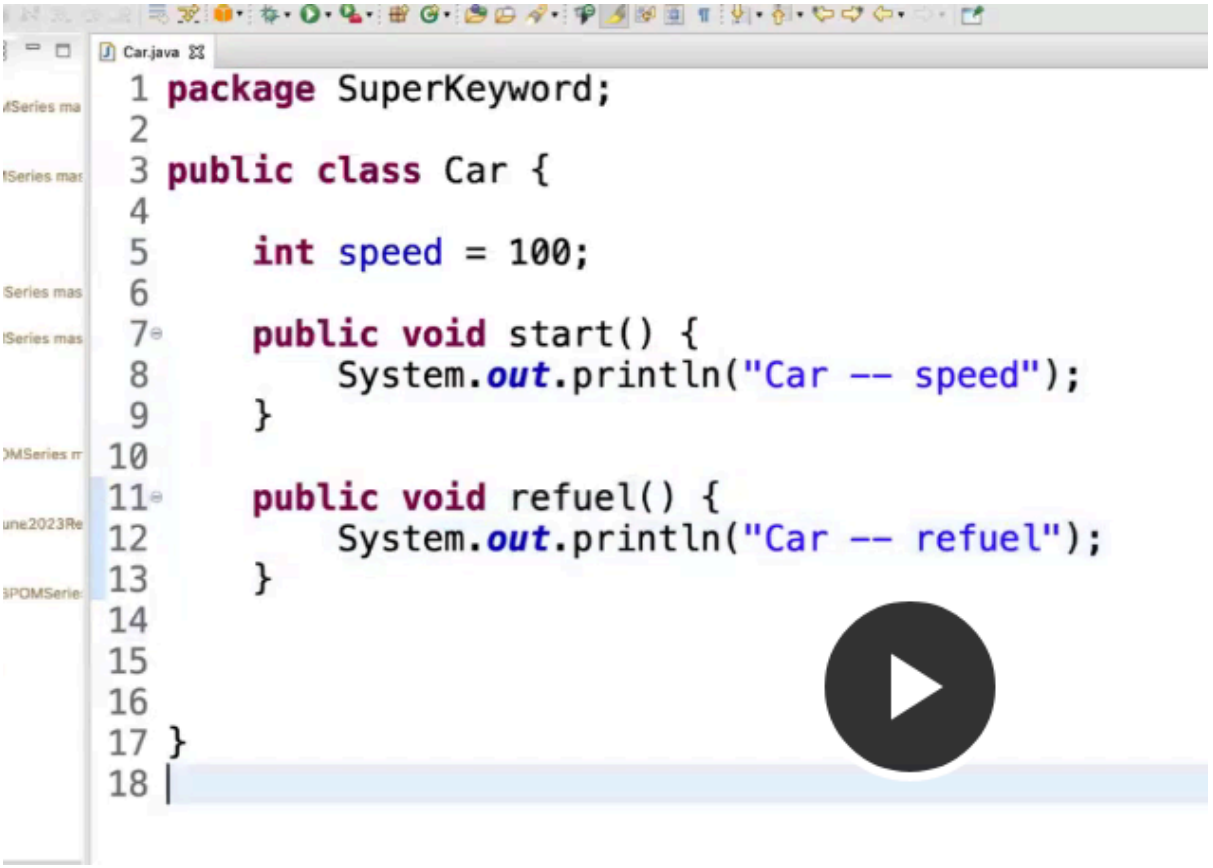
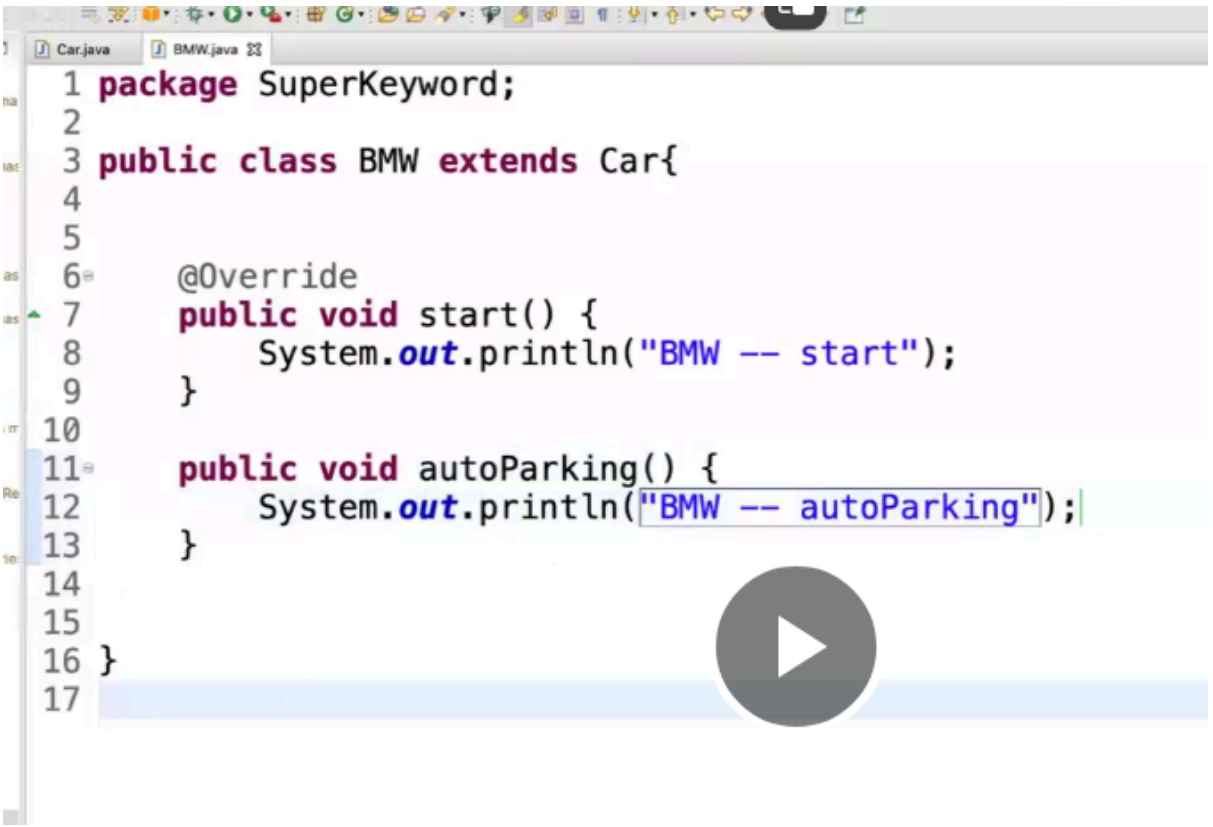
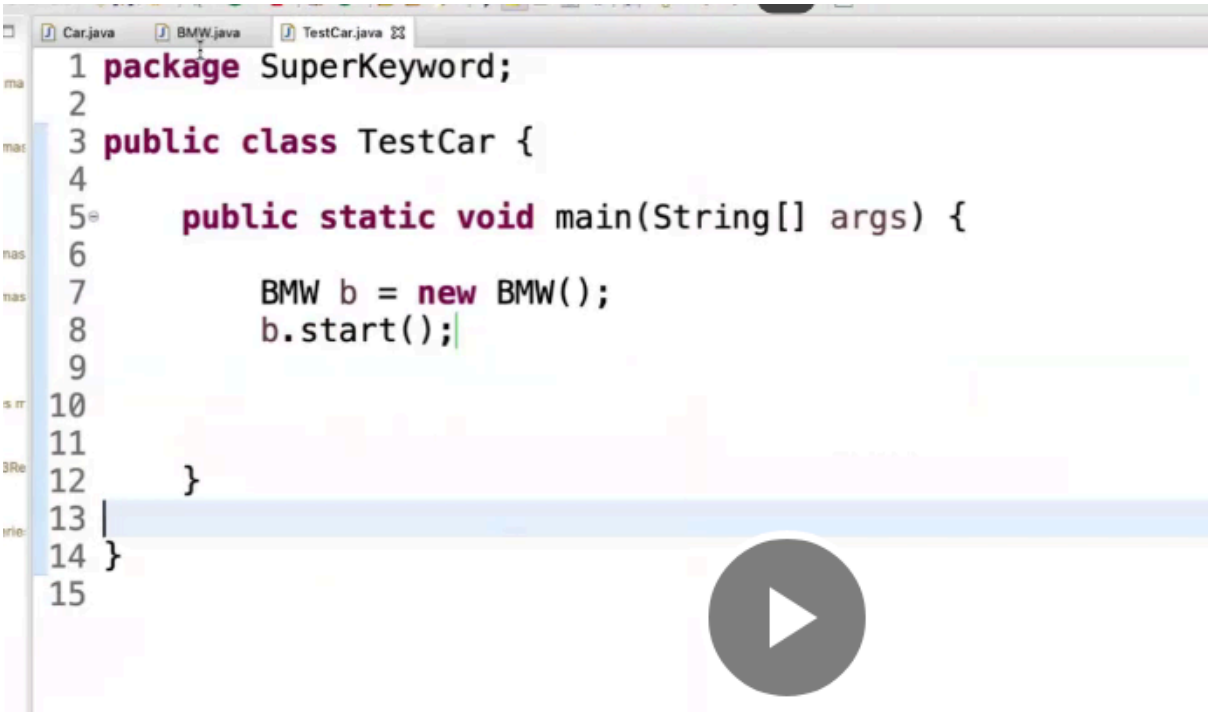
Car-


Car.java
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-



Test-



Bmw method called.

Bmw-

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

Stack overflow.

