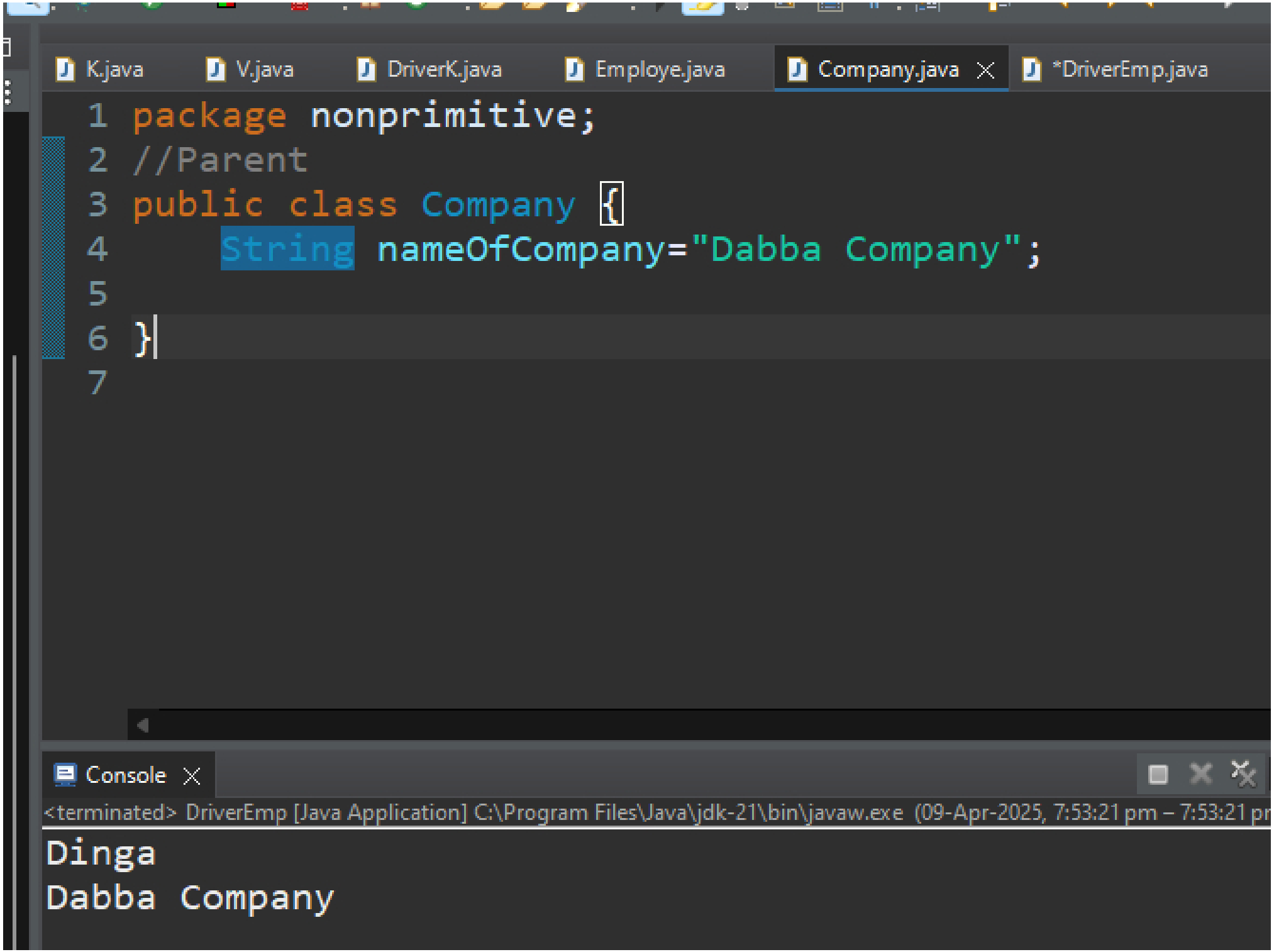
Sopln( ((Mini)c).price);

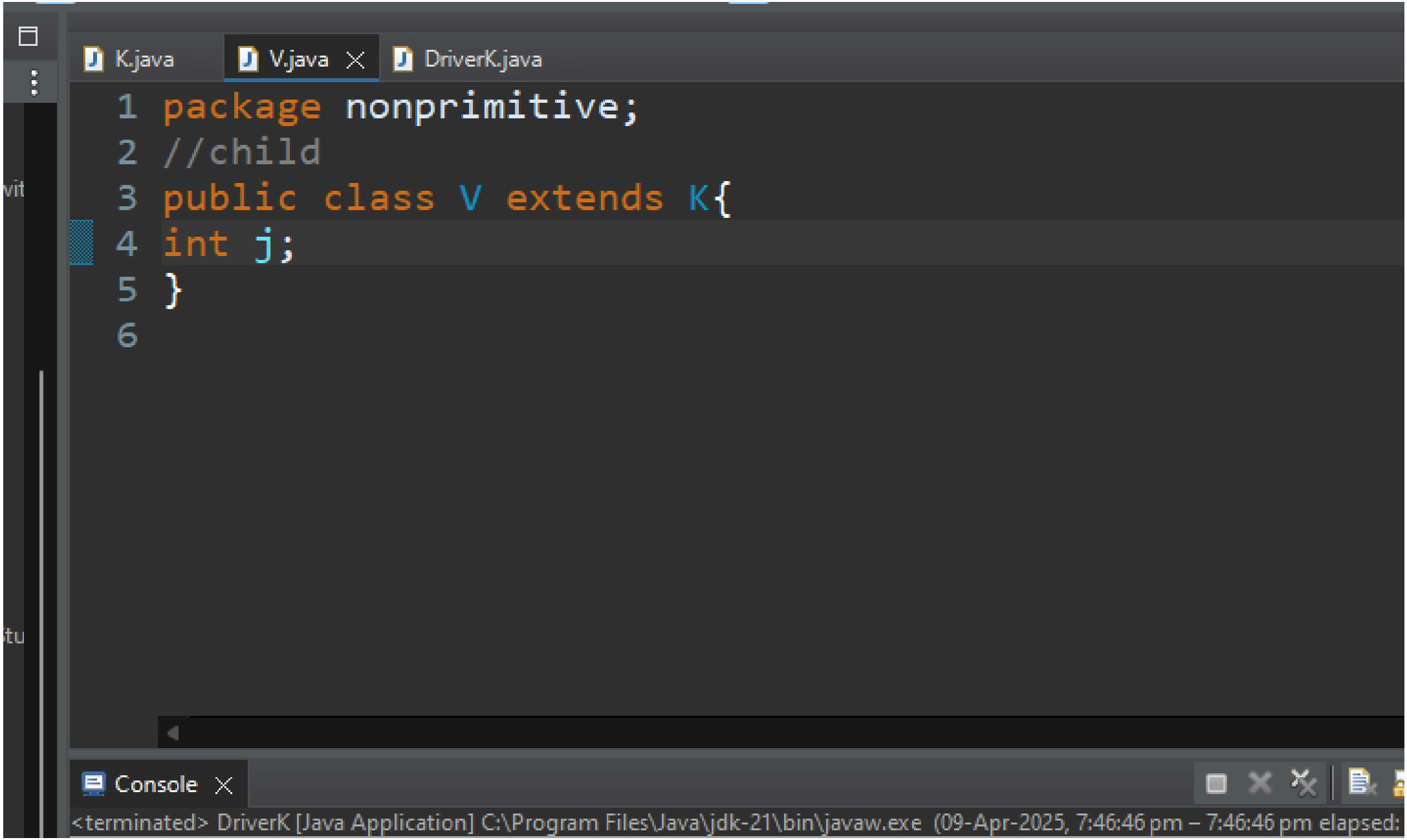


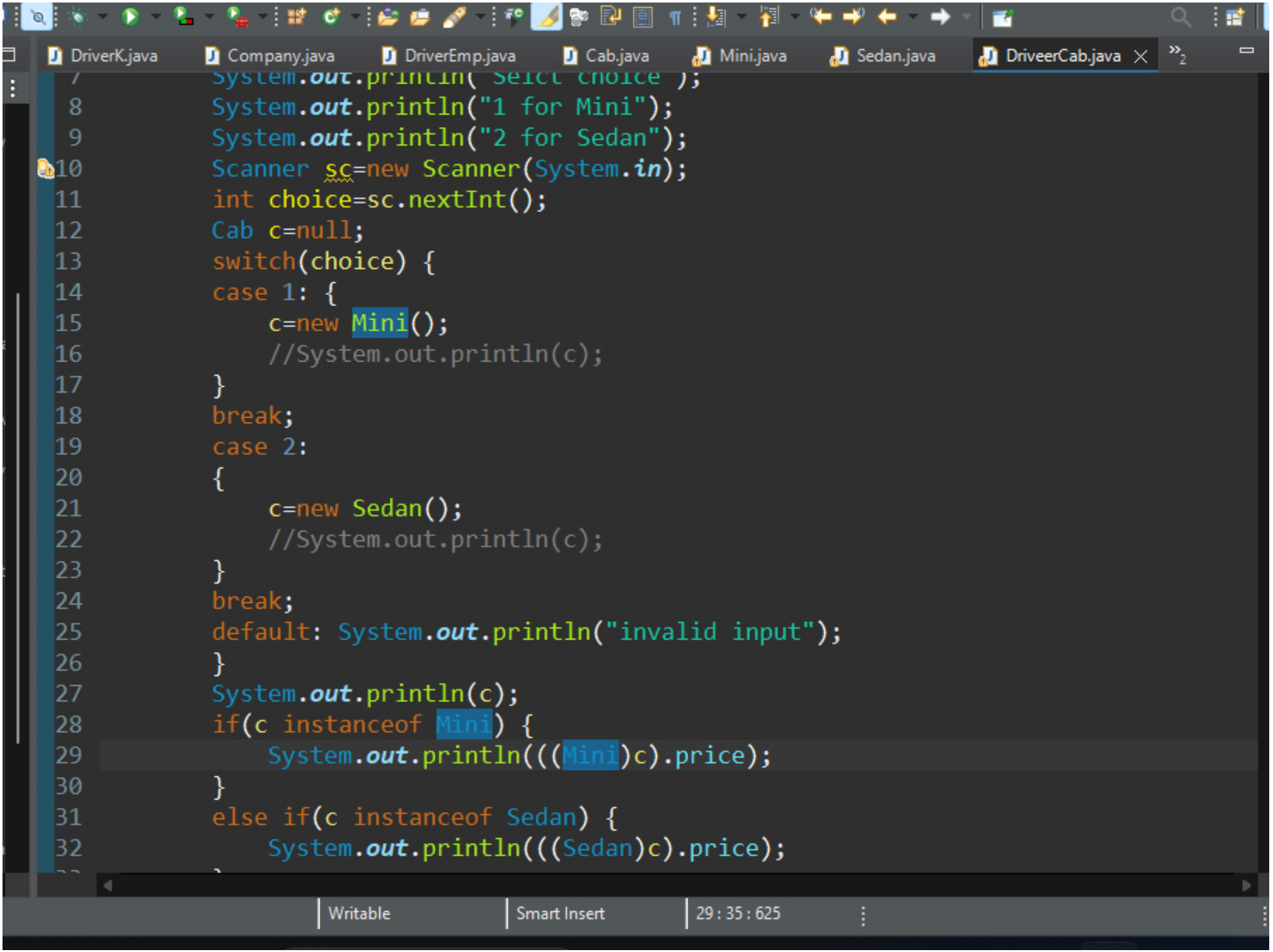

Sopln( ((Sedan)c).price);

Sopln( ((mini)c).price);

CTS


K.java
V.java
DriverK.java
Employe.java
Company.java
*DriverEmp.java
1 package nonprimitive;
2 //Parent
3 public class Company {
4 String nameOfCompany="Dabba Company";
5
6 }
7
Console
<terminated> DriverEmp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 7:53:21 pm – 7:53:21 pr
Dinga
Dabba Company


K.java
V.java
DriverK.java
1 package nonprimitive;
2 //child
3 public class V extends K{
4 int j;
5 }
6
Console
<terminated> DriverK [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 7:46:46 pm – 7:46:46 pm elapsed:


DriverK.java
Company.java
DriverEmp.java
Cab.java
Mini.java
Sedan.java
DriveerCab.java
System.out.println(" Select choice ");
System.out.println("1 for Mini");
System.out.println("2 for Sedan");
Scanner sc=new Scanner(System.in);
int choice=sc.nextInt();
Cab c=null;
switch(choice) {
case 1: {
c=new Mini();
//System.out.println(c);
}
break;
case 2:
{
c=new Sedan();
//System.out.println(c);
}
break;
default: System.out.println("invalid input");
}
System.out.println(c);
if(c instanceof Mini) {
System.out.println(((Mini)c).price);
}
else if(c instanceof Sedan) {
System.out.println(((Sedan)c).price);
Writable
Smart Insert
29 : 35 : 625

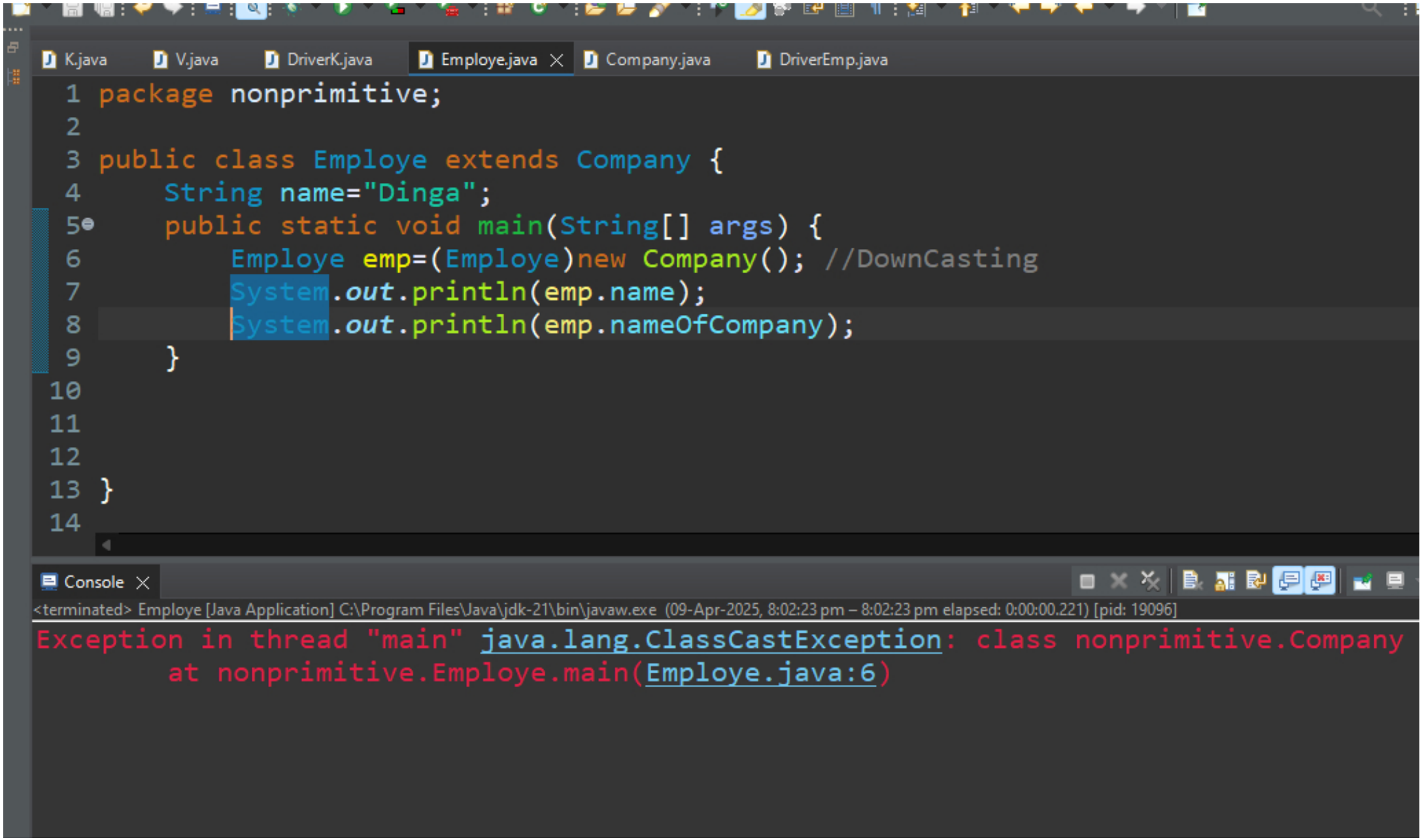
DriverK.java | Company.java | DriverEmp.java | Cab.java | Mini.java | Sedan.java | DriveerCab.java

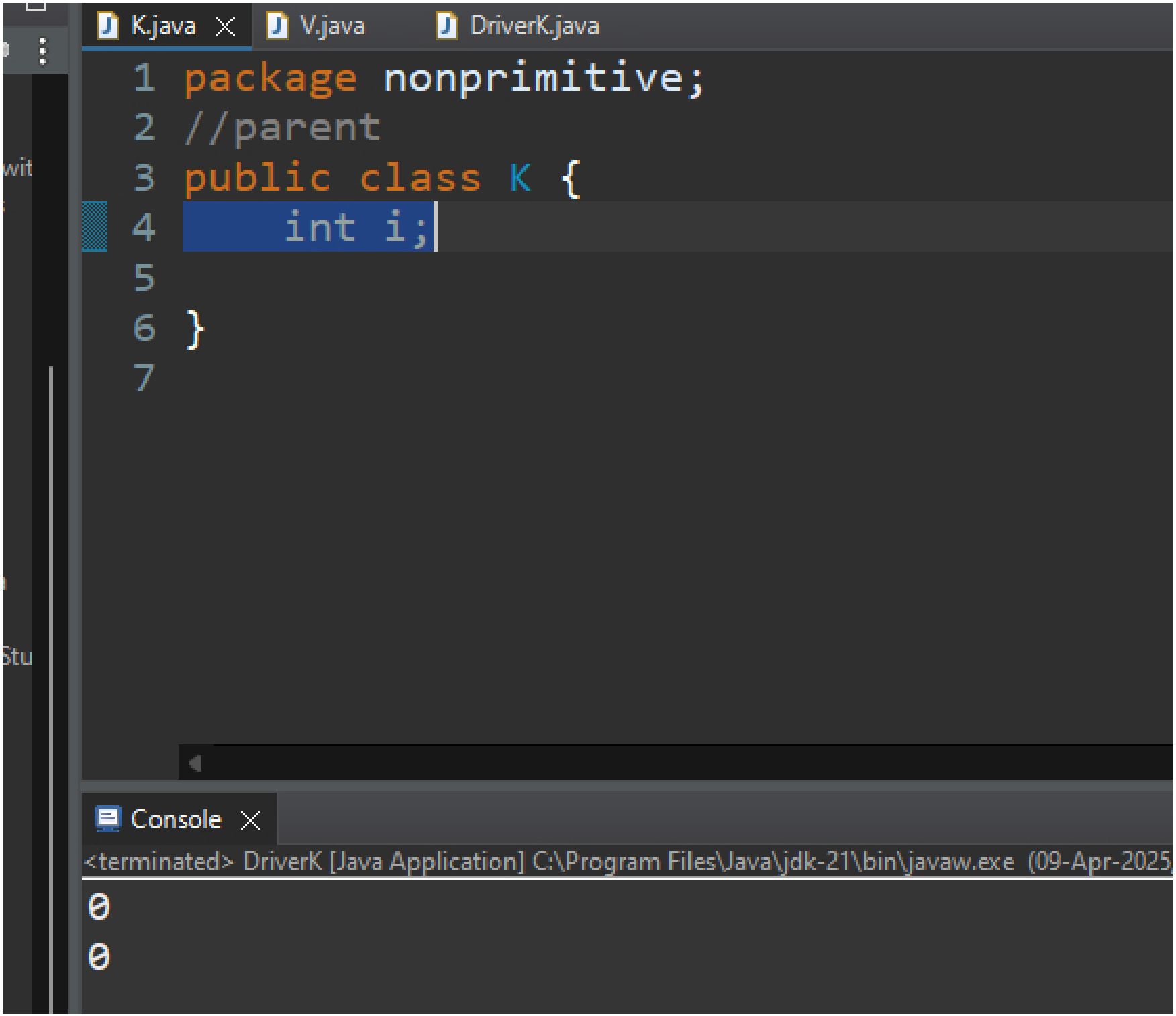
```java
package nonprimitive;

public class DriveerCab {
    public static void main(String[] args) {
        Cab c=new Mini();//upcasted scenario
        Mini m=(Mini)c;
        System.out.println(m.price);
        System.out.println(m.name);

    }

}

```

Console

<terminated> DriveerCab [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 8:46:02 pm – 8:46:02 pm elapsed: 0:00:00.200) [pid: 33700

```
5
Dabba Cab
```


K.java
V.java
DriverK.java
Employe.java
Company.java
DriverEmp.java
1 package nonprimitive;
2
3 public class Employe extends Company {
4 String name="Dinga";
5 public static void main(String[] args) {
6 Employe emp=(Employe)new Company(); //DownCasting
7 System.out.println(emp.name);
8 System.out.println(emp.nameOfCompany);
9 }
10
11
12
13 }
14
Console
<terminated> Employe [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 8:02:23 pm – 8:02:23 pm elapsed: 0:00:00.221) [pid: 19096]
Exception in thread "main" java.lang.ClassCastException: class nonprimitive.Company
at nonprimitive.Employe.main(Employe.java:6)


K.java
V.java
DriverK.java
1 package nonprimitive;
2 //parent
3 public class K {
4 int i;
5
6 }
7
Console
<terminated> DriverK [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025
0
0

K.java | V.java | DriverK.java | Employe.java | Company.java | *DriverEmp.java

```java
package nonprimitive;

public class DriverEmp {
    public static void main(String[] args) {
        Company comp=new Employe();
        //System.out.println(comp.nameOfCompany);//parent class member
        //System.out.println(c.name);//child class member
        Employe emp=(Employe)comp;//DownCasting
        System.out.println(emp.name);//parent class member
        System.out.println(emp.nameOfCompany);//DownCasting
    }

}

```

Console

<terminated> DriverEmp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 7:53:21 pm – 7:53:21 pm elapsed: 0:00:00.226) [pid: 23624]

```
Dinga
Dabba Company
```

K.java | V.java | DriverK.java | Employe.java | Company.java | *DriverEmp.java

```java
package nonprimitive;

public class Employe extends Company {
    String name="Dinga";

}

```

Console

<terminated> DriverEmp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 7:53:21 pm – 7:53:21 pm elapsed: 0:00:00.226) [pid: 23624]

```
Dinga
Dabba Company
```

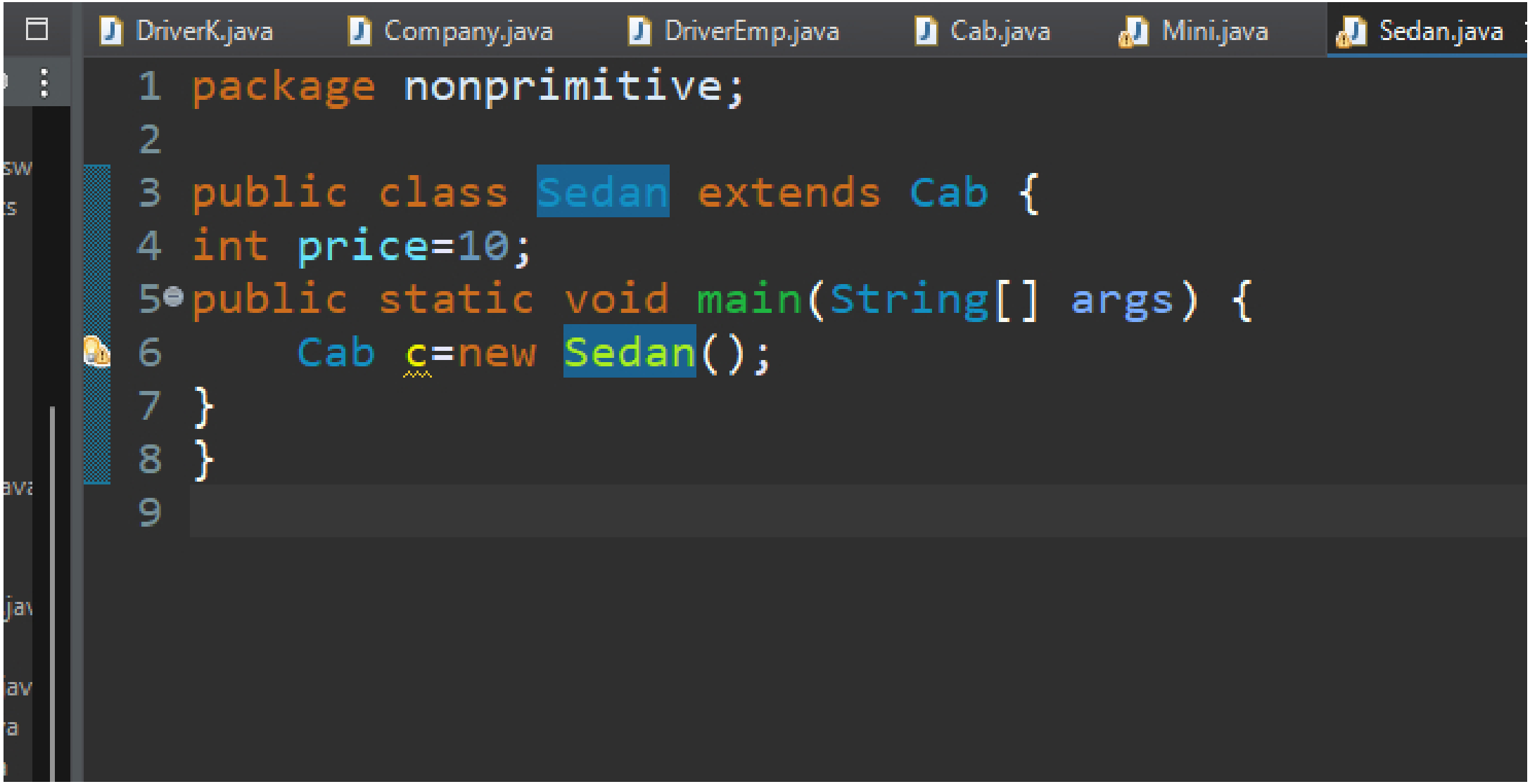
K.java V.java DriverK.java

```java
package nonprimitive;

public class DriverK {
    public static void main(String[] args) {
        K obj1=new V();//upcasting
        //System.out.println(obj1.i);
        //System.out.println(obj1.j);//CTE
        V obj2=(V)obj1;//DownCasting
        System.out.println(obj2.i);
        System.out.println(obj2.j);

    }

}
```

Console

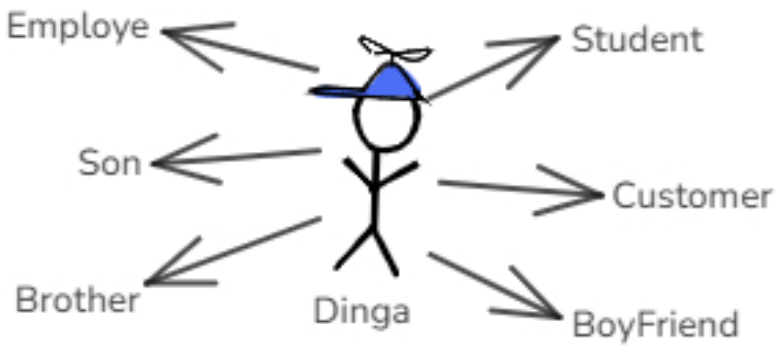
<terminated> DriverK [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (09-Apr-2025, 7:46:46 pm – 7:46:46 pm elapsed: 0:00:00.220) [pid: 20688]

```
0
0
```

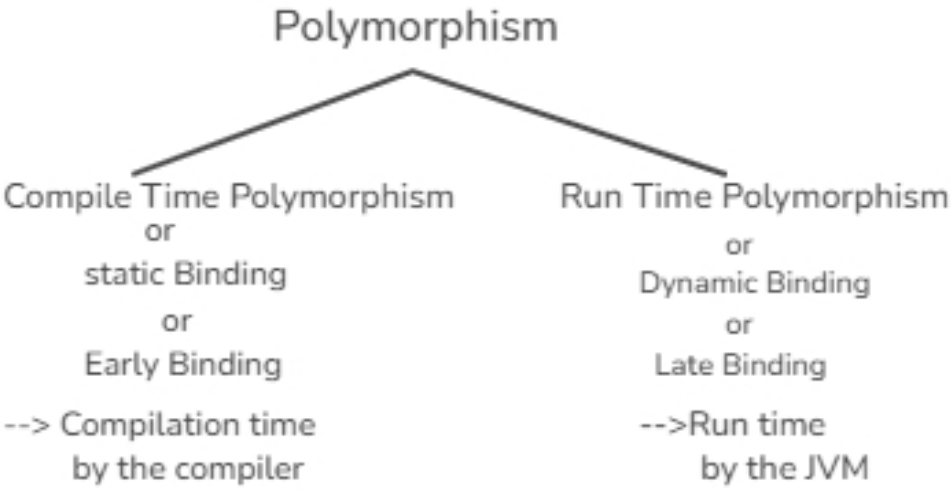

DriverK.java
Company.java
DriverEmp.java
Cab.java
Mini.java
Sedan.java
1 package nonprimitive;
2
3 public class Sedan extends Cab {
4 int price=10;
5 public static void main(String[] args) {
6 Cab c=new Sedan();
7 }
8 }
9

```java
package nonprimitive;

public class Mini extends Cab{
    int price=5;
public static void main(String[] args) {
    Cab c=new Mini();
}

}

```

# Polymorphism

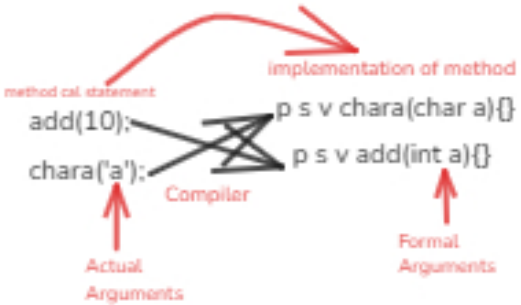



The ability of an Object to exhibit in multiple forms / Behaviours is known as Polymorphism.




