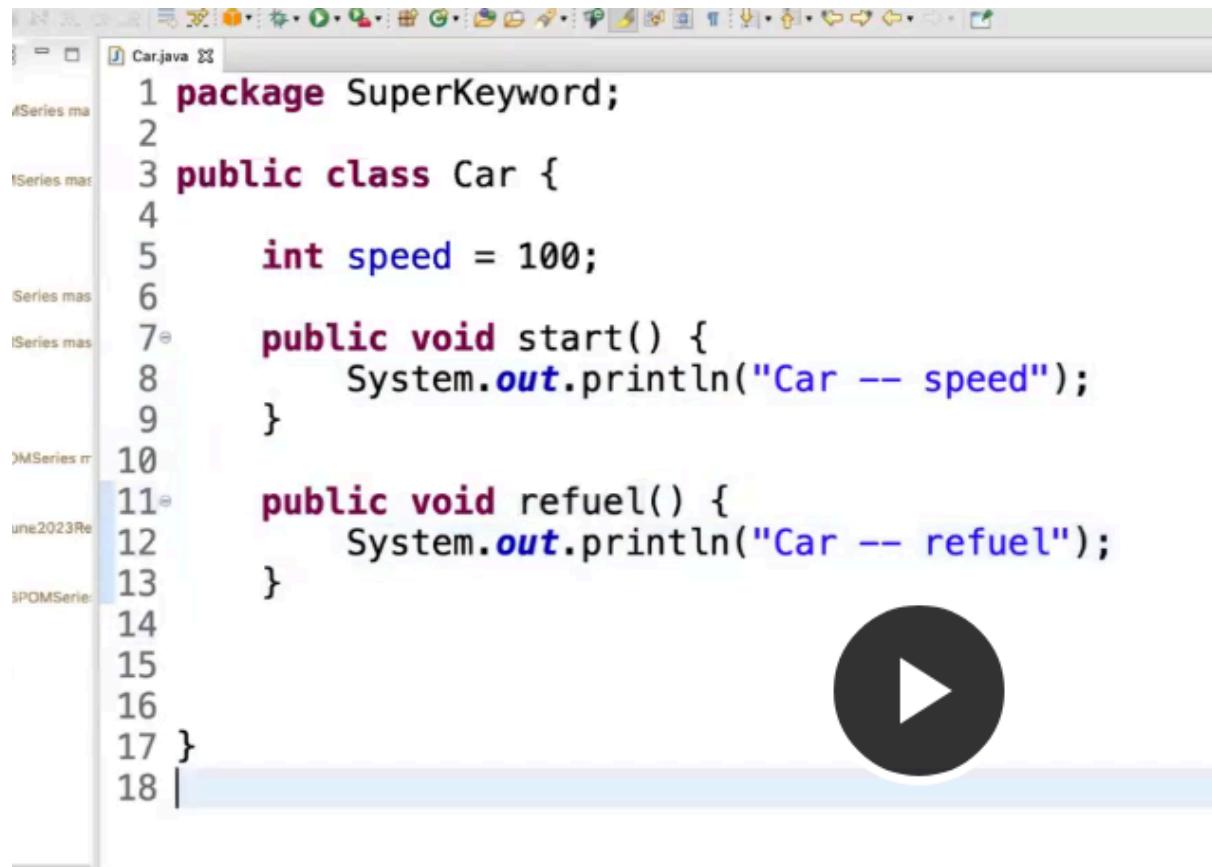


## Car-



```
1 package SuperKeyword;
2
3 public class Car {
4
5     int speed = 100;
6
7     public void start() {
8         System.out.println("Car -- speed");
9     }
10
11    public void refuel() {
12        System.out.println("Car -- refuel");
13    }
14
15
16
17 }
18 |
```



## Bmw-



```
1 package SuperKeyword;
2
3 public class BMW extends Car{
4
5
6     @Override
7     public void start() {
8         System.out.println("BMW -- start");
9     }
10
11    public void autoParking() {
12        System.out.println("BMW -- autoParking");
13    }
14
15
16}
17
```

Test-



```
1 package SuperKeyword;
2
3 public class TestCar {
4
5     public static void main(String[] args) {
6
7         BMW b = new BMW();
8         b.start();
9
10
11    }
12
13}
14
15
```

Bmw method called.

Bmw-

```

5   |
6  @Override
7  public void start() {
8      System.out.println("BMW -- start");
9      start();|
10 }
11

```

Stack overflow.

Call parent class overridden method-

bmw class-

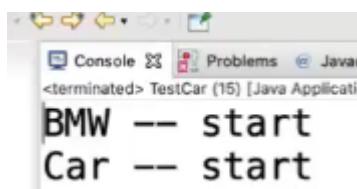
Use super. No need of objects. to call parent class method which has same name.

```

5   |
6  @Override
7  public void start() {
8      System.out.println("BMW -- start");
9      super.start();//car |
10 }
11

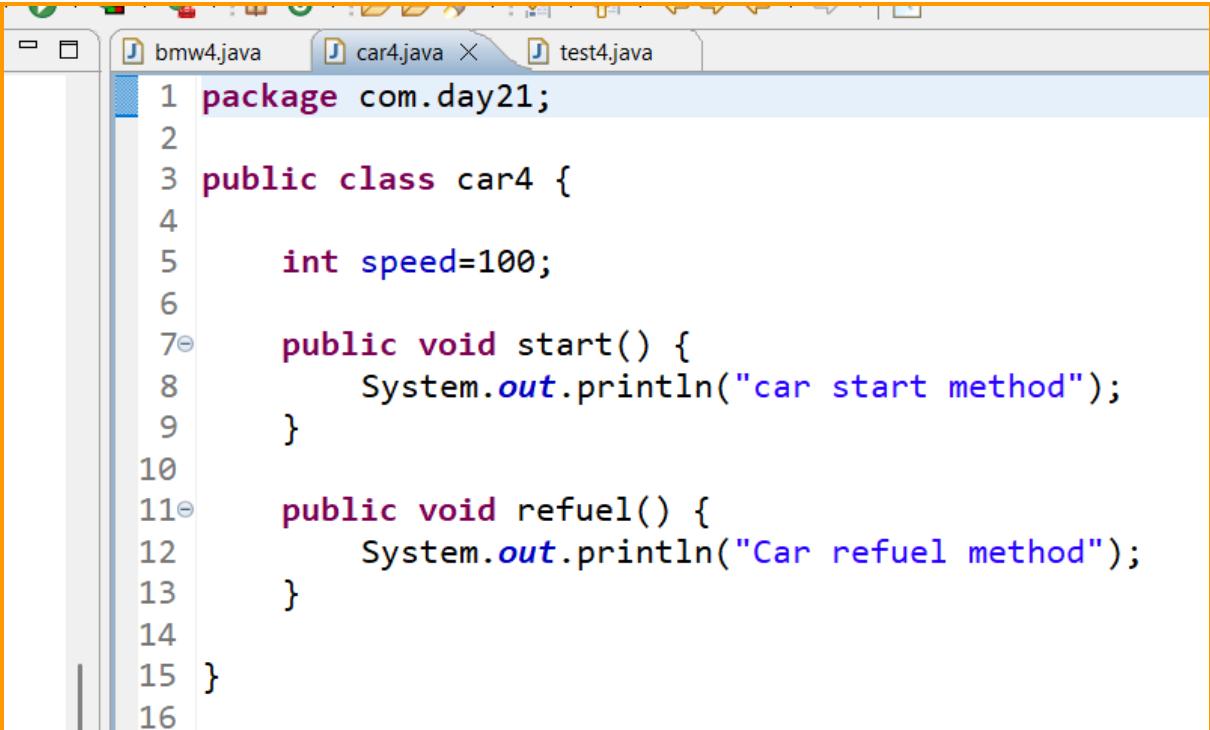
```

Run test-



can call parent class variable also using super-

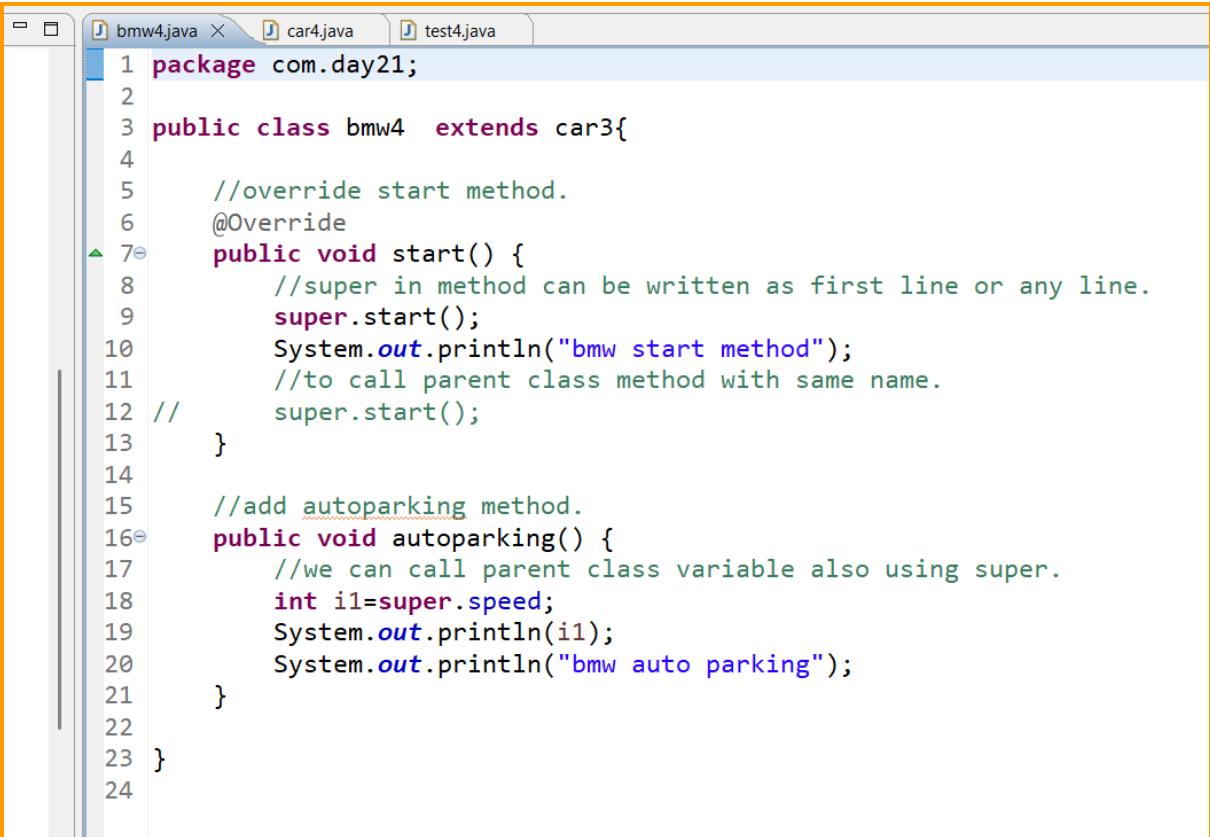
paste car4, bmw4, test4-



```

1 package com.day21;
2
3 public class car4 {
4
5     int speed=100;
6
7     public void start() {
8         System.out.println("car start method");
9     }
10
11    public void refuel() {
12        System.out.println("Car refuel method");
13    }
14
15 }
16

```



```

1 package com.day21;
2
3 public class bmw4 extends car3{
4
5     //override start method.
6     @Override
7     public void start() {
8         //super in method can be written as first line or any line.
9         super.start();
10        System.out.println("bmw start method");
11        //to call parent class method with same name.
12        super.start();
13    }
14
15     //add autoparking method.
16     public void autoparking() {
17         //we can call parent class variable also using super.
18         int i1=super.speed;
19         System.out.println(i1);
20         System.out.println("bmw auto parking");
21     }
22
23 }
24

```

```

1 package com.day21;
2
3 public class test4 {
4
5     public static void main(String[] args) {
6
7         bmw4 b1=new bmw4();
8         b1.autoparking();
9         b1.start();
10
11    }
12
13 }
14
15 //100
16 //bmw auto parking
17 //car start method
18 //bmw start method
19

```

Super can be written at first or end of method-

```

5
6  @Override
7  public void start() {
8      super.start(); //car
9      System.out.println("BMW -- start");
10 }

```

Non overridden method-

Bmw-

```

5
6  @Override
7  public void start() {
8      //start(); //BMW
9      super.start(); //car
10     System.out.println("BMW -- start");
11     refuel();
12 }

```

This is also right.-

```

5
6  @Override
7  public void start() {
8      //start(); //BMW
9      super.start(); //car
10     System.out.println("BMW -- start");
11     super.refuel();
12 }

```

Can be used in non overridden methods-

bmw-

```

13
14  public void autoParking() {
15      System.out.println("BMW -- autoParking");
16      //start(); //bmw -- no
17      super.start();
18 }
19

```



Bmw-

added speed variable (same as in parent). using display speed function to view it.



```

1 package SuperKeyword;
2
3 public class BMW extends Car{
4
5     int speed = 300;
6
7     @Override
8     public void start() {
9         //start(); //BMW
10        super.start(); //car
11        System.out.println("BMW -- start");
12        super.refuel();
13    }
14
15    public void autoParking() {
16        System.out.println("BMW -- autoParking");
17        //start(); //bmw -- no
18        super.start();
19    }
20
21    public void displaySpeed() {
22        System.out.println("speed is : " + speed);
23    }
24
25}
26

```

## Test-



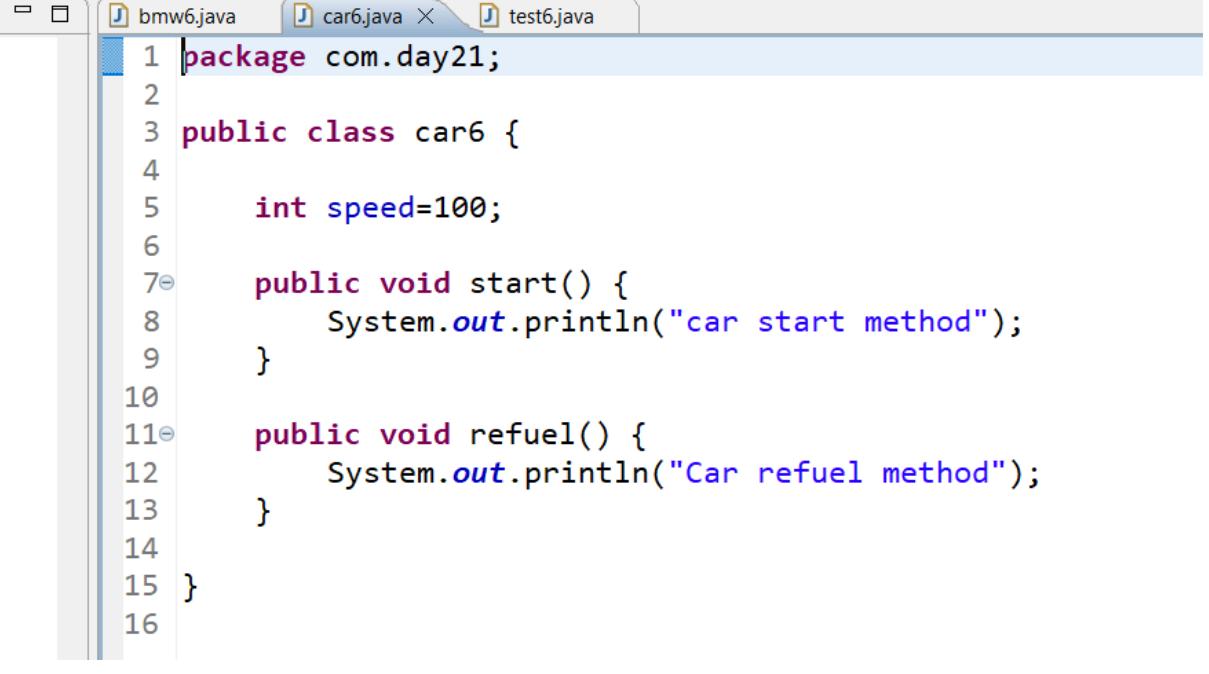
```

1 package SuperKeyword;
2
3 public class TestCar {
4
5     public static void main(String[] args) {
6
7         BMW b = new BMW();
8         b.start();
9
10        b.displaySpeed();
11
12    }
13
14}
15

```

Speed is 300.