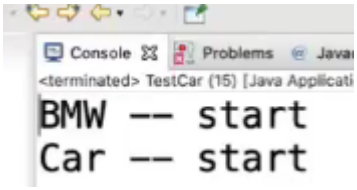
Call parent class overridden method-

bmw class-

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

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

Run test-



can call parent class variable also using super-

paste car4, bmw4, test4-

bmw4.java | car4.java | test4.java

```java
package com.day21;

public class car4 {

    int speed=100;

    public void start() {
        System.out.println("car start method");
    }

    public void refuel() {
        System.out.println("Car refuel method");
    }

}

```

bmw4.java | car4.java | test4.java

```java
package com.day21;

public class bmw4  extends car3{

    //override start method.
    @Override
    public void start() {
        //super in method can be written as first line or any line.
        super.start();
        System.out.println("bmw start method");
        //to call parent class method with same name.
//      super.start();
    }

    //add autoparking method.
    public void autoparking() {
        //we can call parent class variable also using super.
        int i1=super.speed;
        System.out.println(i1);
        System.out.println("bmw auto parking");
    }

}

```

test4.java

```java
package com.day21;

public class test4 {

    public static void main(String[] args) {

        bmw4 b1=new bmw4();
        b1.autoparking();
        b1.start();

    }

}

//100
//bmw auto parking
//car start method
//bmw start method

```

Super can be written at first or end of method-

```java
    @Override
    public void start() {
        super.start();//car
        System.out.println("BMW -- start");
    }
```

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   }
```

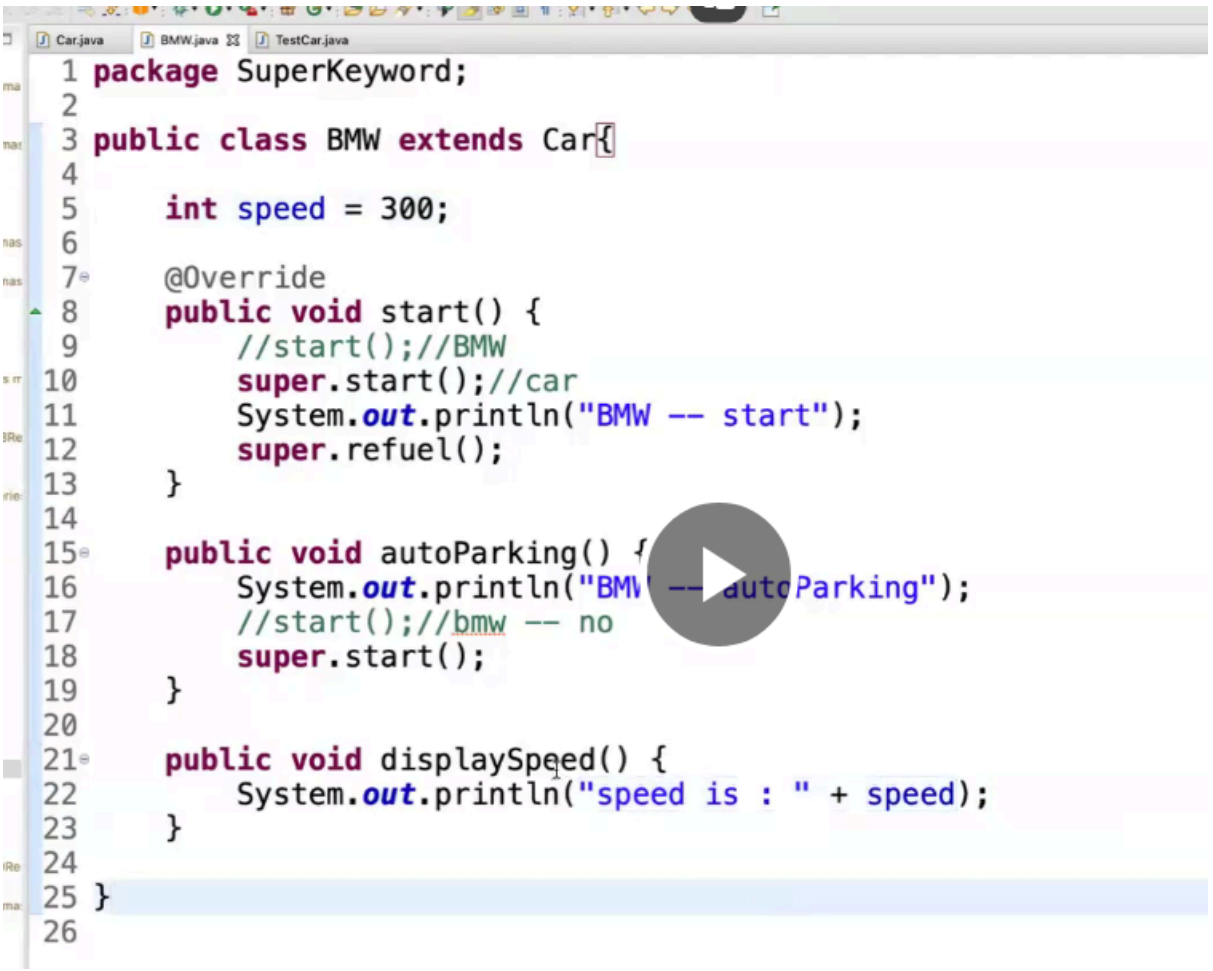
Can be used in non overridden methods-

bmw-

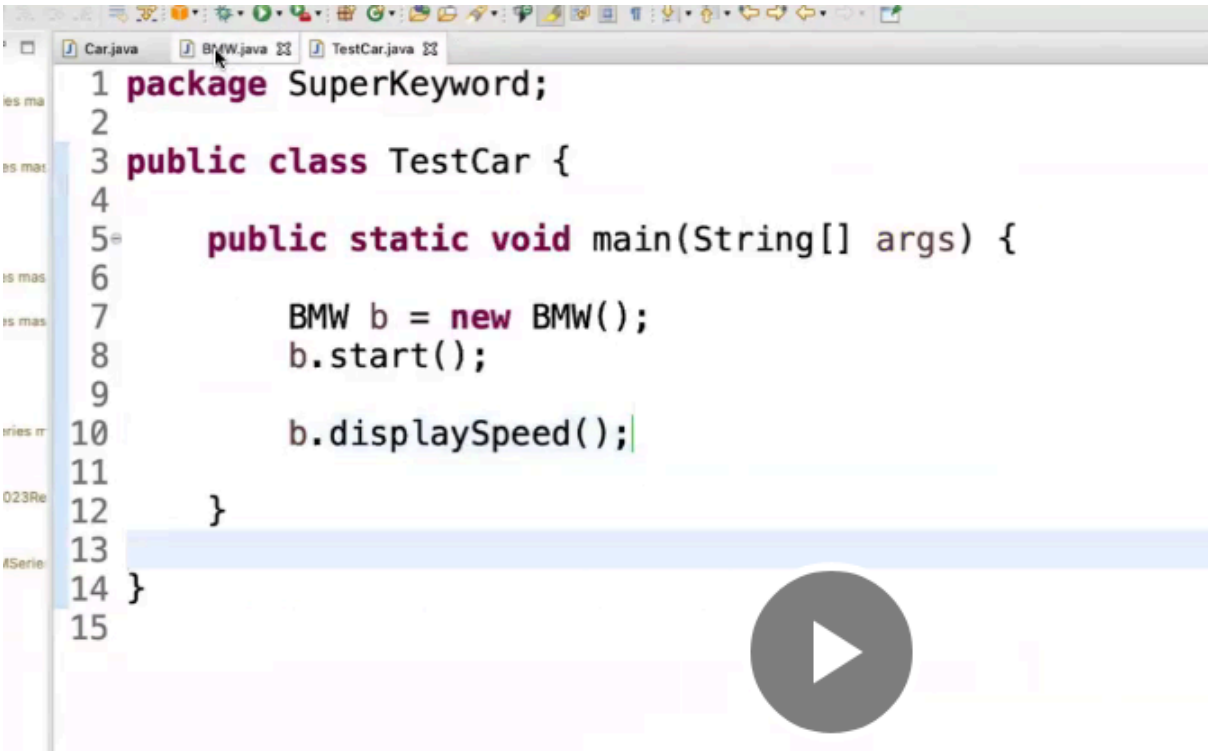
```
13
14   public void autoParking() {
15       System.out.println("BMW -- autoParking");
16       //start();//bmw -- no
17       super.start();
18   }
19
```

Bmw-

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

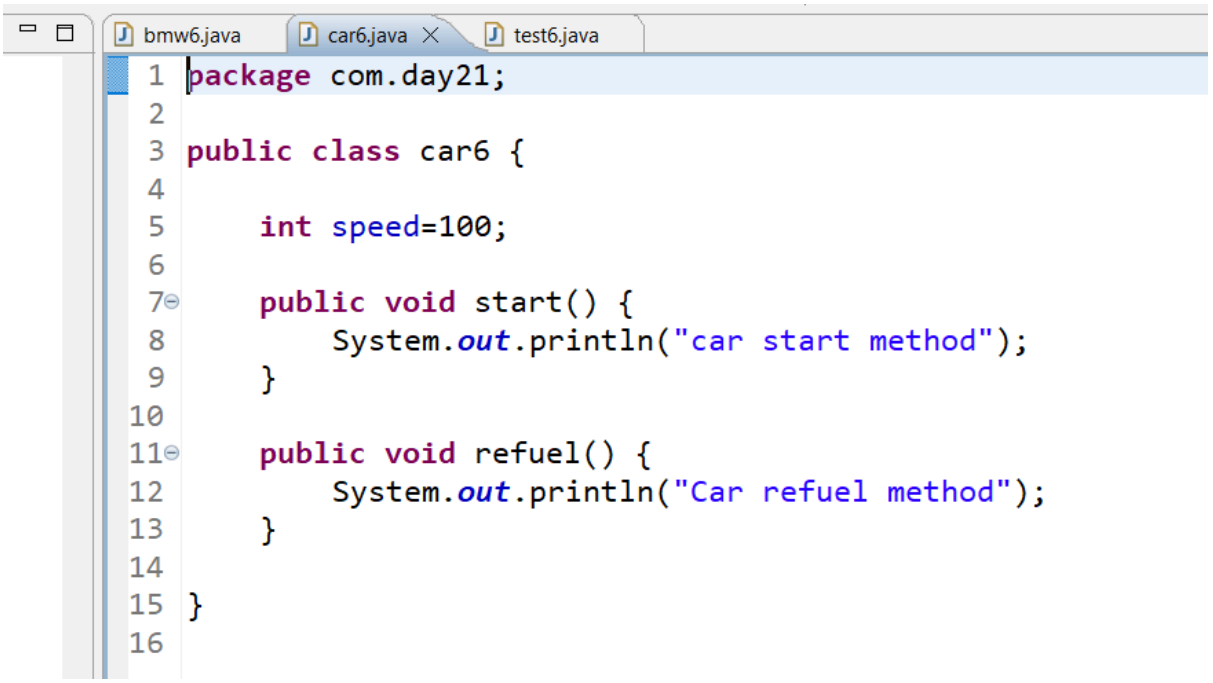

Car.java
BMW.java
TestCar.java
1 package SuperKeyword;
2
3 public class BMW extends Car{
4
5 int speed = 300;
6
7 @Override
8 public void start() {
9 //start();//BMW
10 super.start();//car
11 System.out.println("BMW -- start");
12 super.refuel();
13 }
14
15 public void autoParking() {
16 System.out.println("BMW -- 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-


Car.java
BMW.java
TestCar.java
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 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     }
```

```
28         System.out.println("speed of parent is " +i1);
29     }
30
31 }
32
```