## Compile Time Polymorphism

The Bind between the method call statement and the implementation of the method during the compile time by the compiler by looking at actual and formal arguements is known as CompileTime Polymorphism.



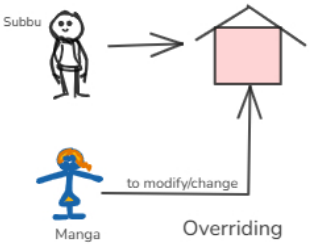

1)Method OverLoading
2)Constructor OverLoading
3)Variable shadowing/Hiding
4)Method shadowing/Hiding

## Run Time Polymorphism :

The Bind between the method call statement
and the implementation of the method during the RunTime
is Known as Run Time Polymorphism.

Ex : Method Overriding

### Method OverRiding :





```
class Rosy{
void demo( ){
Sopln("Classical");
}
}
```

```
class Dingi extends Rosy{
void demo( ){
Sopln("Salsa");
}
```

```
Dingi d=new Dingi( );
d.demo( );
```

→ Salsa

The process of providing new Implementation to the inheirted method or to the super class method in the subclass is known as Method Overriding

-->method overiding depends on / the binding depends on Object Created

Rules :

1)create a method in subclass whose declaration should be the same

```
ex : class Test{
       void print( ){
        Sopln("hai");
       }
     }
```

```
class Demo extends Test{
void print(int a ){
Sopln("Bye");
}
}
```

method overloading ✓

method overriding ✗

2)the access modifier of overriding method(Sub class) should be either same or higher than visibility of Overridden method(Super class).

private < default < protected < public

Ex :

```
class Demo{
  void test( ){
  }
}
class Test extends Demo( ){
 private void test( ){
 }
}
```

→ overridden method (default Scope)

→ overriding method (private Scope)

Ex :

```
class A{
public void demo(){
}
}
```

```
class B extends A{
public void demo(){
}
}
```

Characteristics of method Overriding :

1)Method overriding is achieved at runtime.It follows Run Time Polymorphism and doesnot follows Compile Time polymorphism.

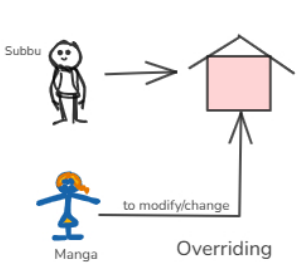
2)Execution of method implementation depends on object Created and doesnot depends on type of reference used to call.

## Run Time Polymorphism :

The Bind between the method call statement
and the implementation of the method during the RunTime
is Known as Run Time Polymorphism.

Ex : Method Overriding

### Method OverRiding :





Rosy

```
class Rosy{
void demo( ){
Sopln("Classical");
}
}
```


Dingi

```
class Dingi extends Rosy{
void demo( ){
Sopln("Salsa");
}
```

```
Dingi d=new Dingi( );
d.demo( );
```

Salsa

The process of providing new Implementation to the inheirted method or to the super class method in the subclass is known as Method Overriding

-->method overiding depends on / the binding depends on Object Created

Rules :

1)create a method in subclass whose declaration should be the same

```
ex : class Test{
       void print( ){
        Sopln("hai");
       }
     }
```

```
class Demo extends Test{
void print(int a ){
Sopln("Bye");
}
}
```

method overloading ✓

method overriding ✗

2)the access modifier of overriding method(Sub class) should be either same or higher than visibility of Overridden method(Super class).

private < default < protected < public

Ex :

```
class Demo{
  void test( ){

  }
}
```

overridden method

(default Scope)

```
class Test extends Demo( ){
  private  void test( ){

  }
}
```

(private Scope)

overriding method

Ex :

```
class A{
public void demo(){
}
}
```

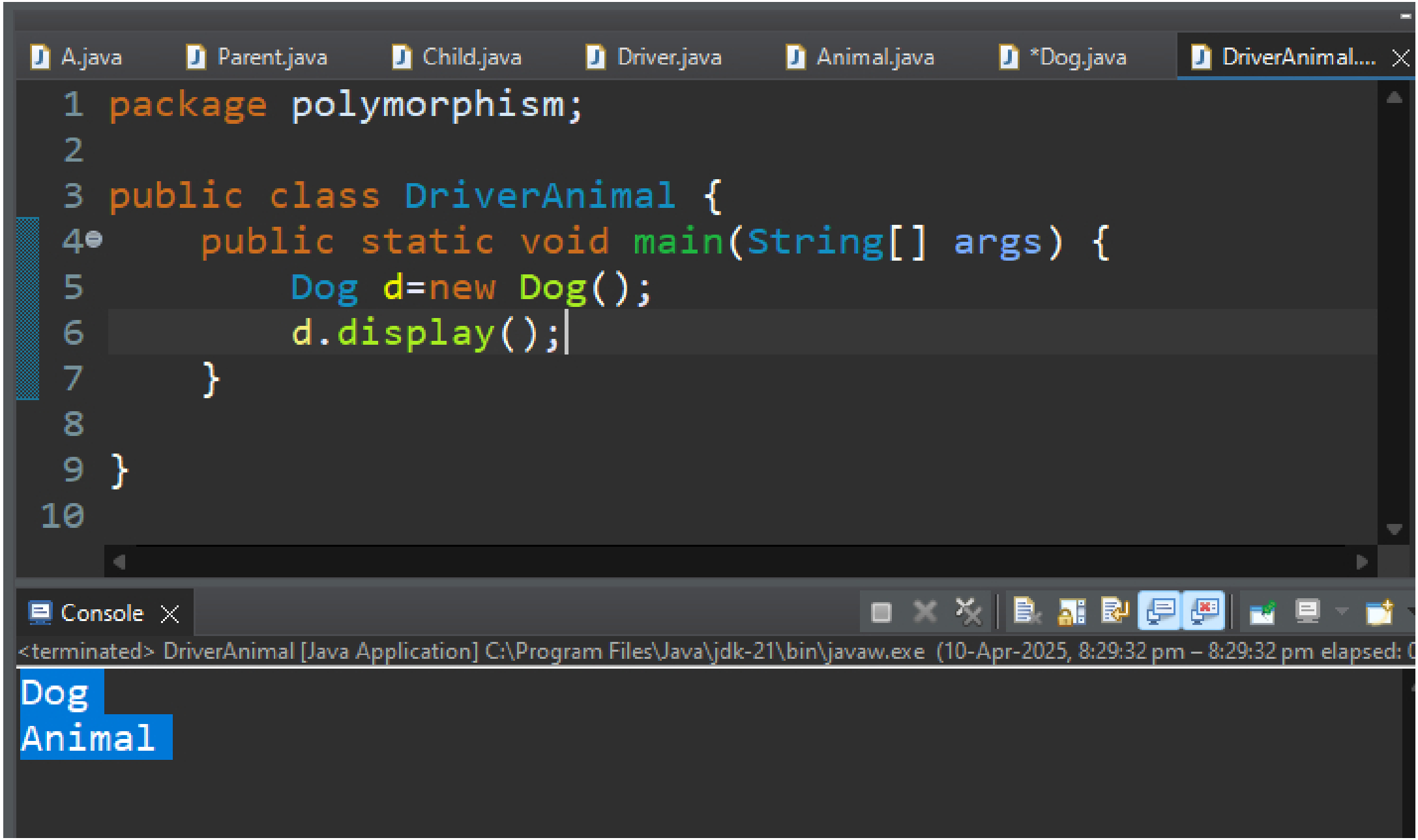
✓

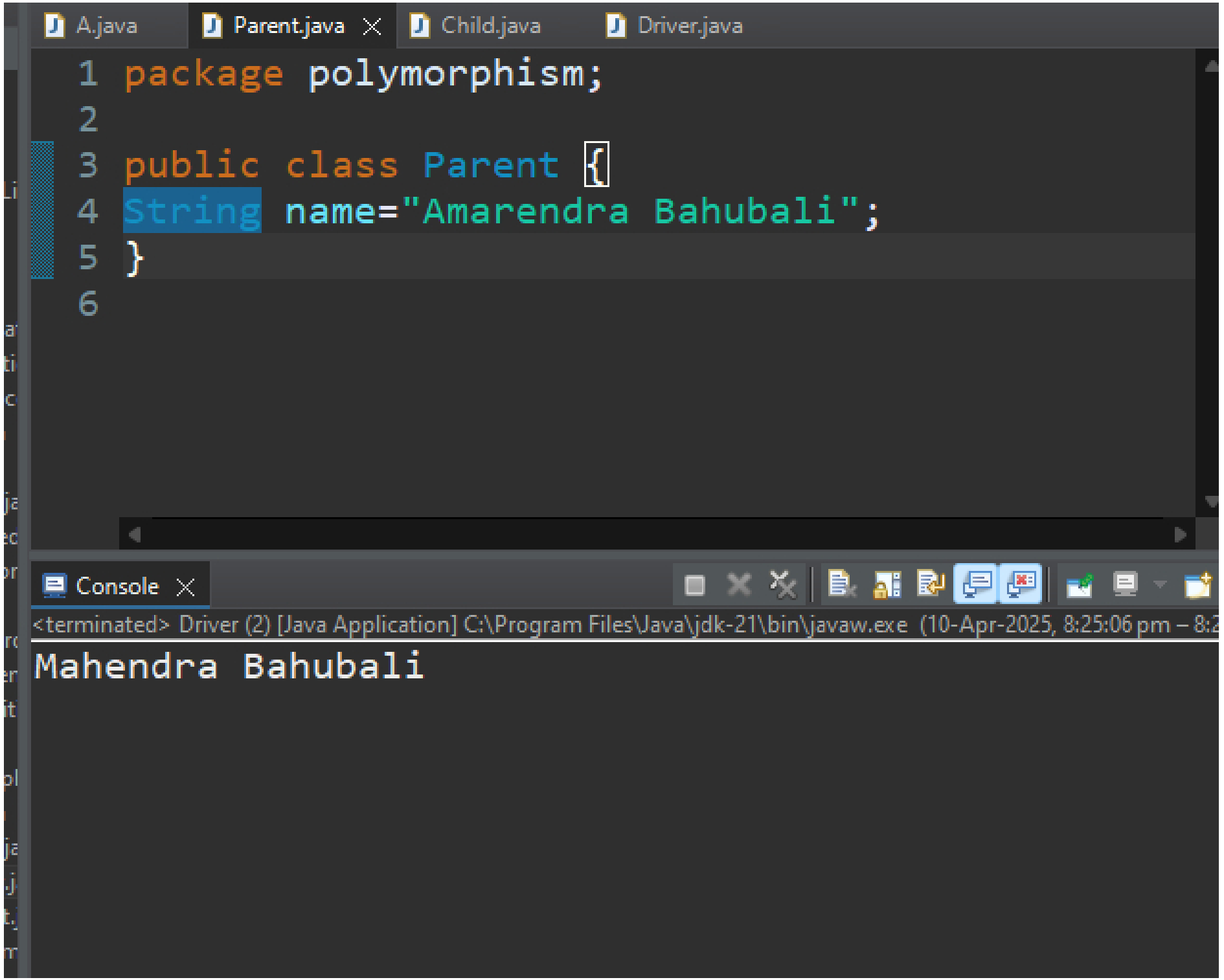
```
class B extends A{
public void demo(){
}
}
```

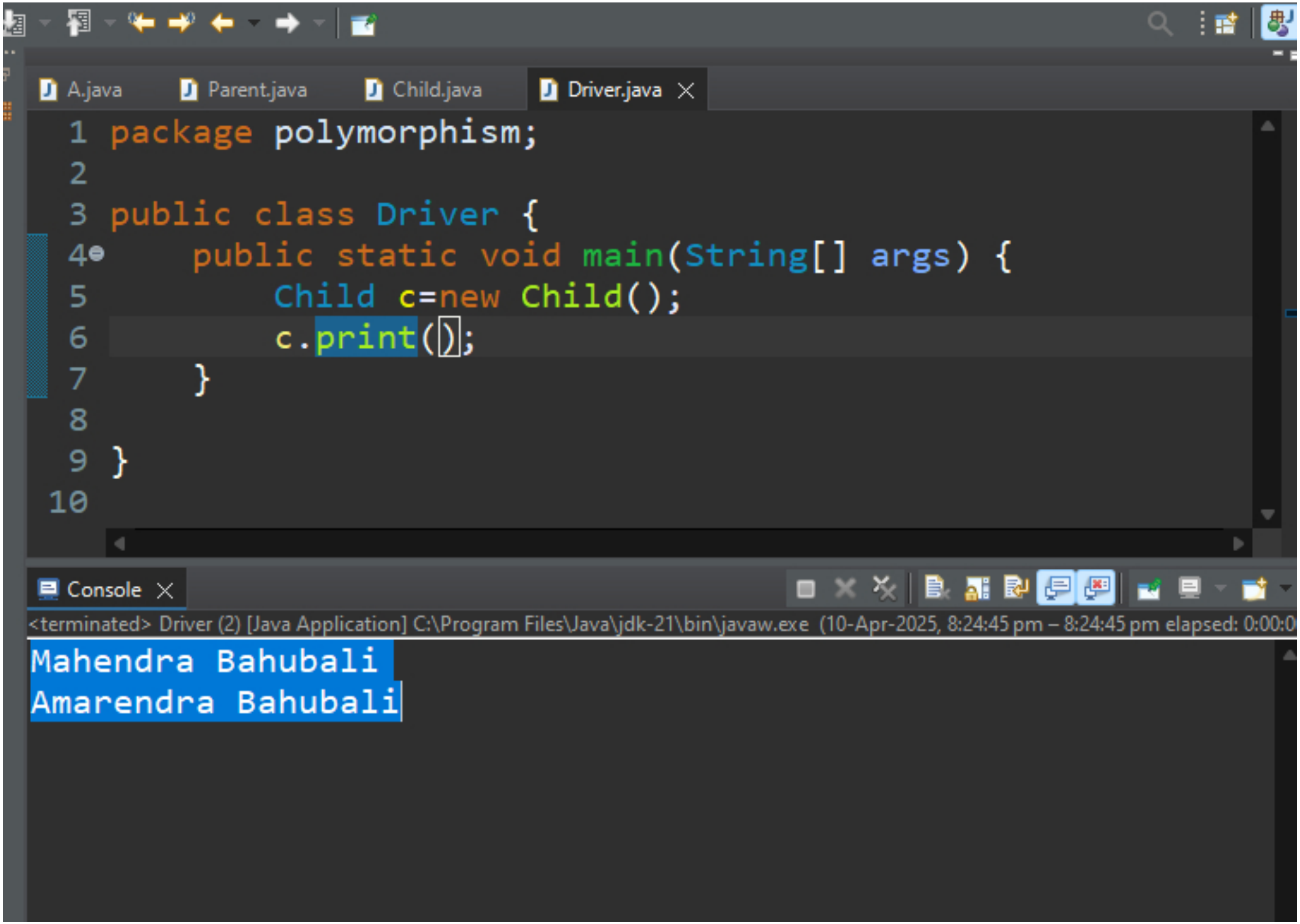
Characteristics of method Overriding :

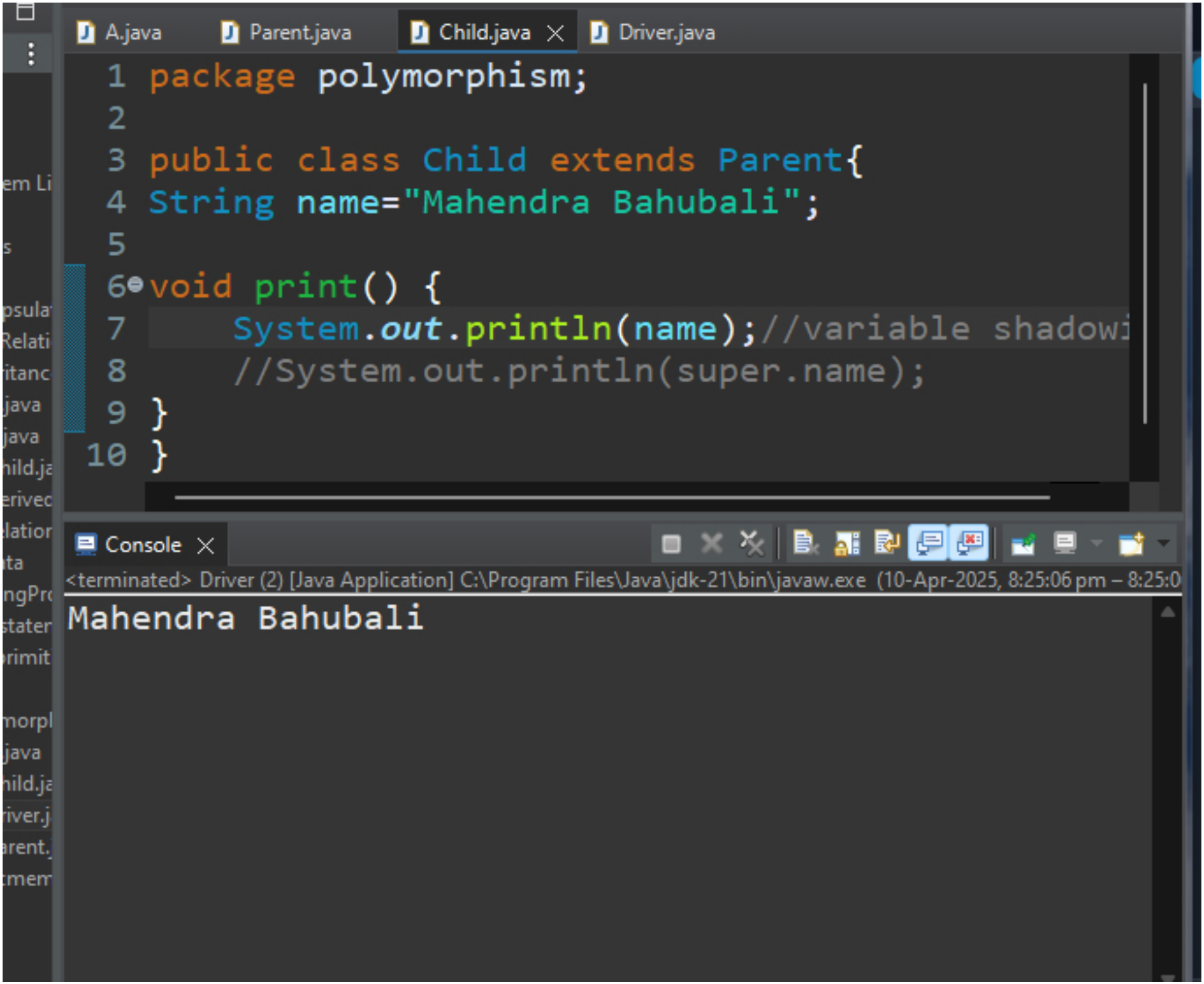
1)Method overriding is achieved at runtime.It follows Run Time Polymorphism and doesnot follows Compile Time polymorphism.

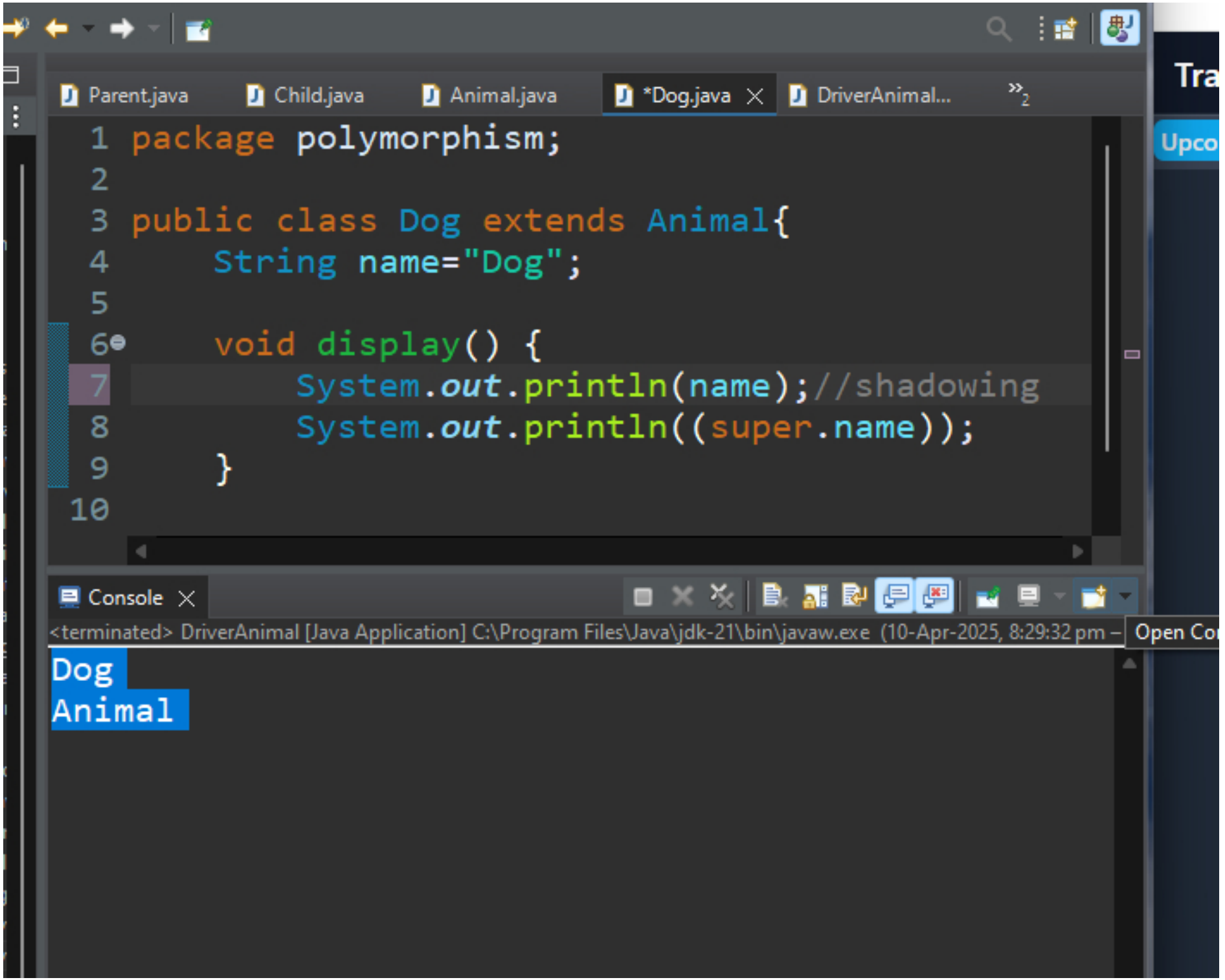
2)Execution of method implementation depends on object Created and doesnot depends on type of reference used to call.