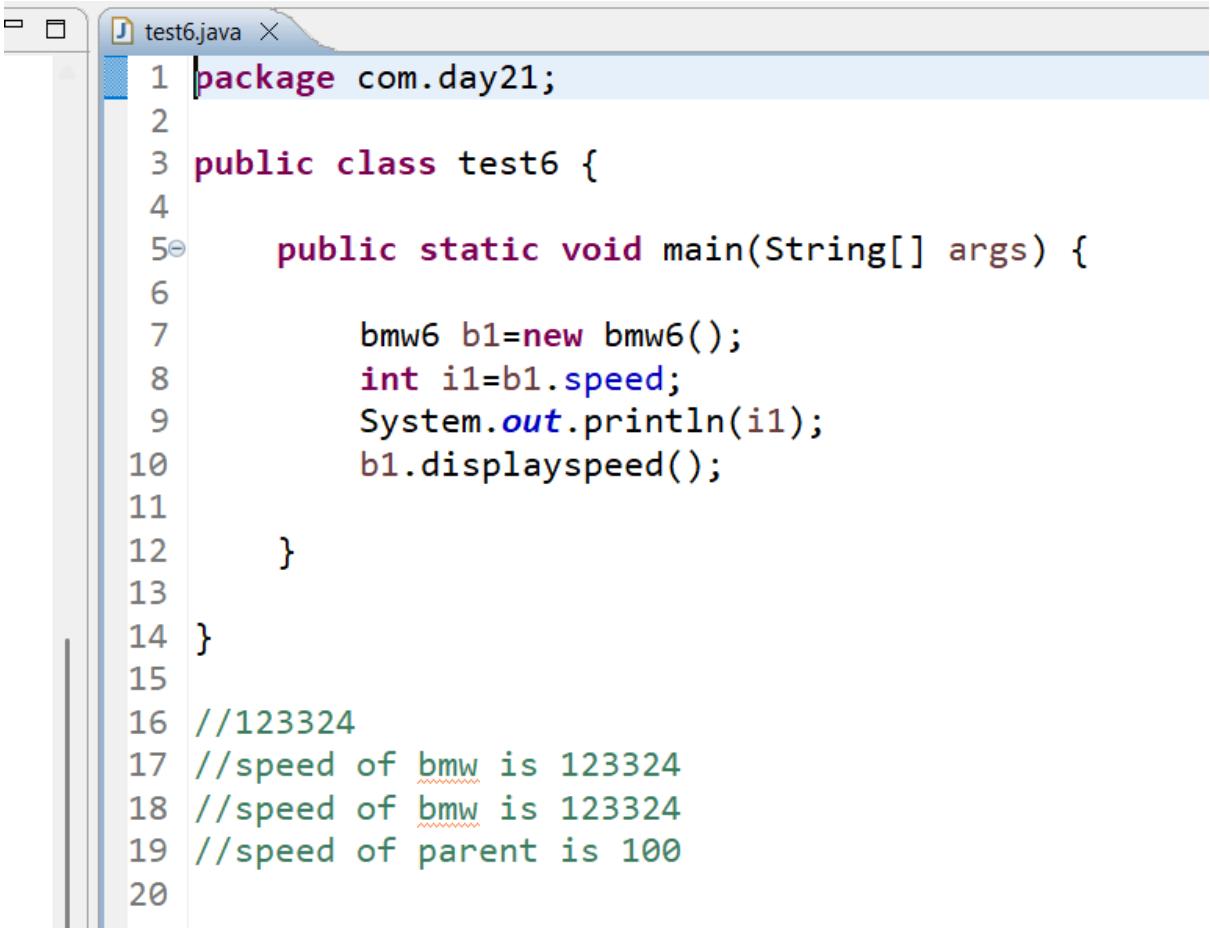
paste car6, bmw6, test6-



```
bmw6.java car6.java test6.java
1 package com.day21;
2
3 public class car6 {
4
5     int speed=100;
6
7     public void start() {
8         System.out.println("car start method");
9     }
10
11    public void refuel() {
12        System.out.println("Car refuel method");
13    }
14
15 }
16
```

The screenshot shows a Java code editor with the file `bmw6.java` open. The code defines a class `bmw6` that extends `car6`. It includes methods for speed, start, autoparking, and displayspeed, demonstrating inheritance and method overriding.

```
1 package com.day21;
2
3 public class bmw6 extends car6{
4
5     //own speed method.
6     int speed=123324;
7
8     //override start method.
9     @Override
10    public void start() {
11        super.start();
12        System.out.println("bmw start method");
13    }
14
15     //add autoparking method.
16    public void autoparking() {
17        System.out.println("bmw auto parking");
18    }
19
20    public void displayspeed() {
21        //can use this also for accessing class variables.
22        System.out.println("speed of bmw is "+ this.speed);
23        System.out.println("speed of bmw is "+ speed);
24        //call parent class speed
25        System.out.println("speed of parent is " +super.this.speed);
26        //Syntax error on token "this", Identifier expected
27        int i1=super.speed;
28        System.out.println("speed of parent is " +i1);
29    }
30
31 }
32 }
```



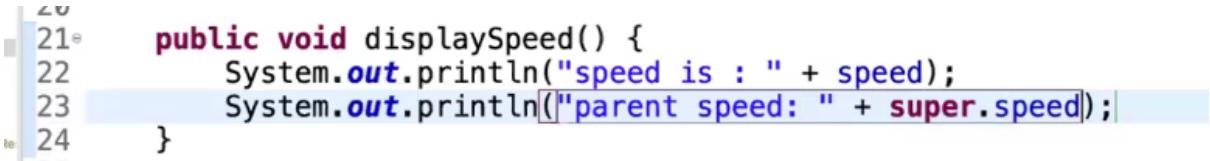
```

1 package com.day21;
2
3 public class test6 {
4
5     public static void main(String[] args) {
6
7         bmw6 b1=new bmw6();
8         int i1=b1.speed;
9         System.out.println(i1);
10        b1.displayspeed();
11    }
12}
13
14}
15
16//123324
17//speed of bmw is 123324
18//speed of bmw is 123324
19//speed of parent is 100
20

```

Parent class speed called-

Bmw-

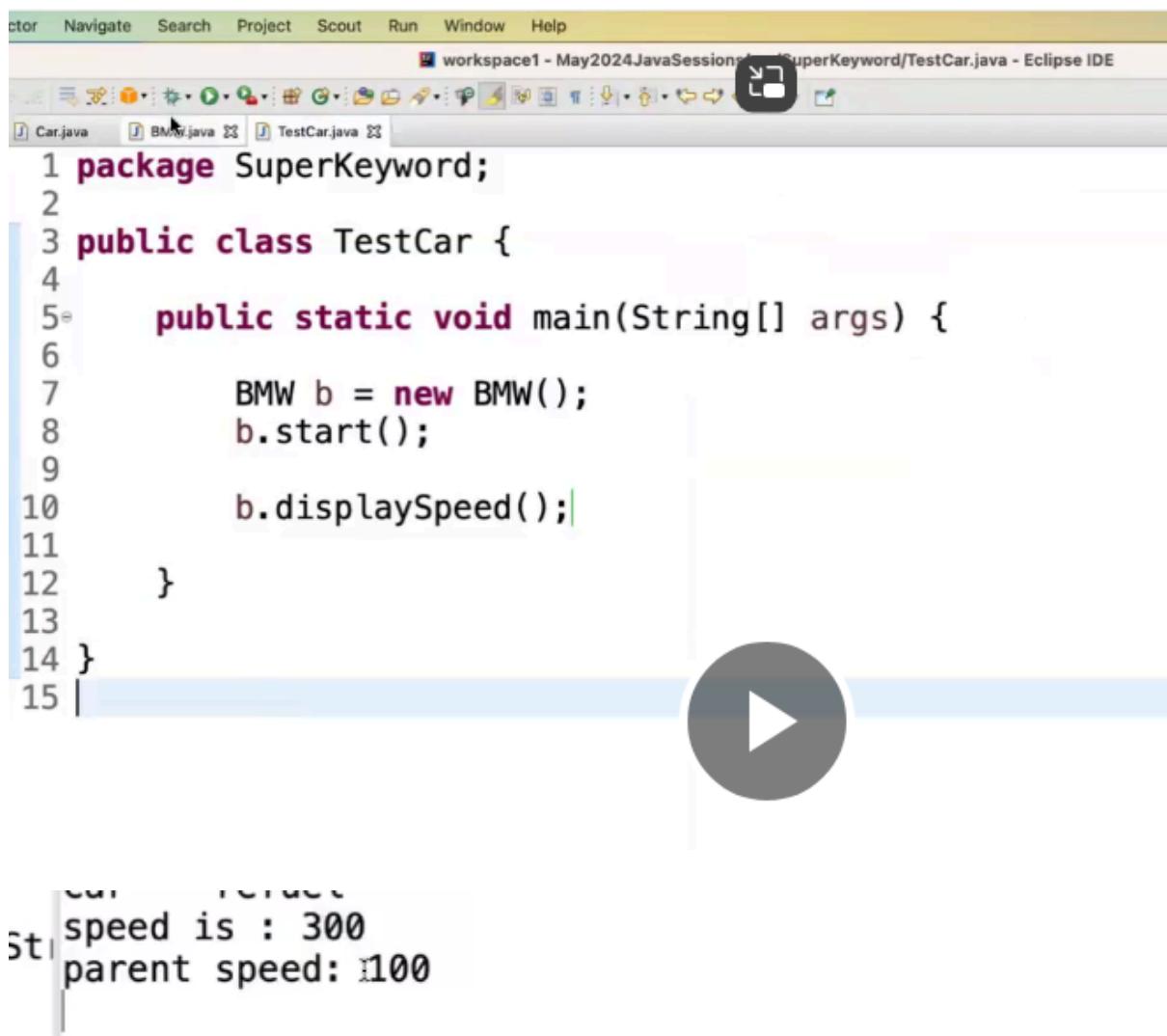


```

21     public void displaySpeed() {
22         System.out.println("speed is : " + speed);
23         System.out.println("parent speed: " + super.speed);
24     }

```

Run test.



```

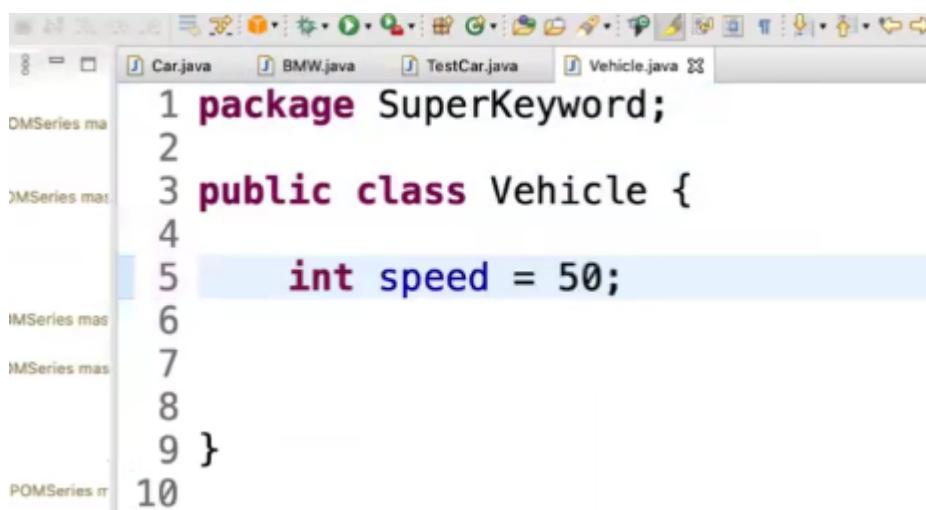
1 package SuperKeyword;
2
3 public class TestCar {
4
5     public static void main(String[] args) {
6
7         BMW b = new BMW();
8         b.start();
9
10        b.displaySpeed();
11    }
12
13}
14
15

```

speed is : 300  
parent speed: 100

Grand parent-

Vehicle-



```

1 package SuperKeyword;
2
3 public class Vehicle {
4
5     int speed = 50;
6
7
8
9 }
10

```

Not right syntax-

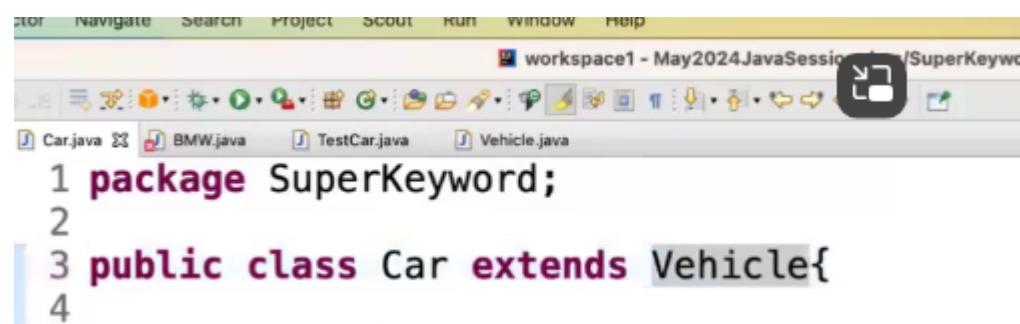
To access speed.