test6.java

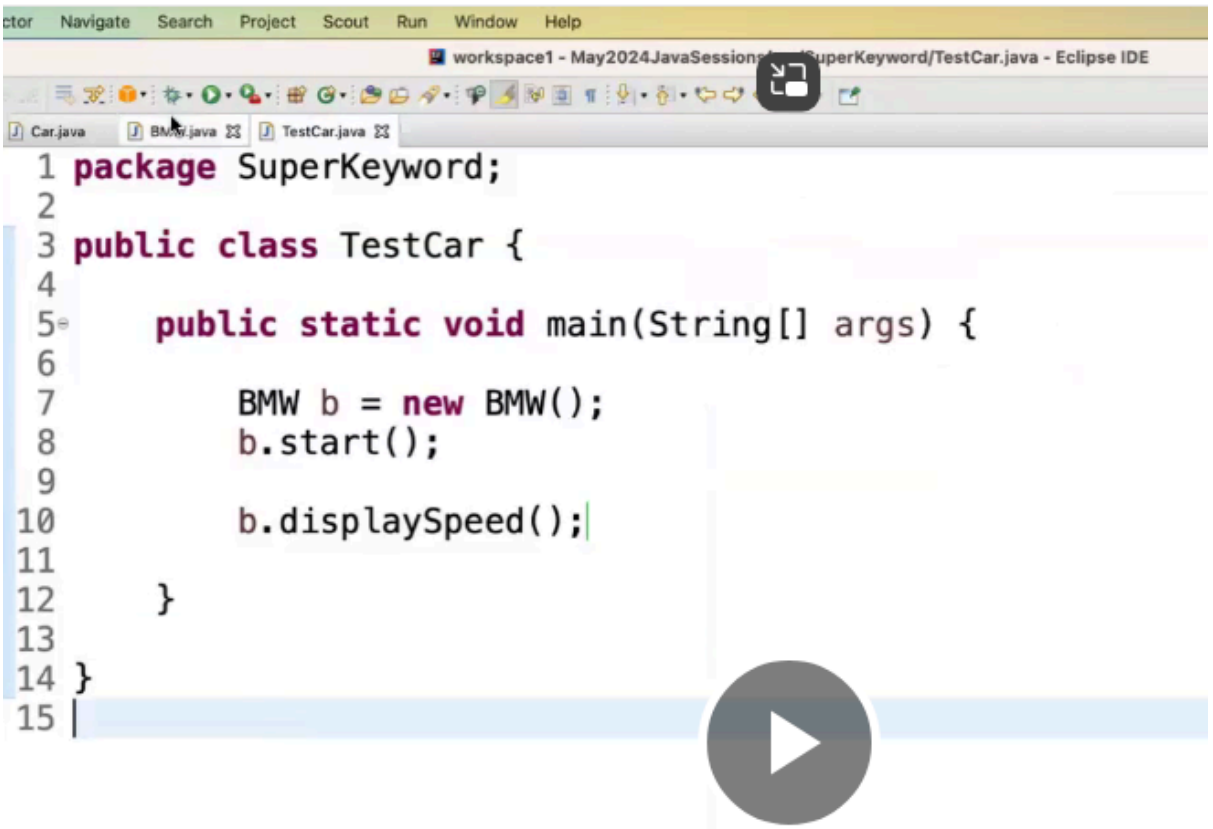
```java
package com.day21;

public class test6 {

    public static void main(String[] args) {

        bmw6 b1=new bmw6();
        int i1=b1.speed;
        System.out.println(i1);
        b1.displayspeed();

    }

}

//123324
//speed of bmw is 123324
//speed of bmw is 123324
//speed of parent is 100

```

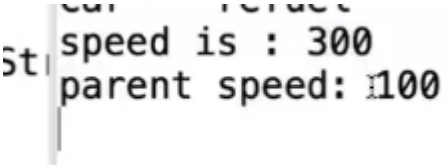
Parent class speed called-

Bmw-

```java
    public void displaySpeed() {
        System.out.println("speed is : " + speed);
        System.out.println("parent speed: " + super.speed);
    }
```

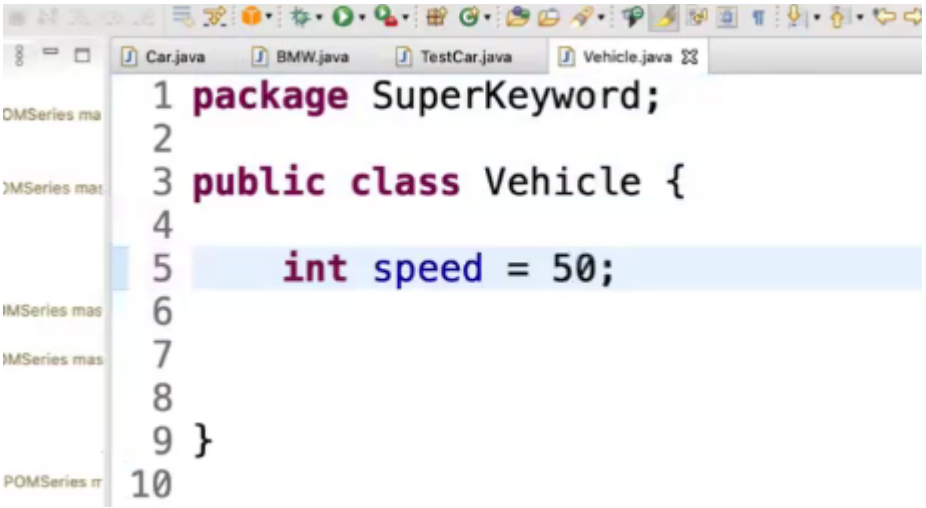
Run test.





Grand parent-

Vehicle-



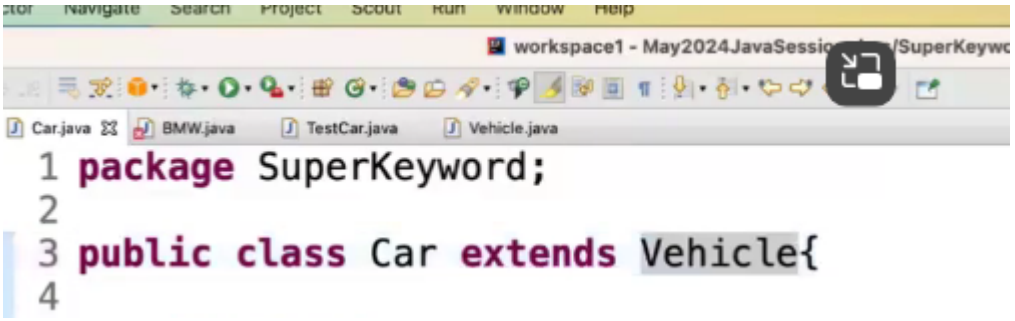
Not right syntax-

To access speed.

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

```
    public void displaySpeed() {
        System.out.println("speed is : " + speed);//300
        System.out.println("parent speed: " + super.speed);//100
        System.out.println(super.super.speed);
    }
```

Car-



bmw class-

```
    public void displaySpeed() {
        System.out.println("speed is : " + speed);//300
        System.out.println("parent speed: " + super.speed);//100
        System.out.println(super.speed);
    }
```

Run test

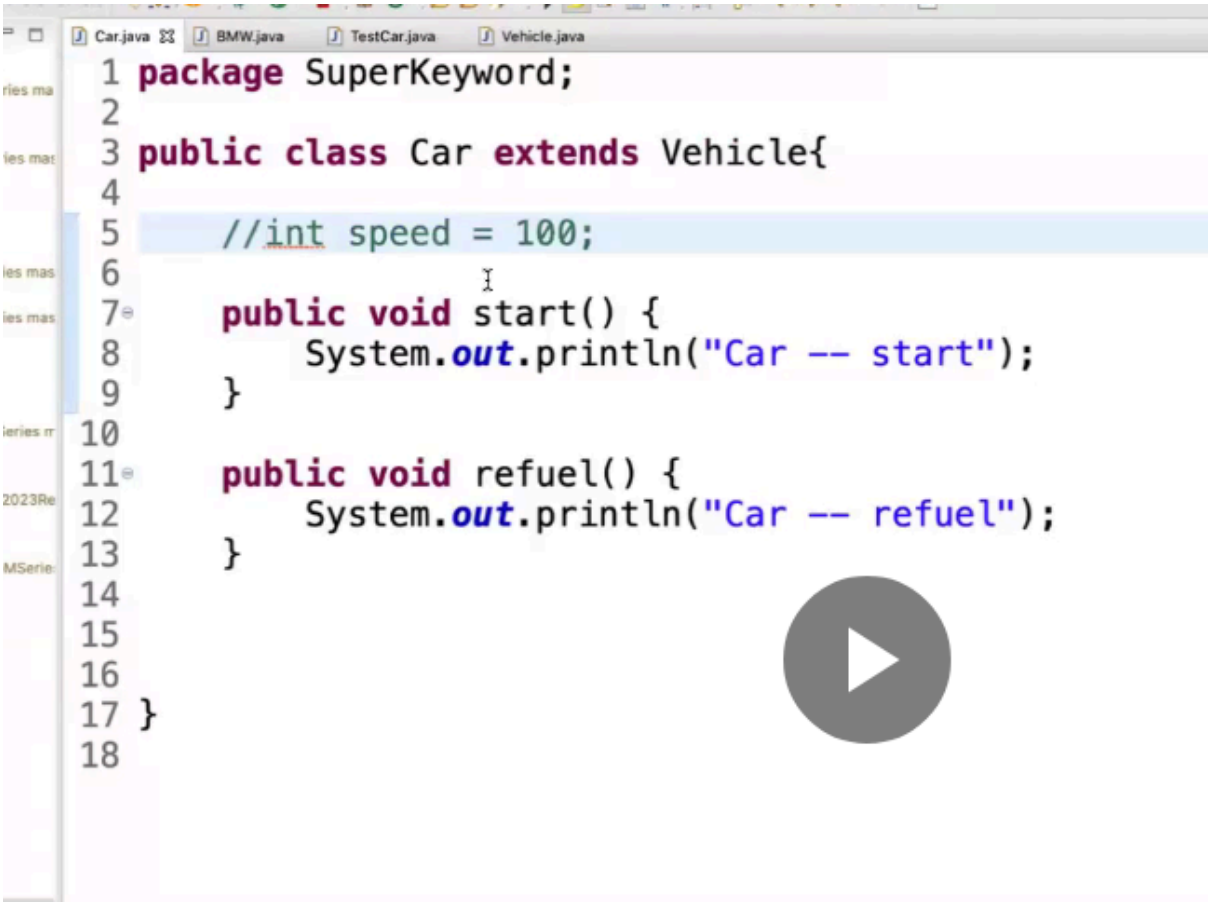
300

100

100

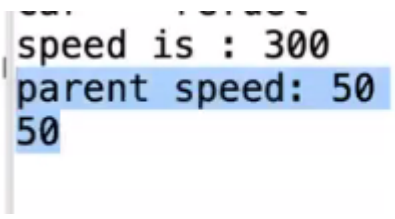
Super for accessing immediate parent class objects only not grandparents.

Car-



Run test.

Parent is not having the variable but grand parent has.