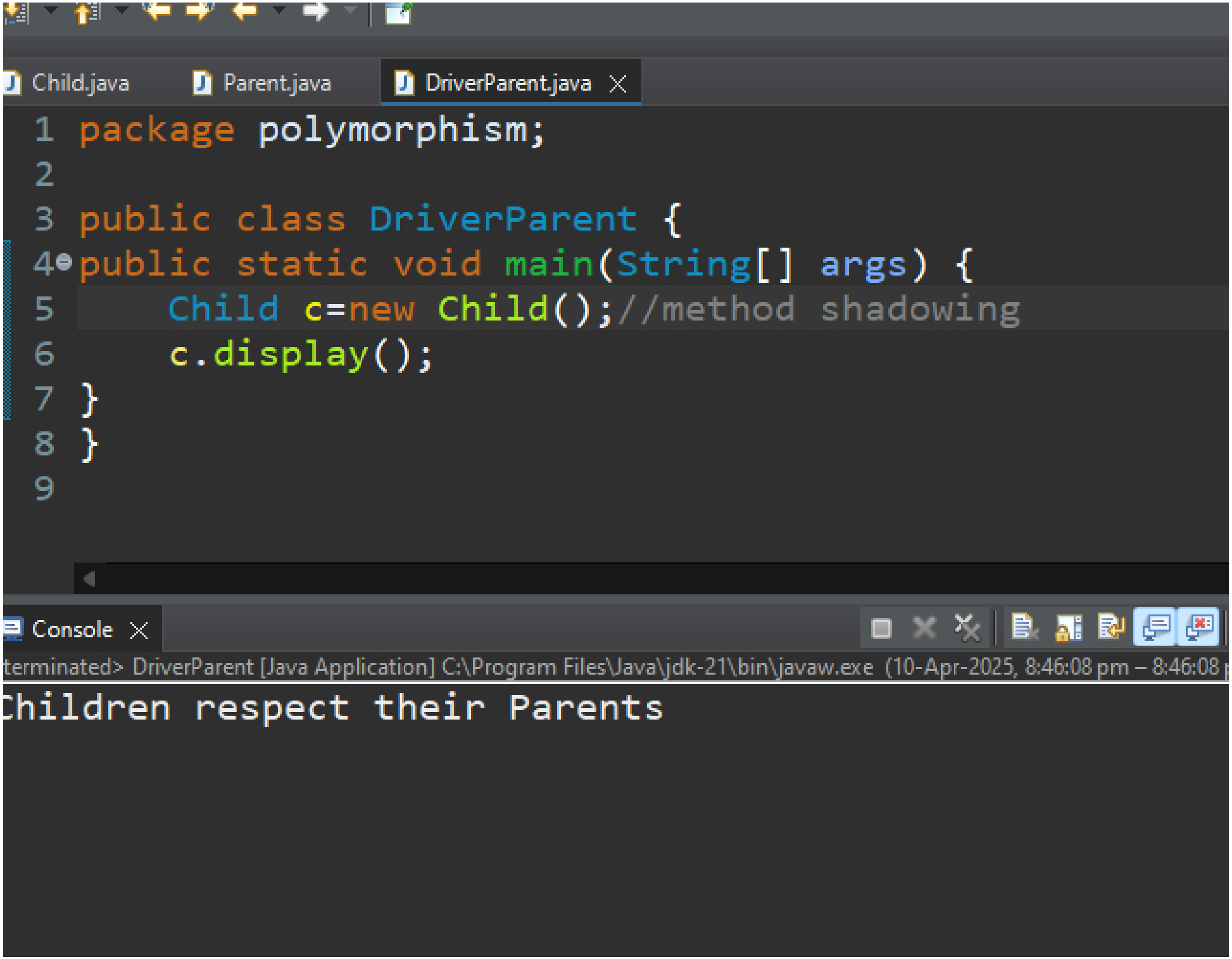

A.java
Parent.java
Child.java
Driver.java
Animal.java
*Dog.java
DriverAnimal....
1 package polymorphism;
2
3 public class DriverAnimal {
4 public static void main(String[] args) {
5 Dog d=new Dog();
6 d.display();
7 }
8
9 }
10
Console
<terminated> DriverAnimal [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:29:32 pm – 8:29:32 pm elapsed:
Dog
Animal

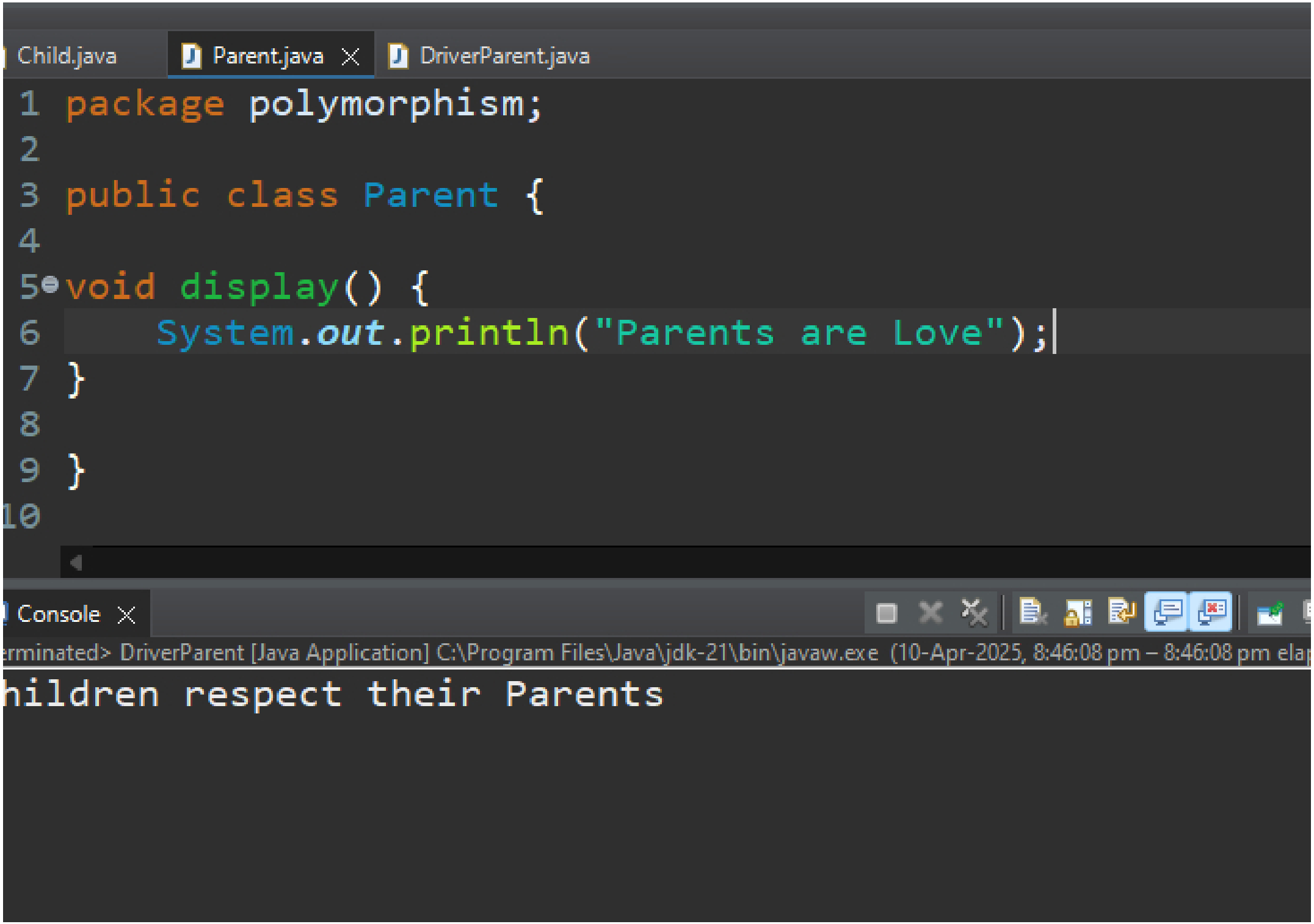

A.java
Parent.java
Child.java
Driver.java
1 package polymorphism;
2
3 public class Parent {
4 String name="Amarendra Bahubali";
5 }
6
Console
<terminated> Driver (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:25:06 pm – 8:2
Mahendra Bahubali

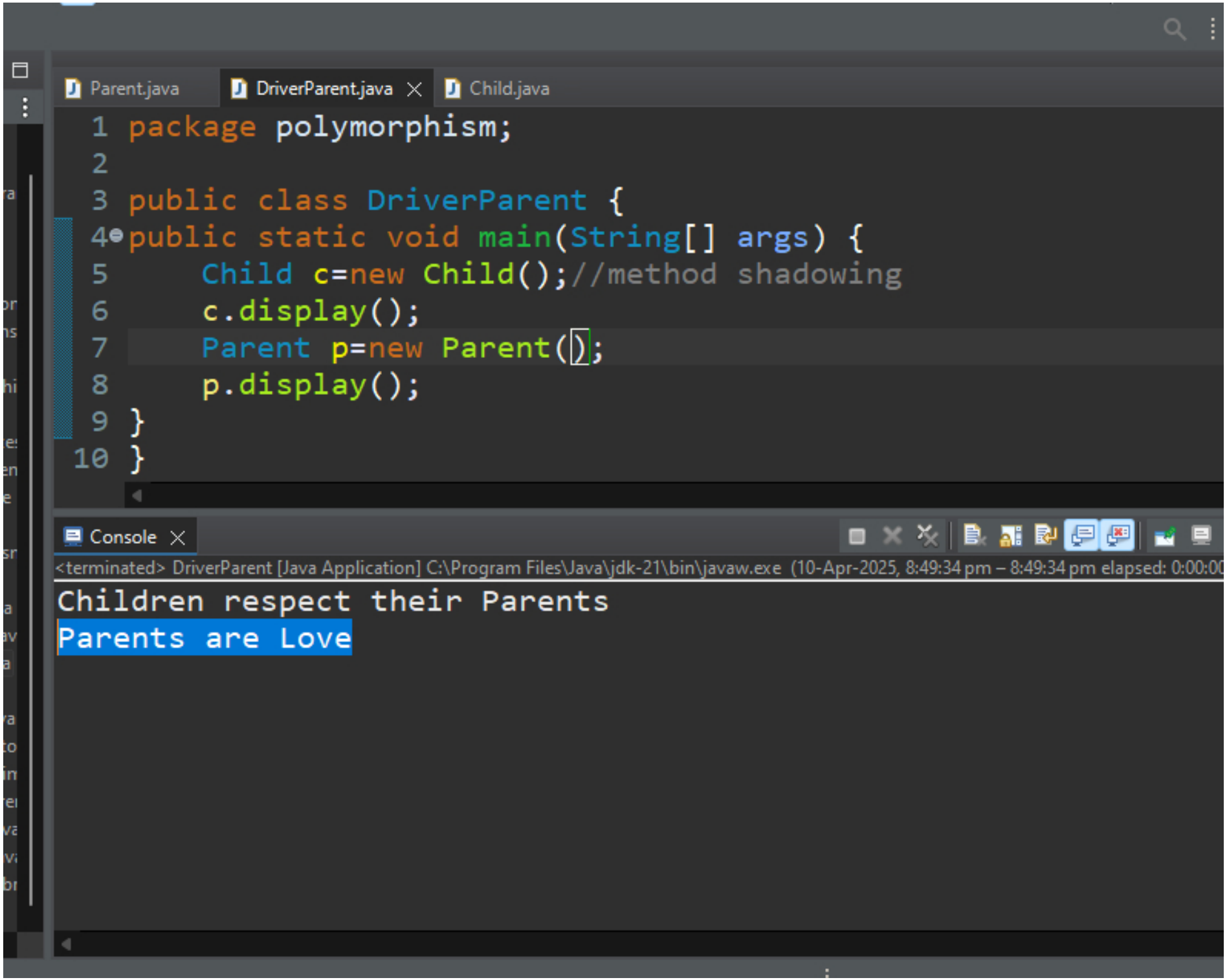

A.java
Parent.java
Child.java
Driver.java
1 package polymorphism;
2
3 public class Driver {
4 public static void main(String[] args) {
5 Child c=new Child();
6 c.print();
7 }
8
9 }
10
Console
<terminated> Driver (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:24:45 pm – 8:24:45 pm elapsed: 0:00:0
Mahendra Bahubali
Amarendra Bahubali

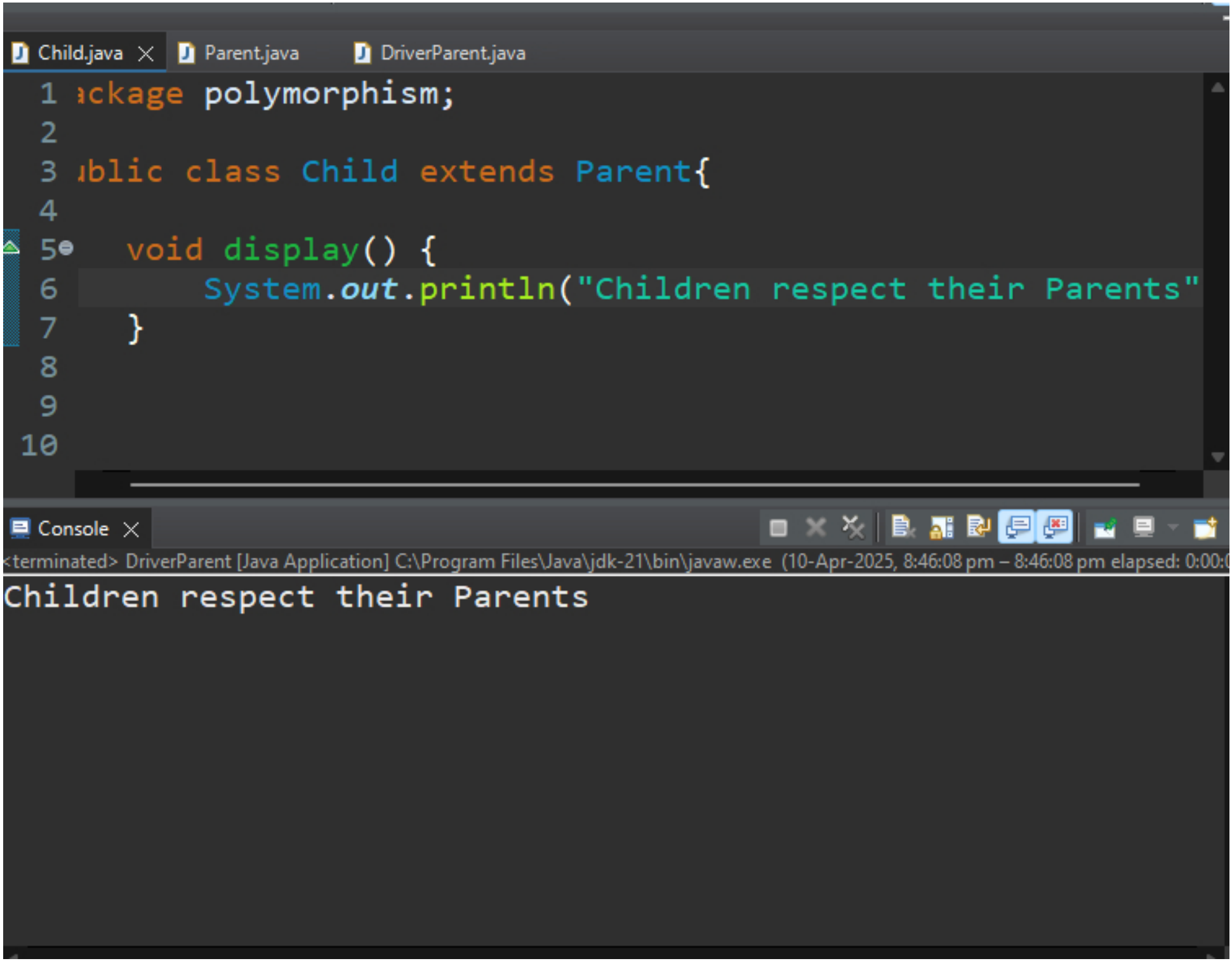

A.java
Parent.java
Child.java
Driver.java
1 package polymorphism;
2
3 public class Child extends Parent{
4 String name="Mahendra Bahubali";
5
6 void print() {
7 System.out.println(name);//variable shadowi
8 //System.out.println(super.name);
9 }
10 }
Console
<terminated> Driver (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:25:06 pm – 8:25:0
Mahendra Bahubali

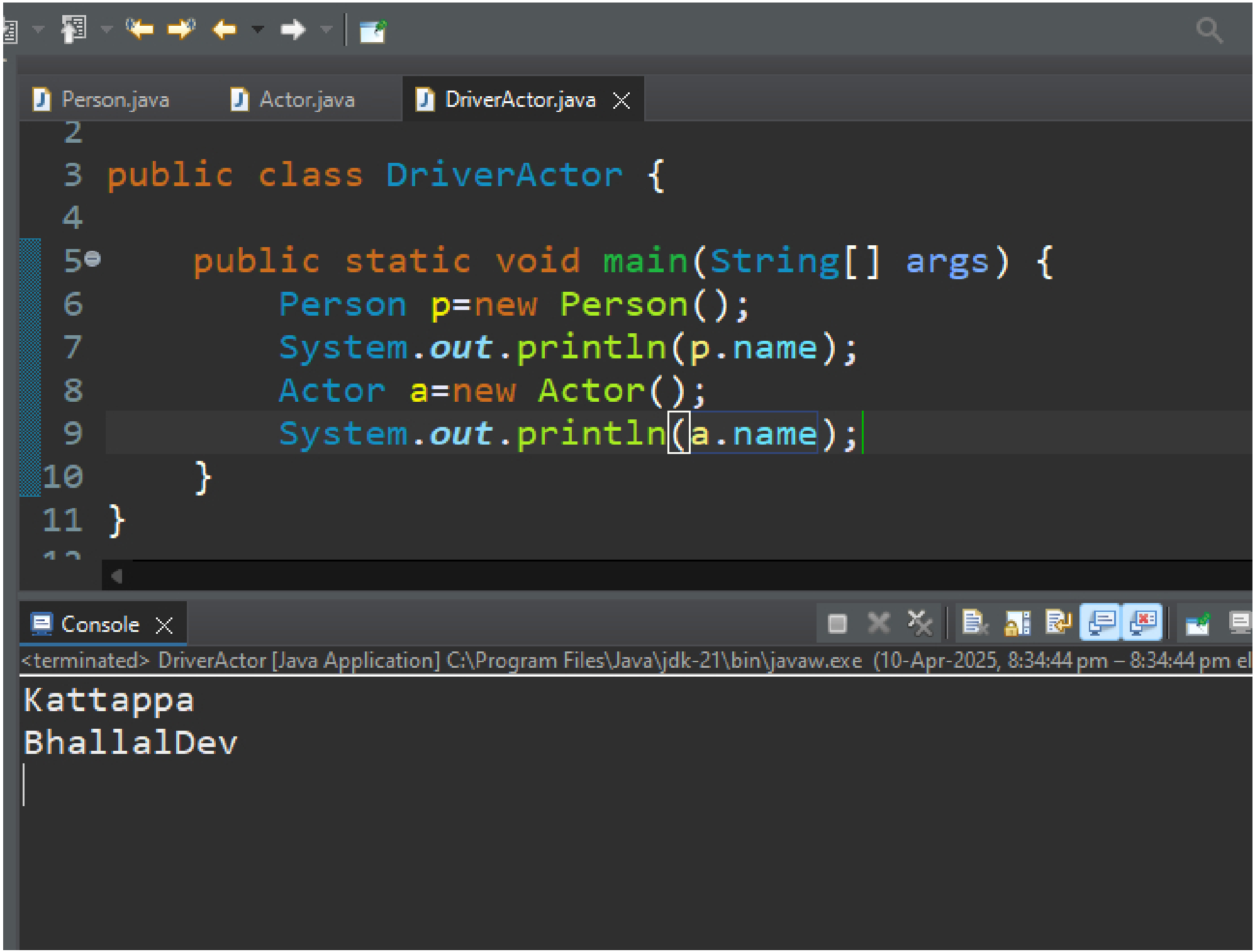

Parent.java
Child.java
Animal.java
*Dog.java
DriverAnimal...
1 package polymorphism;
2
3 public class Dog extends Animal{
4 String name="Dog";
5
6 void display() {
7 System.out.println(name);//shadowing
8 System.out.println((super.name));
9 }
10
Console
<terminated> DriverAnimal [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:29:32 pm –
Dog
Animal


Child.java
Parent.java
DriverParent.java
1 package polymorphism;
2
3 public class DriverParent {
4 public static void main(String[] args) {
5 Child c=new Child();//method shadowing
6 c.display();
7 }
8 }
9
Console
terminated> DriverParent [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:46:08 pm – 8:46:08
Children respect their Parents


Child.java
Parent.java
DriverParent.java
1 package polymorphism;
2
3 public class Parent {
4
5 void display() {
6 System.out.println("Parents are Love");
7 }
8
9 }
10
Console
erminated> DriverParent [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:46:08 pm – 8:46:08 pm ela
hildren respect their Parents


Parent.java
DriverParent.java
Child.java
1 package polymorphism;
2
3 public class DriverParent {
4 public static void main(String[] args) {
5 Child c=new Child();//method shadowing
6 c.display();
7 Parent p=new Parent();
8 p.display();
9 }
10 }
Console
<terminated> DriverParent [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:49:34 pm – 8:49:34 pm elapsed: 0:00:0
Children respect their Parents
Parents are Love


Child.java
Parent.java
DriverParent.java
ackage polymorphism;
blic class Child extends Parent{
void display() {
System.out.println("Children respect their Parents"
}
Console
<terminated> DriverParent [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:46:08 pm – 8:46:08 pm elapsed: 0:00:
Children respect their Parents

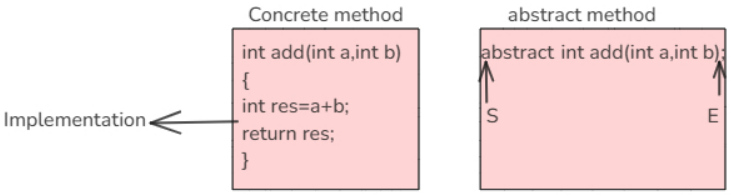

Person.java
Actor.java
DriverActor.java
public class DriverActor {
public static void main(String[] args) {
Person p=new Person();
System.out.println(p.name);
Actor a=new Actor();
System.out.println(a.name);
}
}
Console
<terminated> DriverActor [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (10-Apr-2025, 8:34:44 pm – 8:34:44 pm el
Kattappa
BhallalDev

# Abstraction

It is the process of hiding the actual implementation
and exposing only the necessary information to the user

--.it is an overview
-->hiding the unneccessary and showing the neccessary
implementation

Abstract method :




-->A method which is having implementation
is known as Concrete method

--> A method which is prefixed with abstract keyword
and doesnot have implementation is known as
abstract method.
-->we cannot execute abstract method because it
doesnot have body
--> it should be started with abstract keyword/abstract
modifier and end with semicolon

//concrete class
Ex : class Example {
//create abstract method → CTE
abstract int add(int a,int b);
}

-->An incomplete method is not allowed in regular class
--> class is by default concrete in nature(full/complete)

Abstract Component : An incomplete component is considered
as Abstract Component.

There are 2 Components
1)abstract class
2)interface

abstract class :

-->the main purpose is to support abstract method
Def : it is an abstract component which can have both
abstract and concrete methods in it

Syntax :

```
abstract class ClassName{
//abstract methods
//concrete methods
//variables //initializers
}
```

-->whatever we write in normal class,we can
write here , The extra Component is abstract
method.

Ex :

```
abstract class Example2{ → abstract class
    abstract void demo( ); → abstract method
}
```

When we should make a class abstract ??

```
abstract class Example{
    abstract void demo( );
}

class Example2 extends Example {

}
```

→ CTE

-->abstract method is also inherited
-->it is mandatory to make class as abstract or override the
method.

-->if a class contains atleast one abstract method either declared
or inherited but not overridden, it is mandatory to make the class as
abstract.

Characteristics of abstract class :

1)it can have abstract methods and concrete methods
2)abstract class cannot be Instantiated.
-->creation of object for abstract class is restricted
-->we can have Constructors inside abstract class
extra included is abstract method.

Ex ;
```
abstract class Student{
 String name;
  Student(string name){
   this.name=name;
  }
  void demo( ){
   Sopln("hai from concrete method");
  }
  abstract void bunk( );
}
```
Constructor
Concrete method
abstract method

```
Student s1=new Student( );
Student s2;
```
CTE

```
abstract class Test1{

}
```

```
abstract class Test2{
    abstract void demo( );
}
```

```
abstract class Test3{
   abstract static void demo( );
}
```

-->we cannot make static method as abstract
-->The concept of Abstraction is valid only for non-static members

```
abstract class Test4{
public final static void test( ){
}
}
```
-->it allows more than one modifer

Create a class with following Components
1)abstract method add with 2 int args and return int
2)abstract method div with 2 int args and return int
3)abstract method mul with 2 int args and return int
4)concrete method sub with 2 int args and return int

to inherit abstract class and Implementation of abstract method :

ex ;

```
abstract class Demo{
 abstract void test( );
}

class Example extends Demo{
}
```
abstract method also inherited but CTE

Case 1 : make class example abstract
Case 2 : override the inherited abstract method

```
abstract class Demo2{
   abstract void test( );
}
```
abstract component
abstract layer
-->overview is available

```
class Example extends Demo2{
  //provide the implementation for abstract
     method
   void test( ){
   Sopln("hello");
 }
```
Implementation Layer

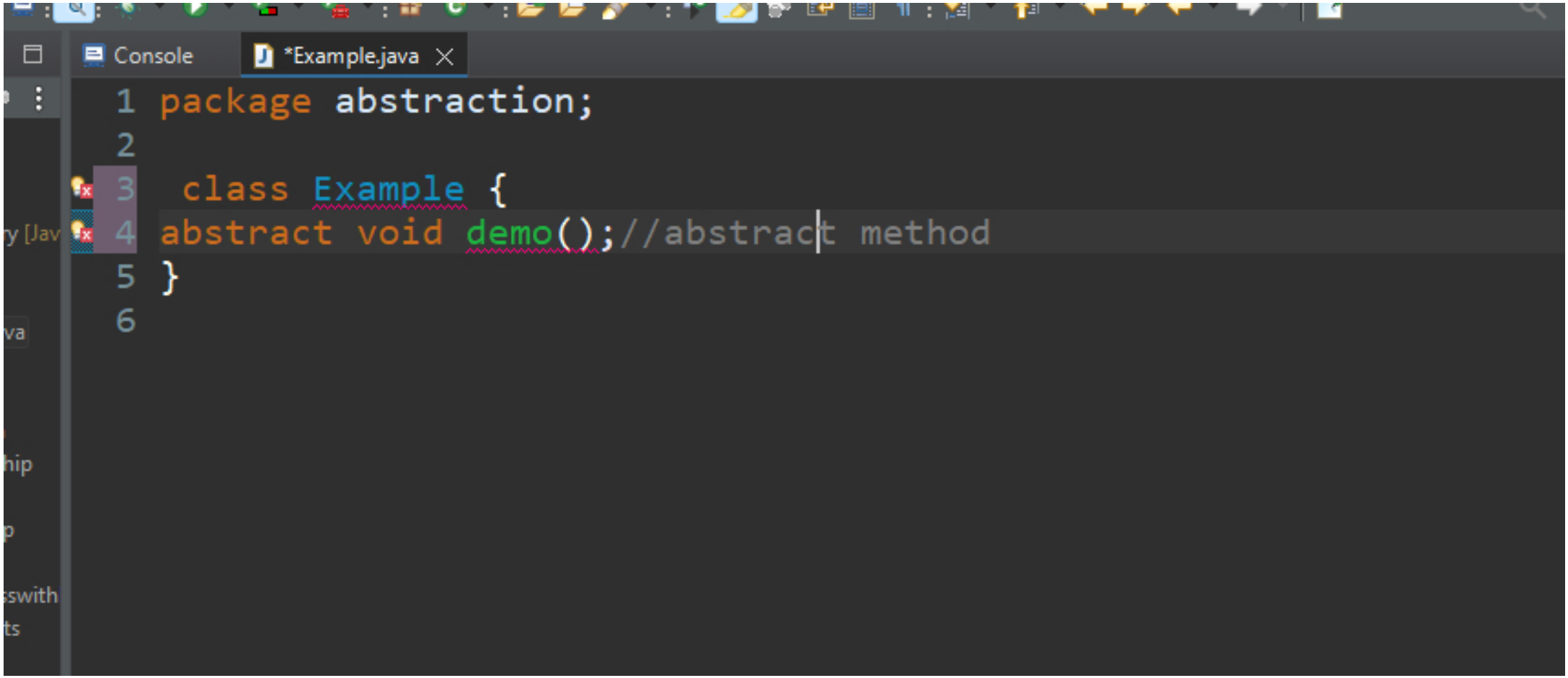
Entire application can be break into 2 layers
1)abstract layer
abstract layer can be developed by one person
2)Implementation Layer
Implementation layer can be developed by another person
-->implementation layer should follow the signature of abstract layer

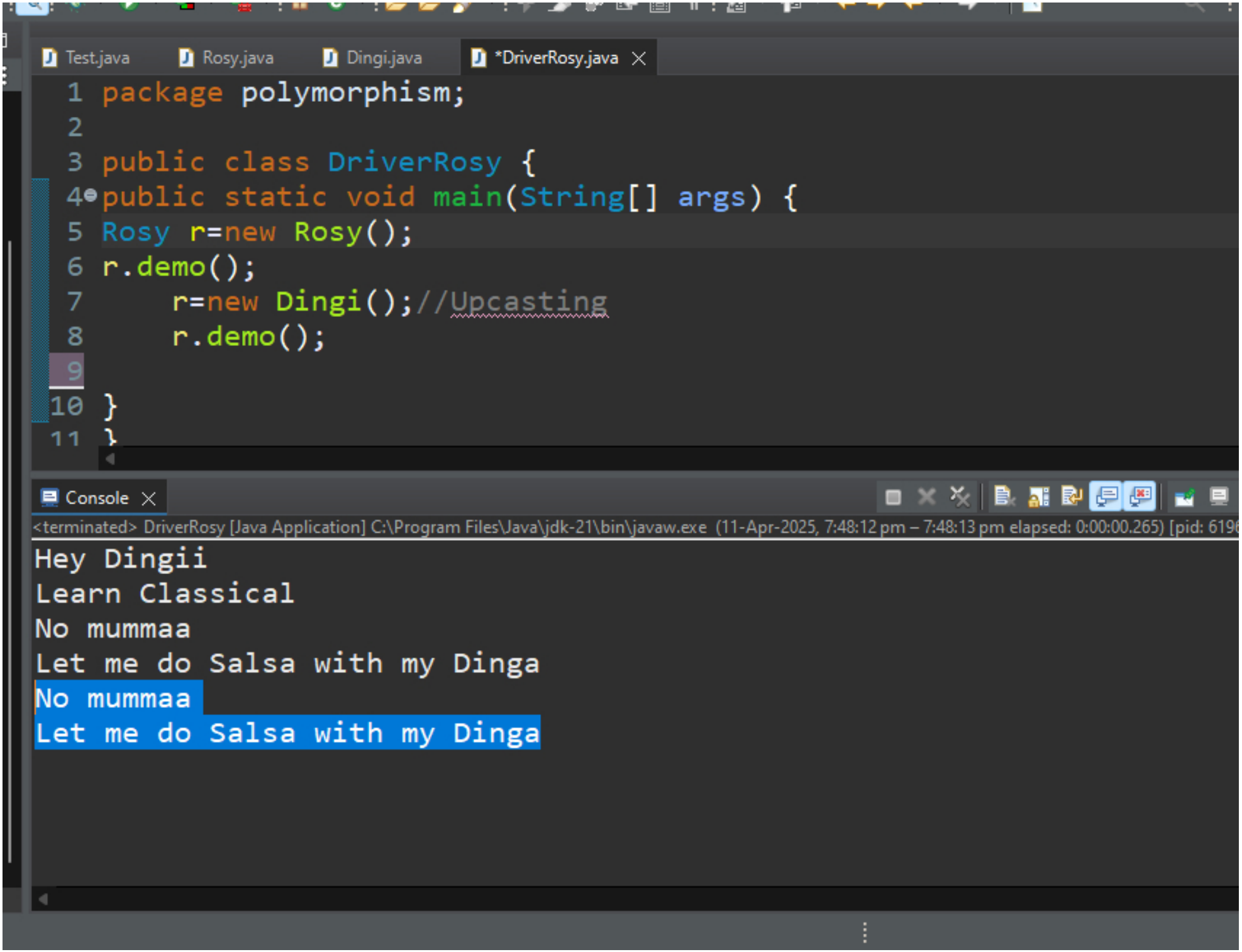
-->Implementation layer contains of Concrete classes
-->Concrete classesn contains only Concrete methods(Implementations)
abstract methods