//Syntax error on token "super", Identifier expected

```

21° public void displaySpeed() {
22     System.out.println("speed is : " + speed);//300
23     System.out.println("parent speed: " + super.speed);//100
24     System.out.println(super.super.speed);
25 }
```

Car-



```

1 package SuperKeyword;
2
3 public class Car extends Vehicle{
4 }
```

bmw class-

```

20
21° public void displaySpeed() {
22     System.out.println("speed is : " + speed);//300
23     System.out.println("parent speed: " + super.speed);//100
24     System.out.println(super.speed);
25 }
26
27 |
```

Run test

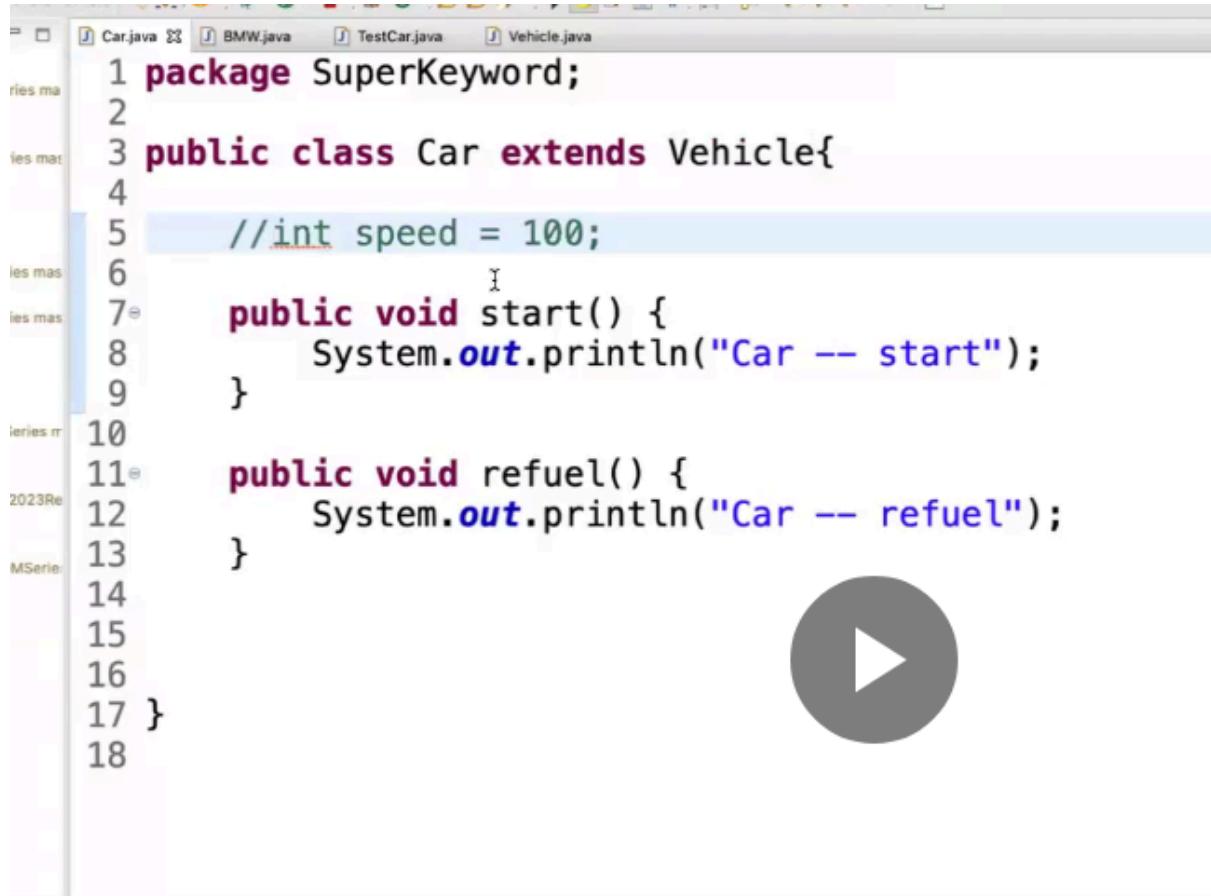
300

100

100

Super for accessing immediate parent class objects only not grandparents.

## Car-



```
1 package SuperKeyword;
2
3 public class Car extends Vehicle{
4
5     //int speed = 100;
6
7     public void start() {
8         System.out.println("Car -- start");
9     }
10
11    public void refuel() {
12        System.out.println("Car -- refuel");
13    }
14
15
16
17 }
18
```

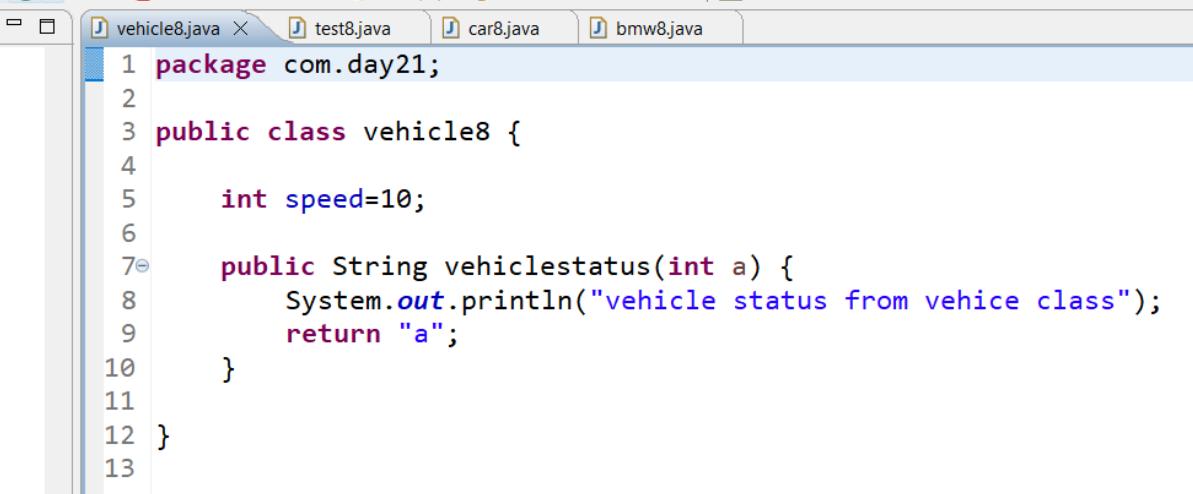
Run test.

Parent is not having the variable but grand parent has.

```
t-----  
t speed is : 300  
t parent speed: 50  
50
```

Super will go to grandparent if immediate parent does not have the variable or method.

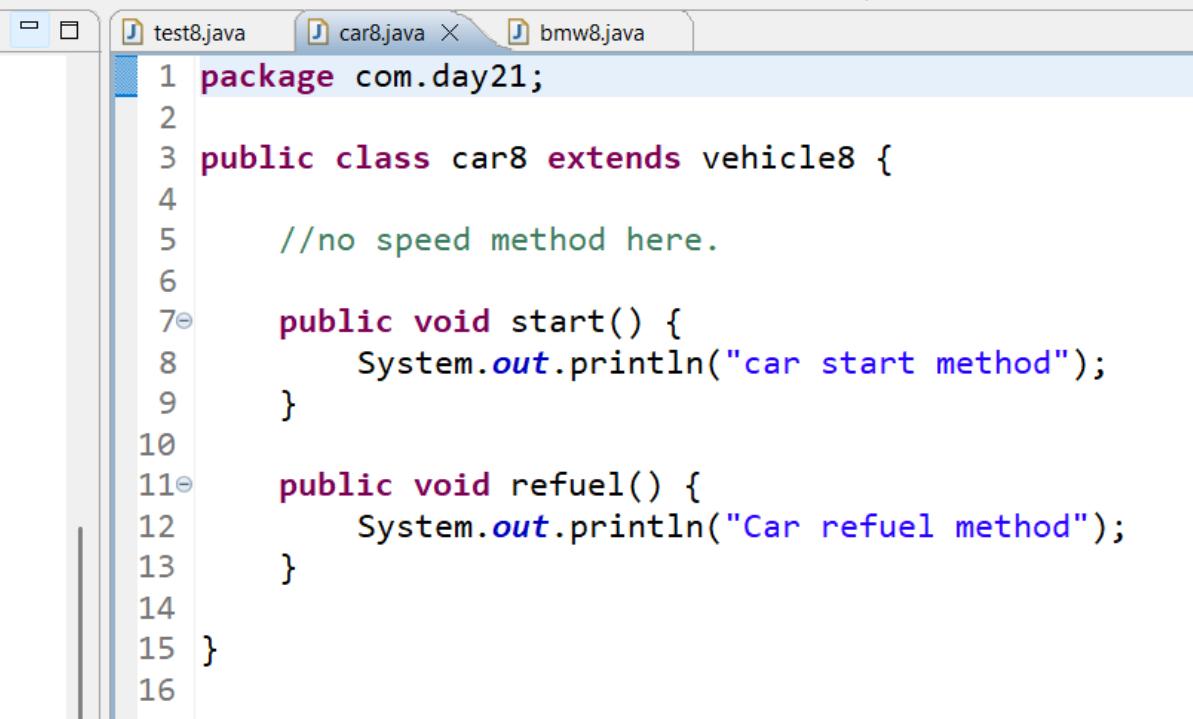
paste vehicle8, car8, bmw8, testfor8-



```

1 package com.day21;
2
3 public class vehicle8 {
4
5     int speed=10;
6
7     public String vehiclestatus(int a) {
8         System.out.println("vehicle status from vehice class");
9         return "a";
10    }
11
12 }
13

```



```

1 package com.day21;
2
3 public class car8 extends vehicle8 {
4
5     //no speed method here.
6
7     public void start() {
8         System.out.println("car start method");
9     }
10
11     public void refuel() {
12         System.out.println("Car refuel method");
13     }
14
15 }
16

```

```
1 package com.day21;
2
3 public class bmw8 extends car8{
4
5     //Super will go to grandparent if immediate parent does not have the variable or method.
6
7     //own speed method.
8     int speed=123324;
9
10    //override start method.
11    @Override
12    public void start() {
13        super.start();
14        System.out.println("bmw start method");
15    }
16
17    //add autoparking method.
18    public void autoparking() {
19        System.out.println("bmw auto parking");
20    }
21
22    public void displayspeed() {
23        System.out.println("speed of bmw is "+ this.speed);
24        System.out.println("speed of bmw is "+ speed);
25        //call parent class speed
26        // System.out.println("speed of parent is " +super.this.speed);
27        //Syntax error on token "this", Identifier expected
28        int i1=super.speed;
29        System.out.println("speed of parent is " +i1);
30
31        //try accessing grand parent speed.
32        //Syntax error on token "super", Identifier expected
33        super.super.speed;
34        String s1=super.vehiclestatus(10);
35        System.out.println(s1);
36    }
37
38 }
39
```

```
test8.java
1 package com.day21;
2
3 public class test8 {
4
5     public static void main(String[] args) {
6
7         bmw8 b1=new bmw8();
8         int i1=b1.speed;
9         System.out.println(i1);
10        b1.displayspeed();
11    }
12
13 }
14
15
16 //123324
17 //speed of bmw is 123324
18 //speed of bmw is 123324
19 //speed of parent is 10
20 //vehicle status from vehice class
21 //a
22
23
24
```

Car class –

With more constructor.

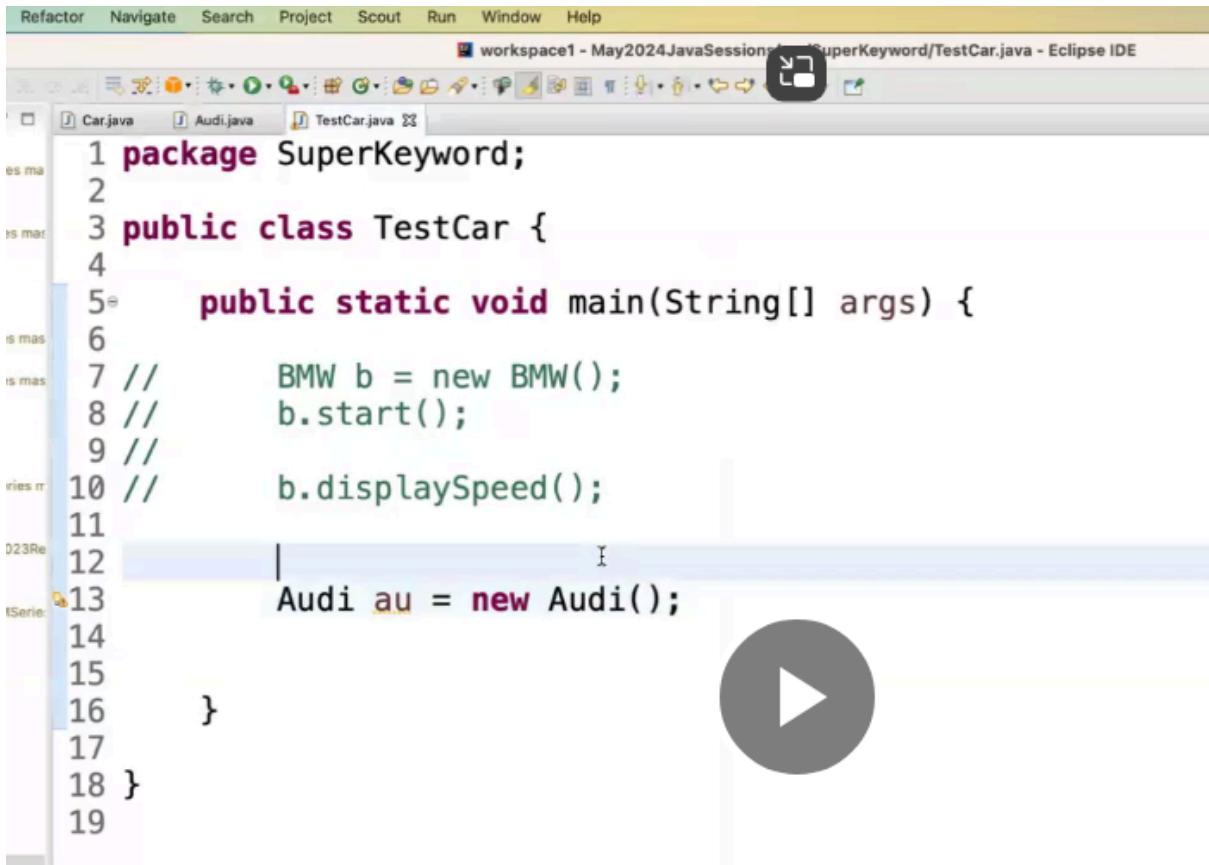


```
1 package SuperKeyword;
2
3 public class Car extends Vehicle{
4
5     int speed = 100;
6
7     public Car() {
8         System.out.println("Car -- default const...");
9     }
10
11    public Car(int a) {
12        System.out.println("Car -- const..." + a);
13    }
14
15    public Car(int a, int b) {
16        System.out.println("Car -- const..." + (a+b));
17    }
18
19
20    public void start() {
21        System.out.println("Car has started");
22    }
23}
```

Audi-

```
1 package SuperKeyword;
2
3 public class Audi extends Car{
4
5     public Audi() {
6         System.out.println("Audi -- default const...");
7     }
8
9     public Audi(int a) {
10        System.out.println("Audi -- const..." + a);
11    }
12
13    public Audi(int a, int b) {
14        System.out.println("Audi -- const..." + (a+b));
15    }
16
17
18 }
```

Test-

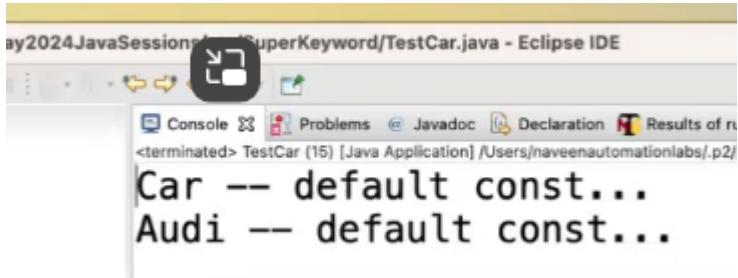


```

1 package SuperKeyword;
2
3 public class TestCar {
4
5     public static void main(String[] args) {
6
7         BMW b = new BMW();
8         b.start();
9
10        b.displaySpeed();
11
12        Audi au = new Audi();
13
14
15
16    }
17
18 }
19

```

Default constructor of parent called, then child class relevant constructor.