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

paste vehicle8, car8, bmw8, testfor8-

vehicle8.java

```java
package com.day21;

public class vehicle8 {

    int speed=10;

    public String vehiclestatus(int a) {
        System.out.println("vehicle status from vehice class");
        return "a";
    }

}

```

car8.java

```java
package com.day21;

public class car8 extends vehicle8 {

    //no speed method here.

    public void start() {
        System.out.println("car start method");
    }

    public void refuel() {
        System.out.println("Car refuel method");
    }

}

```

test8.java bmw8.java

```java
package com.day21;

public class bmw8  extends car8{

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

    //own speed method.
    int speed=123324;

    //override start method.
    @Override
    public void start() {
        super.start();
        System.out.println("bmw start method");
    }

    //add autoparking method.
    public void autoparking() {
        System.out.println("bmw auto parking");
    }
```

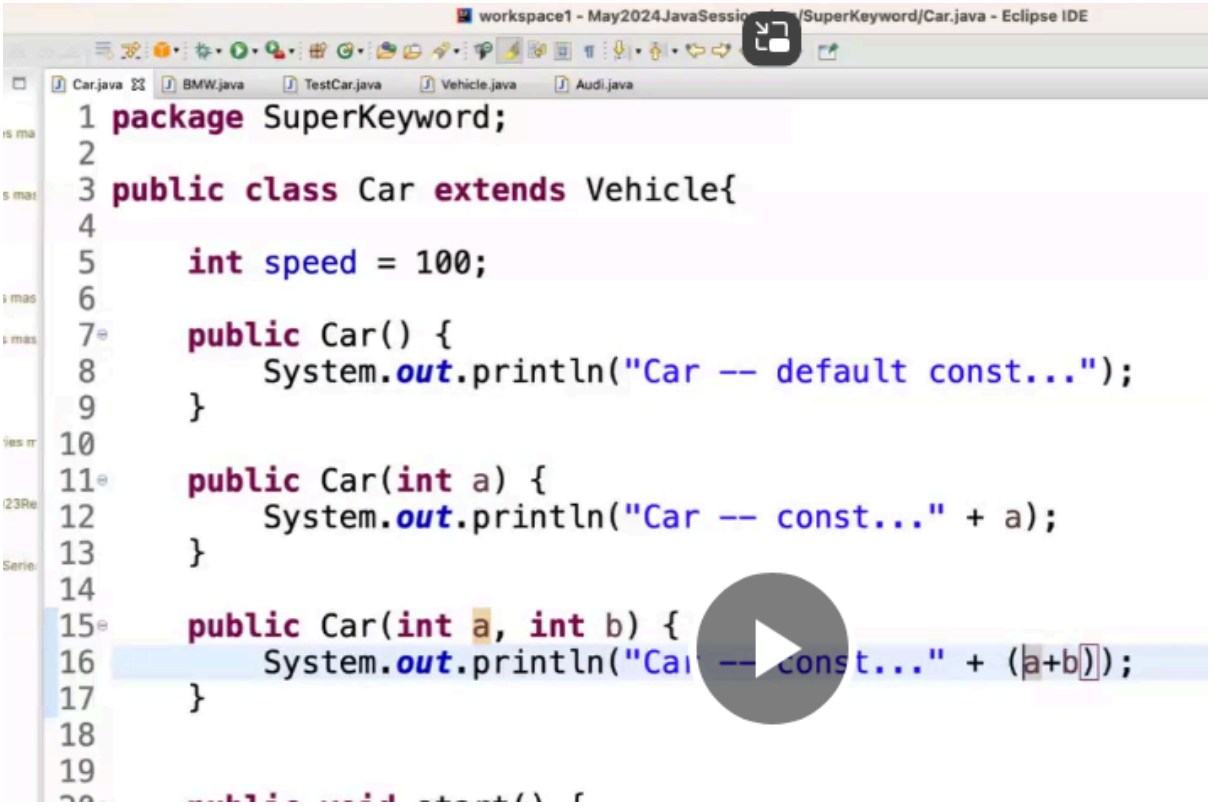
```java
    public void autoparking() {
        System.out.println("bmw auto parking");
    }

    public void displayspeed() {
        System.out.println("speed of bmw is "+ this.speed);
        System.out.println("speed of bmw is "+ speed);
        //call parent class speed
//      System.out.println("speed of parent is " +super.this.speed);
        //Syntax error on token "this", Identifier expected
        int i1=super.speed;
        System.out.println("speed of parent is " +i1);

        //try accessing grand parent speed.
        //Syntax error on token "super", Identifier expected
//      super.super.speed;
        String s1=super.vehiclestatus(10);
        System.out.println(s1);
    }

}
```

test8.java

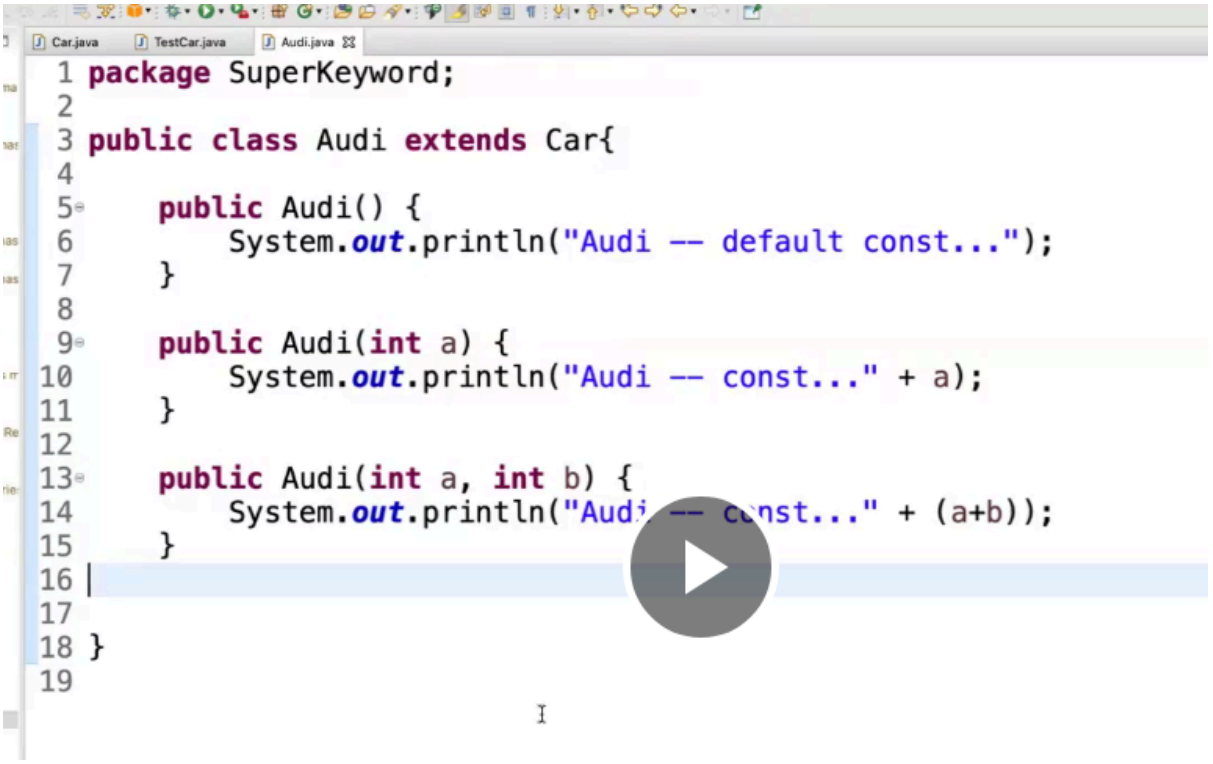
```java
package com.day21;

public class test8 {

    public static void main(String[] args) {

        bmw8 b1=new bmw8();
        int i1=b1.speed;
        System.out.println(i1);
        b1.displayspeed();

    }

}

//123324
//speed of bmw is 123324
//speed of bmw is 123324
//speed of parent is 10
//vehicle status from vehice class
//a
```

Car class –

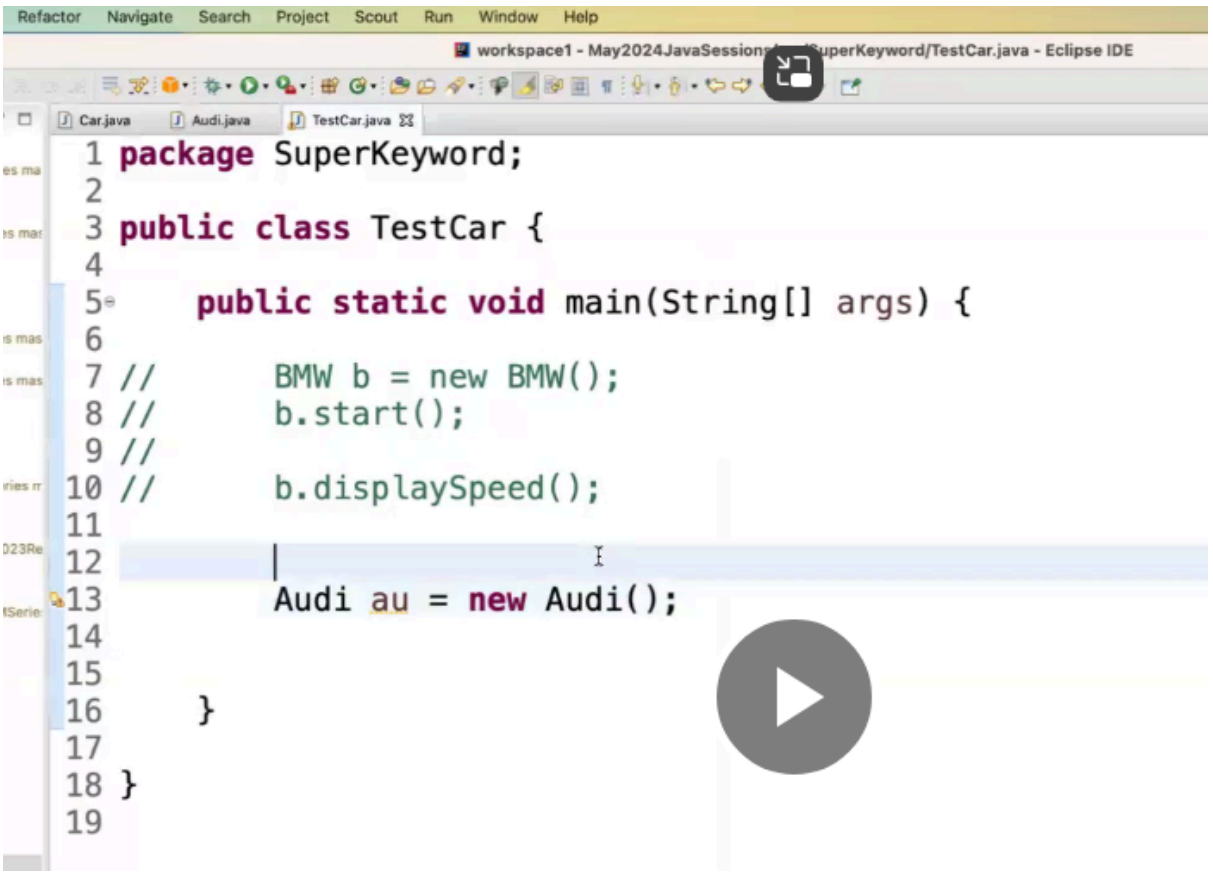
With more constructor.