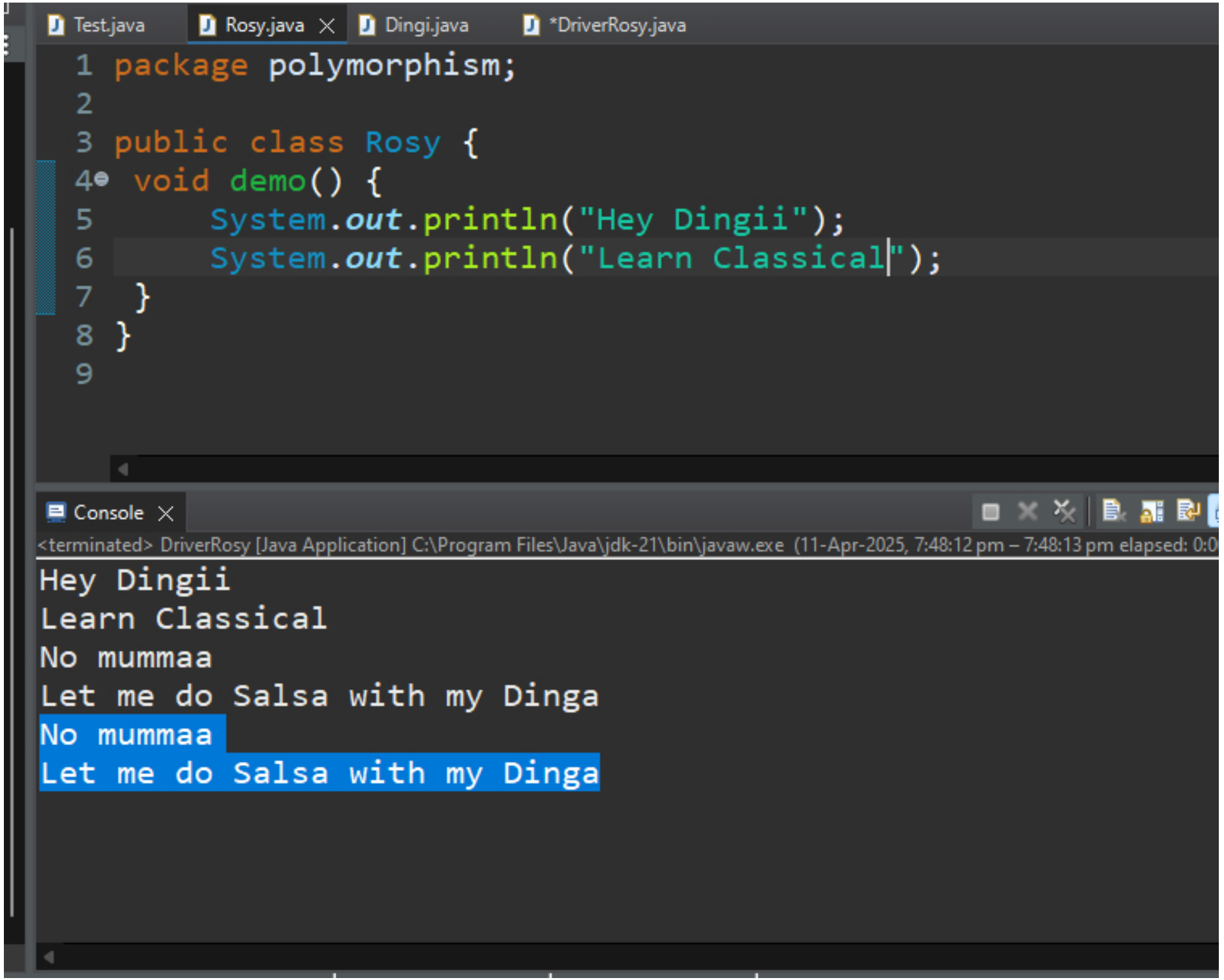
ex :
```
abstract class Student{
   void attendClass( ){
     Sopln("Gain Knowledge");
   }
}
```
-->can we create directly an object

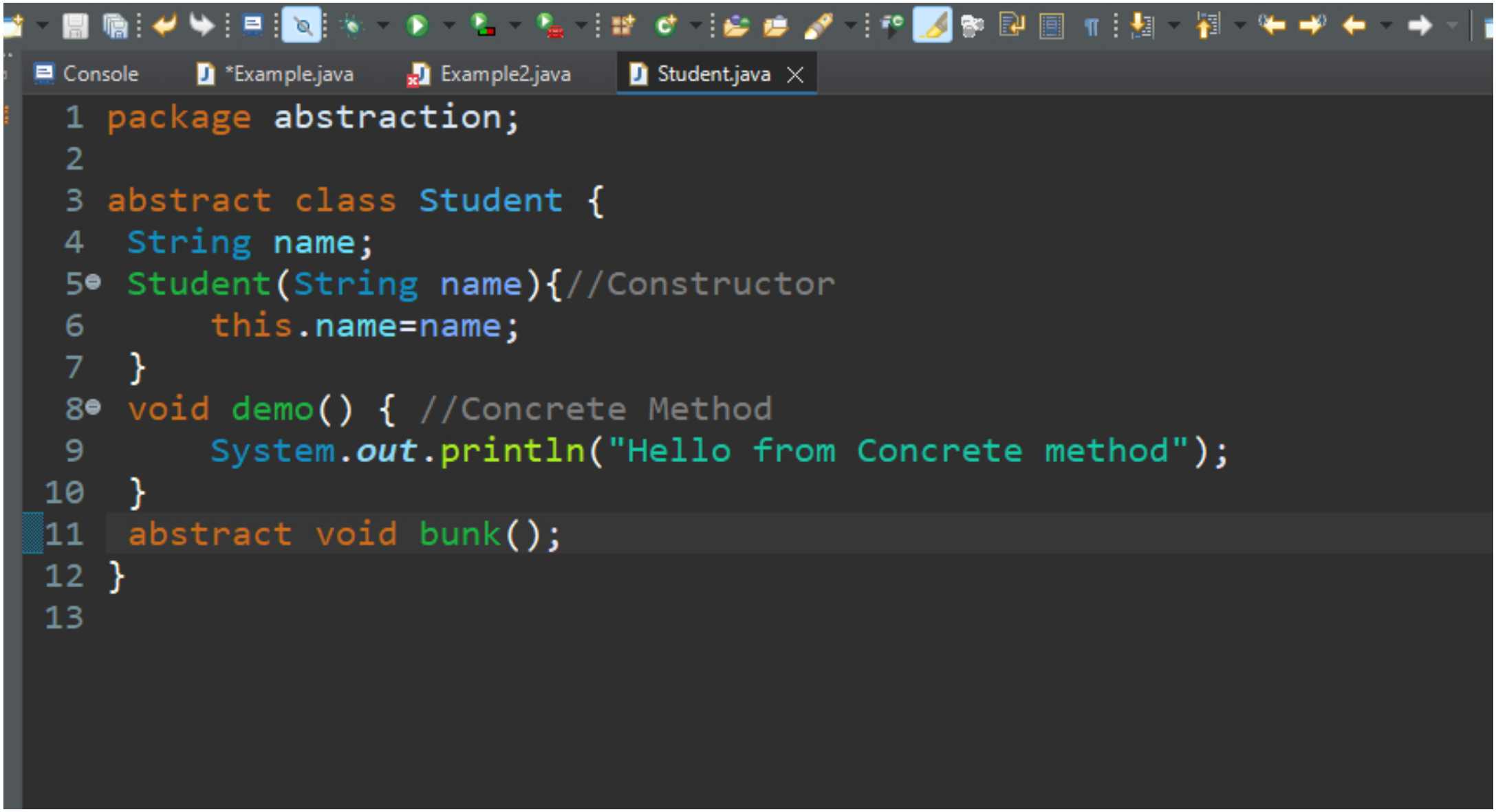
-->it is a non-static method , Object creation is compulsory
-->if we want to use a non-static method of abstract class, then we need a concrete implementing class

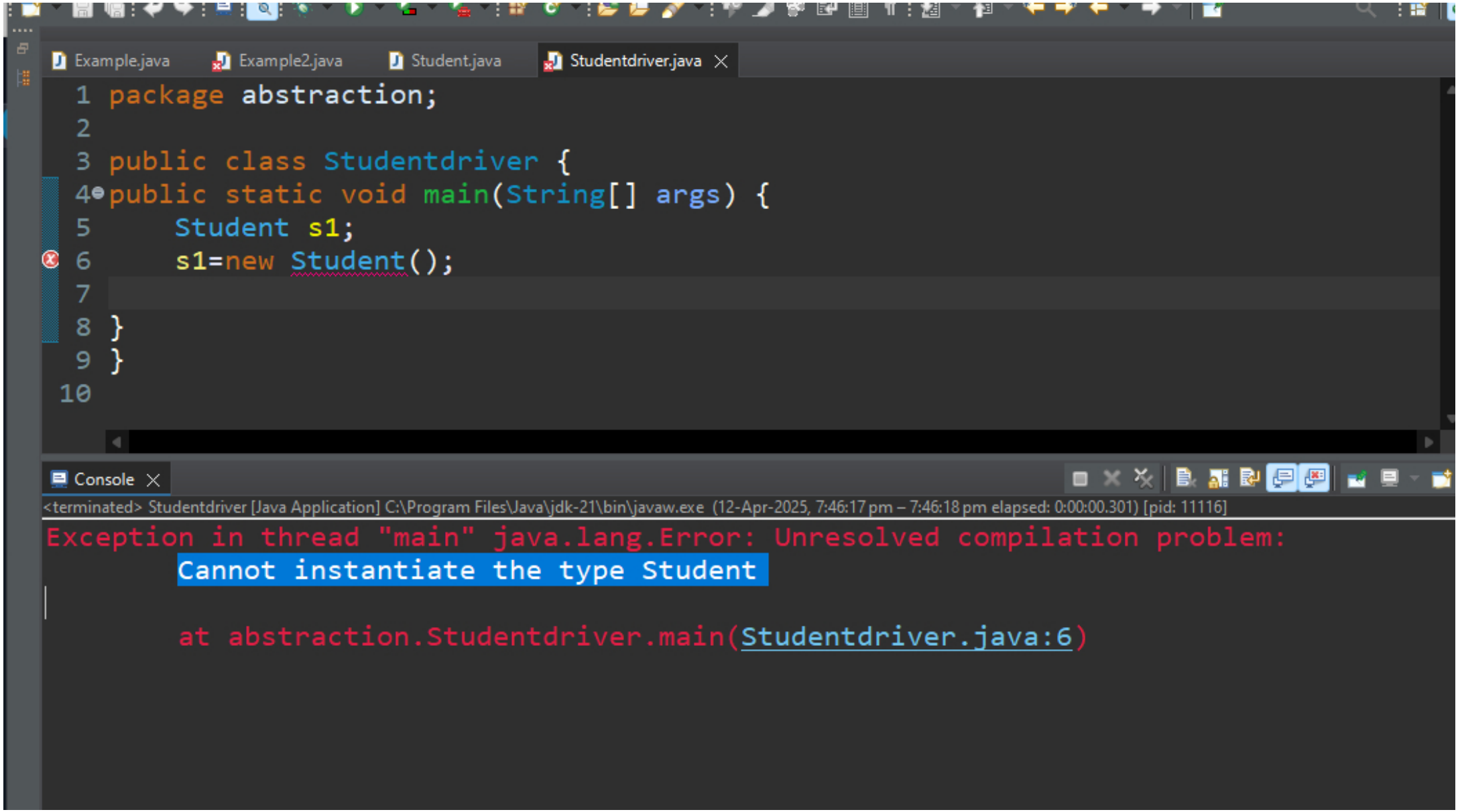
```
class StudentImplemented extends Student{
     p s v m(s[] args){
     new StudentImplemented( ).attendClass( );
}
}
```


Console
*Example.java
1 package abstraction;
2
3 class Example {
4 abstract void demo();//abstract method
5 }
6


Test.java
Rosy.java
Dingi.java
*DriverRosy.java
1 package polymorphism;
2
3 public class DriverRosy {
4 public static void main(String[] args) {
5 Rosy r=new Rosy();
6 r.demo();
7 r=new Dingi();//Upcasting
8 r.demo();
9
10 }
11 }
Console
<terminated> DriverRosy [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (11-Apr-2025, 7:48:12 pm – 7:48:13 pm elapsed: 0:00:00.265) [pid: 619
Hey Dingii
Learn Classical
No mummaa
Let me do Salsa with my Dinga
No mummaa
Let me do Salsa with my Dinga

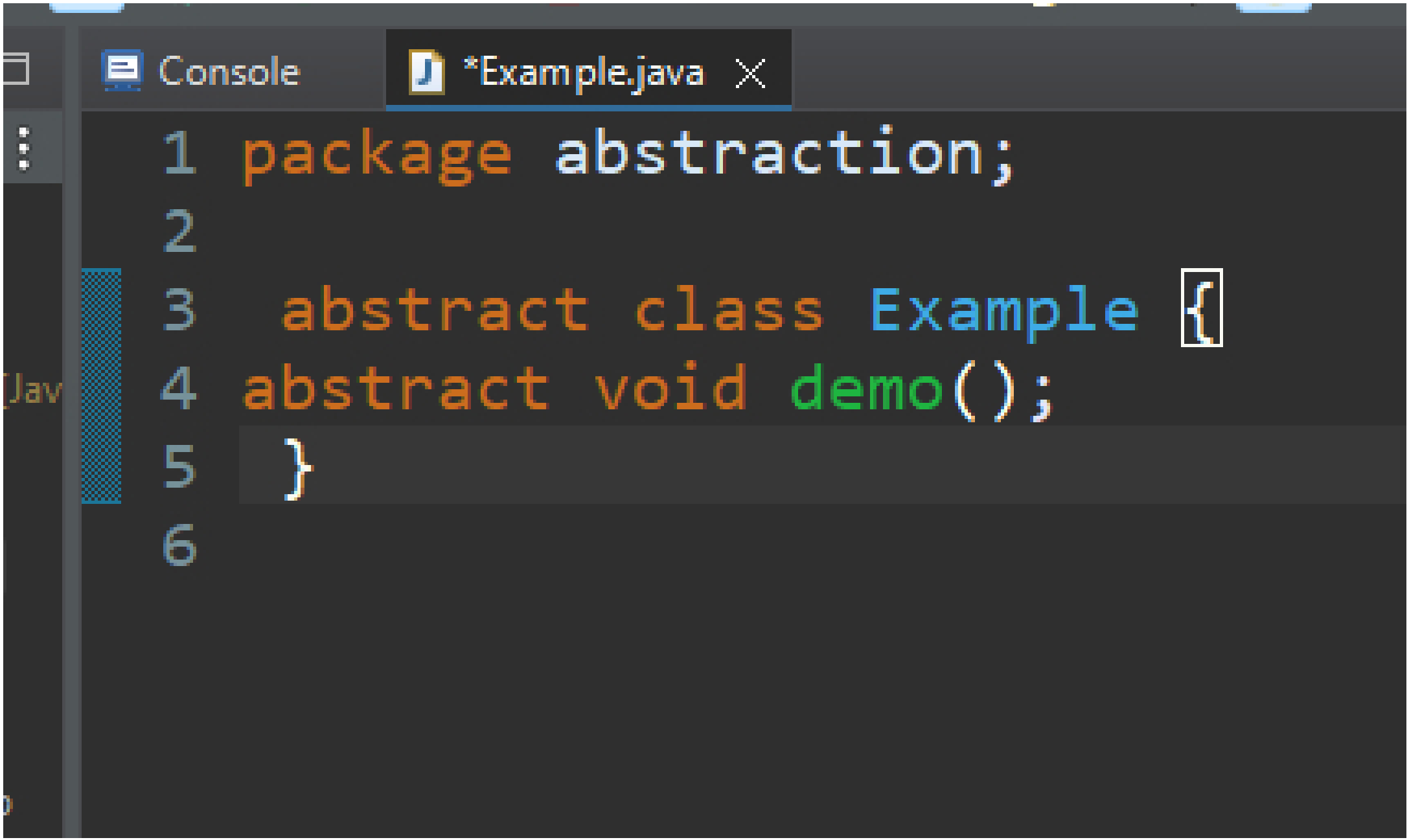

Test.java
Rosy.java
Dingi.java
*DriverRosy.java
1 package polymorphism;
2
3 public class Rosy {
4 void demo() {
5 System.out.println("Hey Dingii");
6 System.out.println("Learn Classical");
7 }
8 }
9
Console
<terminated> DriverRosy [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (11-Apr-2025, 7:48:12 pm – 7:48:13 pm elapsed: 0:0
Hey Dingii
Learn Classical
No mummaa
Let me do Salsa with my Dinga
No mummaa
Let me do Salsa with my Dinga

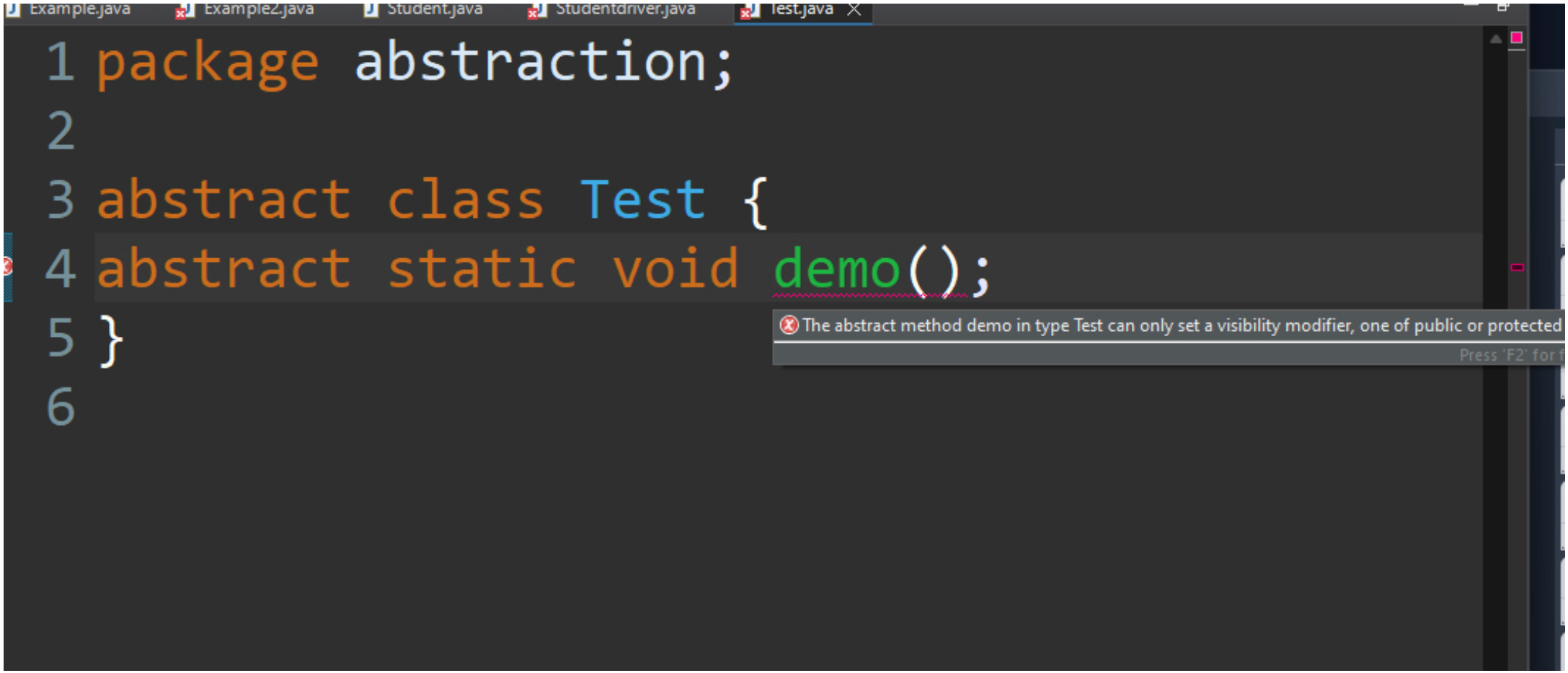

Console
*Example.java
Example2.java
Student.java
1 package abstraction;
2
3 abstract class Student {
4 String name;
5 Student(String name){//Constructor
6 this.name=name;
7 }
8 void demo() { //Concrete Method
9 System.out.println("Hello from Concrete method");
10 }
11 abstract void bunk();
12 }
13


Example.java
Example2.java
Student.java
Studentdriver.java
1 package abstraction;
2
3 public class Studentdriver {
4 public static void main(String[] args) {
5 Student s1;
6 s1=new Student();
7
8 }
9 }
10
Console
<terminated> Studentdriver [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (12-Apr-2025, 7:46:17 pm – 7:46:18 pm elapsed: 0:00:00.301) [pid: 11116]
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Cannot instantiate the type Student
at abstraction.Studentdriver.main(Studentdriver.java:6)

Example.java | Example2.java | Student.java | Studentdriver.java | Test.java | Testdriver.java

```
1 package abstraction;
2
3 class Testdriver extends Test {
4 void demo() {//Overriding
5     System.out.println("Hai");
6 }
7 }
8
```


Console
*Example.java
1 package abstraction;
2
3 abstract class Example {
4 abstract void demo();
5 }
6


Example.java
Example2.java
Student.java
Studentdriver.java
Test.java
1 package abstraction;
2
3 abstract class Test {
4 abstract static void demo();
5 }
6
The abstract method demo in type Test can only set a visibility modifier, one of public or protected
Press 'F2' for

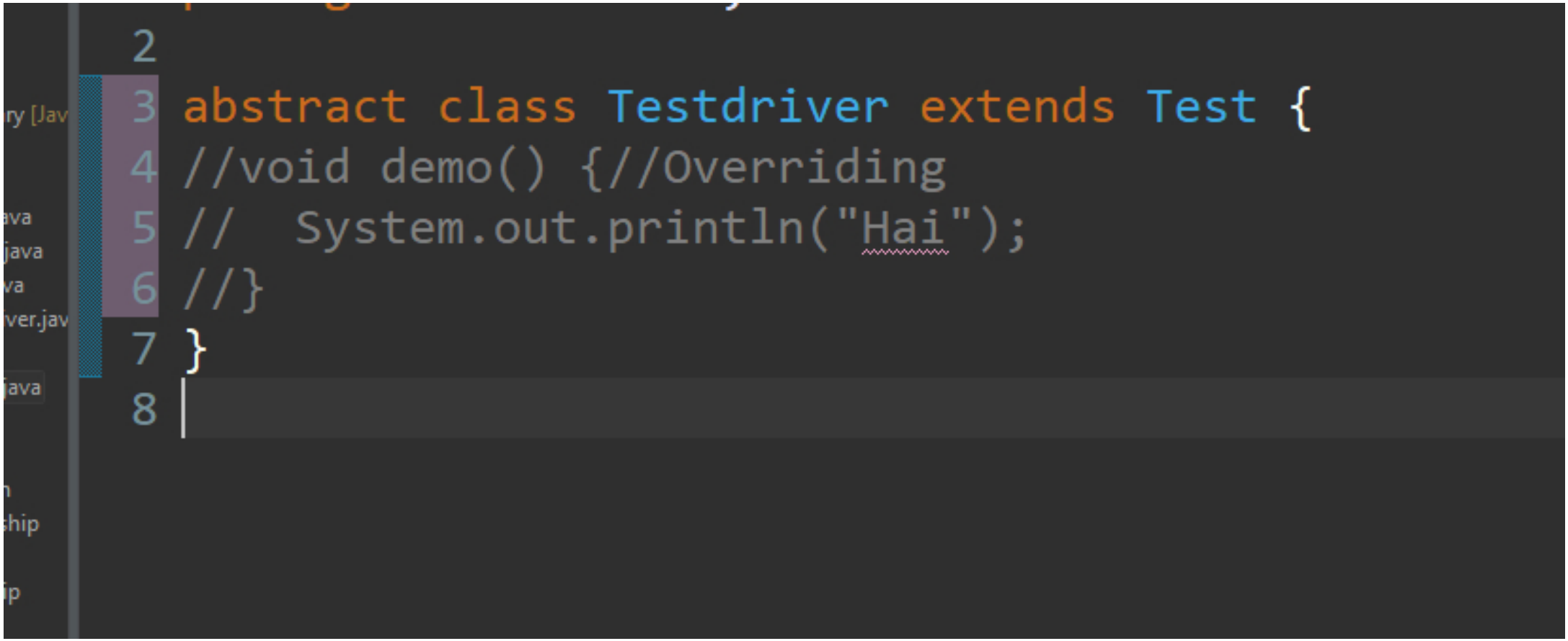
Test.java | Rosy.java | Dingi.java | *DriverRosy.java