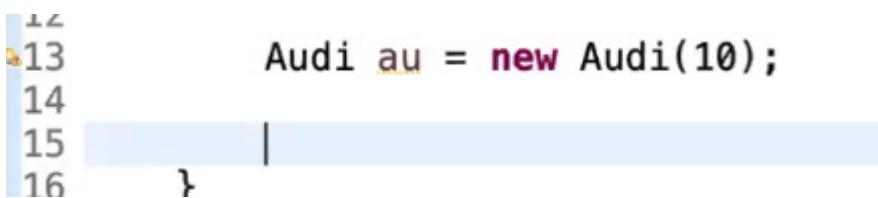


```

ay2024JavaSessions/SuperKeyword/TestCar.java - Eclipse IDE
Console Problems Javadoc Declaration Results of run
<terminated> TestCar [15] [Java Application] /Users/haveenautomationlabs/.p2/
|Car -- default const...
|Audi -- default const...

```

Test-



```

13     Audi au = new Audi(10);
14
15
16 }

```

```
<terminated> TestCar (15) [Java Application] /Users/naveenautomationlabs/.p2/po
Car -- default const...
Audi -- const...10
```

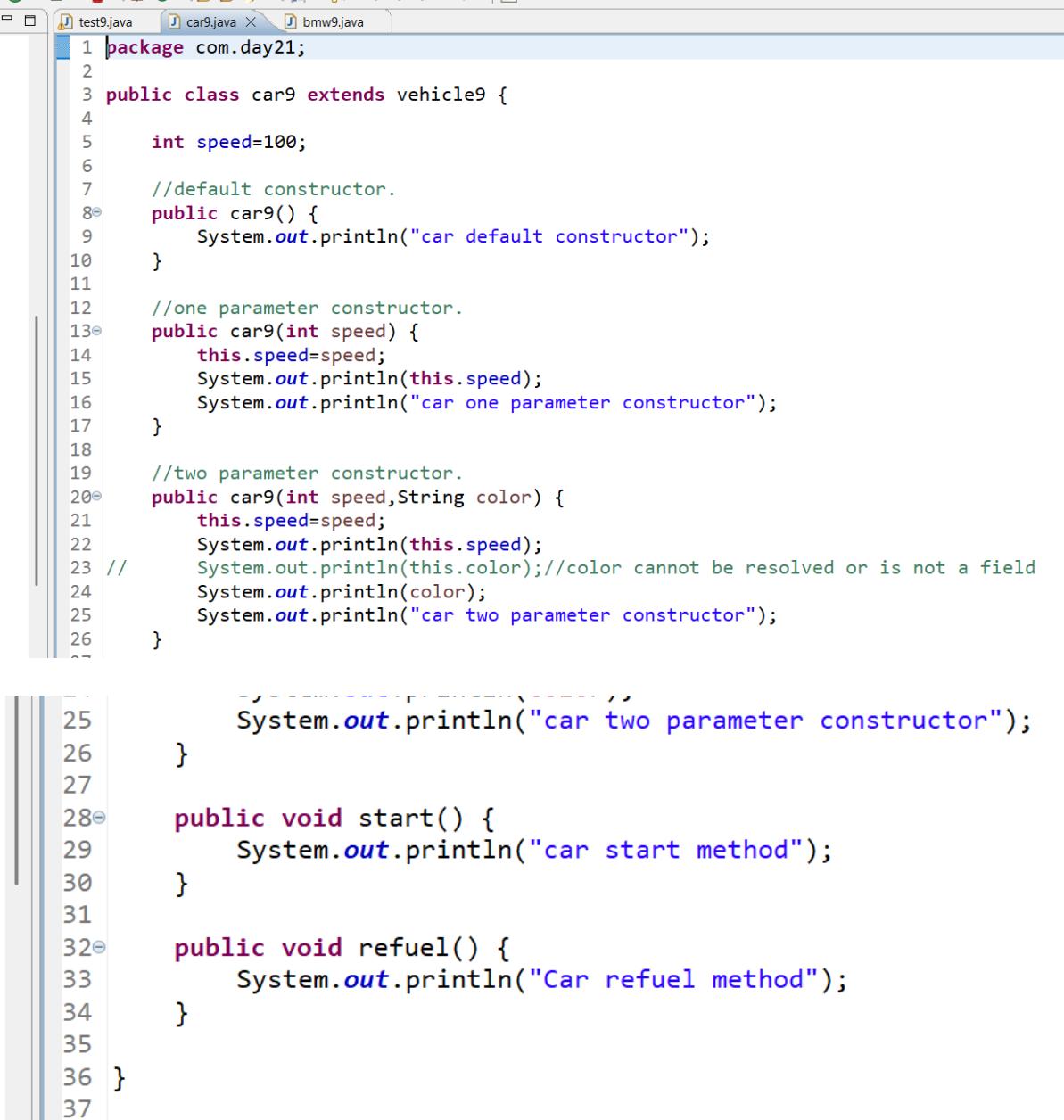
## Test-

```
12
13     Audi au = new Audi(10,20);
14
15
```

Console Problems Javadoc Declaration Results  
<terminated> TestCar (15) [Java Application] /Users/naveenautomationlabs  
Car -- default const...  
Audi -- const...30

paste vehicle9, bmw9, car9, testfor9-

```
1 package com.day21;
2
3 public class vehicle9 {
4
5     int speed=10;
6
7     public String vehiclestatus(int a) {
8         System.out.println("vehicle status from vehice class");
9         return "a";
10    }
11
12 }
13
```



The screenshot shows a Java code editor with three tabs at the top: test9.java, car9.java (which is the active tab), and bmw9.java. The code in car9.java is as follows:

```
1 package com.day21;
2
3 public class car9 extends vehicle9 {
4     int speed=100;
5
6     //default constructor.
7     public car9() {
8         System.out.println("car default constructor");
9     }
10
11    //one parameter constructor.
12    public car9(int speed) {
13        this.speed=speed;
14        System.out.println(this.speed);
15        System.out.println("car one parameter constructor");
16    }
17
18    //two parameter constructor.
19    public car9(int speed, String color) {
20        this.speed=speed;
21        System.out.println(this.speed);
22        System.out.println(this.color); //color cannot be resolved or is not a field
23        System.out.println(color);
24        System.out.println("car two parameter constructor");
25    }
26
27
28    public void start() {
29        System.out.println("car start method");
30    }
31
32    public void refuel() {
33        System.out.println("Car refuel method");
34    }
35
36 }
37
```

```
1 package com.day21;
2
3 public class bmw9 extends car9{
4
5     //Super will go to grandparent if immediate parent does not have the variable or method.
6
7     //own speed method.
8     int speed=123324;
9
10    //default constructor.
11    public bmw9() {
12        System.out.println("bmw default constructor");
13    }
14
15    //one parameter constructor.
16    public bmw9(int speed) {
17        this.speed=speed;
18        System.out.println(this.speed);
19        System.out.println("bmw one parameter constructor");
20    }
21
22    //two parameter constructor.
23    public bmw9(int speed,String color) {
24        this.speed=speed;
25        System.out.println(this.speed);
26        System.out.println(this.color);//color cannot be resolved or is not a field
27        System.out.println(color);
28        System.out.println("bmw two parameter constructor");
29    }
30
31        System.out.println("bmw two parameter constructor");
32    }
33
34    //override start method.
35    @Override
36    public void start() {
37        super.start();
38        System.out.println("bmw start method");
39    }
40
41    //add autoparking method.
42    public void autoparking() {
43        System.out.println("bmw auto parking");
44    }
45
46    public void displayspeed() {
47        System.out.println("speed of bmw is "+ this.speed);
48        System.out.println("speed of bmw is "+ speed);
49        //call parent class speed
50        System.out.println("speed of parent is " +super.this.speed);
51        //Syntax error on token "this", Identifier expected
52        int i1=super.speed;
53        System.out.println("speed of parent is " +i1);
```

```

49     int i1=super.speed;
50     System.out.println("speed of parent is " +i1);
51
52     //try accessing grand parent speed.
53     //Syntax error on token "super", Identifier expected
54     //super.super.speed;
55     String s1=super.vehiclestatus(10);
56     System.out.println(s1);
57 }
58
59 }
60

```

```

1 package com.day21;
2
3 public class test9 {
4
5     public static void main(String[] args) {
6
7         //default constructor of immediate parent called when mentioned.
8         bmw9 b1=new bmw9();
9
10        bmw9 b11=new bmw9(10);
11
12        bmw9 b111=new bmw9(10,"blue");
13    }
14
15 }
16
17 //car default constructor
18 //bmw default constructor
19 //car default constructor
20 //10
21 //bmw one parameter constructor
22 //car default constructor
23 //10
24 //blue
25 //bmw two parameter constructor
26
27
28
29
30

```

Call the super class constructor-

Audi-

```

9     public Audi(int a) {
10    super(100);
11    System.out.println("Audi -- const..." + a);
12 }
13
14
15
16
17
18

```

## Test-

The screenshot shows the Eclipse IDE interface. At the top, there's a toolbar with icons for file operations like New, Open, Save, and Run. Below the toolbar is a menu bar with 'File', 'Edit', 'Select', 'Run', 'View', 'Search', 'Help', and 'Project'. The main area has tabs for 'JavaSession1' and 'SuperKeyword/TestCar.java - Eclipse IDE'. The code editor shows the following Java code:

```

13     Audi au = new Audi(10);
14
15
16
17
18

```

To the right of the code editor is a terminal window titled 'Console' which displays the output of the program:

```

Car -- const...100
Audi -- const...10

```

Now default wont be called, as we specified which constructor to call using super keyword.

## Audi-

```

13
14  public Audi(int a, int b) {
15    super(10, 20);
16    System.out.println("Audi -- const..." + (a+b));
17 }
18

```

## Test-

The screenshot shows the Eclipse IDE interface. The code editor shows the same Java code as before, but with a different constructor call:

```

12
13     Audi au = new Audi(10,20);
14

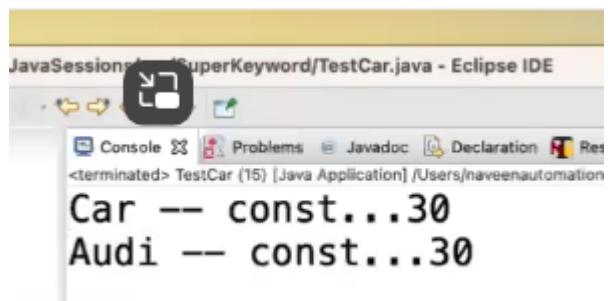
```

The terminal window shows the output:

```

Audi -- const...30

```



```
JavaSessions / SuperKeyword/TestCar.java - Eclipse IDE
Console Problems Javadoc Declaration Results of
<terminated> TestCar (15) [Java Application] /Users/naveenautomation
Car -- const...30
Audi -- const...30
```

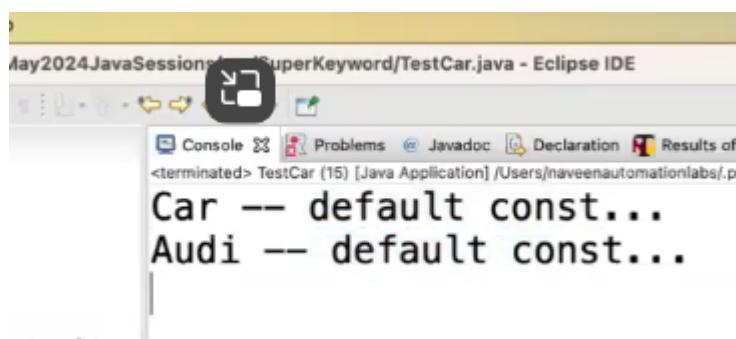
Audi-

```
4
5  public Audi() {
6      super();
7      System.out.println("Audi -- default const...");
```

Test-



```
11
12
13     Audi au = new Audi();
14
15
```



```
JavaSessions / SuperKeyword/TestCar.java - Eclipse IDE
Console Problems Javadoc Declaration Results of
<terminated> TestCar (15) [Java Application] /Users/naveenautomationlabs/.p
Car -- default const...
Audi -- default const...
```

Super should be the first statement in child constructor, else error-

//Constructor call must be the first statement in a constructor

```

4
5  public Audi() {
6      //first statement in child class const...
7      System.out.println("Audi -- default const..."); 
8      super();| 
9  }

```

Cannot use two super at a time-

//Constructor call must be the first statement in a constructor

```

JAVASERIES 10
June2023Re 11  public Audi(int a) {
12      super();
13      super(100); | 
14      System.out.println("Audi -- const..." + a);
15  }
16

```

Only one parent constructor can be called. In above case, the super becomes second sentence.