workspace1 - May2024JavaSessio /SuperKeyword/Car.java - Eclipse IDE
Car.java
BMW.java
TestCar.java
Vehicle.java
Audi.java
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

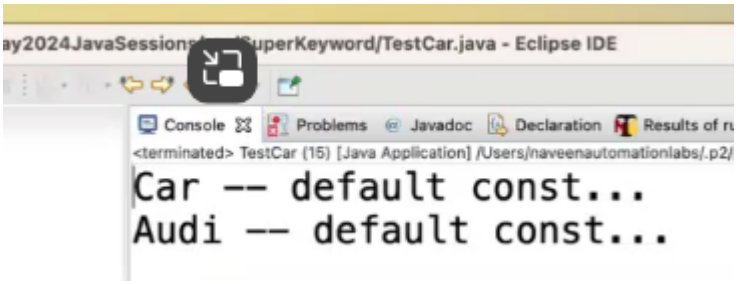
Audi-


Car.java
TestCar.java
Audi.java
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 }
19

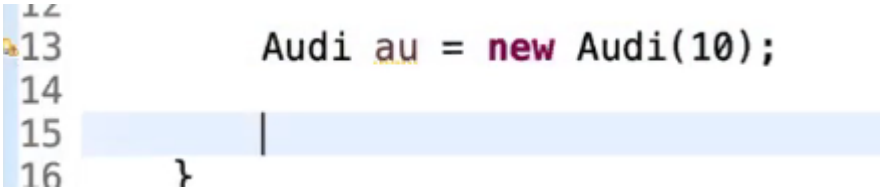
Test-

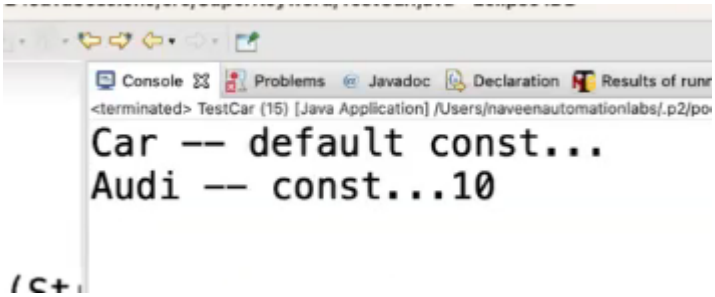


Default constructor of parent called, then child class relevant constructor.

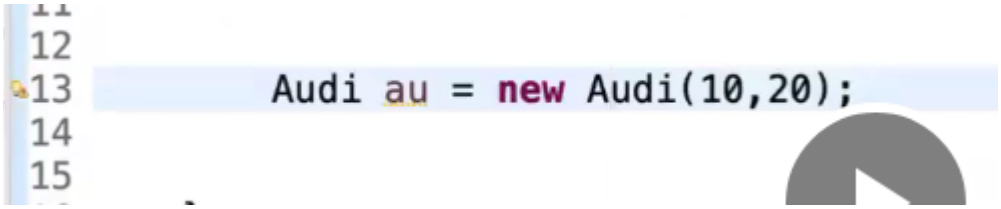


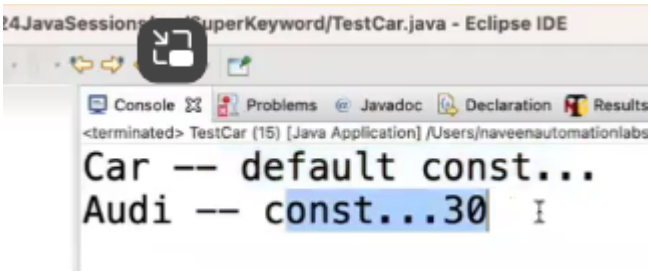
Test-



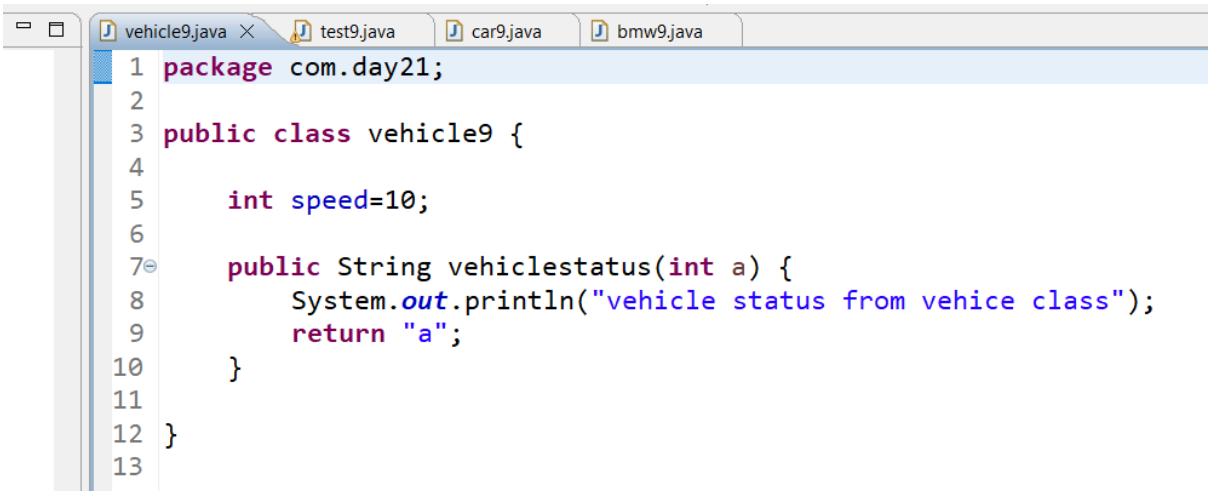


Test-





paste vehicle9, bmw9, car9, testfor9-



test9.java | car9.java | bmw9.java

```java
package com.day21;

public class car9 extends vehicle9 {

    int speed=100;

    //default constructor.
    public car9() {
        System.out.println("car default constructor");
    }

    //one parameter constructor.
    public car9(int speed) {
        this.speed=speed;
        System.out.println(this.speed);
        System.out.println("car one parameter constructor");
    }

    //two parameter constructor.
    public car9(int speed,String color) {
        this.speed=speed;
        System.out.println(this.speed);
//      System.out.println(this.color);//color cannot be resolved or is not a field
        System.out.println(color);
        System.out.println("car two parameter constructor");
    }
```

```java
        System.out.println("car two parameter constructor");
    }

    public void start() {
        System.out.println("car start method");
    }

    public void refuel() {
        System.out.println("Car refuel method");
    }

}
```

test9.java bmw9.java

```java
package com.day21;

public class bmw9  extends car9{

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

    //own speed method.
    int speed=123324;

    //default constructor.
    public bmw9() {
        System.out.println("bmw default constructor");
    }

    //one parameter constructor.
        public bmw9(int speed) {
            this.speed=speed;
            System.out.println(this.speed);
            System.out.println("bmw one parameter constructor");
        }

        //two parameter constructor.
        public bmw9(int speed,String color) {
            this.speed=speed;
            System.out.println(this.speed);
//          System.out.println(this.color);//color cannot be resolved or is not a field
            System.out.println(color);
            System.out.println("bmw two parameter constructor");
        }
```

```java
             System.out.println("bmw two parameter constructor");
         }

    //override start method.
    @Override
    public void start() {
        super.start();
        System.out.println("bmw start method");
    }

    //add autoparking method.
    public void autoparking() {
        System.out.println("bmw auto parking");
    }

    public void displayspeed() {
        System.out.println("speed of bmw is "+ this.speed);
        System.out.println("speed of bmw is "+ speed);
        //call parent class speed
//      System.out.println("speed of parent is " +super.this.speed);
        //Syntax error on token "this", Identifier expected
        int i1=super.speed;
        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
```

test9.java

```
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
18 //car default constructor
19 //bmw default constructor
20 //car default constructor
21 //10
22 //bmw one parameter constructor
23 //car default constructor
24 //10
25 //blue
26 //bmw two parameter constructor
27
28
29
30
```

Call the super class constructor-

Audi-

```
    public Audi(int a) {
        super(100);
        System.out.println("Audi -- const..." + a);
    }
```

Test-

```
        Audi au = new Audi(10);
```

```
JavaSessions/SuperKeyword/TestCar.java - Eclipse ID
Console  Problems  Javadoc  Declaration
<terminated> TestCar (15) [Java Application] /Users/naveenau
Car -- const...100
Audi -- const...10
```

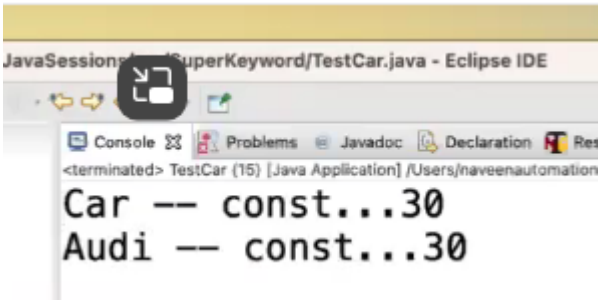
Now default wont be called, as we specified which constructor to call using super keyword.