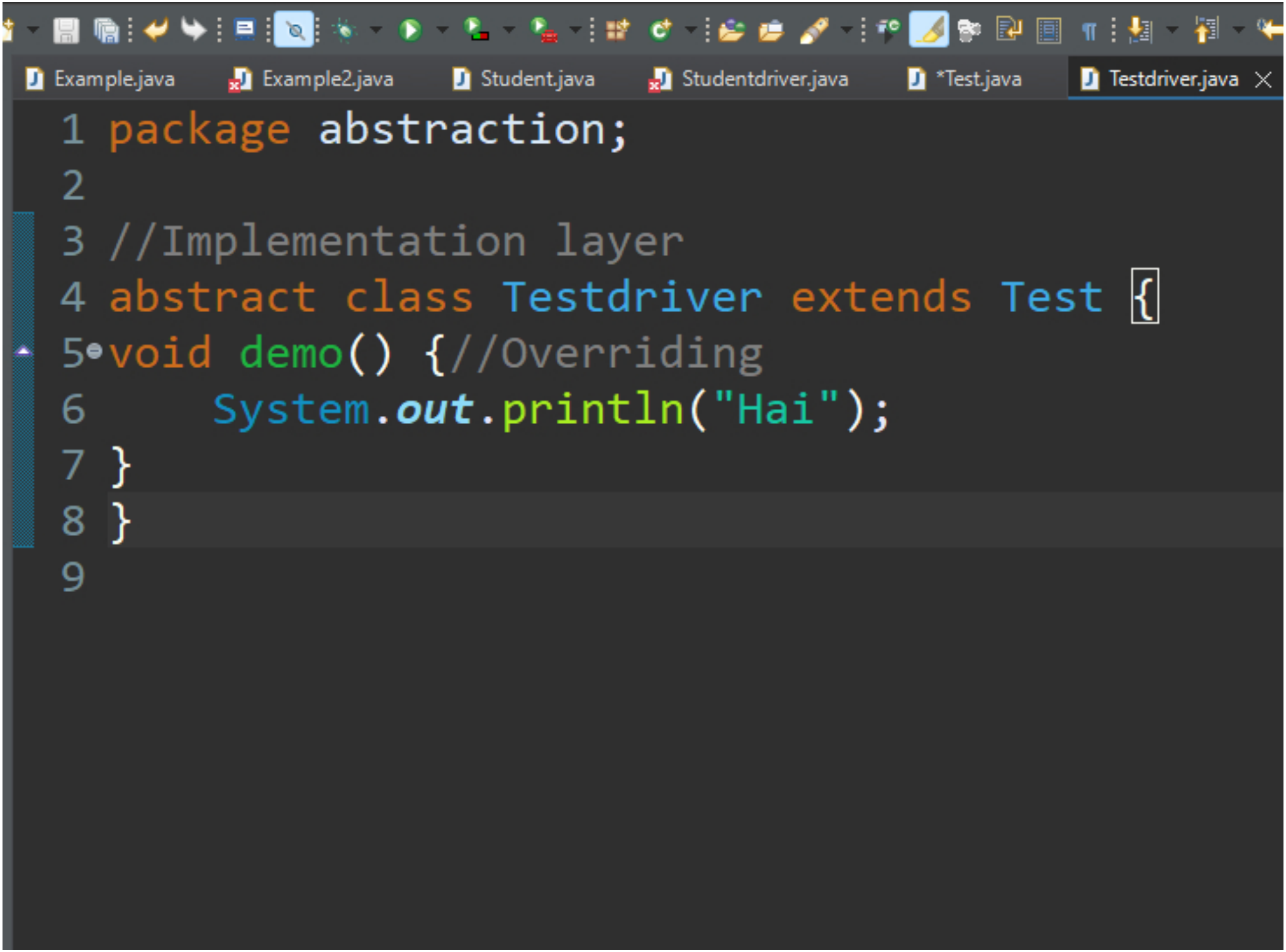
```java
package polymorphism;

public class Dingi extends Rosy {
void demo() {
    System.out.println("No mummaa");
    System.out.println("Let me do Salsa with my Dinga");
}
}

```

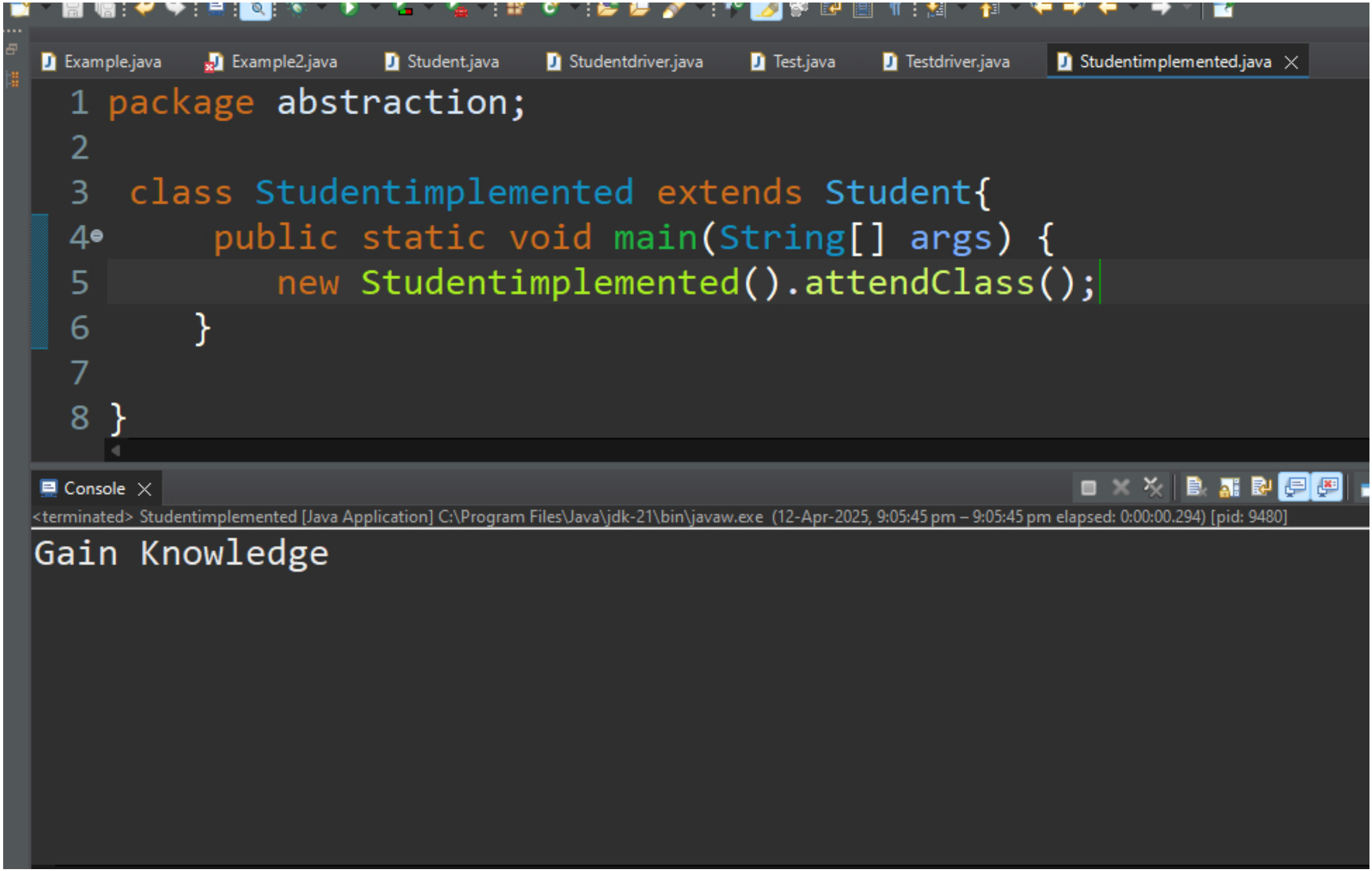
Console

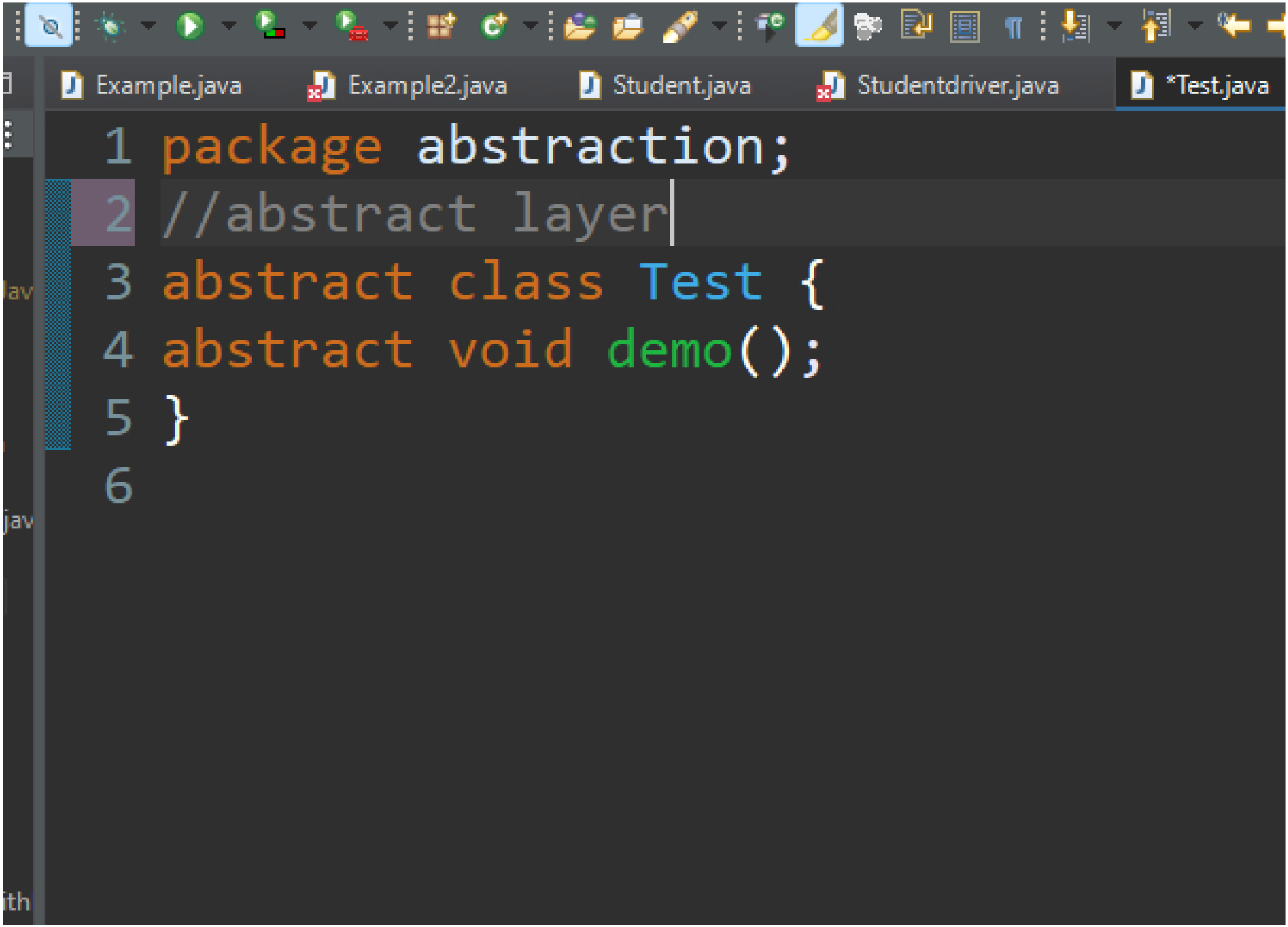
<terminated> DriverRosy [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (11-Apr-2025, 7:48:12 pm – 7:48:13 pm elapsed: 0:00:00.265) [pid: 6196]

```
Hey Dingii
Learn Classical
No mummaa
Let me do Salsa with my Dinga
No mummaa
Let me do Salsa with my Dinga
```


2
abstract class Testdriver extends Test {
//void demo() {//Overriding
//  System.out.println("Hai");
//}
}
8


Example.java
Example2.java
Student.java
Studentdriver.java
*Test.java
Testdriver.java
1 package abstraction;
2
3 //Implementation layer
4 abstract class Testdriver extends Test {
5 void demo() {//Overriding
6 System.out.println("Hai");
7 }
8 }
9

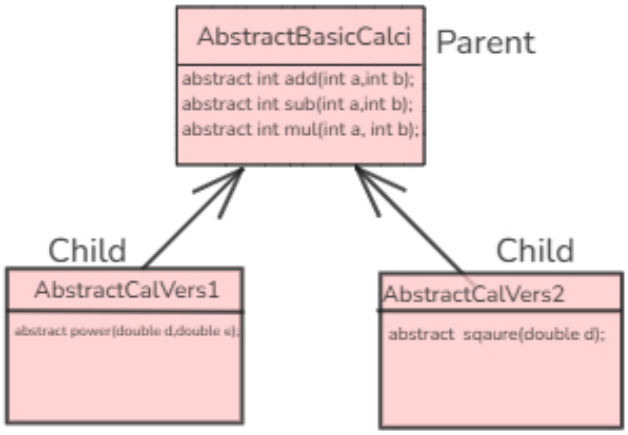

Example.java
Example2.java
Student.java
Studentdriver.java
Test.java
Testdriver.java
Studentimplemented.java
1 package abstraction;
2
3 class Studentimplemented extends Student{
4 public static void main(String[] args) {
5 new Studentimplemented().attendClass();
6 }
7
8 }
Console
<terminated> Studentimplemented [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (12-Apr-2025, 9:05:45 pm – 9:05:45 pm elapsed: 0:00:00.294) [pid: 9480]
Gain Knowledge

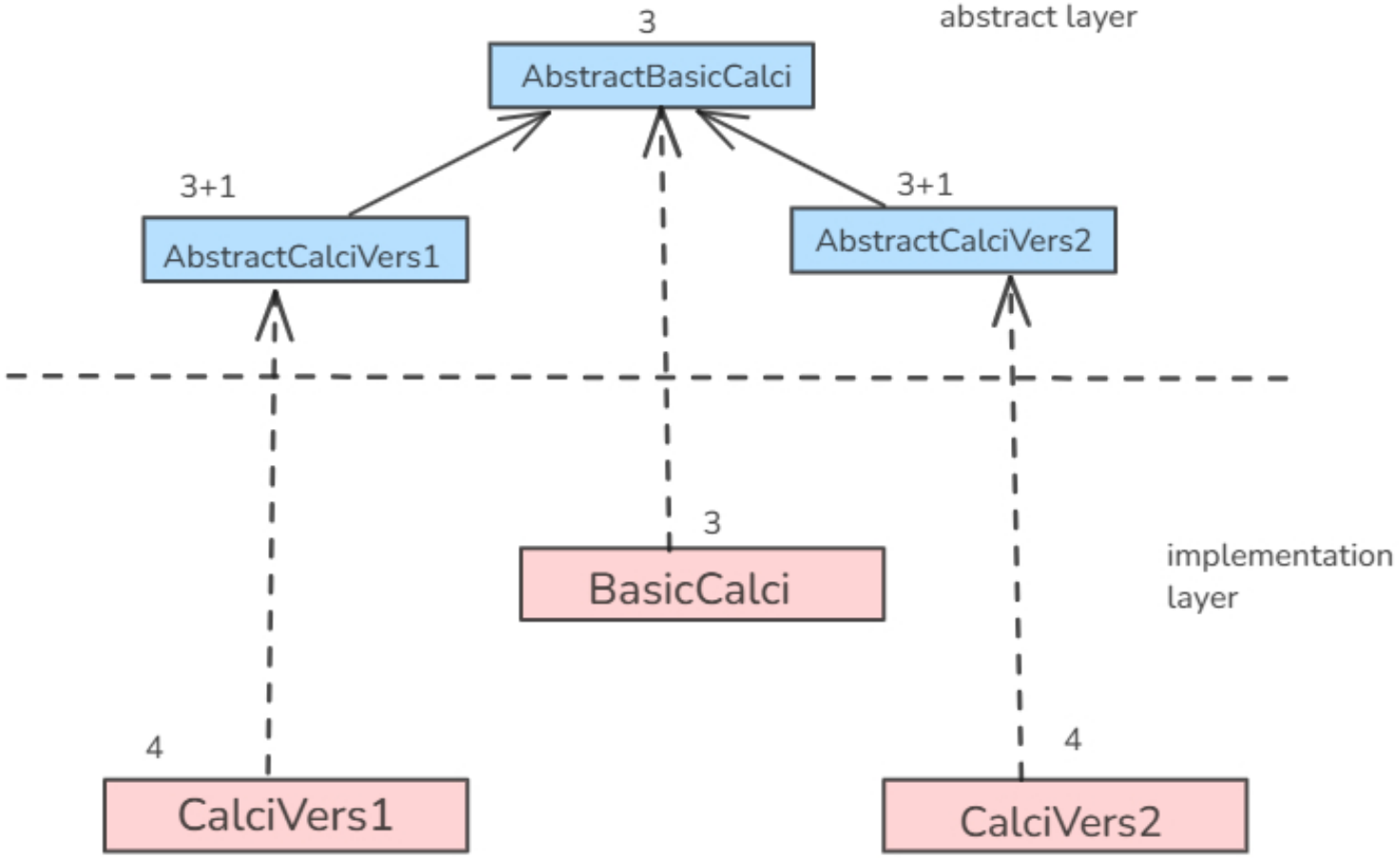

Example.java
Example2.java
Student.java
Studentdriver.java
*Test.java
1 package abstraction;
2 //abstract layer
3 abstract class Test {
4 abstract void demo();
5 }
6

Example.java | Example2.java | Student.java | Studentdriver.java | *Test.java | *Testdriver.java

```
1 package abstraction;
2 //make class as abstract
3 //Implementation layer
4 abstract class Testdriver extends Test {
5
6 }
7
```

Example :


AbstractBasicCalci
abstract int add(int a,int b);
abstract int sub(int a,int b);
abstract int mul(int a, int b);
Parent
Child
AbstractCalVers1
abstract power(double d,double e);
Child
AbstractCalVers2
abstract sqaure(double d);

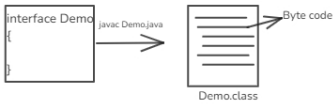

3
abstract layer
AbstractBasicCalci
3+1
AbstractCalciVers1
3+1
AbstractCalciVers2
3
BasicCalci
implementation
layer
4
CalciVers1
4
CalciVers2

Interface : it is a keyword which is used to create an abstract component

Syntax : interface Name{

}

-->we can create interface inside Java File
-->if we compile the interface , we will get class file itself




-->The name of the classfile is same as Interface name

Examples :

```
interface Demo1
{
  void test( );
}
```

-->it is a non-static abstract method.
-->test( ) is by default abstract and public in interface

```
interface Demo2{
  void test( ){
    Sopln("Hai");
  }
}
```

-->it is a concrete method
-->whenever we create non-static method by default it will be abstract , hence we cannot create body

-->interface cannot have concrete non-static methods

```
interface Demo3{
 p s v m(S[] args){
 Sopln("Hai");
 }
}
```

-->it is successfully compiled and executed
-->interface allows static methods
-->public static methods are allowed
-->we can use interface for execution

```
interface Demo4{
public static final int a=10;
}
```

--> a is a variable which is static and visibility is public
-->final means we cannot modify

```
interface Demo1 {
    public static final int a=10;
    int b=20;
    public static void main(String[] args) {
        System.out.println(a);
        Demo1.a=20;
    }
}
```

-->we cannot re-intialize the final variables

```
interface Demo1 {
    int b=20;
    public static void main(String[] args) {
        System.out.println(b);
        Demo1.b=15;
    }
}
```

-->whenever we create variable non-static inside the interface , by default it is static public and final
-->we have to intialize the variable because by default t is final
-->final variables must be intialized

```
interface Demo5{
    Demo5( ){

}
}
```

-->Constructors are not allowed

What can we Create Inside Interface ???

1)public abstract non-static methods is allowed
2)public static final variables is allowed
3)static method is allowed(from jdk 8)

What are not allowed inside Interface ???

1)non-static variables are not allowed
2)initializers are not allowed
3)non-public abstract methods are not allowed
4)non-public static methods are not allowed

with respect to interfaces , we can achieve inheritance in 2 ways

1)inheritance between interface and interface
2)inheritance between class and interface

Note : we can achieve inheritance between class and interface

Parent → Interface ✓
Child → class

Parent → class ✗
Child → Interface

a class can inherit an interface but an interface cannot inherit a class

What can we Create Inside Interface ???

1)public abstract non-static methods is allowed
2)public static final variables is allowed
3)static method is allowed(from jdk 8)

What are not allowed inside Interface ???

1)non-static variables are not allowed
2)initializers are not allowed
3)non-public abstract methods are not allowed
4)non-public static methods are not allowed

with respect to interfaces , we can achieve inheritance in 2 ways

1)inheritance between interface and interface
2)inheritance between class and interface

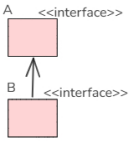
Note : we can achieve inheritance between class and interface



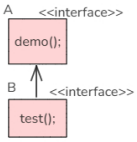
a class can inherit an interface but an interface cannot inherit a class

--> we can achieve inheritance between 2 interfaces using extends keyword

Ex :



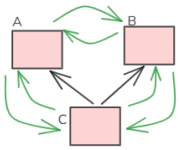

```
interface A
{

}
interface B extends A
{

}
```




```
interface A
{
   void demo();
}
interface B extends A
{
void test();
}
```

2 abstract methods

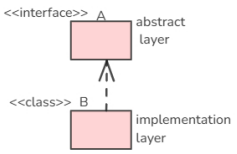
Note : with the help of interface we can achieve multiple inheritance




```
interface A{
}
interface B{
}
interface C extends A,B
{

}
```

class and interface :

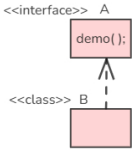
-->we can achieve inheritance between class and interface with the help of implements keyword




```
interface A{
}

class B implements A
{

}
```

Ex :

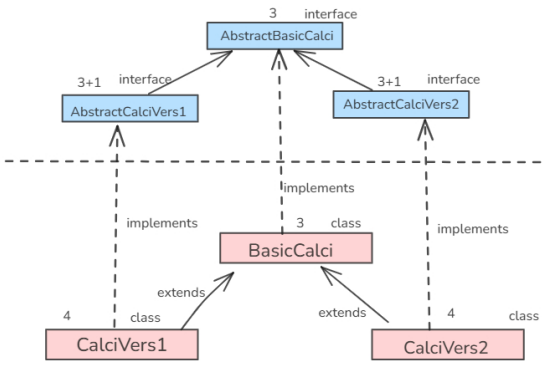
case 1




```
interface A{
void demo( );
}
abstract class B implements A
{

}
```

case 2

```
interface A {
  void demo( );
}
class B implements A
{
void demo( ){
  ---------
  ---------
}
}
```



## Object Class

-->built in class
-->java.lang package
-->by default Object class members are
imported by the compiler
-->Object class is the supermost parent
class for all the classes in Java
-->in object class we have 11 non-static methods

Ex

```
class Book
{

}
```

Book b1=new Book( );

--> inside Book 11 non-sttic methods aare available
-->we can change the implementation of those methods by
Overriding

```
class Book {
 void read( ){
}
}
```

11 inherited methods
1 incorporated method

### 11 non-static methods :

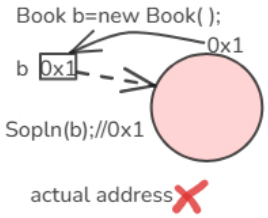
1. public String toString( )
2.public boolean equals(Object o)
3.public int hashCode( )
4.protected void finalize( )
5.final public void wait( )
6.final public void wait(long l)
7.final public void wait(long l,int i)
8.final public void notify( )
9.final publoic void notifyAll( )
10.protected Object clone( )
11.final public void getClass( )

### toString( ) : public String toString( )

-->tostring( ) returns String
-->it is a no argument method
-->toString( ) give you the logical address
in String Format
-->it modifies the address into hexadecimal value

ClassName@HexaDecimal

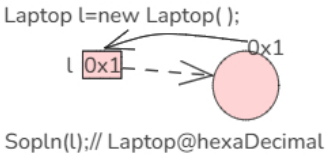
### What happens when Object reference is printed in Java ???




--> we dont get the actual address , we get some data
which is returned by the toString( )

-->toString( ) will give the address
->it takes name of the class
->converts real address to hexadecimal number
->concatinate both ClassName and hexadecimal number
and gives String

```
class Laptop{

}
```



B.java

```
1 package interfacee;
2
3 interface B {
4
5 }
6
```

C.java

```
1 package interfacee;
2
3  interface C extends A,B{
4
5 }
6
```

A.java

```
1 package interfacee;
2
3 interface A {
4
5 }
6
```

Writable | Smart Insert | 3 : 1 : 23

29°C

equals(Object o) : public boolean equals(Object o)

-->after execution it return true or false
-->the type of formal argument is object

Generalized Container

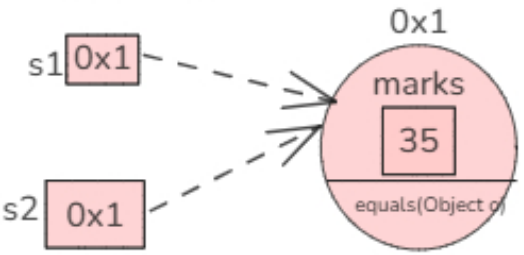
-->to compare the reference of current object with passed object
-->if the reference is same , it returns true
-->if the reference is not same, it returns false
-->it can accept only 1 argument

```
Current_Obj_Ref . equals(Other_Obj_Ref)
```

Ex :
```
class Student{
    double marks;
     Student(double marks){
      this.marks=marks;
       }
     }
```

Case 1 :

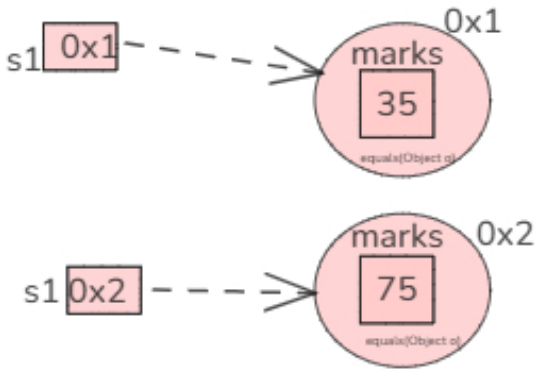
```
Student s1=new Student(35);
Student s2=s1;
```




```
Sopln(s1==s2); //true
Sopln(s1.equals(s2));//true
```

Case 2:

```
Student s1=new Student(35);
Student s2=new Student(75);
```




```
Sopln(s1==s2); // false
Sopln(s1.equals(s2)); // false
```

Why to override the equals( ):

-->To compare the states of an Object , we need to override equals method.