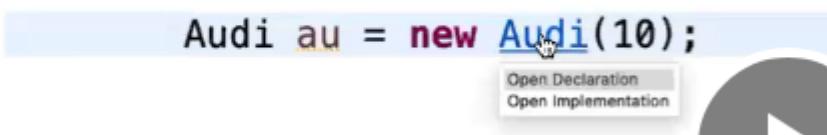
We can pass parameter also inside super.-

Test-

```

11
12
13  Audi au = new Audi(10); 
14
15

```



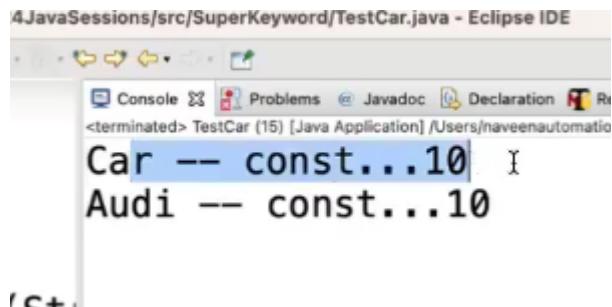
Audi-

```

11 public Audi(int a) {
12     super(a);
13     System.out.println("Audi -- const..." + a);
14 }
```

## Run test-

JavaSessions/src/SuperKeyword/TestCar.java - Eclipse IDE



```

<terminated> TestCar (15) [Java Application] /Users/naveenautomationlaptop
Car -- const...10
Audi -- const...10
```

## Audi-

```

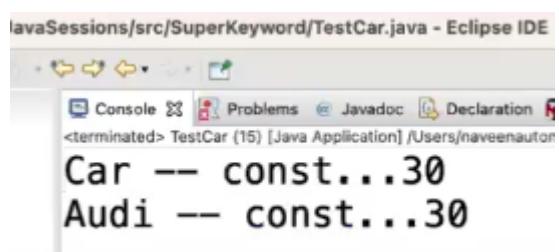
15
16 public Audi(int a, int b) {
17     super(a,b); |
18     System.out.println("Audi -- const..." + (a+b));
19 }
20
```

## Test-

```

13 Audi au = new Audi(10,20);
14
15
```

JavaSessions/src/SuperKeyword/TestCar.java - Eclipse IDE



```

<terminated> TestCar (15) [Java Application] /Users/naveenautomationlaptop
Car -- const...30
Audi -- const...30
```

## Note-

Super first statement only when calling constructor.

For variables and methods it can be any line inside the calling method.

This is allowed-

Super for variables and methods can be anywhere.

Audi-

```
16 public Audi(int a, int b) {
17     super(a,b);
18     System.out.println("Audi -- const..." + (a+b));
19     System.out.println(super.speed); //100
20 }
```

```
15
16 public Audi(int a, int b) {
17     super(a,b);
18     System.out.println("Audi -- const..." + (a+b));
19     System.out.println(super.speed); //100
20     super.start();           |
21 }
```

Calling static using super-

Car-

```
27
28 public static void testing() {
29     System.out.println("car -- testing");
30 }
31
```

Audi-

Can be accessed due to inheritance.

```

16  public Audi(int a, int b) {
17      super(a,b);
18      System.out.println("Audi -- const..." + (a+b));
19      System.out.println(super.speed);//100
20      super.start();
21      super.testing();
22  }
23

```

Note-

Better to access static using class name always.

Good way to access static-

```

15
16  public Audi(int a, int b) {
17      super(a,b);
18      System.out.println("Audi -- const..." + (a+b));
19      System.out.println(super.speed);//100
20      super.start();
21      Car.testing();
22  }
23
24

```

paste vehicle13, test13, car13, bmw13-

```
car13.java vehicle13.java test13.java bmw13.java
1 package com.day21;
2
3 public class vehicle13 {
4
5     int speed=10;
6
7     public String vehiclestatus(int a) {
8         System.out.println("vehicle status from vehice class");
9         return "a";
10    }
11
12 }
13
```

```
car13.java test13.java bmw13.java
1 package com.day21;
2
3 public class car13 extends vehicle13 {
4
5     int speed=100;
6     static int duration=100000;
7
8     //default constructor.
9     public car13() {
10         System.out.println("car default constructor");
11     }
12
13     //one parameter constructor.
14     public car13(int speed) {
15         this.speed=speed;
16         System.out.println(this.speed);
17         System.out.println("car one parameter constructor");
18     }
19
20     //two parameter constructor.
21     public car13(int speed,String color) {
22         this.speed=speed;
23         System.out.println(this.speed);
24 //        System.out.println(this.color);//color cannot be resolved or is not a field
25         System.out.println(color);
26         System.out.println("car two parameter constructor");
27     }
28 }
```

```

26     System.out.println("car two parameter constructor");
27 }
28
29 public void start() {
30     System.out.println("car start method");
31 }
32
33 public void refuel() {
34     System.out.println("Car refuel method");
35 }
36
37 //static method in car class.
38 public static String testing(int a) {
39     System.out.println("static method in car class");
40     return "a";
41 }
42
43 }
44

```

```

1 package com.day21;
2
3 public class bmw13 extends car13{
4
5     //Super will go to grandparent if immediate parent does not have the variable or method.
6
7     //own speed method.
8     int speed=123324;
9
10    //default constructor.
11    public bmw13() {
12        super(100);
13        System.out.println("bmw default constructor");
14        //Super for variables and methods can be anywhere.
15        System.out.println(super.speed);
16    }
17
18    //one parameter constructor.
19    public bmw13(int speed) {
20        super(speed); //can pass parameter also to parent.
21        //not necessary to hard code everytime.
22        this.speed=speed;
23        System.out.println(this.speed);
24        System.out.println("bmw one parameter constructor");
25        //Super for variables and methods can be anywhere.
26        super.start();
27        System.out.println("-----");
28    }
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

```

```

26     super.start();
27     System.out.println("-----");
28
29     //can access static methods also this way.
30     String s1=super.testing(10);
31     System.out.println(s1);
32
33     //access static variable.
34     int i1=super.duration;
35     System.out.println(i1);
36 }
37
38 //two parameter constructor.
39 public bmw13(int speed,String color) {
40     super(speed, color); //can pass parameter also to parent.
41     //not necessary to hard code everytime.
42     this.speed=speed;
43     System.out.println(this.speed);
44     System.out.println(this.color);//color cannot be resolved or is not a field
45     System.out.println(color);
46     System.out.println("bmw two parameter constructor");
47     System.out.println("-----");
48
49     System.out.println("bmw two parameter constructor");
50     System.out.println("-----");
51
52     //can access static methods also this way.
53     //this is better way to access static using class name.
54     String s2=car13.testing(20);
55     System.out.println(s2);
56
57     //access static variables.
58     //best way using class name.
59     int i1=car13.duration;
60     System.out.println(i1);
61 }
62
63 //override start method.
64 @Override
65 public void start() {
66     super.start();
67     System.out.println("bmw start method");
68 }

```

```

64         System.out.println("bmw start method");
65     }
66
67     //add autoparking method.
68     public void autoparking() {
69         System.out.println("bmw auto parking");
70     }
71
72     public void displayspeed() {
73         System.out.println("speed of bmw is "+ this.speed);
74         System.out.println("speed of bmw is "+ speed);
75         //call parent class speed
76         System.out.println("speed of parent is " +super.this.speed);
77         //Syntax error on token "this", Identifier expected
78         int i1=super.speed;
79         System.out.println("speed of parent is " +i1);
80
81         //try accessing grand parent speed.
82         //Syntax error on token "super", Identifier expected
83     //    super.super.speed;
84     String s1=super.vehiclestatus(10);
85     System.out.println(s1);
86 }
87
88 }
89

```

```

1 package com.day21;
2
3 public class test13 {
4
5     public static void main(String[] args) {
6
7         //default constructor of immediate parent called when mentioned.
8         bmw13 b1=new bmw13();
9
10        bmw13 b11=new bmw13(10);
11
12        bmw13 b111=new bmw13(10,"blue");
13
14    }
15
16 }
17
18
19

```

```
18
19
20 //100
21 //car one parameter constructor
22 //bmw default constructor
23 //100
24 //10
25 //car one parameter constructor
26 //10
27 //bmw one parameter constructor
28 //car start method
29 //-----
30 //static method in car class
31 //a
32 //100000
33 //10
34 //blue
35 //car two parameter constructor
36 //10
37 //blue
38 //bmw two parameter constructor
39 //-----
40 //static method in car class
41 //a
42 //100000
```

```

41 //a
42 //100000
43
44
45
46
47
48
49
50
51
52
53

```

Inside method we cannot use super as first line  
and call parent class constructor-

audi class-

//Constructor call must be the first statement in a constructor

```

23
24     //method
25     public void billing() {
26         super(10);
27     }
28

```

Error.

Method cannot call constructor.

Constructor calls another constructor.

```

34 //super:
35 //1. is used to call parent class const... from the child class const...
36 //2. but it should be used as first statement in child class const...
37 //3. is used to call parent class vars/methods in the child class
38 //4. if child class has overridden method (start), now If I want to call parent class (start) method:
39 //we have to super.start()

```

constructor can call method-

paste bmw15, test15-

```

bmw15.java  test15.java
1 package com.day21;
2
3 public class bmw15 extends car13{
4
5     //Super will go to grandparent if immediate parent does not have the variable or method.
6
7     //own speed method.
8     int speed=123324;
9
10    //default constructor.
11    public bmw15() {
12        super(100);
13        System.out.println("bmw default constructor");
14        //Super for variables and methods can be anywhere.
15        System.out.println(super.speed);
16
17        //can constructor call method.
18        //yes allowed.
19        refuel();
20    }
21
22    //one parameter constructor.
23    public bmw15(int speed) {
24        super(speed); //can pass parameter also to parent.
25        //not necessary to hard code everytime.
26        this.speed=speed;
27        System.out.println(this.speed);
28        System.out.println("bmw one parameter constructor");
29        //Super for variables and methods can be anywhere.
30        super.start();
31        System.out.println("-----");
32
33        //can access static methods also this way.
34        String s1=super.testing(10);
35        System.out.println(s1);
36
37        //access static variable.
38        int i1=super.duration;
39        System.out.println(i1);
40    }

```

```

39         System.out.println(i1);
40     }
41
42     //two parameter constructor.
43     public bmw15(int speed, String color) {
44         super(speed, color); //can pass parameter also to parent.
45         //not necessary to hard code everytime.
46         this.speed=speed;
47         System.out.println(this.speed);
48     //    System.out.println(this.color);//color cannot be resolved or is not a field
49     //    System.out.println(color);
50     //    System.out.println("bmw two parameter constructor");
51     //    System.out.println("-----");
52
53     //can access static methods also this way.
54     //this is better way to access static using class name.
55     String s2=car13.testing(20);
56     System.out.println(s2);
57
58     //access static variables.
59     //best way using class name.
60     int i1=car13.duration;
61     System.out.println(i1);
62 }

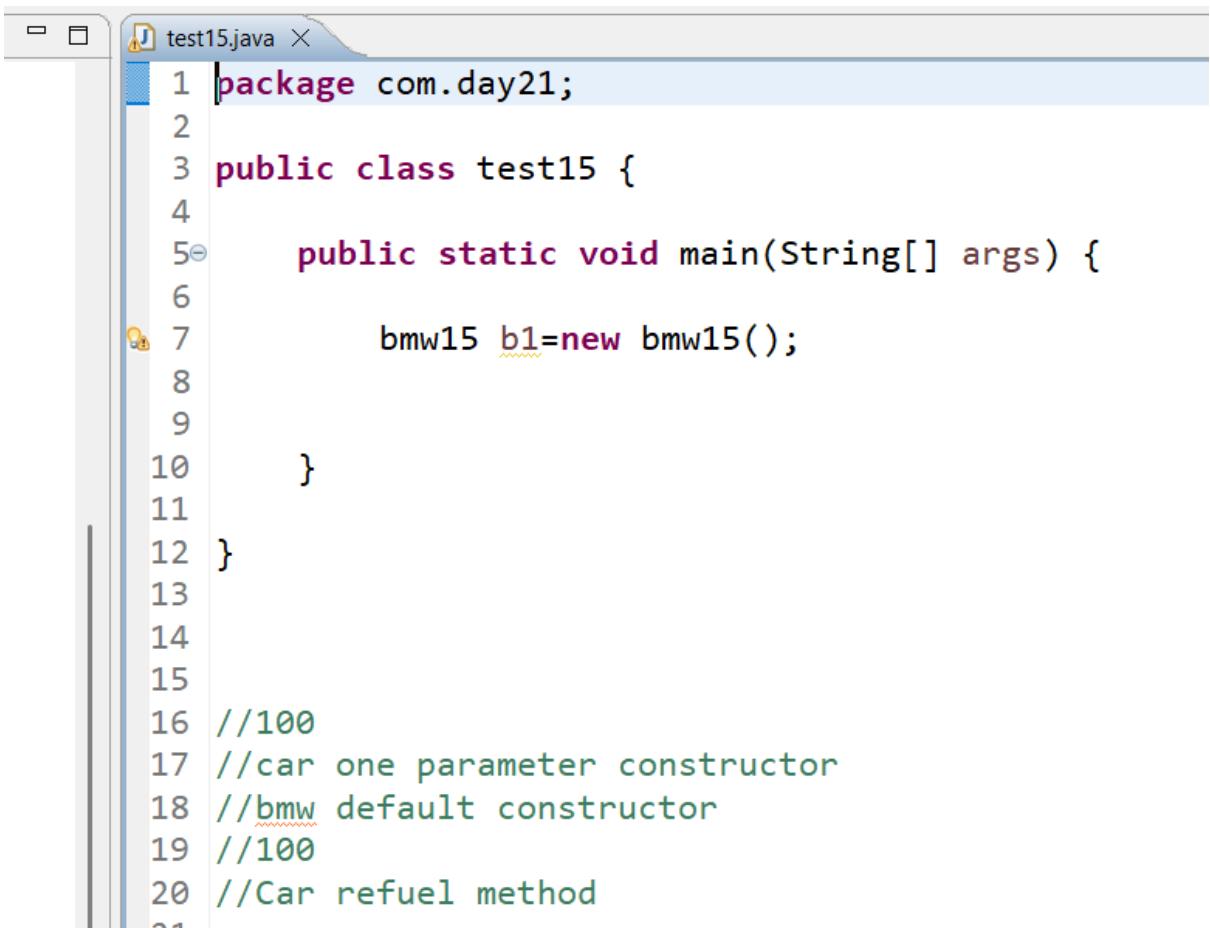
-----,
61     System.out.println(i1);
62 }
63
64 //override start method.
65 @Override
66 public void start() {
67     super.start();
68     System.out.println("bmw start method");
69 }
70
71 //add autoparking method.
72 public void autoparking() {
73     System.out.println("bmw auto parking");
74 }
75
76 public void displayspeed() {
77     System.out.println("speed of bmw is "+ this.speed);
78     System.out.println("speed of bmw is "+ speed);
79     //call parent class speed
80 //    System.out.println("speed of parent is " +super.this.speed);
81     //Syntax error on token "this", Identifier expected
82     int i1=super.speed;
83     System.out.println("speed of parent is " +i1);
84