Audi-

```
    public Audi(int a, int b) {
        super(10, 20);
        System.out.println("Audi -- const..." + (a+b));
    }
```

Test-

```
        Audi au = new Audi(10,20);
```

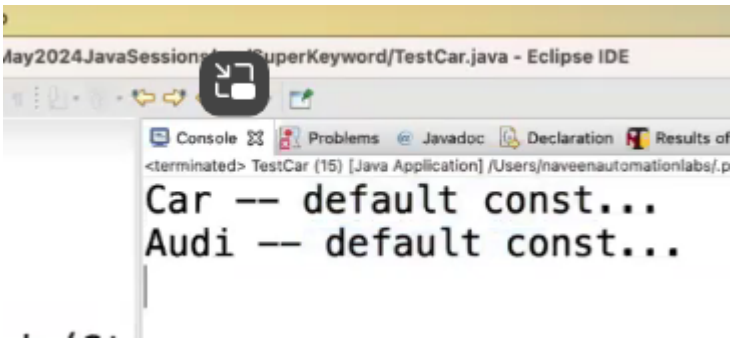


Audi-

```
4
5    public Audi() {
6        super();
7        System.out.println("Audi -- default const...");
8    }
```

Test-

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



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

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

```
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

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

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

We can pass parameter also inside super.-

Test-

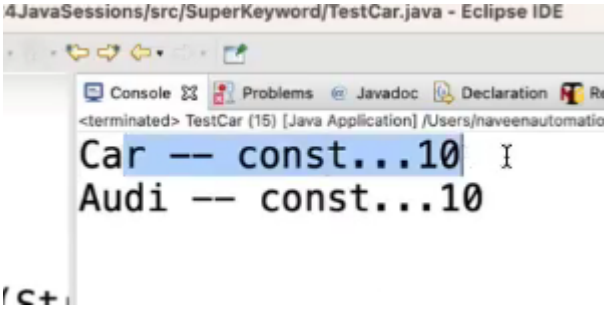
```
11
12
13       Audi au = new Audi(10);
14
15
```

Open Declaration
Open Implementation

Audi-

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

Run test-



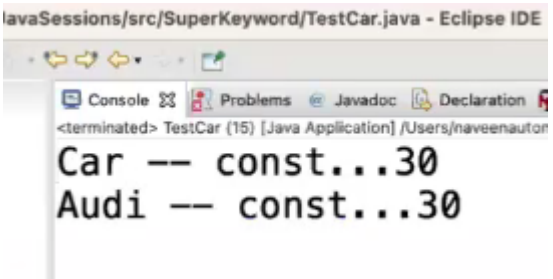
Audi-

```
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
```

Open Declaration
Open Implementation



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    }
```

```
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

```java
package com.day21;

public class vehicle13 {

    int speed=10;

    public String vehiclestatus(int a) {
        System.out.println("vehicle status from vehice class");
        return "a";
    }

}

```

car13.java | test13.java | bmw13.java

```java
package com.day21;

public class car13 extends vehicle13 {

    int speed=100;
    static int duration=100000;

    //default constructor.
    public car13() {
        System.out.println("car default constructor");
    }

    //one parameter constructor.
    public car13(int speed) {
        this.speed=speed;
        System.out.println(this.speed);
        System.out.println("car one parameter constructor");
    }

    //two parameter constructor.
    public car13(int speed,String color) {
        this.speed=speed;
        System.out.println(this.speed);
//      System.out.println(this.color);//color cannot be resolved or is not a field
        System.out.println(color);
        System.out.println("car two parameter constructor");
    }
```

```
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
```

test13.java bmw13.java

```
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("----------");
```

```
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("----------");
```

```
46              System.out.println("bmw two parameter constructor");
47              System.out.println("----------");
48
49              //can access static methods also this way.
50              //this is better way to access static using class name.
51              String s2=car13.testing(20);
52              System.out.println(s2);
53
54              //access static variables.
55              //best way using class name.
56              int i1=car13.duration;
57              System.out.println(i1);
58          }
59
60      //override start method.
61      @Override
62      public void start() {
63          super.start();
64          System.out.println("bmw start method");
65      }
```

```
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
```

test13.java

```
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

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

Error.

Method cannot call constructor.

Constructor calls another constructor.

```
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
```

constructor can call method-

paste bmw15, test15-

bmw15.java test15.java

```java
package com.day21;

public class bmw15  extends car13{

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

    //own speed method.
    int speed=123324;

    //default constructor.
    public bmw15() {
        super(100);
        System.out.println("bmw default constructor");
        //Super for variables and methods can be anywhere.
        System.out.println(super.speed);

        //can construtor call method.
        //yes allowed.
        refuel();
    }
```

```java
        refuel();
    }

    //one parameter constructor.
        public bmw15(int speed) {
            super(speed); //can pass parameter also to parent.
            //not necessary to hard code everytime.
            this.speed=speed;
            System.out.println(this.speed);
            System.out.println("bmw one parameter constructor");
            //Super for variables and methods can be anywhere.
            super.start();
            System.out.println("----------");

            //can access static methods also this way.
            String s1=super.testing(10);
            System.out.println(s1);

            //access static variable.
            int i1=super.duration;
            System.out.println(i1);
        }
```

```
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          }
```

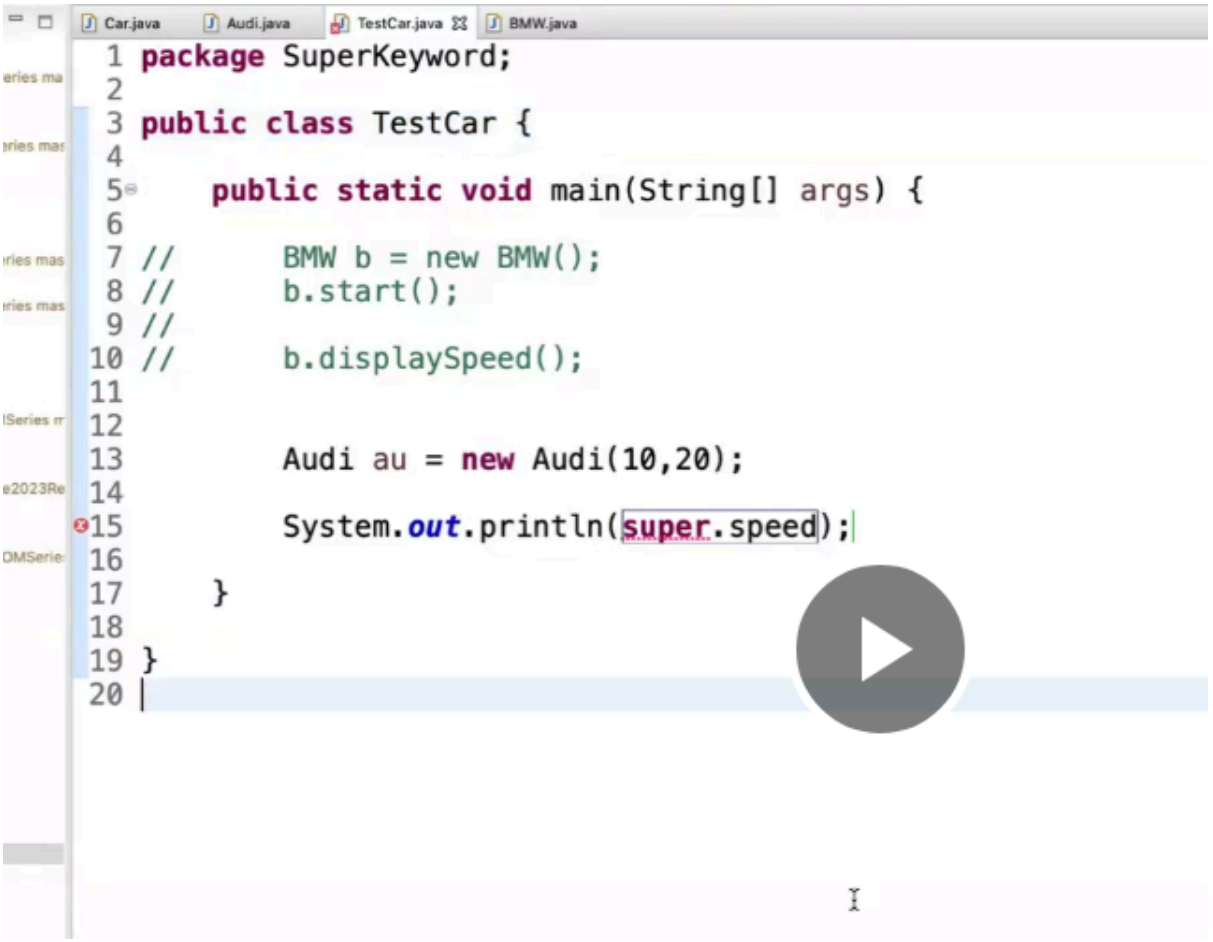
```
61              System.out.println(i1);
62          }
63
64      //override start method.
65      @Override
66      public void start() {
67          super.start();
68          System.out.println("bmw start method");
69      }
70
71      //add autoparking method.
72      public void autoparking() {
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);
```

```
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
```

test15.java

```
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
```

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

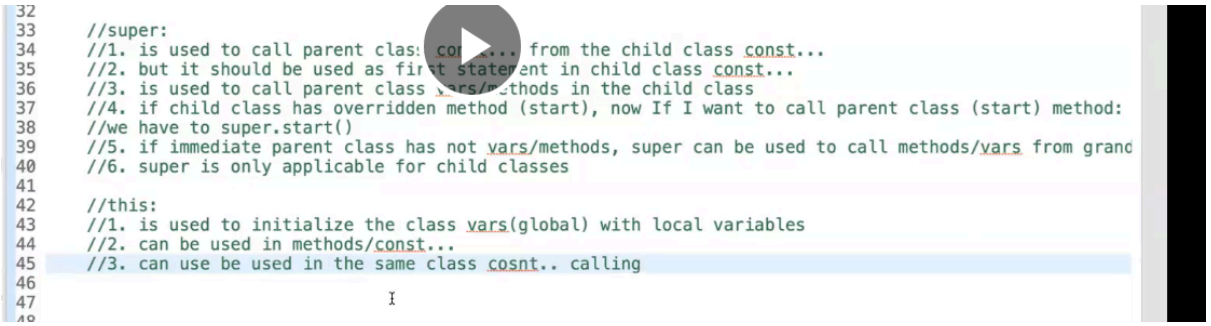


//super cannot be used for class which does not extend anything.
// we dont get options after dot.

paste test16-

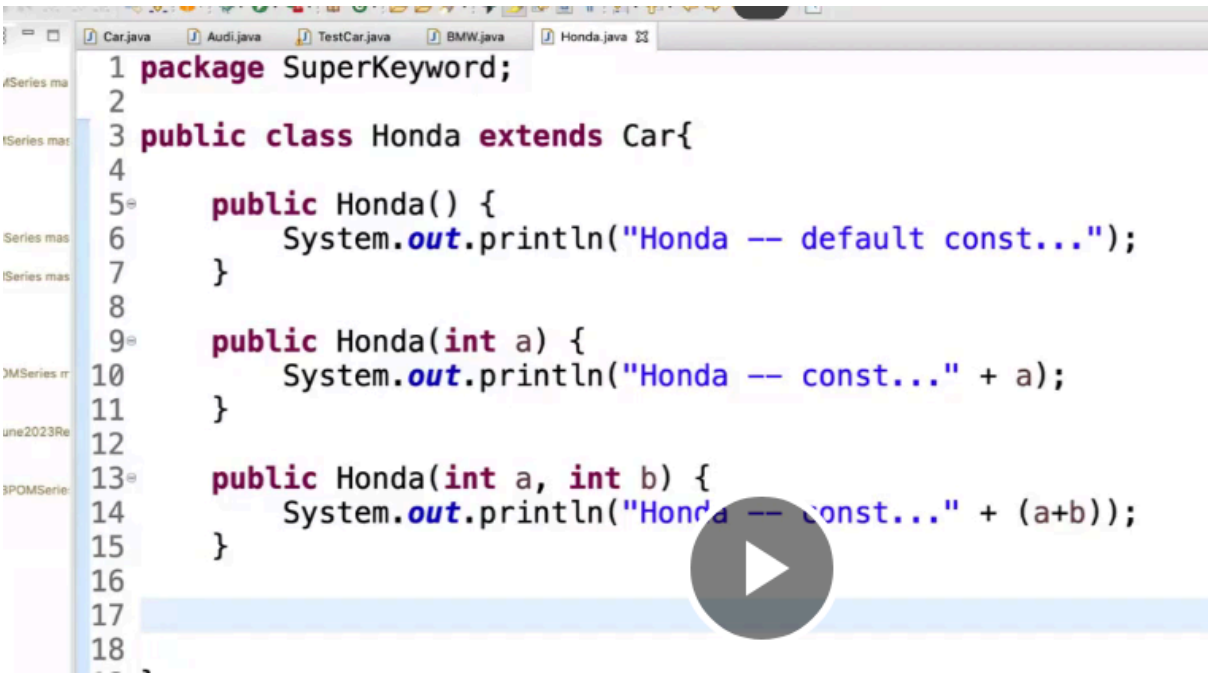
test16.java

```java
package com.day21;

public class test16 {

    public static void main(String[] args) {

        bmw15 b1=new bmw15();

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


    }

}



//100
//car one parameter constructor
//bmw default constructor
//100
//Car refuel method
```