-->it should compare the properties of current object with properties of passed object

hashCode( ): public int hasCode( )

-->hashCode( ) is to check hash values

-->if 2 objects are equal (obj1.equals(obj2)) returns true , then thier hashcode must be same

# STRINGS

--> enclosed within double quotes

-->String is a group of Characters

--> in java we represent String with " "

-->In Java , String is stored using Object


-->In Java , to store the String we have 3 Built-in Classes

1)java.lang.String
2)java.lang.StringBuilder
3)java.lang.StringBuffer

java.lang.String :

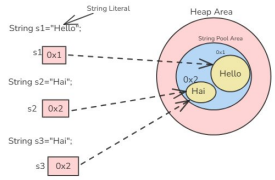
-->it is a non-primitive Datatype
-->The object class methods such as
toString()
equals(Object o)
hashCode( ) is overridden

Constructors of String Class :

1) String( )
2)String(String)
3)String(char[ ])
4)String(StringBuilder)
5)String(StringBuffer)

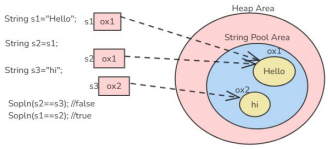
-->any data written within " " is known as String Literal

Ex :




Sopln(s2==s3) ;// true

Ex :



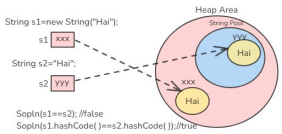

Sopln(s2==s3); //false
Sopln(s1==s2); //true

String Object using new keyword :

Syntax :

new String( )

new String(String)




Sopln(s1==s2); //false
Sopln(s1.hashCode( )==s2.hashCode( ));//true

Characteristics of String :

1)String is a Sequence of Characters
you must use a String DataType, even the characters all are numeric
int a=0056;
Sopln(a);

2)we can find the length of the String by some inbuilt method
String name="Chinki";
Sopln(name.length( )); // 6

3)
An empty String can be Created by giving only double quotes
-->the length of an empty String is Zero
-->the space is a character and it has length of 1

```
String s=" ";
System.out.println(s.length());
```

4)String is Immutable

-->Once String Object is created , it cannot be modified
-->any operation on String , actually creates a new String Object

5)Characters in a String are Indexed Starting from 0


6)String is a Sequence of Characters or array of Characters

char[ ] c={'d','i','n','g','a'};
String s=new String(c); //dinga

String Comparision:

In Java we can compare the String in 3 ways
1.equals( )
2.compareTo( )
3.== operator

1.equals( ) : to compare the content of 2 Strings is same or not
-->if content is same it returns True, else it returns false
-->the return type is boolean

String s1="hai";
String s2="bye";
boolean a=s1.equals(s2);
Sopln(a); //false

String s1="Hai";
String s2="hai";
boolean b=s1.equals(s2);
Sopln(b);❌

s1.equalsIgnoreCase(s2)

it will ignore the case of letters

2.compareTo( ):

it compares the two Strings
-->if they are same /equal it returns 0
-->if s1 > s2 +ve Value
-->if s1 < s2 -ve Value

Syntax : int compareTo(string)

String s1="hai";
String s2="hai";
int a=s1.compareTo(s2); //0

String s1="hai";
String s2="Hai";
int a=s1.compareTo(s2); //32

s1.compareToIgnoreCase(s2)

it ignorescorresponding case values

int a=s1.compareToIgnoreCase(s2);

3.== operator :

it checks whether the reference of the 2 Strings is same or not

-->if reference is equal it returns true, else it returns false

String s1="hai";
String s2="hai";
Sopln(s1==s2); // true

Methods of String Class :

1.concat( )
2.length( )
3.charAt( )
4.compareTo( )
5.equals( )
6.startsWith( )
7.endsWith( )
8.indexOf( )
9.lastIndexOf( )
10.subString( )
11.toLowerCase( )
12.toUpperCase( )
13.trim( )
14.split( )
15.replace( )

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//split( ) -
//Syntax : String split(String)
public class Test {
public static void main(String[] args) {
    String s1="Hello to You";
    String[] a=s1.split("lo");
    System.out.println(a[0]);
    System.out.println(a[1]);


}
}
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 9:06:16 pm – 9:06:16 pm elapsed: 0:00:00.185) [pid

```
Hel
 to You
```

Test.java

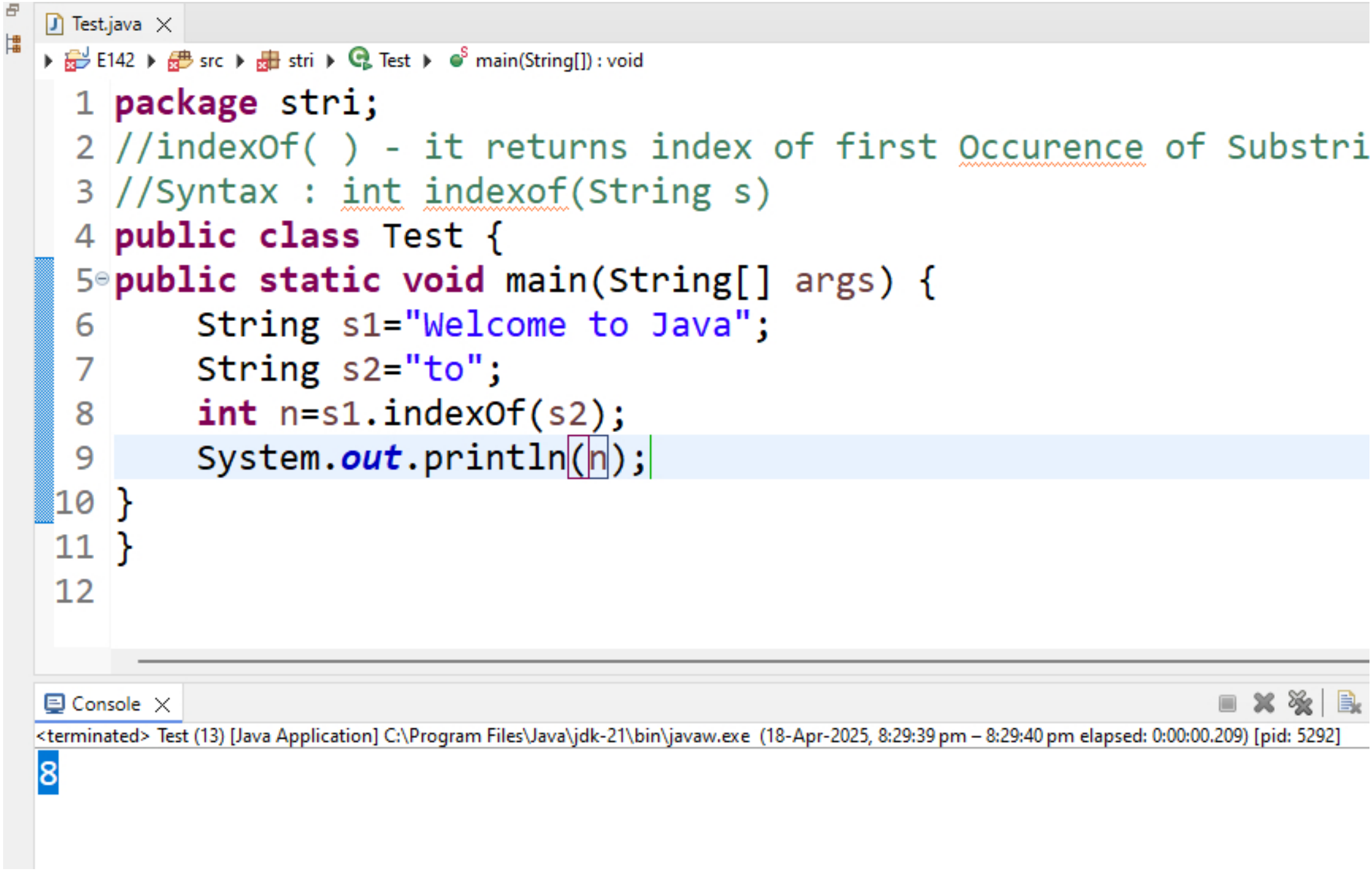
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//subString( ) - it returns subString of Characters
//Syntax - String sunString(int i)
public class Test {
public static void main(String[] args) {
    String s1="Truth is God";
    String s2=s1.substring(6);
    System.out.println(s2);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:46:38 pm – 8:46:38 pm elapsed: 0:00:00.169) [pid: 2820]

```
is God
```


Test.java
E142 src stri Test main(String[]) : void
1 package stri;
2 //indexOf( ) - it returns index of first Occurence of Substri
3 //Syntax : int indexof(String s)
4 public class Test {
5 public static void main(String[] args) {
6 String s1="Welcome to Java";
7 String s2="to";
8 int n=s1.indexOf(s2);
9 System.out.println(n);
10 }
11 }
12
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:29:39 pm – 8:29:40 pm elapsed: 0:00:00.209) [pid: 5292]
8

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//length( )
//Syntax : int length()
public class Test {
public static void main(String[] args) {
    String s1="Hello World";
    int l=s1.length();
    System.out.println(l);

}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:15:32 pm – 8:15:32 pm elapsed: 0:00:00.274) [pid: 1

```
11
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//trim( )-it removes the blank spaces at begining and ending
//Syntax : String trim( )
public class Test {
public static void main(String[] args) {
    String s1=" Network is Weak ";
    String s2=s1.trim();
    System.out.println(s2);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:59:29 pm – 8:59:29 pm elapsed: 0:00:00.172) [pid: 16016]

```
Network is Weak
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//indexOf( ) - it returns index of first Occurence of Substring in r
//Syntax : int indexof(String s)
public class Test {
public static void main(String[] args) {
    String s1="Welcome to Java";
    String s2="s";//if it is not present it returns -1
    int n=s1.indexOf(s2);
    System.out.println(n);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:30:54 pm – 8:30:54 pm elapsed: 0:00:00.173) [pid: 10332]

```
-1
```

va

l2 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```
package stri;
//indexOf( ) - it returns index of first Occurence c
//Syntax : int indexof(String s)
public class Test {
public static void main(String[] args) {
    String s1="Welcome to Java";
    String s2="s";//if it is not present it returns
    int n=s1.indexOf(s2);
    System.out.println(n);
}
}
```

ole

ed> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:30:54 pm – 8:30:54 pm elapsed: 0:0

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//toUppercase( )-it converts lowercase characters of a String UpperCase ch
//Syntax : String toUpperCase( );
public class Test {
public static void main(String[] args) {
    String s1="dinga";
    String s2=s1.toUpperCase();
    System.out.println(s2);

}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:56:06 pm – 8:56:06 pm elapsed: 0:00:00.186) [pid: 9380]

```
DINGA
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```
1  package stri;
2  //toLowercase( )-it converts uppercase characters of a String lowerCase ch
3  //Syntax : String toLowerCase( );
4  public class Test {
5  public static void main(String[] args) {
6      String s1="KaRmA";
7      String s2=s1.toLowerCase();
8      System.out.println(s2);
9
10 }
11 }
12
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:54:43 pm – 8:54:43 pm elapsed: 0:00:00.185) [pid: 16580]

```
karma
```

```java
package stri;
//lastIndexOf( ) - it returns index of last Occurence of Substring in mai
//Syntax : int lastIndexof(String s)
public class Test {
public static void main(String[] args) {
    String s1="Hakuna Matata";
    String s2="ta";
    int a=s1.lastIndexOf(s2);
    System.out.println(a);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:39:30 pm – 8:39:31 pm elapsed: 0:00:00.180) [pid: 14556]

```
11
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//indexOf( ) - it returns index of first Occurence of Substring in
//Syntax : int indexof(String s)
public class Test {
public static void main(String[] args) {
    String s1="Welcome to Java";
    String s2="e";
    int n=s1.indexOf(s2);
    System.out.println(n);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:30:04 pm – 8:30:04 pm elapsed: 0:00:00.186) [pid: 10436]

```
1
```

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;

public class Test {
public static void main(String[] args) {
    char[] c= {'j','a','v','a'};
    String s=new String(c);
    System.out.println(s);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:01:29 pm – 8:01:29 pm elapsed: 0:00:00.213) [pid: 16308]

```
java
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//concat( )
//Syntax : concat(String)
public class Test {
public static void main(String[] args) {
    String s1="Hello";
    String s2="World";
    String s3=s1.concat(s2);
    System.out.println(s3);
}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:13:09 pm – 8:13:10 pm elapsed: 0:00:00.175) [pid: 17324]

```
HelloWorld
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```
1  package stri;
2
3  public class Test {
4  public static void main(String[] args) {
5      String s1="haI";
6      String s2="haii";
7      int b=s1.compareToIgnoreCase(s2);
8      System.out.println(b);
9  }
10 }
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 7:50:18 pm – 7:50:18 pm elapsed: 0:00:00.258) [pid: 18184]

```
-1
```

Test.java

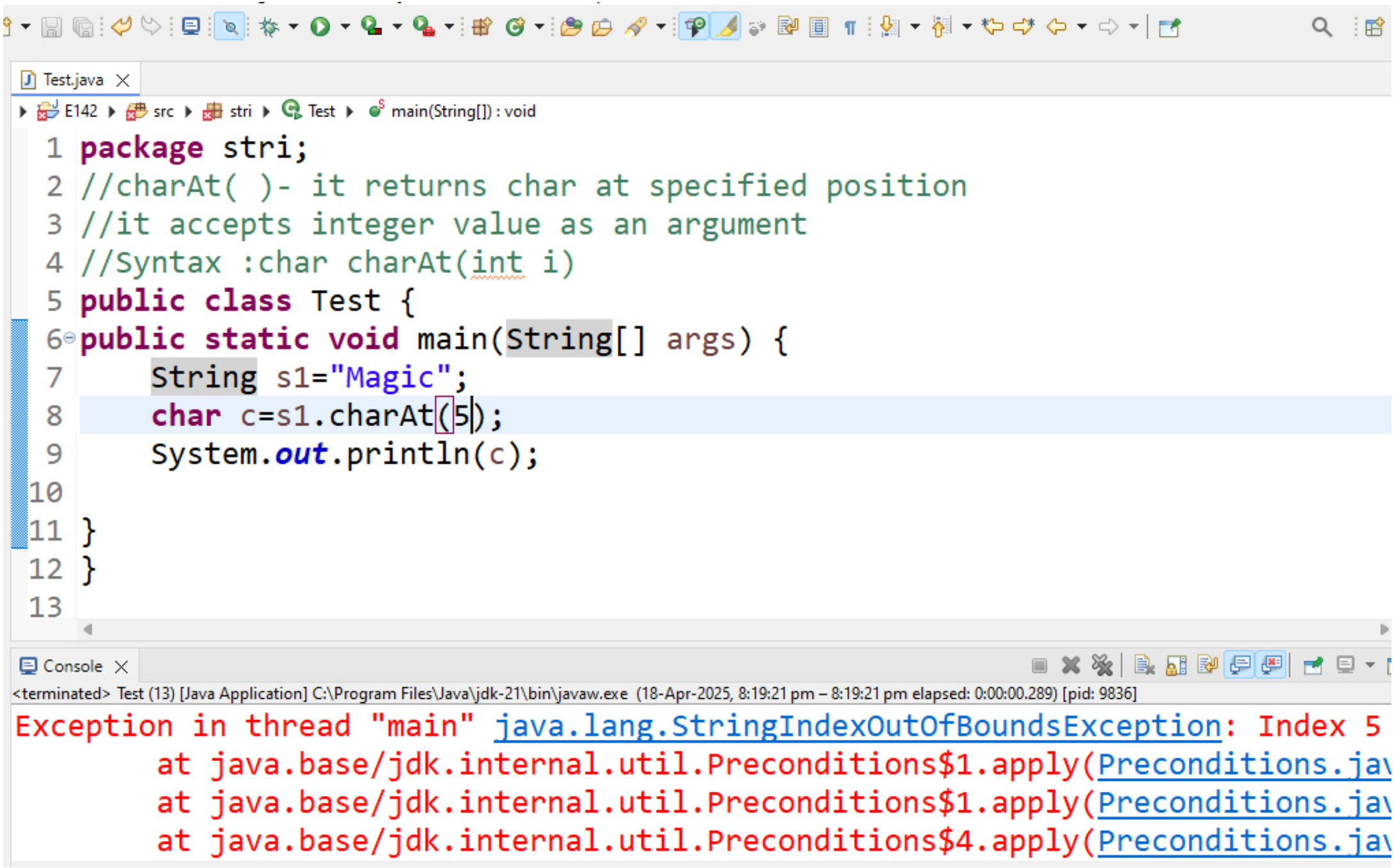
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

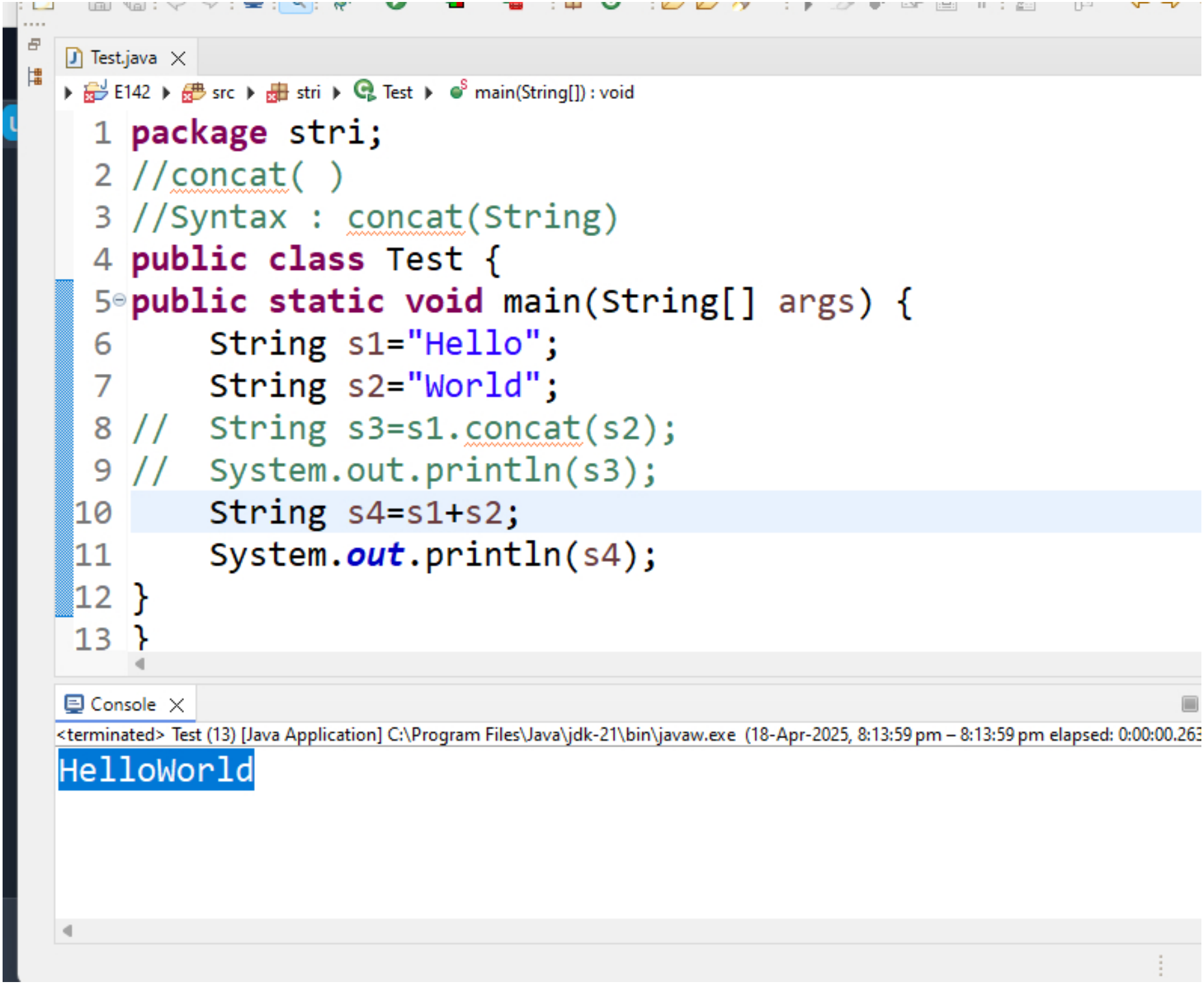
```java
package stri;
//startsWith( ) - it checks whether the String starts with Spec
//String or not
//Syntax : boolean startsWith(String s)
public class Test {
public static void main(String[] args) {
    String s1="Hai Guys";
    String s2="Hai";
    boolean b=s1.startsWith(s2);
    System.out.println(b);

}
}
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:24:09 pm – 8:24:09 pm elapsed: 0:00:00.205) [pid: 8256]

```
true
```


Test.java
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void
1 package stri;
2 //charAt( )- it returns char at specified position
3 //it accepts integer value as an argument
4 //Syntax :char charAt(int i)
5 public class Test {
6 public static void main(String[] args) {
7 String s1="Magic";
8 char c=s1.charAt(5);
9 System.out.println(c);
10
11 }
12 }
13
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:19:21 pm – 8:19:21 pm elapsed: 0:00:00.289) [pid: 9836]
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: Index 5
at java.base/jdk.internal.util.Preconditions$1.apply(Preconditions.jav
at java.base/jdk.internal.util.Preconditions$1.apply(Preconditions.jav
at java.base/jdk.internal.util.Preconditions$4.apply(Preconditions.jav


Test.java
E142
src
stri
Test
main(String[]) : void
1 package stri;
2 //concat( )
3 //Syntax : concat(String)
4 public class Test {
5 public static void main(String[] args) {
6 String s1="Hello";
7 String s2="World";
8 // String s3=s1.concat(s2);
9 // System.out.println(s3);
10 String s4=s1+s2;
11 System.out.println(s4);
12 }
13 }
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:13:59 pm – 8:13:59 pm elapsed: 0:00:00.263
HelloWorld

```java
package stri;
//charAt( )- it returns char at specified position
//it accepts integer value as an argument
//Syntax :char charAt(int i)
public class Test {
public static void main(String[] args) {
    String s1="Magic";
    char c=s1.charAt(3);
    System.out.println(c);

}
}

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:18:57 pm – 8:18:57 pm elapsed: 0:00:00.234) [pid: 10624]

```
i
```

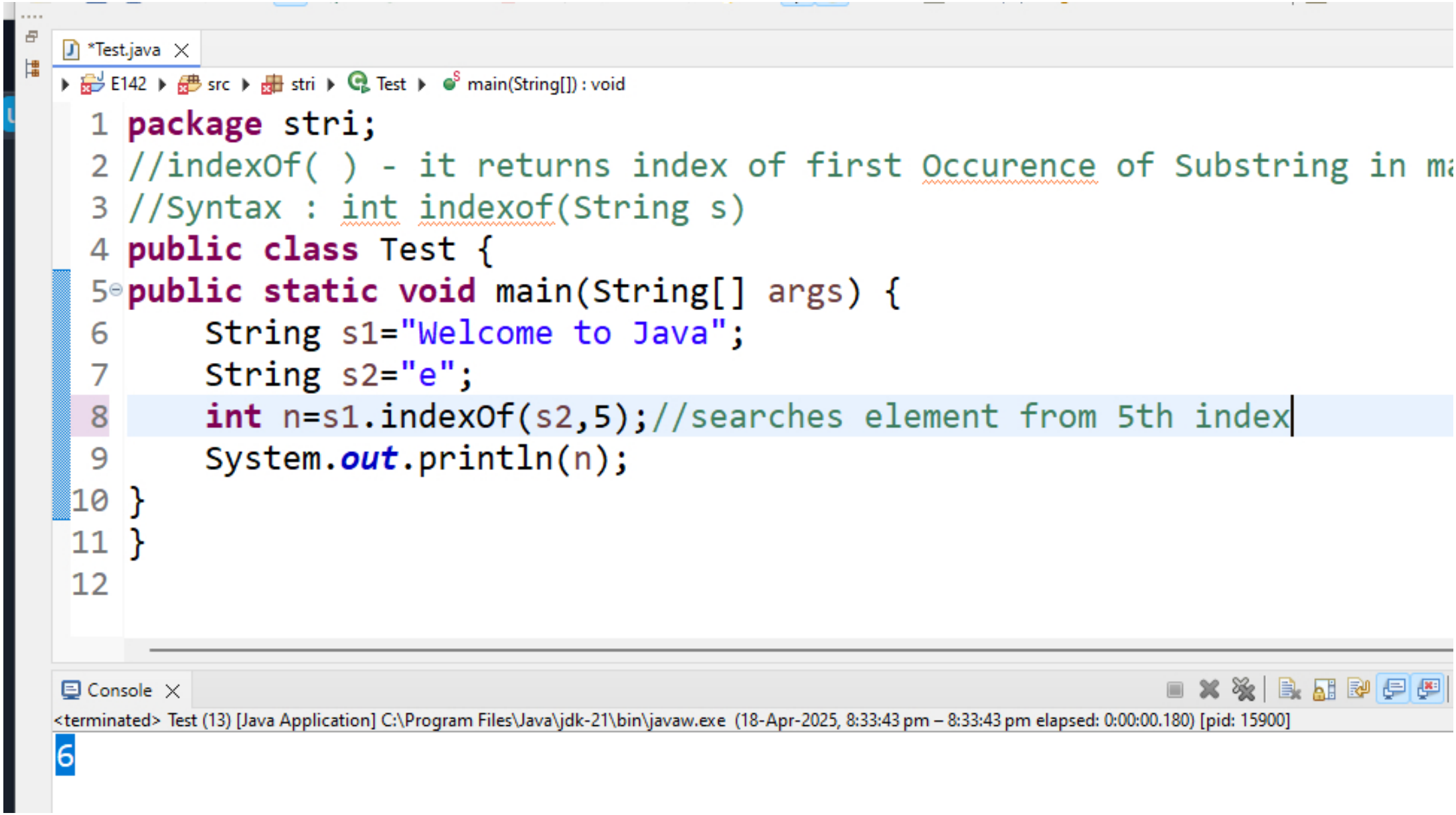
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//endsWith( ) - it checks whether the String ends with Specified
//String or not
//Syntax : boolean endsWith(String s)
public class Test {
public static void main(String[] args) {
    String s1="Hello Buddies";
    String s2="ies";
    boolean b=s1.endsWith(s2);
    System.out.println(b);

}
}
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:25:35 pm – 8:25:35 pm elapsed: 0:00:00.211) [pid: 1912]

```
true
```


*Test.java
E142 src stri Test main(String[]) : void
1 package stri;
2 //indexOf( ) - it returns index of first Occurence of Substring in ma
3 //Syntax : int indexof(String s)
4 public class Test {
5 public static void main(String[] args) {
6 String s1="Welcome to Java";
7 String s2="e";
8 int n=s1.indexOf(s2,5);//searches element from 5th index
9 System.out.println(n);
10 }
11 }
12
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:33:43 pm – 8:33:43 pm elapsed: 0:00:00.180) [pid: 15900]
6

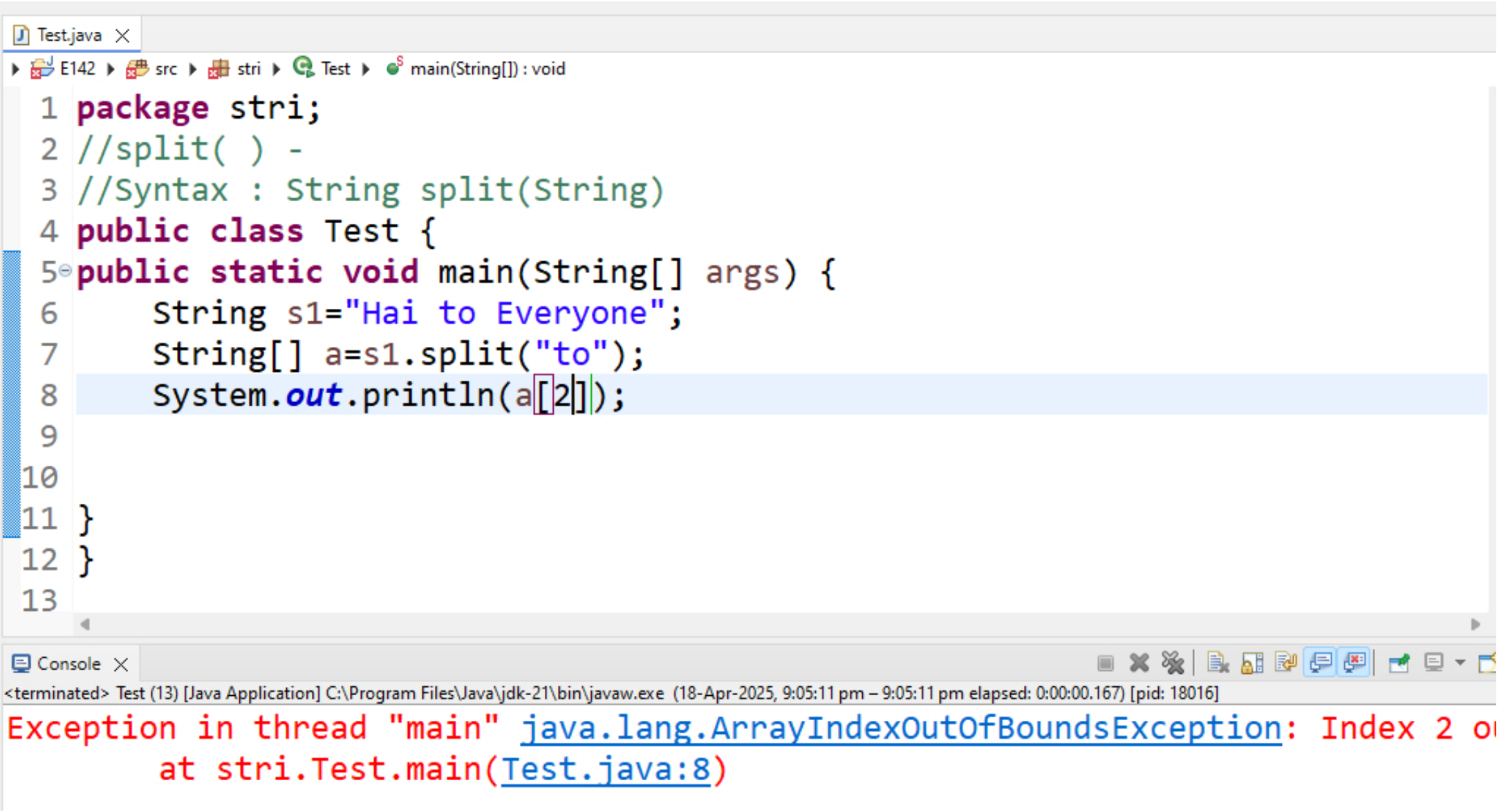
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