```

```

82     int i1=super.speed;
83     System.out.println("speed of parent is " +i1);
84
85     //try accessing grand parent speed.
86     //Syntax error on token "super", Identifier expected
87 //    super.super.speed;
88     String s1=super.vehiclestatus(10);
89     System.out.println(s1);
90 }
91
92 // //inside method we cannot use super and call constructor
93 // //method cannot call constructor.
94 // public void billing() {
95 //     super(10); //Constructor call must be the first statement in a constructor
96 // }
97
98 }
99

```



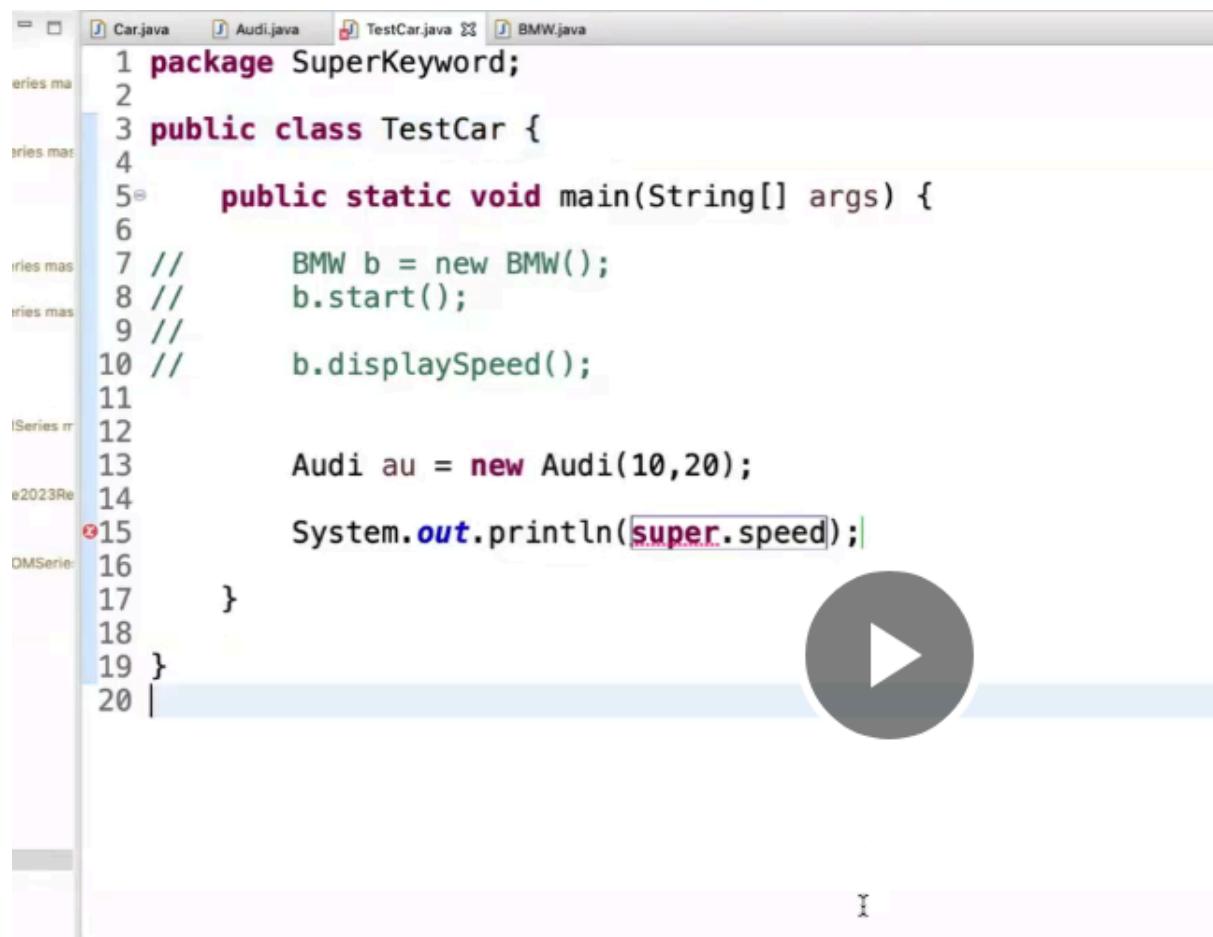
The screenshot shows a Java code editor with the file `test15.java` open. The code defines a class `test15` with a main method. It creates an object `b1` of type `bmw15`. The code includes several comments and blank lines.

```

1 package com.day21;
2
3 public class test15 {
4
5     public static void main(String[] args) {
6
7         bmw15 b1=new bmw15();
8
9     }
10
11 }
12
13
14
15
16 //100
17 //car one parameter constructor
18 //bmw default constructor
19 //100
20 //Car refuel method
21

```

You cannot use super for class which does not extend anything-



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

```
1 package SuperKeyword;
2
3 public class TestCar {
4
5     public static void main(String[] args) {
6
7         // BMW b = new BMW();
8         b.start();
9
10        // b.displaySpeed();
11
12
13        Audi au = new Audi(10,20);
14
15        System.out.println(super.speed);
16
17    }
18
19 }
20
```

A red error squiggle is underlined over the word "super" in the line `System.out.println(super.speed);`. A tooltip or status bar at the bottom right of the editor window displays the message "I".

//super cannot be used for class which does not extend anything.  
// we don't get options after dot.

paste test16-

```
test16.java X
1 package com.day21;
2
3 public class test16 {
4
5     public static void main(String[] args) {
6
7         bmw15 b1=new bmw15();
8
9         //super cannot be used for class which does not extend anything.
10 // we dont get options after dot.
11
12     }
13
14 }
15
16
17
18
19 //100
20 //car one parameter constructor
21 //bmw default constructor
22 //100
23 //Car refuel method
24
25
26
27
28
29
30
31
32
33 //super:
34 //1. is used to call parent class const... from the child class const...
35 //2. but it should be used as first statement in child class const...
36 //3. is used to call parent class vars/methods in the child class
37 //4. if child class has overridden method (start), now If I want to call parent class (start) method:
38 //we have to super.start()
39 //5. if immediate parent class has not vars/methods, super can be used to call methods/vars from grand
40 //6. super is only applicable for child classes
41
42 //this:
43 //1. is used to initialize the class vars(global) with local variables
44 //2. can be used in methods/const...
45 //3. can be used in the same class const.. calling
46
47
48
```

Honda-



```

1 package SuperKeyword;
2
3 public class Honda extends Car{
4
5     public Honda() {
6         System.out.println("Honda -- default const...");}
7
8
9     public Honda(int a) {
10        System.out.println("Honda -- const..." + a);}
11
12
13    public Honda(int a, int b) {
14        System.out.println("Honda -- const..." + (a+b));}
15
16
17
18

```

## Honda-

```

5     public Honda() {
6         this(10);|
7         System.out.println("Honda -- default const...");}
8

```



```

3 public class Honda extends Car{
4
5     public Honda() {
6         this(10); //is calling same class const..|
7         System.out.println("Honda -- default const...");}
8
9
10    public Honda(int a) {
11        System.out.println("Honda -- const..." + a);}
12
13
14    public Honda(int a, int b) {
15        System.out.println("Honda -- const..." + (a+b));}
16
17
18
19
20 }
21

```

## Test-

```

1 package SuperKeyword;
2
3 public class TestCar {
4
5     public static void main(String[] args) {
6
7         //        BMW b = new BMW();
8         //        b.start();
9         //
10        //       b.displaySpeed();
11
12
13        //Audi au = new Audi(10,20);
14
15
16        Honda h = new Honda();
17
18    }
19
20 }
21

```

```

May2024JavaSession / SuperKeyword/Honda.java - Eclipse IDE
Car -- default const...
is Car Honda -- const...10
Honda -- default const...

```

paste car18, honda18, test18-

The screenshot shows a Java code editor with three tabs at the top: car18.java, honda18.java, and test18.java. The car18.java tab is active, displaying the following Java code:

```
1 package com.day21;
2
3 public class car18 extends vehicle13 {
4
5     int speed=100;
6     static int duration=100000;
7
8     //one parameter constructor.
9     public car18(int speed) {
10         this.speed=speed;
11         System.out.println(this.speed);
12         System.out.println("car one parameter constructor");
13     }
14
15     //two parameter constructor.
16     public car18(int speed, String color) {
17         this.speed=speed;
18         System.out.println(this.speed);
19         // System.out.println(this.color); //color cannot be resolved or is not a field
20         System.out.println(color);
21         System.out.println("car two parameter constructor");
22     }
23
24     public void start() {
25         System.out.println("car start method");
26     }
27
28     public void refuel() {
29         System.out.println("Car refuel method");
30     }
31
32     //static method in car class.
33     public static String testing(int a) {
34         System.out.println("static method in car class");
35         return "a";
36     }
37
38 }
39
```

```

1 package com.day21;
2
3 public class honda18 extends car18{
4
5     public honda18() {
6         //to call same class constructor.
7         this(10);
8         System.out.println("Honda -- default const...");
9     }
10
11    public honda18(int a) {
12        this(1,2);
13        System.out.println("Honda -- const..." + a);
14    }
15
16    //Implicit super constructor car18() is undefined. Must explicitly invoke another constructor
17    //we have to call another constructor explicitly.
18    //public honda18(int a, int b) {
19    //    System.out.println("Honda -- const..." + (a + b));
20    //}
21
22    public honda18(int a, int b) {
23        super(a);
24        System.out.println("Honda -- const..." + (a + b));
25    }
26
27 }

```

```

1 package com.day21;
2
3 public class test18 {
4
5     public static void main(String[] args) {
6
7         //parent class default constructor if present will always be called.
8         //when we dont use super.
9
10        honda18 h1=new honda18();
11
12    }
13
14 }
15
16 //1
17 //car one parameter constructor
18 //Honda -- const...3
19 //Honda -- const...10
20 //Honda -- default const...
21
22
23

```

“This” also can't be second sentence-

Constructor call has to happen at first line.

//Constructor call must be the first statement in a constructor

```

4
5  public Honda() {
6      //is calling same class const..
7      System.out.println("Honda -- default const..."); 
8      this(10);
9  }
10

```

Can this and super be together-

//Constructor call must be the first statement in a constructor

Honda-

Not allowed as super becomes second statement.

```

5  public Honda() {
6      this(10); //is calling same class const..
7      super(10); //not allowed
8      System.out.println("Honda -- default const..."); 
9  }
10

```

Now this is in second statement-

```

4
5  public Honda() {
6      //is calling same class const..
7      super(10); //not allowed
8      this(10);
9      System.out.println("Honda -- default const..."); 
10
11

```

Super and this can never come for constructor together.

Honda-

```

10
11  public Honda(int a) {
12      this(10, 20);
13      System.out.println("Honda -- const..." + a);
14  }
15
16  public Honda(int a, int b) {
17      System.out.println("Honda -- const..." + (a+b));
18  }

```

## Test-




```

13
14
15
16     Honda h = new Honda(10);
17
18 }

```

JavaSession > SuperKeyword/TestCar.java - Eclipse IDE

Console Problems Javadoc Declaration Run

<terminated> TestCar (15) [Java Application] /Users/neveenautomatio

Car -- default const...  
Honda -- const...30  
Honda -- const...10

Recursive constructor not allowed-

Recursion only for methods not for constructors.

Honda-

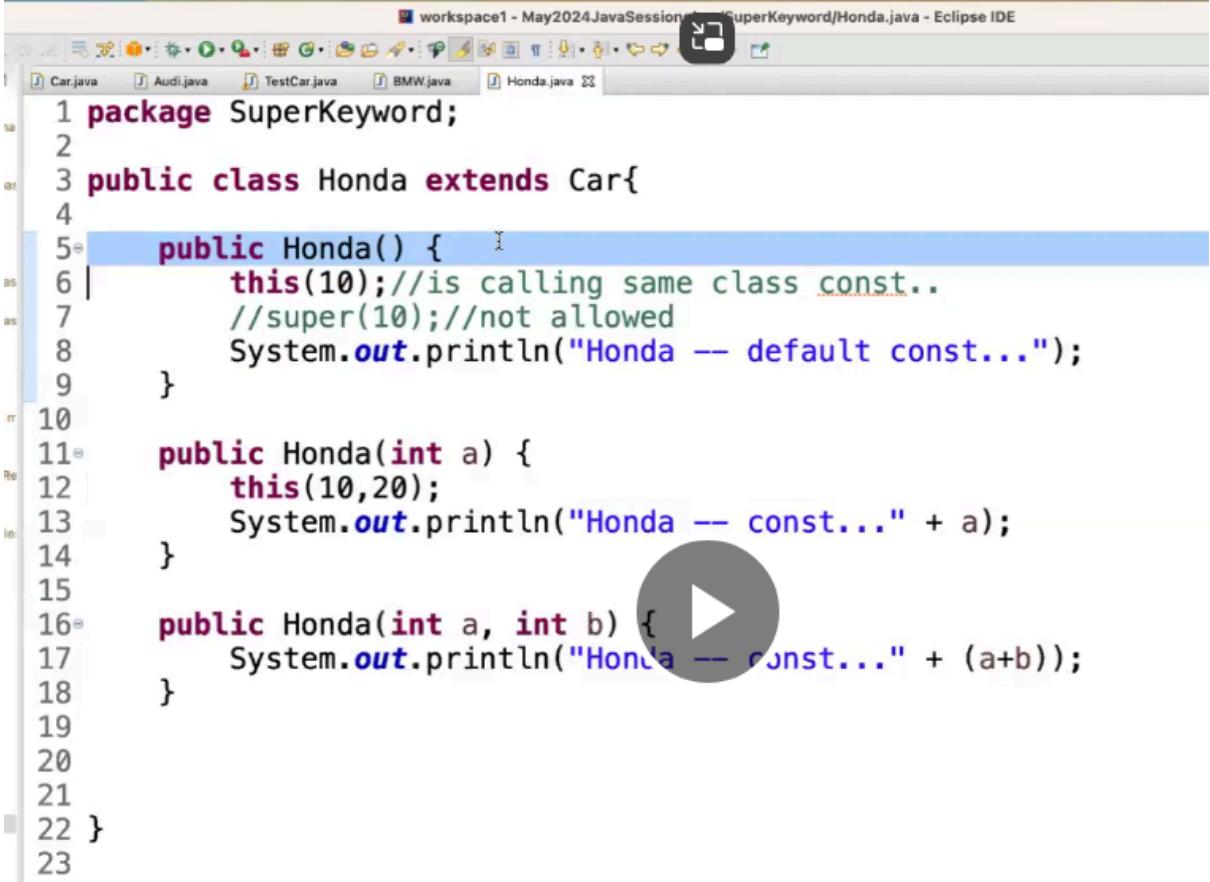
//Recursive constructor invocation honda21(int)

```

10
11  public Honda(int a) {
12      this(10);
13      System.out.println("Honda -- const..." + a);
14  }

```

## honda class-



```

1 package SuperKeyword;
2
3 public class Honda extends Car{
4
5     public Honda() {
6         this(10); //is calling same class const..
7         //super(10); //not allowed
8         System.out.println("Honda -- default const...");
9     }
10
11    public Honda(int a) {
12        this(10,20);
13        System.out.println("Honda -- const..." + a);
14    }
15
16    public Honda(int a, int b) {
17        System.out.println("Honda -- const..." + (a+b));
18    }
19
20
21
22 }
23

```

see how data flows from one const to another const ....