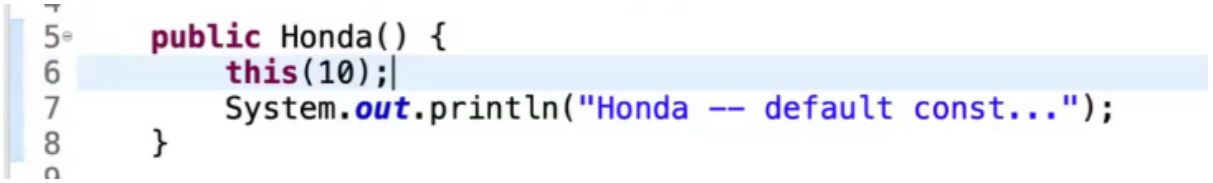


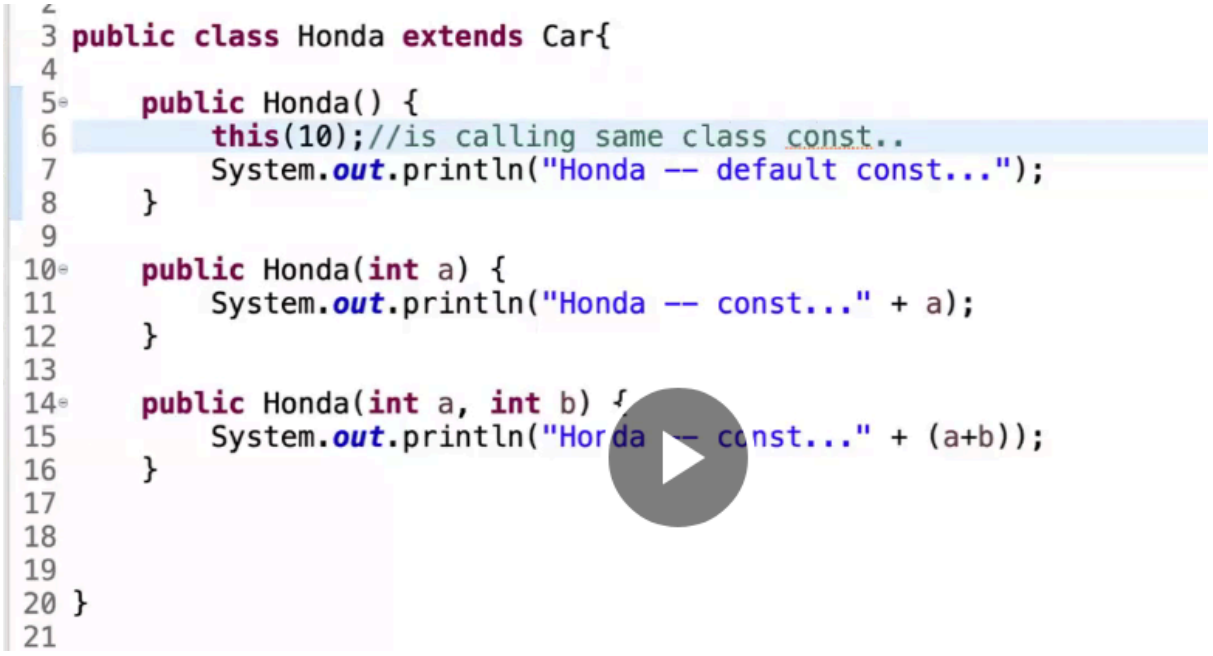
```java
//super:
//1. is used to call parent clas: co... from the child class const...
//2. but it should be used as fir t statement in child class const...
//3. is used to call parent class vars/methods in the child class
//4. if child class has overridden method (start), now If I want to call parent class (start) method:
//we have to super.start()
//5. if immediate parent class has not vars/methods, super can be used to call methods/vars from granc
//6. super is only applicable for child classes

//this:
//1. is used to initialize the class vars(global) with local variables
//2. can be used in methods/const...
//3. can use be used in the same class cosnt.. calling
```

Honda-

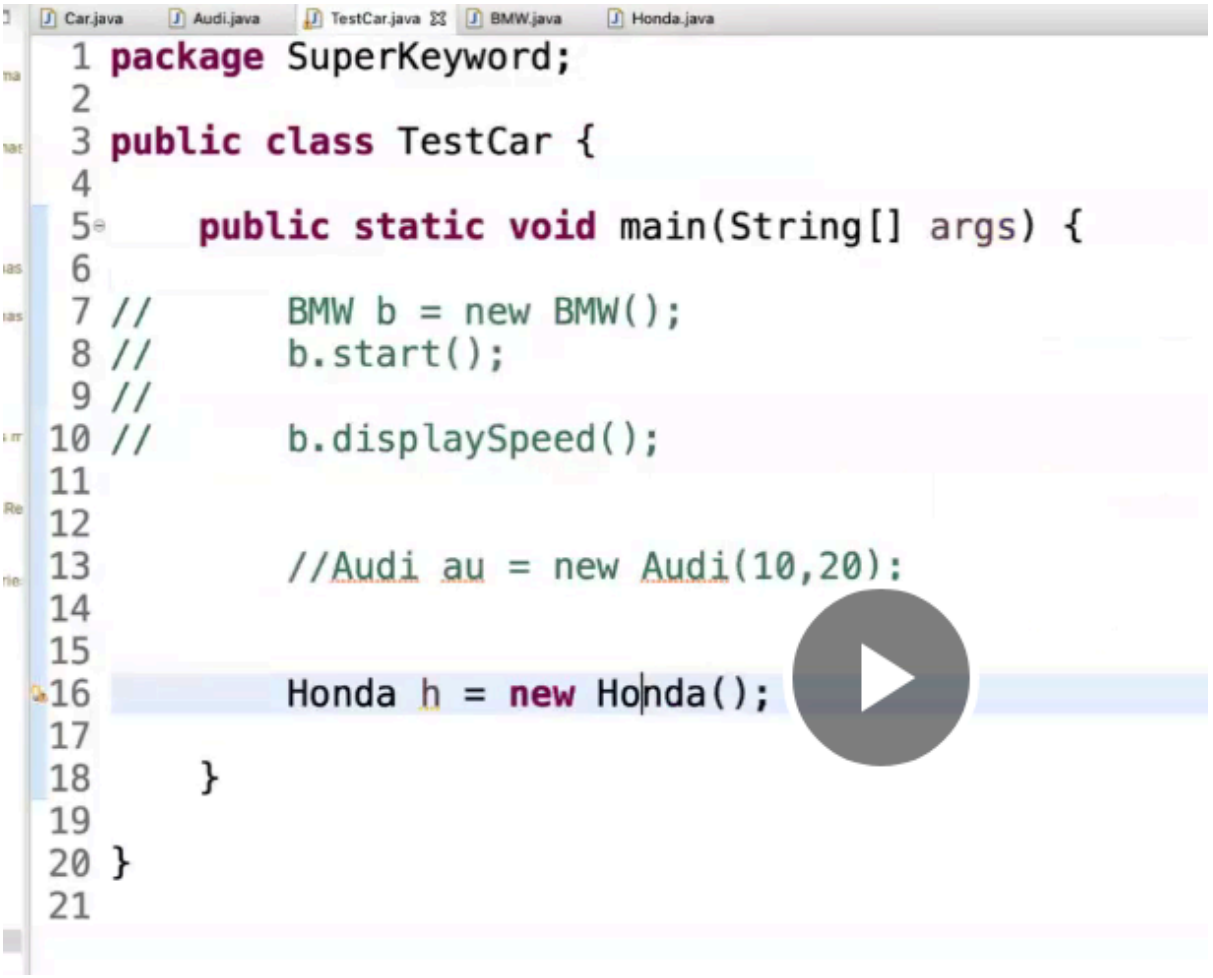

Car.java
Audi.java
TestCar.java
BMW.java
Honda.java
1 package SuperKeyword;
2
3 public class Honda extends Car{
4
5 public Honda() {
6 System.out.println("Honda -- default const...");
7 }
8
9 public Honda(int a) {
10 System.out.println("Honda -- const..." + a);
11 }
12
13 public Honda(int a, int b) {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

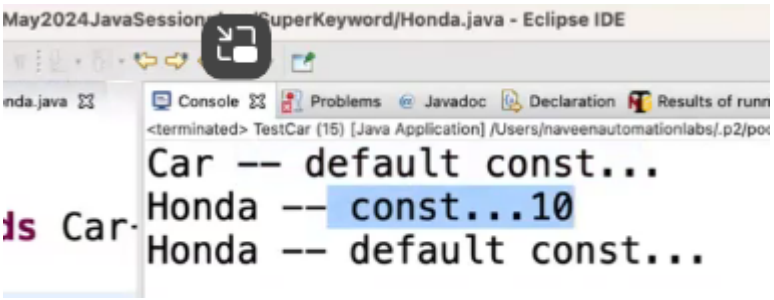
Honda-


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-





paste car18, honda18, test18-

car18.java | honda18.java | test18.java

```java
package com.day21;

public class car18 extends vehicle13 {

    int speed=100;
    static int duration=100000;

    //one parameter constructor.
    public car18(int speed) {
        this.speed=speed;
        System.out.println(this.speed);
        System.out.println("car one parameter constructor");
    }

    //two parameter constructor.
    public car18(int speed,String color) {
        this.speed=speed;
        System.out.println(this.speed);
//      System.out.println(this.color);//color cannot be resolved or is not a field
        System.out.println(color);
        System.out.println("car two parameter constructor");
    }
```

```java
        System.out.println("car two parameter constructor");
    }

    public void start() {
        System.out.println("car start method");
    }

    public void refuel() {
        System.out.println("Car refuel method");
    }

    //static method in car class.
    public static String testing(int a) {
        System.out.println("static method in car class");
        return "a";
    }

}
```

test18.java honda18.java

```java
package com.day21;

public class honda18 extends car18{

    public honda18() {
        //to call same class constructor.
        this(10);
           System.out.println("Honda -- default const...");
        }

        public honda18(int a) {
        this(1,2);
            System.out.println("Honda -- const..." + a);
        }

        //Implicit super constructor car18() is undefined. Must explicitly invoke another constructor
        //we have to call another constructor explicitly.
//      public honda18(int a, int b) {
//          System.out.println("Honda -- const..." + (a + b));
//      }

        public honda18(int a, int b) {
        super(a);
        System.out.println("Honda -- const..." + (a + b));
        }

}

```

test18.java

```java
package com.day21;

public class test18 {

    public static void main(String[] args) {

        //parent class default constructor if present will always be called.
        //when we dont use super.

        honda18 h1=new honda18();

    }

}

//1
//car one parameter constructor
//Honda -- const...3
//Honda -- const...10
//Honda -- default const...