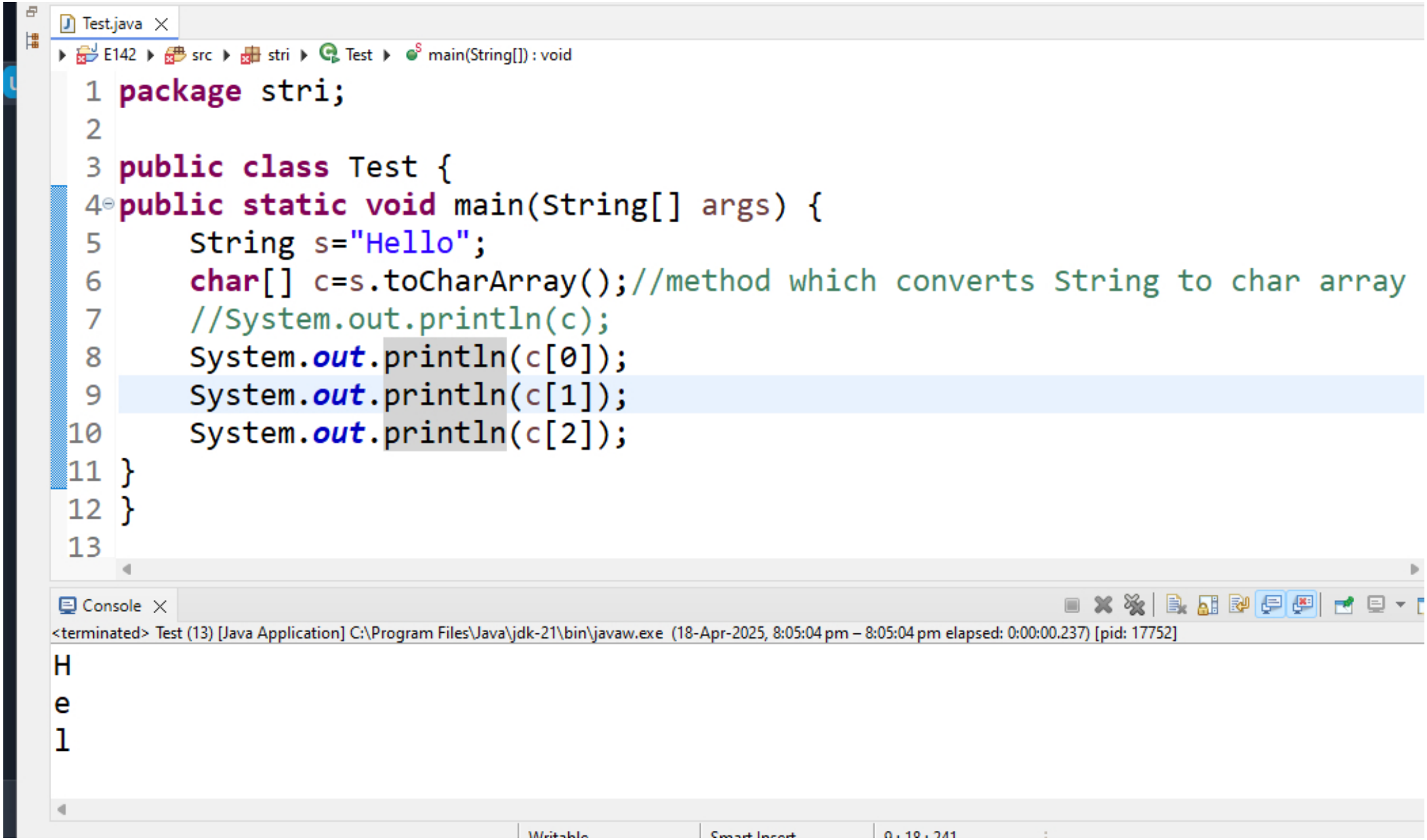
```java
package stri;

public class Test {
public static void main(String[] args) {
    String s1="haII";
    String s2="hai";
    int b=s1.compareToIgnoreCase(s2);
    System.out.println(b);
}
}
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 7:52:33 pm – 7:52:33 pm elapsed: 0:00:00.230) [pid: 16716]

```
1
```


Test.java
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void
1 package stri;
2 //split( ) -
3 //Syntax : String split(String)
4 public class Test {
5 public static void main(String[] args) {
6 String s1="Hai to Everyone";
7 String[] a=s1.split("to");
8 System.out.println(a[2]);
9
10
11 }
12 }
13
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 9:05:11 pm – 9:05:11 pm elapsed: 0:00:00.167) [pid: 18016]
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 2 ou
at stri.Test.main(Test.java:8)


Test.java
E142 src stri Test main(String[]) : void
1 package stri;
2
3 public class Test {
4 public static void main(String[] args) {
5 String s="Hello";
6 char[] c=s.toCharArray();//method which converts String to char array
7 //System.out.println(c);
8 System.out.println(c[0]);
9 System.out.println(c[1]);
10 System.out.println(c[2]);
11 }
12 }
13
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 8:05:04 pm – 8:05:04 pm elapsed: 0:00:00.237) [pid: 17752]
H
e
l

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//replaces( ) - it replace the old character with new String
//Syntax : String replace(char,char);
public class Test {
public static void main(String[] args) {
    String s1="Hello You";
    String s2=s1.replace('H','Y');
    System.out.println(s2);



}
}
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 9:10:55 pm – 9:10:55 pm elapsed: 0:00:00.187) [pid: 14952]

```
Yello You
```

Test.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```java
package stri;
//replaces( ) - it replace the old character with new String
//Syntax : String replace(char,char);
public class Test {
public static void main(String[] args) {
    String s1="Hakuna Matata";
    String s2=s1.replace('a','e');
    System.out.println(s2);



}
}
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (18-Apr-2025, 9:11:38 pm – 9:11:39 pm elapsed: 0:00:00.231) [pid: 10236]

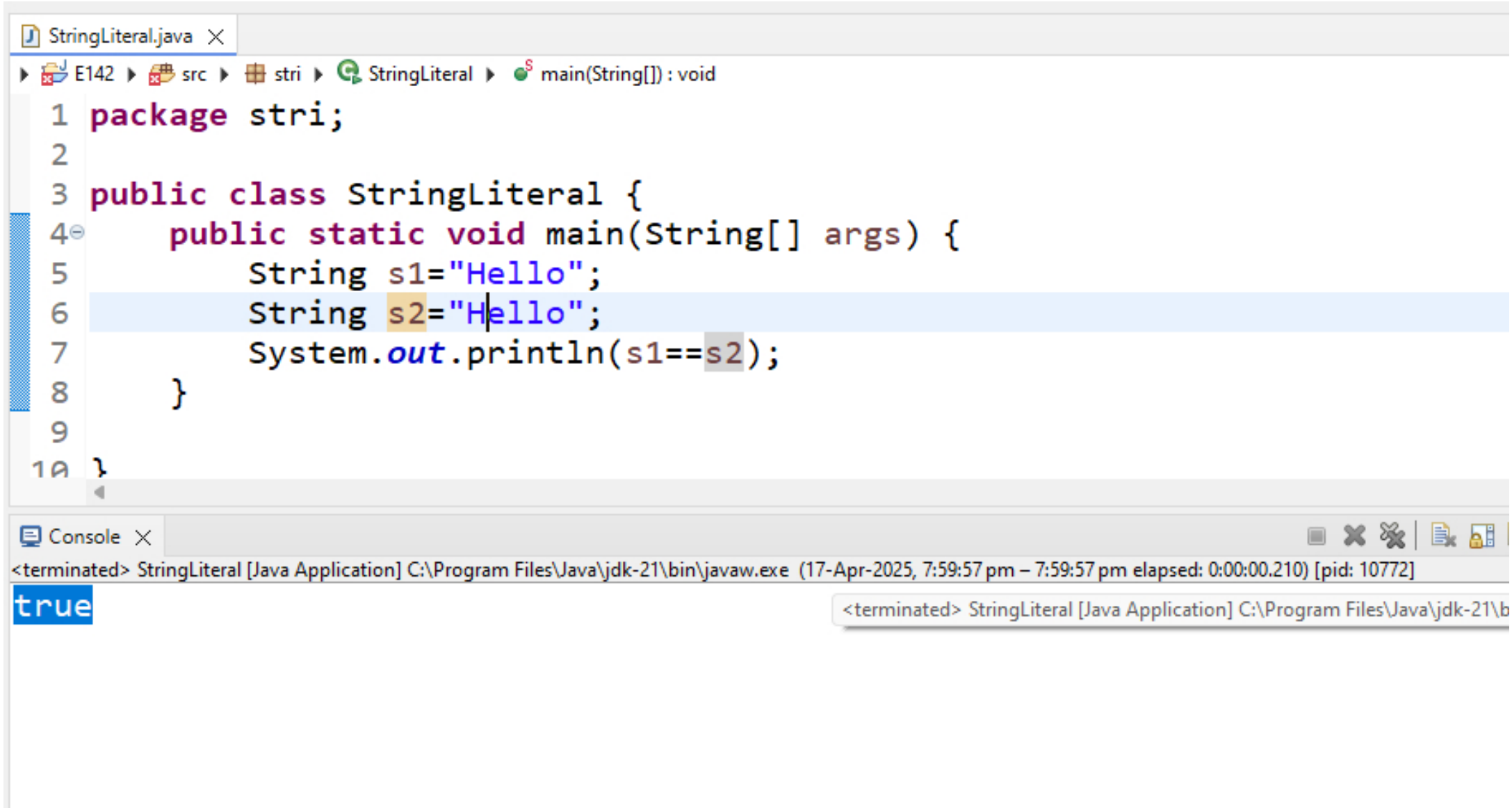
```
Hekune Metete
```

```java
package str1;

public class StringLiteral {
    public static void main(String[] args) {
        String s1="Hello";
        String s2="Hai";
        String s3=s2;
        System.out.println(s2=="Hello");
        System.out.println(s2==s3);
    }
```

Console

<terminated> StringLiteral [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Apr-2025, 8:04:33 pm – 8:04:33 pm elapsed: 0:00:00.200) [pid: 7728]

```
false
true
```


StringLiteral.java
E142
src
stri
StringLiteral
main(String[]) : void
1 package stri;
2
3 public class StringLiteral {
4 public static void main(String[] args) {
5 String s1="Hello";
6 String s2="Hello";
7 System.out.println(s1==s2);
8 }
9
10 }
Console
<terminated> StringLiteral [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Apr-2025, 7:59:57 pm – 7:59:57 pm elapsed: 0:00:00.210) [pid: 10772]
true
<terminated> StringLiteral [Java Application] C:\Program Files\Java\jdk-21\b

StringBuffer :

-->java.lang package
-->it allows us to create Mutable String
-->we can perform modifications of a String without creating the new object everytime

Characteristics :

1)Mutable : it provides the Flexibility to change the content

2)Synchronized :

3)performance Efficient -

4)memory : less memory consumption and improves performance

Constructors :

1) StringBuffer s=new StringBuffer( );

( ) creates an empty StringBuffer with the initial capacity of 16/ it will have the capacity to store 16 characters

2)StringBuffer s=new StringBuffer(10);

(int capacity) - it creates a StringBuffer with the specified capacity as length

3)StringBuffer s=new StringBuffer("welcome");

(String s) - creates a StringBuffer with the specified String

methods of StringBuffer Class :

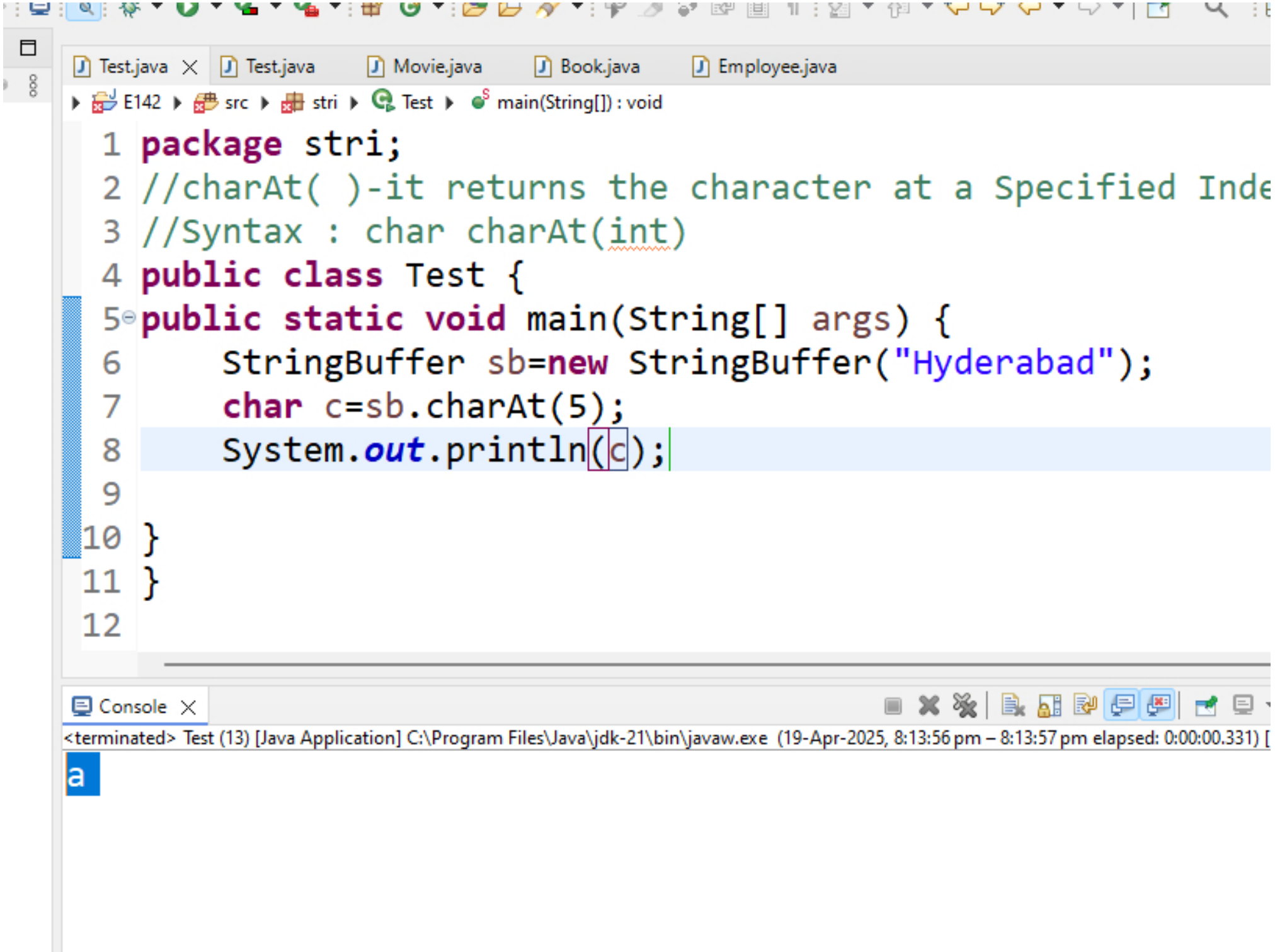
1)length( )
2)capacity( )
3)append( )
4)insert( )
5)reverse( )
6)delete( )
7)deleteCharAt( )
8)replace( )
9)charAt( )
10)indexOf( )
11)lastIndexOf( )
12)setCharAt( )

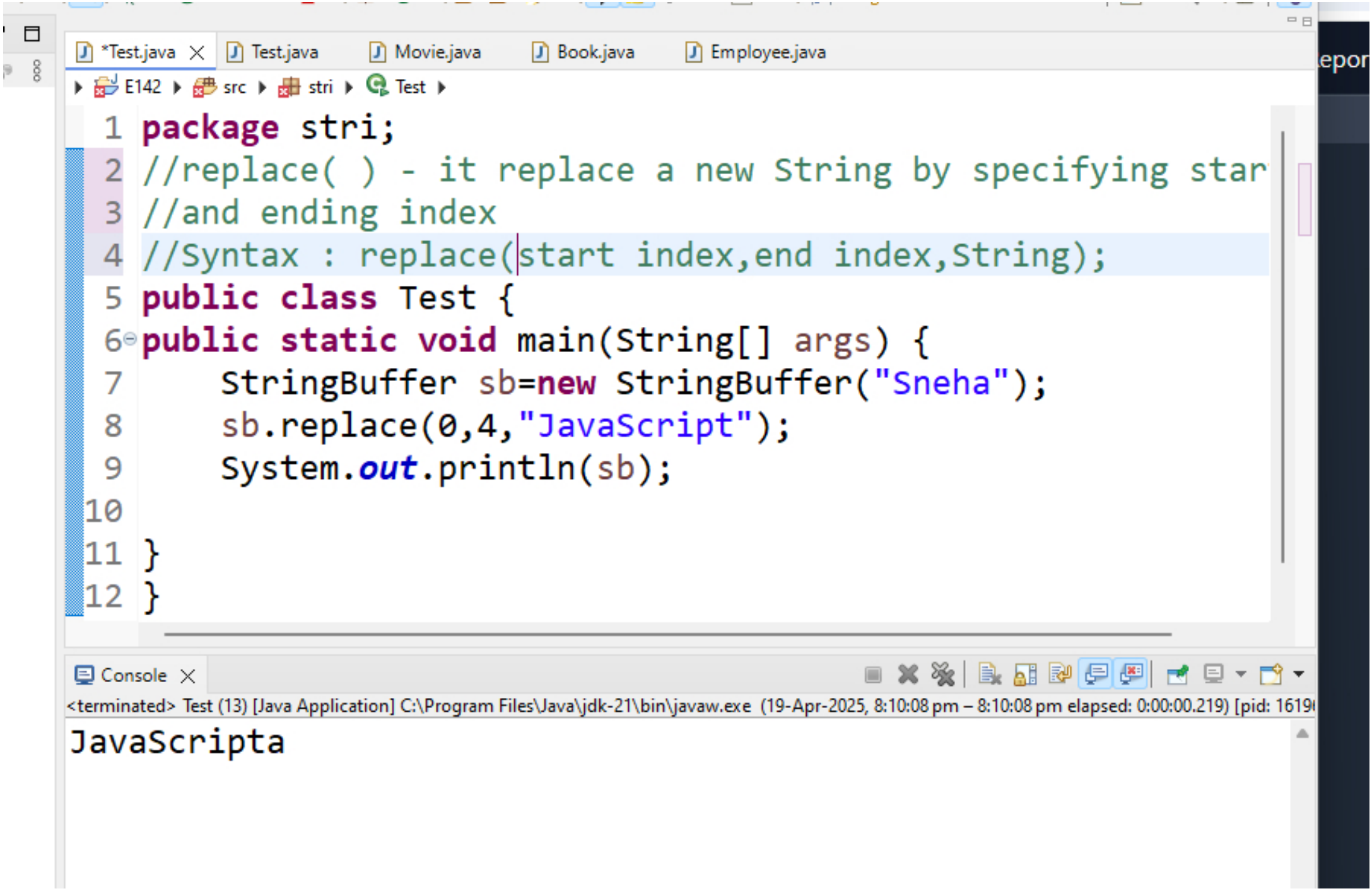
StringBuilder( ) :

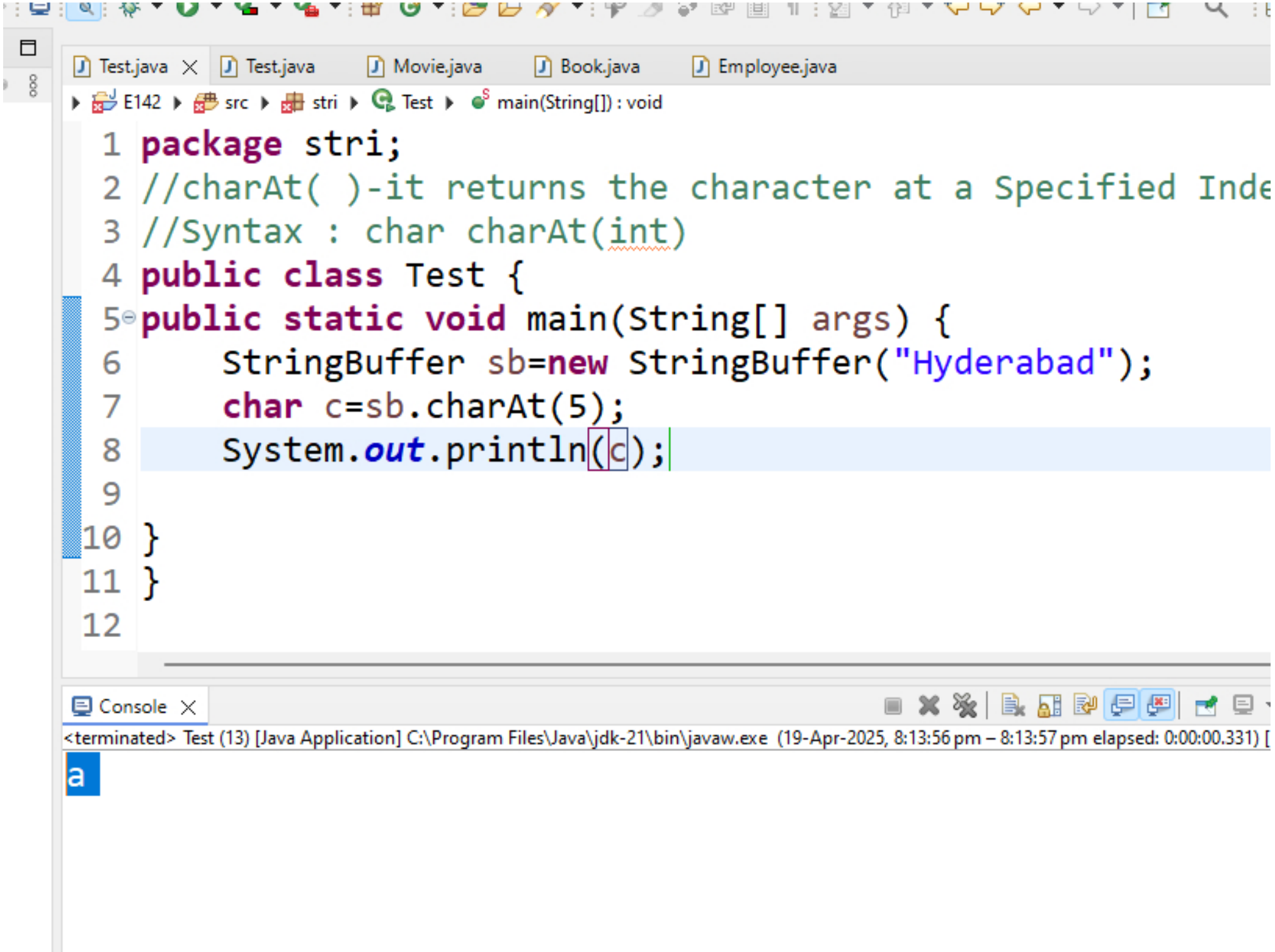
-->it is mutable
-->Stringbuilder is same as StringBuffer except that it is non-Synchronized

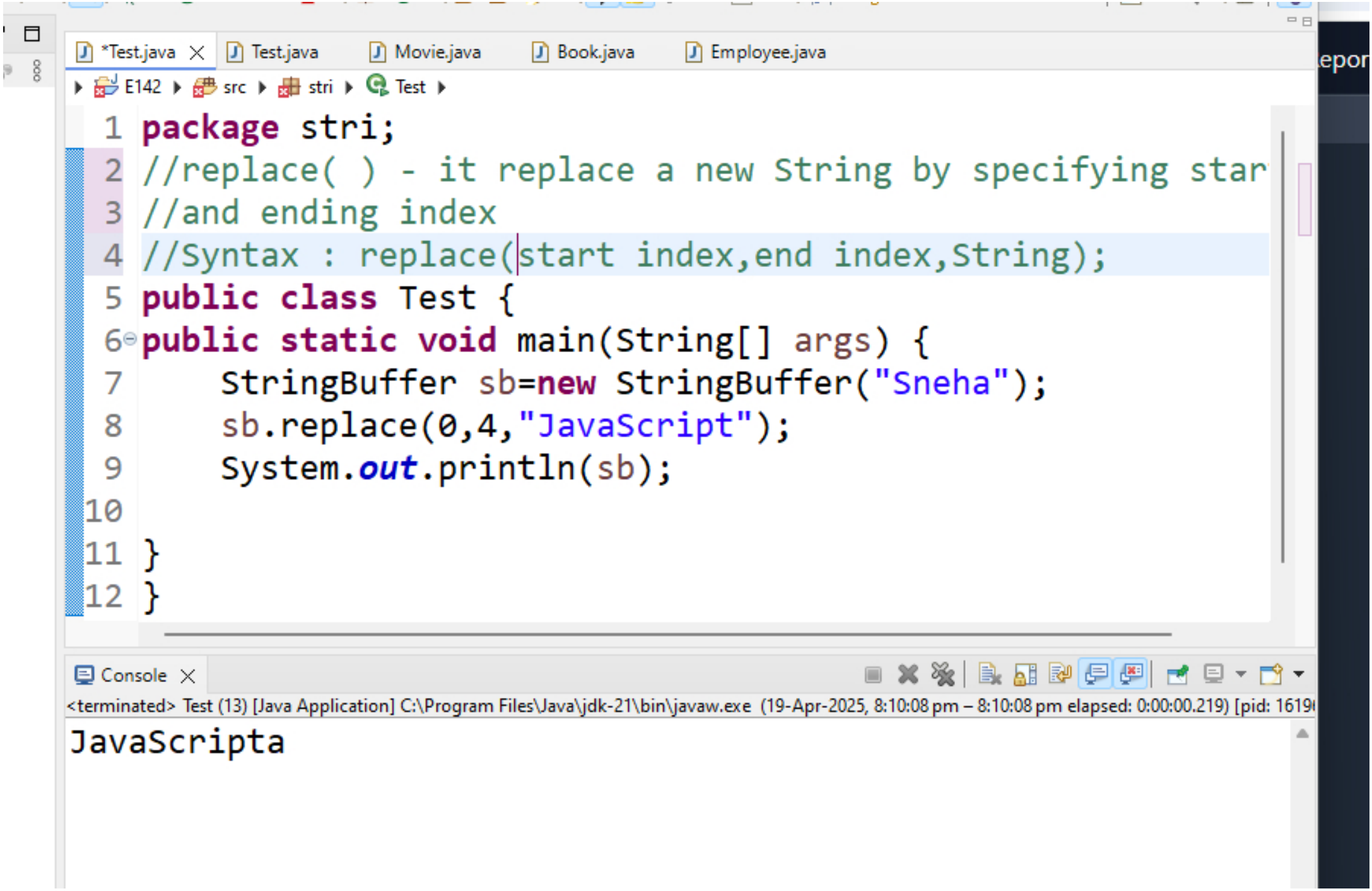
non-Synchronized :

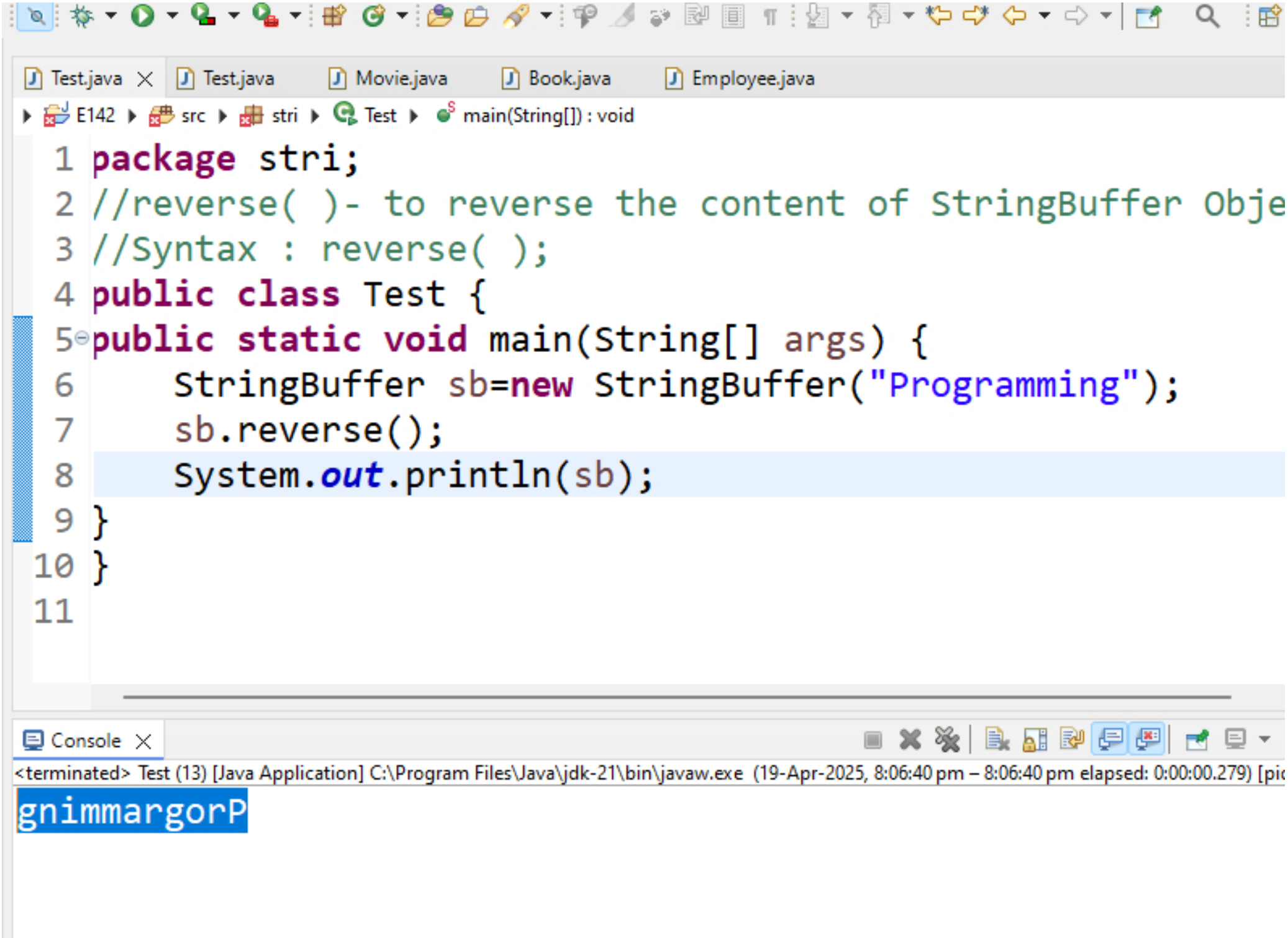
in StringBuilder we can access more than 1 thread at same time
-->it is thread safe

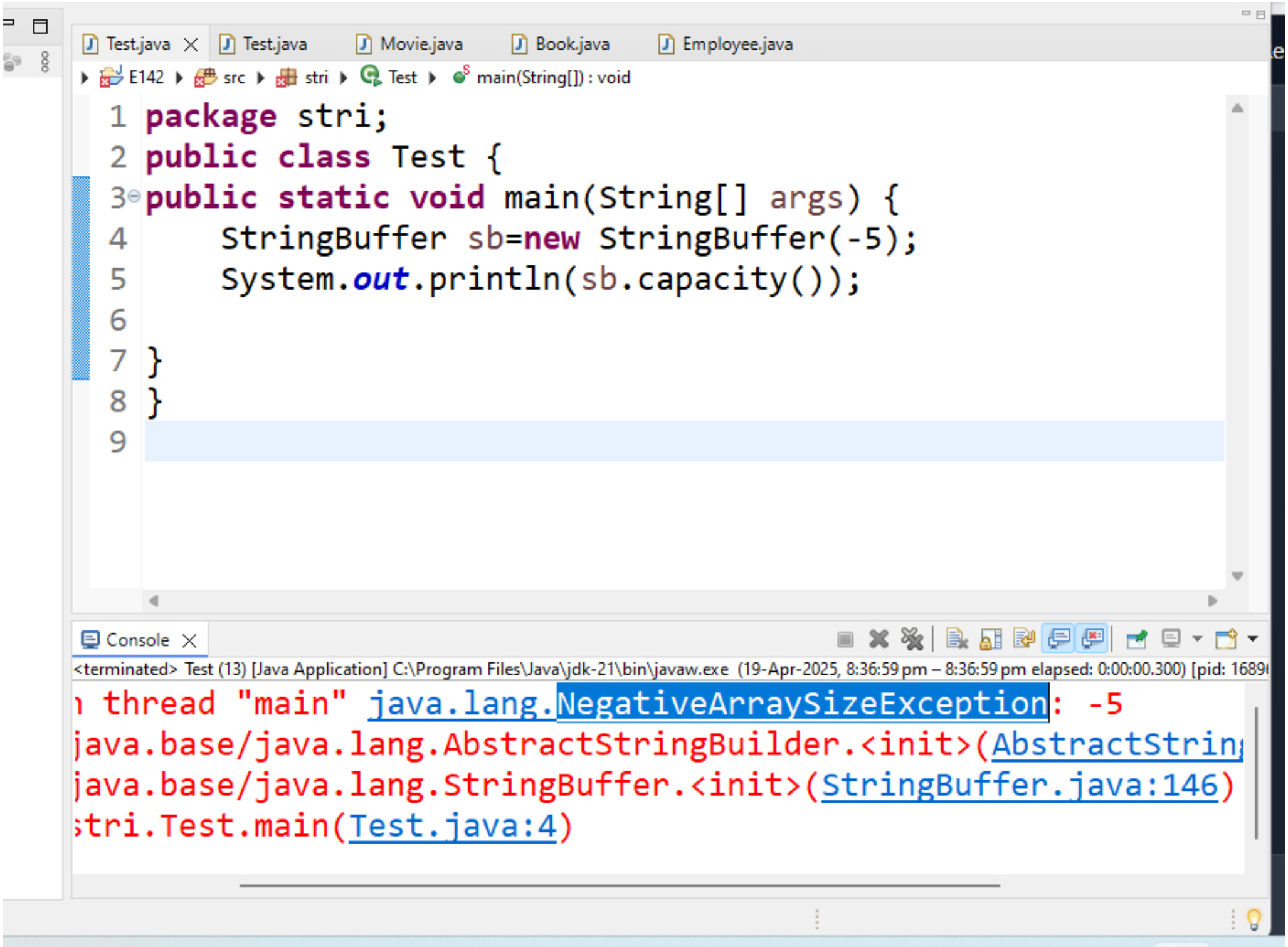

Test.java
Test.java
Movie.java
Book.java
Employee.java
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void
1 package stri;
2 //charAt( )-it returns the character at a Specified Inde
3 //Syntax : char charAt(int)
4 public class Test {
5 public static void main(String[] args) {
6 StringBuffer sb=new StringBuffer("Hyderabad");
7 char c=sb.charAt(5);
8 System.out.println(c);
9
10 }
11 }
12
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:13:56 pm – 8:13:57 pm elapsed: 0:00:00.331) [
a


*Test.java
Test.java
Movie.java
Book.java
Employee.java
E142
src
stri
Test
1 package stri;
2 //replace( ) - it replace a new String by specifying star
3 //and ending index
4 //Syntax : replace(start index,end index,String);
5 public class Test {
6 public static void main(String[] args) {
7 StringBuffer sb=new StringBuffer("Sneha");
8 sb.replace(0,4,"JavaScript");
9 System.out.println(sb);
10
11 }
12 }
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:10:08 pm – 8:10:08 pm elapsed: 0:00:00.219) [pid: 1619
JavaScripta


Test.java
Test.java
Movie.java
Book.java
Employee.java
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void
1 package stri;
2 //charAt( )-it returns the character at a Specified Inde
3 //Syntax : char charAt(int)
4 public class Test {
5 public static void main(String[] args) {
6 StringBuffer sb=new StringBuffer("Hyderabad");
7 char c=sb.charAt(5);
8 System.out.println(c);
9
10 }
11 }
12
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:13:56 pm – 8:13:57 pm elapsed: 0:00:00.331) [
a


*Test.java
Test.java
Movie.java
Book.java
Employee.java
E142
src
stri
Test
1 package stri;
2 //replace( ) - it replace a new String by specifying star
3 //and ending index
4 //Syntax : replace(start index,end index,String);
5 public class Test {
6 public static void main(String[] args) {
7 StringBuffer sb=new StringBuffer("Sneha");
8 sb.replace(0,4,"JavaScript");
9 System.out.println(sb);
10
11 }
12 }
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:10:08 pm – 8:10:08 pm elapsed: 0:00:00.219) [pid: 1619
JavaScripta


Test.java
Test.java
Movie.java
Book.java
Employee.java
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void
1 package stri;
2 //reverse( )- to reverse the content of StringBuffer Obje
3 //Syntax : reverse( );
4 public class Test {
5 public static void main(String[] args) {
6 StringBuffer sb=new StringBuffer("Programming");
7 sb.reverse();
8 System.out.println(sb);
9 }
10 }
11
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:06:40 pm – 8:06:40 pm elapsed: 0:00:00.279) [pi
gnimmargorP


Test.java
Test.java
Movie.java
Book.java
Employee.java
E142 src stri Test main(String[]) : void
1 package stri;
2 public class Test {
3 public static void main(String[] args) {
4 StringBuffer sb=new StringBuffer(-5);
5 System.out.println(sb.capacity());
6
7 }
8 }
9
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:36:59 pm – 8:36:59 pm elapsed: 0:00:00.300) [pid: 1689
thread "main" java.lang.NegativeArraySizeException: -5
java.base/java.lang.AbstractStringBuilder.<init>(AbstractStrin
java.base/java.lang.StringBuffer.<init>(StringBuffer.java:146)
stri.Test.main(Test.java:4)

Test.java | Test.java | Movie.java | Book.java | Employee.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

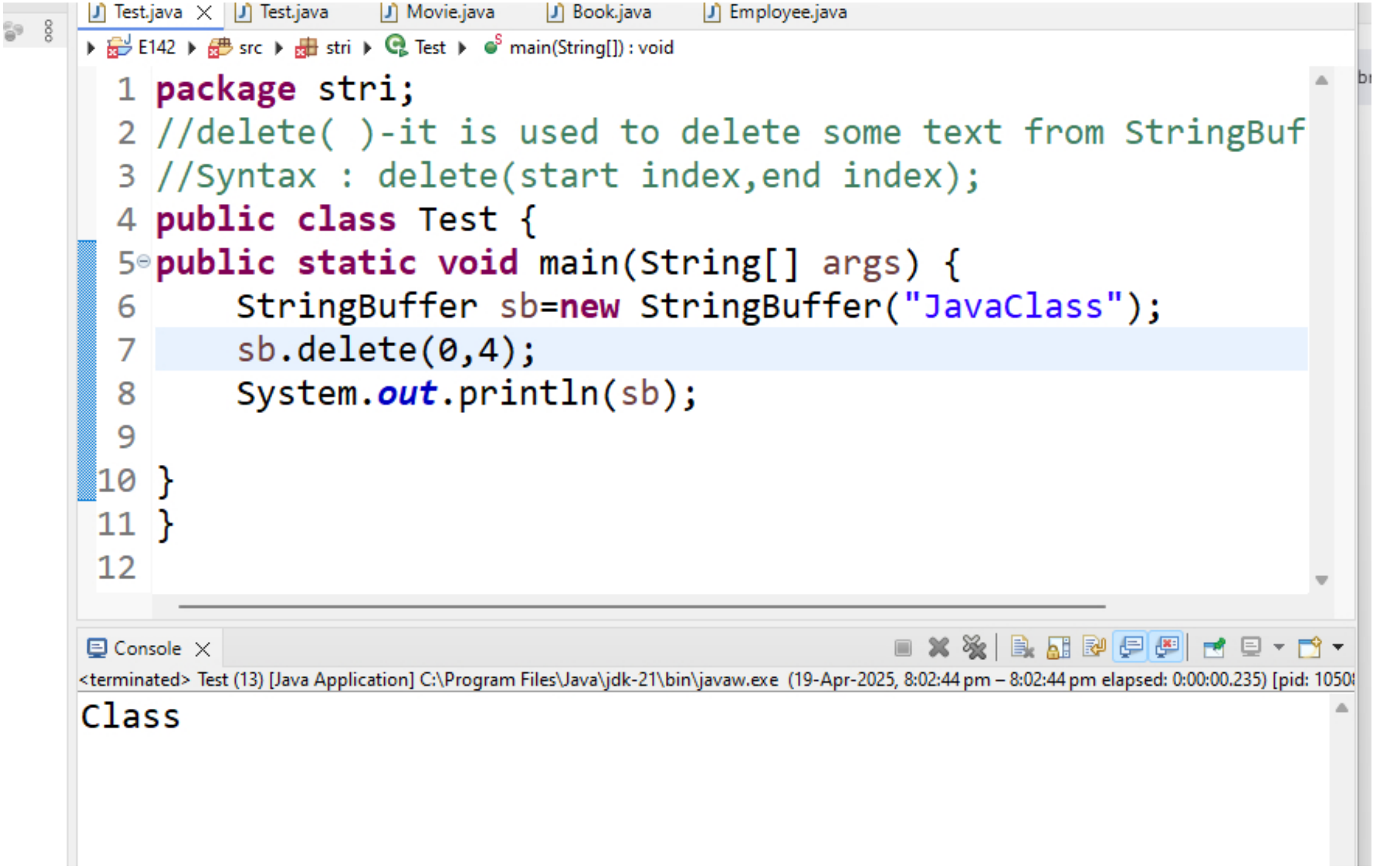
```java
package stri;
//setCharAt( )-it set the character at a Specified Index
//Syntax : setCharAt(index,char)
public class Test {
public static void main(String[] args) {
    StringBuffer sb=new StringBuffer("Banglore");
    sb.setCharAt(0, 'M');
    System.out.println(sb);
}
}

```

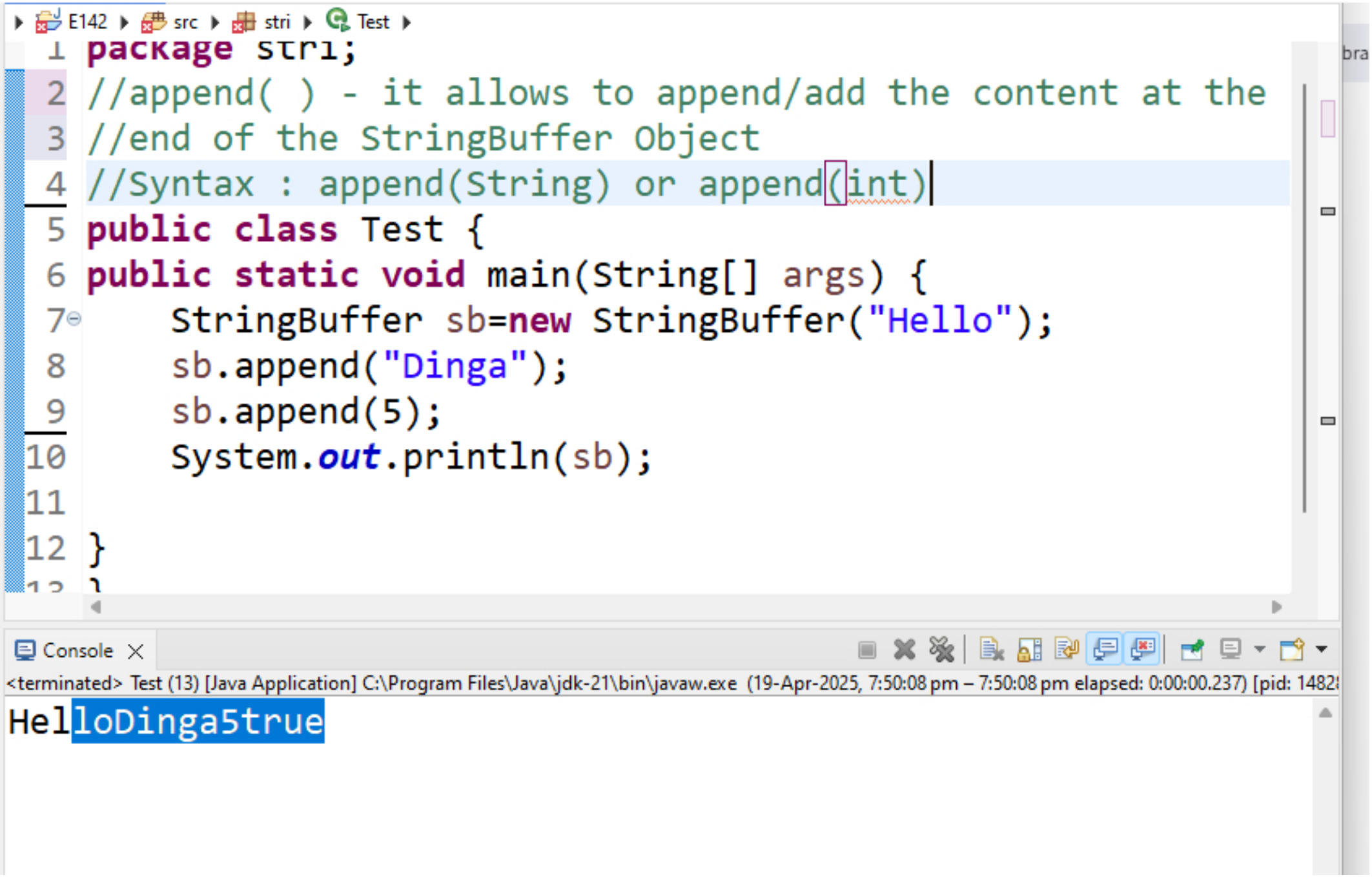
Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:16:59 pm – 8:16:59 pm elapsed: 0:00:00.202) [pid: 1123