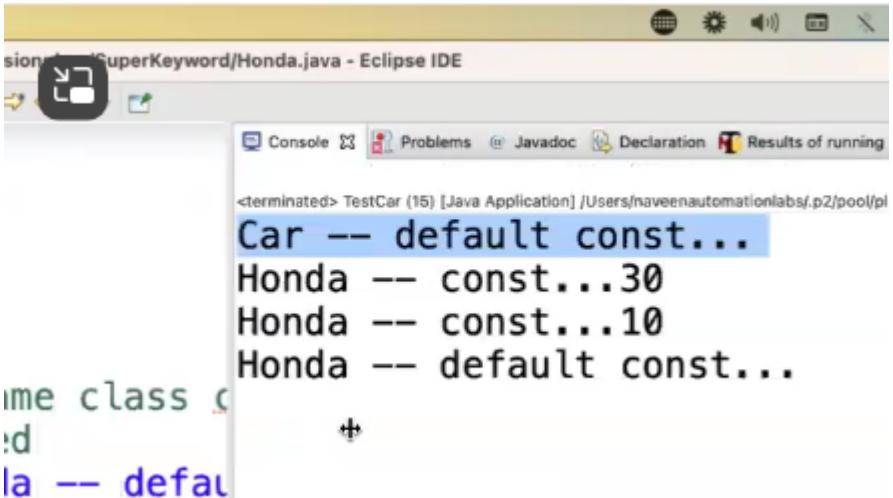
## Test –



```

14
15
16     Honda h = new Honda();
17
18

```



The screenshot shows the Eclipse IDE interface. The title bar says "siony / SuperKeyword/Honda.java - Eclipse IDE". The toolbar includes icons for file operations like New, Open, Save, and Run. Below the toolbar is a menu bar with "Console", "Problems", "Javadoc", "Declaration", and "Results of running". The main area shows Java code and its execution output. The code includes a class definition and some print statements. The output window shows the results of the program's execution.

```

<terminated> TestCar (15) [Java Application] /Users/naveenautomationlabs/p2/pool/pl
Car -- default const...
Honda -- const...30
Honda -- const...10
Honda -- default const...
ime class C
:d
la -- defau

```

```

42 //this:
43 //1. is used to initialize the class vars(global) with local variables
44 //2. can be used in methods/const...
45 //3. can use be used in the same class for the same class cosnt.. calling
46 //4. this() in const... should be the first statement
47 //5. this() and super() can not be together
48 //6. Recursive const.. calling not allowed
49 //7. we can return this keyword from a method/function: Builder Pattern
50

```

“This” keyword can be used inside getter, setter, constructor, method to initialise.

Same for the super keyword – can be used.

“This” cannot be used inside static methods. “This” refers to the current class object. “This” deals with objects only.

```

1 package SuperKeyword;
2
3 public class Browser {
4
5     String name;
6     double version;
7
8     public void addBrowser(String name, double version) {
9         this.name = name;
10        this.version = version;
11    }
12
13
14     public static void main(String[] args) {
15         Browser br = new Browser();
16         br.addBrowser("chrome", 124.55);
17     }
18
19 }
20
21
22 }
23
24

```

## Interface-

```

1 package SuperKeyword;
2
3 public interface Automation {
4
5     int time = 10;
6
7 }
8
9 }
10

```

## Browser-

```

1 package SuperKeyword;
2
3 public class Browser implements Automation{
4
5     String name;
6     double version;
7
8
9     public void addBrowser(String name, double version) {
10         this.name = name;
11         this.version = version;
12
13         System.out.println(time);
14
15     }
16

```

10

Made it public to see if the super.time works in browser -

```

2
3 public interface Automation {
4
5     public int time = 10;
6

```

Browser-

Cannot access using super.

```
//      super./cannot use super with interface.
//we wont get options after super.
```

```

8
9  public void addBrowser(String name, double version) {
10    this.name = name;
11    this.version = version;
12
13    System.out.println(super.time);
14
15  }
16

```

A tooltip box appears at the bottom right of the code editor, containing the text "time cannot be resolved or is not a field" and "Press 'F2' for focus".

Super used to access only other class, the class can be abstract or normal.

Interface variables are static final by default.

Use class names-

```

5
6  public void addBrowser(String name, double version) {
7    this.name = name;
8    this.version = version;
9
10   System.out.println(Automation.time);
11
12 }
13
14
15
16
17

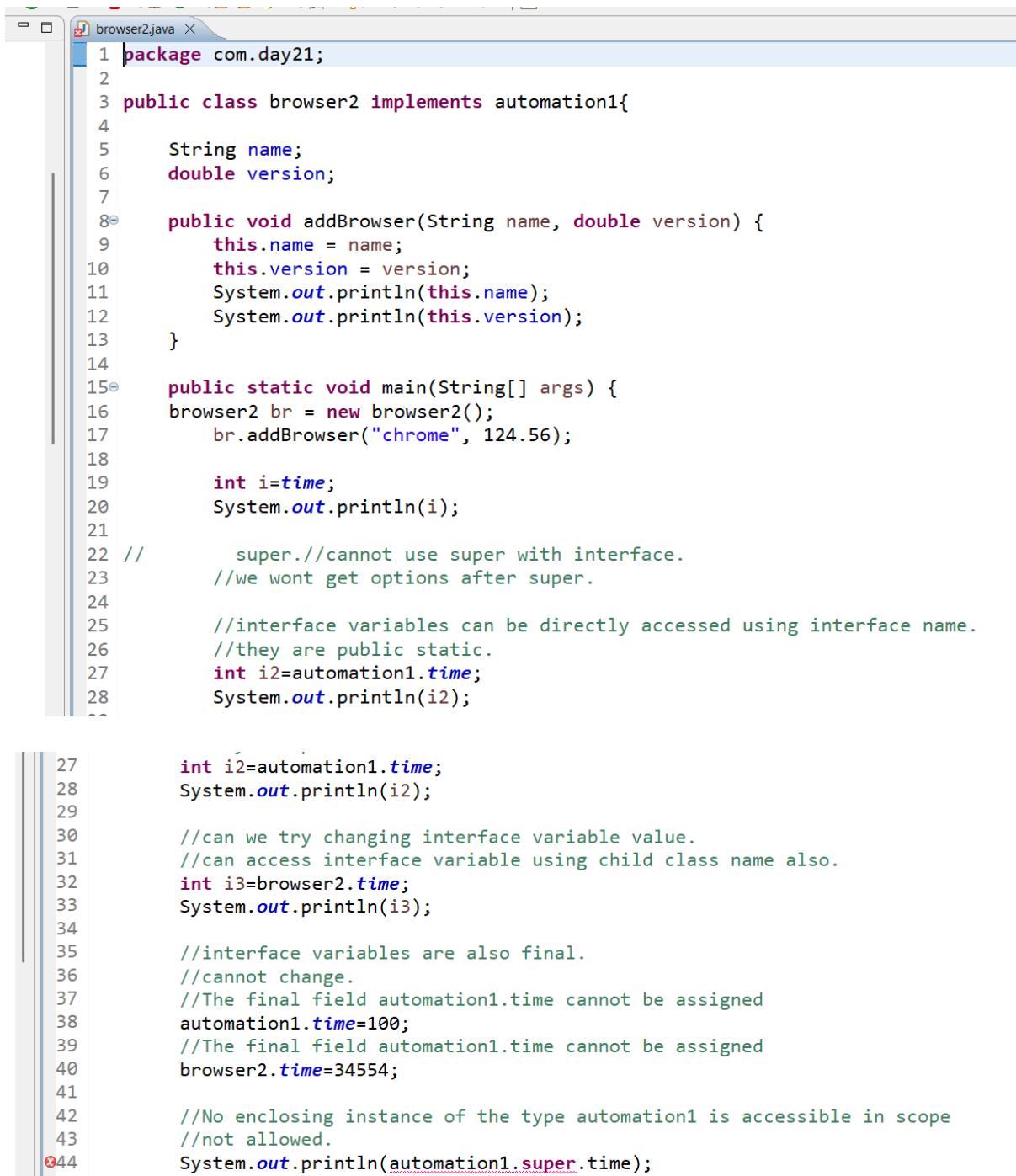
```

paste browser2, automation1-

```

1 package com.day21;
2
3 public interface automation1 {
4
5   int time=10;
6
7 }
8

```



```

1 package com.day21;
2
3 public class browser2 implements automation1{
4
5     String name;
6     double version;
7
8     public void addBrowser(String name, double version) {
9         this.name = name;
10        this.version = version;
11        System.out.println(this.name);
12        System.out.println(this.version);
13    }
14
15    public static void main(String[] args) {
16        browser2 br = new browser2();
17        br.addBrowser("chrome", 124.56);
18
19        int i=time;
20        System.out.println(i);
21
22 //        super.//cannot use super with interface.
23 //        //we wont get options after super.
24
25        //interface variables can be directly accessed using interface name.
26        //they are public static.
27        int i2=automation1.time;
28        System.out.println(i2);
29
30        int i2=automation1.time;
31        System.out.println(i2);
32
33        //can we try changing interface variable value.
34        //can access interface variable using child class name also.
35        int i3=browser2.time;
36        System.out.println(i3);
37
38        //interface variables are also final.
39        //cannot change.
40        //The final field automation1.time cannot be assigned
41        automation1.time=100;
42        //The final field automation1.time cannot be assigned
43        browser2.time=34554;
44
45        //No enclosing instance of the type automation1 is accessible in scope
46        //not allowed.
47        System.out.println(automation1.super.time);
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2
```

```

43 //not allowed.
44 System.out.println(automation1.super.time);
45
46 //Cannot use super in a static context
47 //time cannot be resolved or is not a field
48 System.out.println(browser2.super.time);
49 }
50 }
51
52 //chrome
53 //124.56
54 //10
55 //10
56 //10
57
58

```

This is also incorrect syntax-

browser class - add browser method-

```
//No enclosing instance of the type automation1 is accessible in scope
```

```

13     System.out.println(Automation.time);
14     System.out.println(Automation.super.time);|

```

This is also not allowed-

browser class - add browser method-

```
//Cannot use super in a static context
```

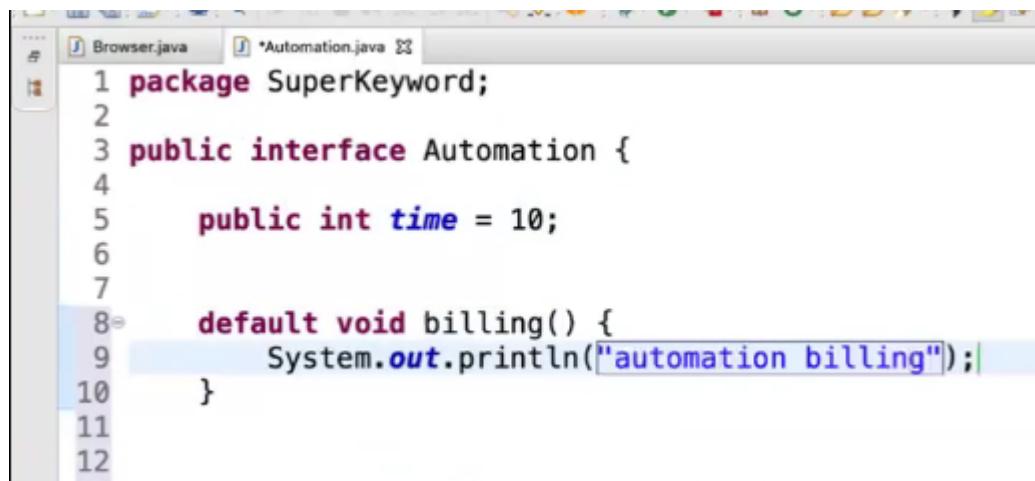
```
//time cannot be resolved or is not a field
```

```

13     System.out.println(Automation.time);
14     System.out.println(Browser.super.time);|

```

Added default method in interface-



```

1 package SuperKeyword;
2
3 public interface Automation {
4
5     public int time = 10;
6
7
8     default void billing() {
9         System.out.println("automation billing");
10    }
11
12

```

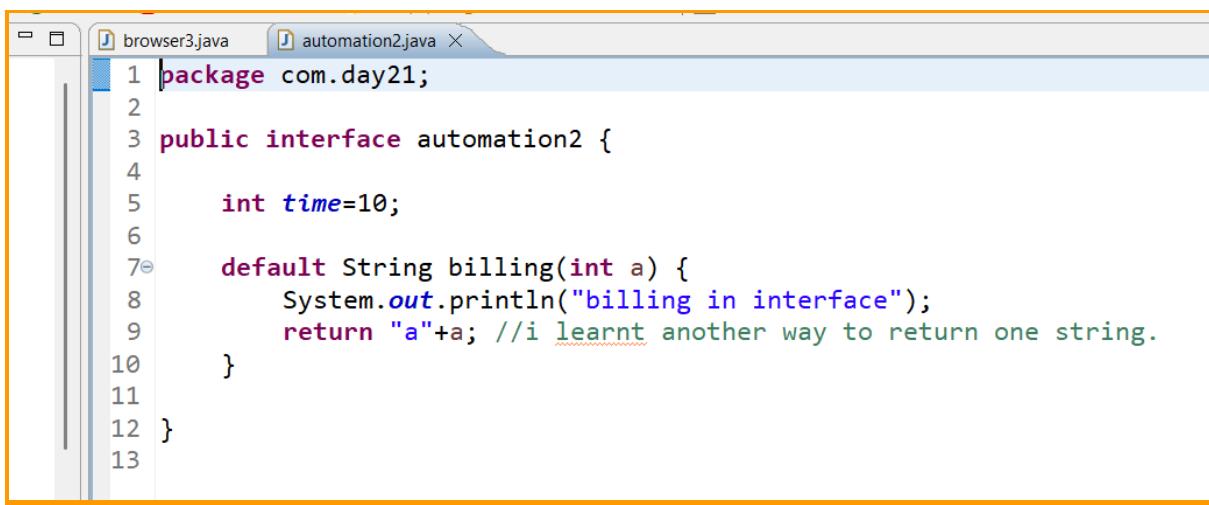
## Browser-

```

8
9     public void addBrowser(String name, double version) {
10         this.name = name;
11         this.version = version;
12
13         System.out.println(Automation.time);
14
15         billing();
16