```
Manglore
```

Writable | Smart Insert | 8 : 26 : 264


Test.java
Test.java
Movie.java
Book.java
Employee.java
E142 src stri Test main(String[]) : void
1 package stri;
2 //delete( )-it is used to delete some text from StringBuf
3 //Syntax : delete(start index,end index);
4 public class Test {
5 public static void main(String[] args) {
6 StringBuffer sb=new StringBuffer("JavaClass");
7 sb.delete(0,4);
8 System.out.println(sb);
9
10 }
11 }
12
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:02:44 pm – 8:02:44 pm elapsed: 0:00:00.235) [pid: 1050
Class


E142 src stri Test
package stri;
//append( ) - it allows to append/add the content at the
//end of the StringBuffer Object
//Syntax : append(String) or append(int)
public class Test {
public static void main(String[] args) {
StringBuffer sb=new StringBuffer("Hello");
sb.append("Dinga");
sb.append(5);
System.out.println(sb);
}
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 7:50:08 pm – 7:50:08 pm elapsed: 0:00:00.237) [pid: 1482
HelloDinga5true

```java
package str1;
//insert( )-it will insert text into the SB object at
//Specified Position
//Syntax : insert(index,String)
//         insert(index,char)
//         insert(index,int)
public class Test {
public static void main(String[] args) {
    StringBuffer sb=new StringBuffer("Java");
    sb.insert(1,"ii");
    sb.insert(2, 'a');
    sb.insert(0, 5);
    System.out.println(sb);

```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 7:59:32 pm – 7:59:32 pm elapsed: 0:00:00.232) [pid:

```
5Jiaiava
```

Test.java | Test.java | Movie.java | Book.java | Employee.java

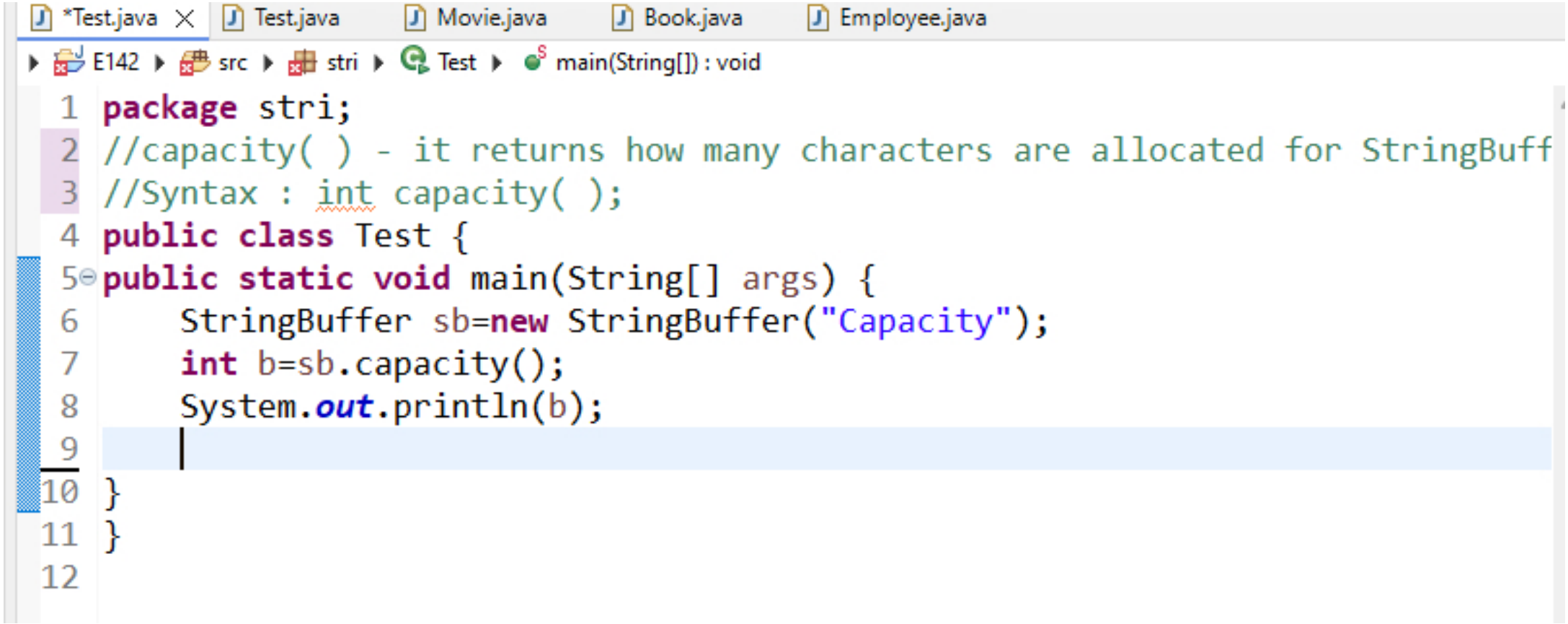
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```
1 ckage stri;
2 deleteCharAt( )-it is used to delete at Specified Positio
3 Syntax : deleteCharAt(index);
4 blic class Test {
5 blic static void main(String[] args) {
6   StringBuffer sb=new StringBuffer("JavaClass");
7   sb.deleteCharAt(0);
8   System.out.println(sb);
9
10
11
12
```

Console

<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 8:04:24 pm – 8:04:24 pm elapsed: 0:00:00.190) [pid: 9

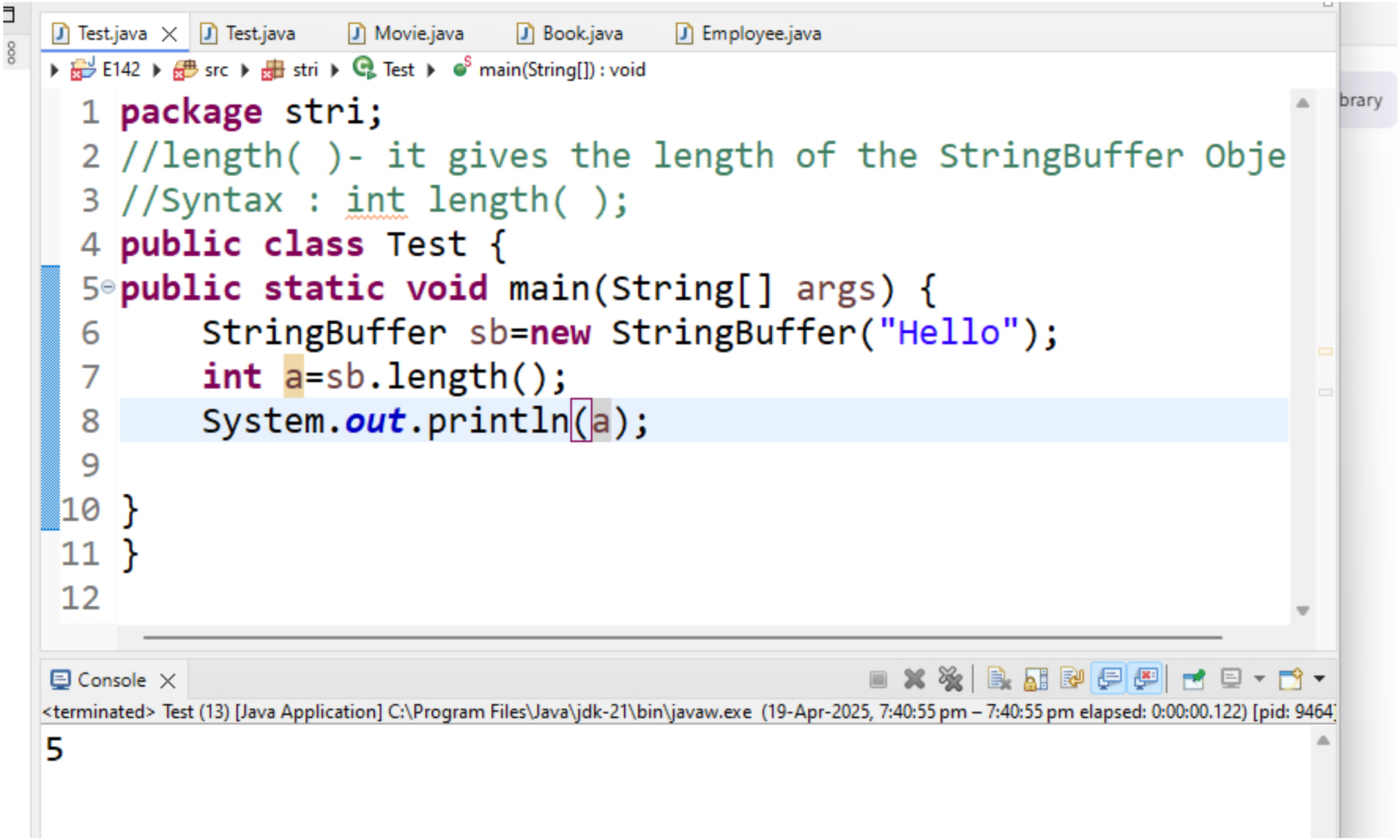
```
avaClass
```


*Test.java
Test.java
Movie.java
Book.java
Employee.java
E142
src
stri
Test
main(String[]) : void
1 package stri;
2 //capacity( ) - it returns how many characters are allocated for StringBuff
3 //Syntax : int capacity( );
4 public class Test {
5 public static void main(String[] args) {
6 StringBuffer sb=new StringBuffer("Capacity");
7 int b=sb.capacity();
8 System.out.println(b);
9
10 }
11 }
12

Test.java | Test.java | Movie.java | Book.java | Employee.java

E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void

```
1  package stri;
2  //split( ) - it is used to split the given String into an array of Strings
3  //Syntax : String split(String)
4  public class Test {
5  public static void main(String[] args) {
6      String s1="Welcome to Java";
7      String[] a=s1.split("to");
8      System.out.println(a[0]);
9      System.out.println(a[1]);
10 }
11 }
12
```


Test.java
Test.java
Movie.java
Book.java
Employee.java
E142 ▸ src ▸ stri ▸ Test ▸ main(String[]) : void
1 package stri;
2 //length( )- it gives the length of the StringBuffer Obje
3 //Syntax : int length( );
4 public class Test {
5 public static void main(String[] args) {
6 StringBuffer sb=new StringBuffer("Hello");
7 int a=sb.length();
8 System.out.println(a);
9
10 }
11 }
12
Console
<terminated> Test (13) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (19-Apr-2025, 7:40:55 pm – 7:40:55 pm elapsed: 0:00:00.122) [pid: 9464]
5