```

paste automation2, browser3-



```

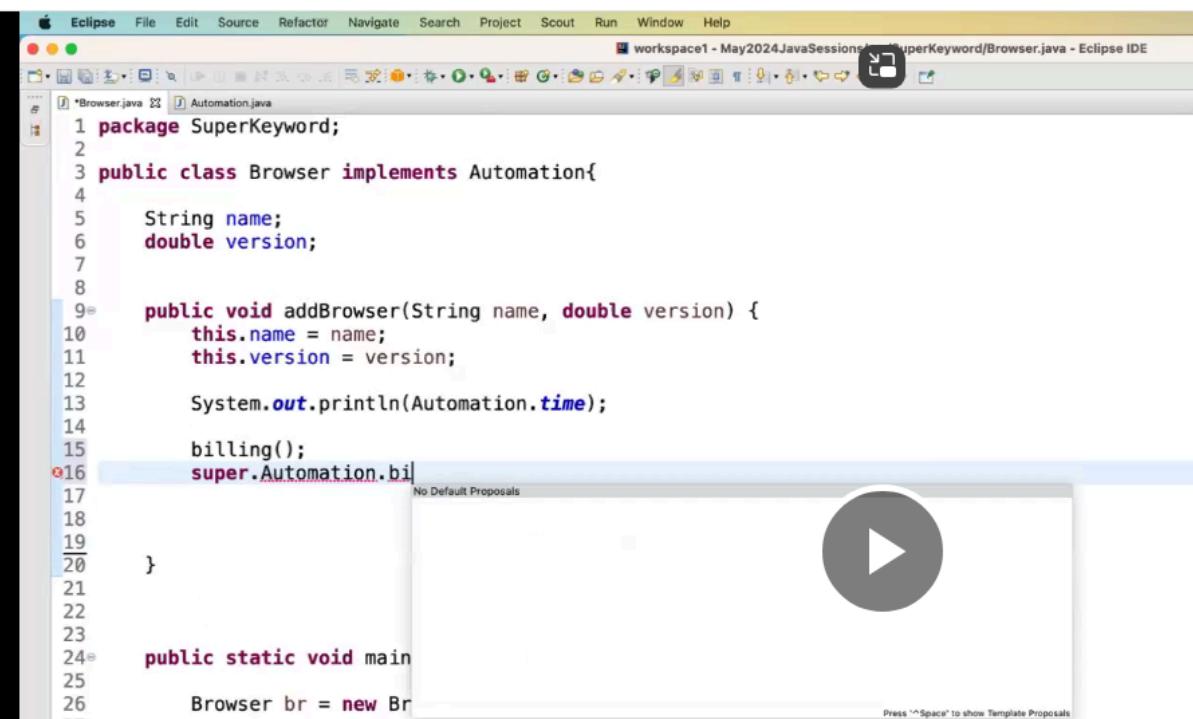
1 package com.day21;
2
3 public interface automation2 {
4
5     int time=10;
6
7     default String billing(int a) {
8         System.out.println("billing in interface");
9         return "a"+a; //i learnt another way to return one string.
10    }
11
12 }
13

```

```
browser3.java
1 package com.day21;
2
3 public class browser3 implements automation2{
4
5     String name;
6     double version;
7
8     public void addBrowser(String name, double version) {
9         this.name = name;
10        this.version = version;
11        System.out.println(this.name);
12        System.out.println(this.version);
13
14        String s1= billing(10);
15        System.out.println(s1);
16
17 //    super.//no options after super to call parent.
18
19        automation2.super.billing(25);
20
21        String s11=automation2.super.billing(67);
22        System.out.println(s11);
23
24
25        //No enclosing instance of the type automation2 is accessible in scope
26 //    int i1=automation2.super.time;
27    }
28
29
30    public static void main(String[] args) {
31
32        browser3 b1=new browser3();
33        String s2=b1.billing(45);
34        System.out.println(s2);
35 //    super.//no options after super.
36        b1.addBrowser("chrome", 435435);
37
38 //    //Cannot use super in a static context
39 //    automation2.super.billing(10);
40
41        //No enclosing instance of the type automation2 is accessible in scope
42 //    int i1=automation2.super.time;
43    }
}
```

```
42
43      }
44 }
45
46 //billing in interface
47 //a45
48 //chrome
49 //435435.0
50 //billing in interface
51 //a10
52 //billing in interface
53 //billing in interface
54 //a67
55
56
57
58
59
60
61
```

Cannot use super to access the method of interface.-



```

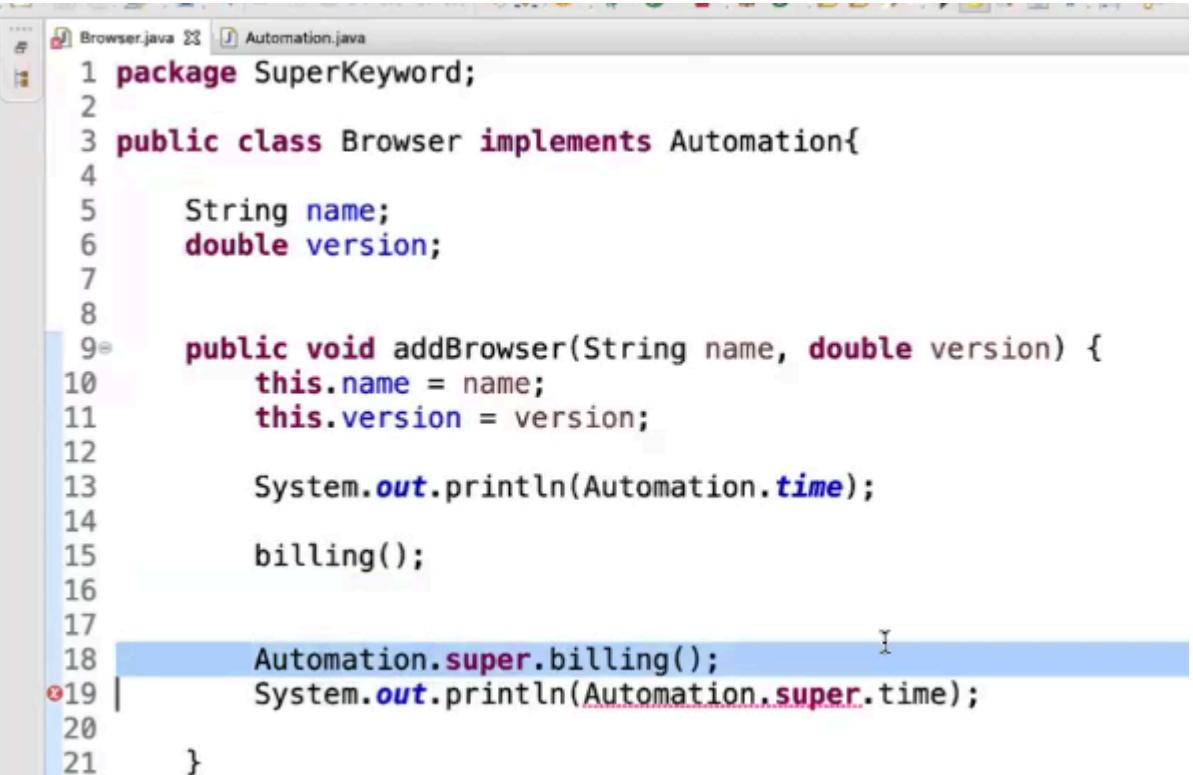
1 package SuperKeyword;
2
3 public class Browser implements Automation{
4
5     String name;
6     double version;
7
8
9     public void addBrowser(String name, double version) {
10         this.name = name;
11         this.version = version;
12
13         System.out.println(Automation.time);
14
15         billing();
16         super.Automation.bil
17
18     }
19
20
21
22
23
24     public static void main
25
26         Browser br = new Br
27

```

No Default Proposals

Press '^Space' to show Template Proposals

See unique way to access default methods-  
Variables still not allowed.



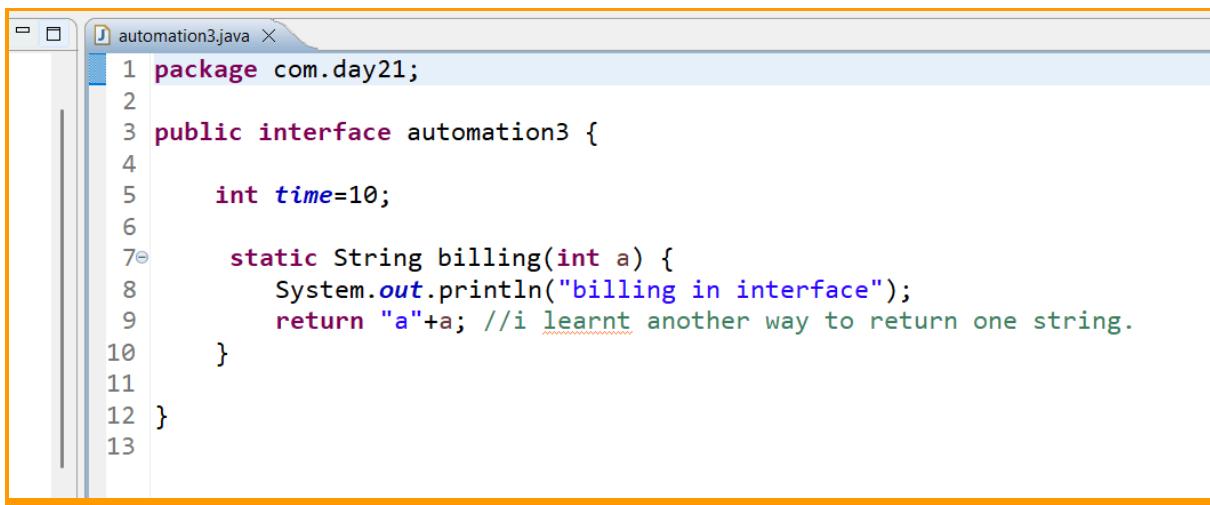
```

1 package SuperKeyword;
2
3 public class Browser implements Automation{
4
5     String name;
6     double version;
7
8
9     public void addBrowser(String name, double version) {
10         this.name = name;
11         this.version = version;
12
13         System.out.println(Automation.time);
14
15         billing();
16
17
18         Automation.super.billing();
19         System.out.println(Automation.super.time);
20
21     }
22

```

Used when multiple interfaces are implemented by one class and all interfaces are having same **default** method name (billing). if usmedical interface has billing, then we can call usmedical.super.billing. (1.28.04)

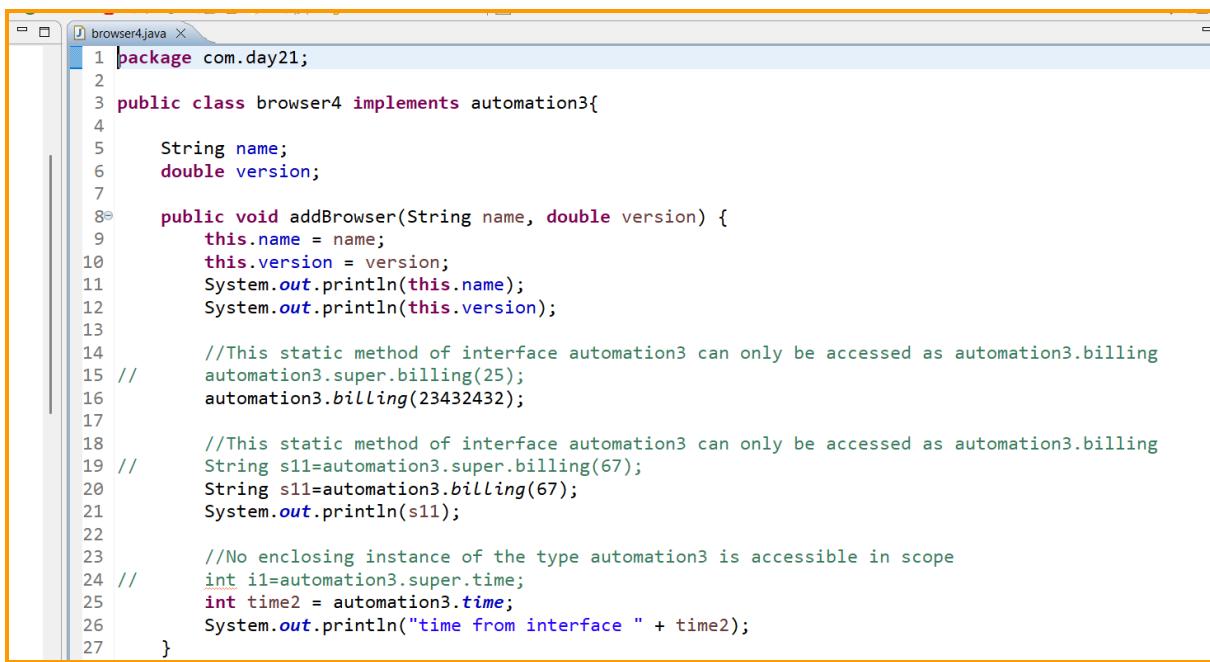
paste automation3, browser4-



```

1 package com.day21;
2
3 public interface automation3 {
4
5     int time=10;
6
7     static String billing(int a) {
8         System.out.println("billing in interface");
9         return "a"+a; //i learnt another way to return one string.
10    }
11
12 }
13

```



```

1 package com.day21;
2
3 public class browser4 implements automation3{
4
5     String name;
6     double version;
7
8     public void addBrowser(String name, double version) {
9         this.name = name;
10        this.version = version;
11        System.out.println(this.name);
12        System.out.println(this.version);
13
14        //This static method of interface automation3 can only be accessed as automation3.billing
15 //        automation3.super.billing(25);
16        automation3.billing(23432432);
17
18        //This static method of interface automation3 can only be accessed as automation3.billing
19 //        String s11=automation3.super.billing(67);
20        String s11=automation3.billing(67);
21        System.out.println(s11);
22
23        //No enclosing instance of the type automation3 is accessible in scope
24 //        int i1=automation3.super.time;
25        int time2 = automation3.time;
26        System.out.println("time from interface " + time2);
27    }

```

```
27 }
28
29 public static void main(String[] args) {
30
31     browser4 b1=new browser4();
32     b1.addBrowser("chrome", 435435);
33 }
34
35
36 //chrome
37 //435435.0
38 //billing in interface
39 //billing in interface
40 //a67
41 //time from interface 10
```



```
1 package SuperKeyword;
2
3 public class Browser implements Automation{
4
5     String name;
6     double version;
7
8     //setter
9     public void addBrowser(String name, double version) {
10         this.name = name;
11         this.version = version;
12
13         System.out.println(Automation.time);
14
15         billing();
16
17
18         Automation.super.billing();
19
20     }
21     //getter
22     public void getInfo() {
23         System.out.println(name + version);
24     }
25
26     public static void main(String[] args) {
27
28         Browser br = new Browser();
29
30         br.addBrowser("chrome", 124.55);
31
32         br.getInfo();
33
34
35
36     }
37 }
```

Chrome 124