```

“This” also cant be second sentence-

Constructor call has to happen at first line.

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

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

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    }
```

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-

```
14
15
16        Honda h = new Honda(10);
17
18    }
19
```

```
JavaSession.../SuperKeyword/TestCar.java - Eclipse IDE
Console  Problems  Javadoc  Declaration
<terminated> TestCar (15) [Java Application] /Users/naveenautomatio
Car -- default const...
Honda -- const...30
Honda -- const...10
```

Recursive constructor not allowed-

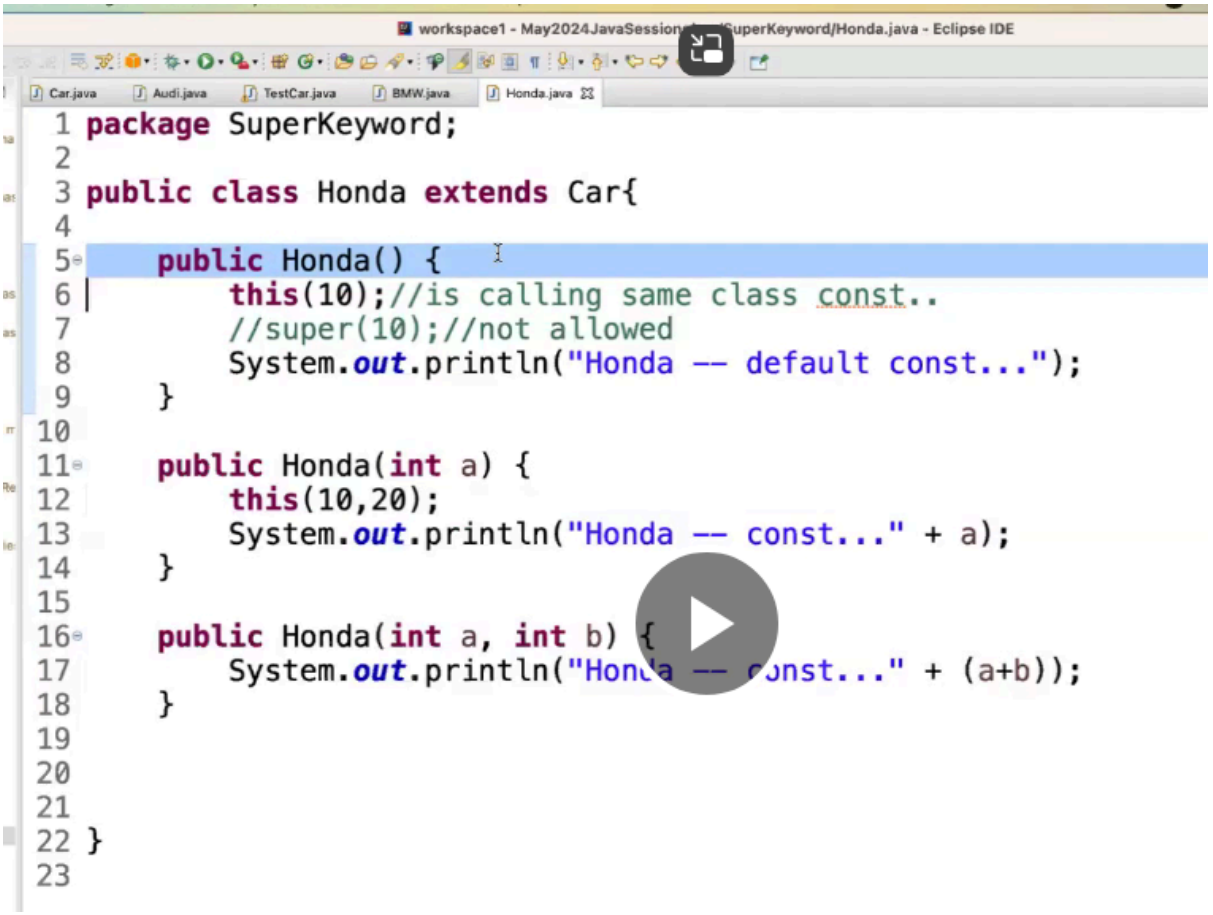
Recursion only for methods not for constructors.

Honda-

//Recursive constructor invocation honda21(int)

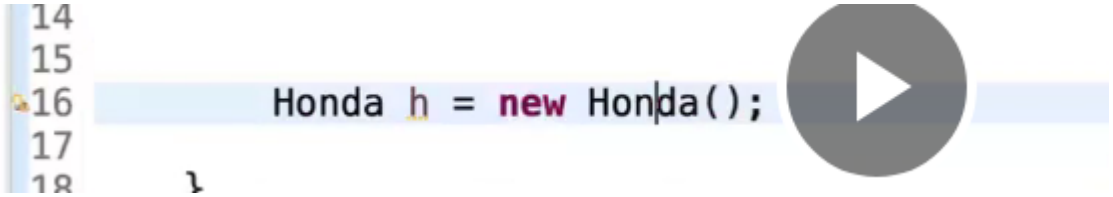
```
10
11    public Honda(int a) {
12        this(10);
13        System.out.println("Honda -- const..." + a);
14    }
```

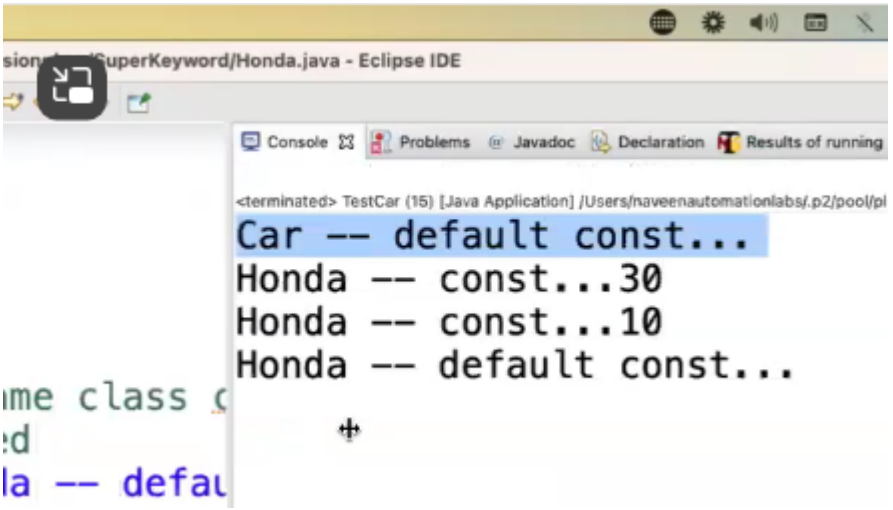
honda class-

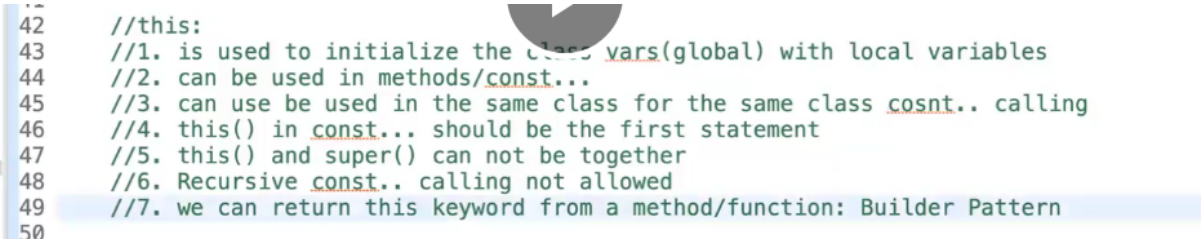


see how data flows from one const to another const ….

Test –



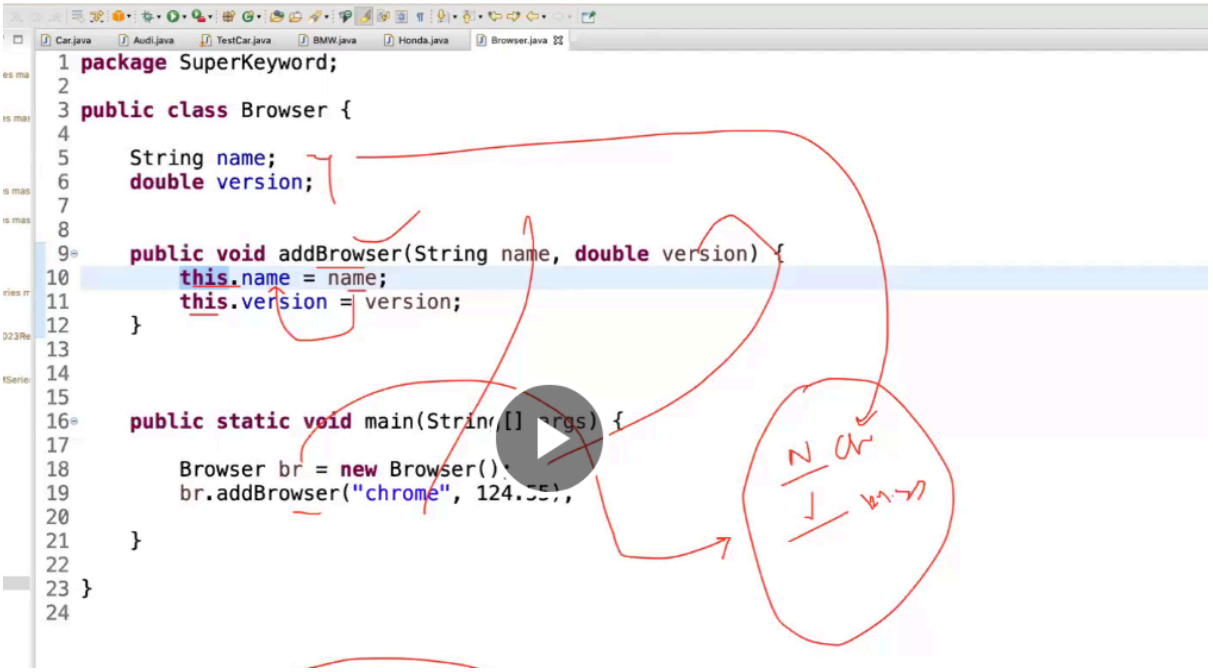




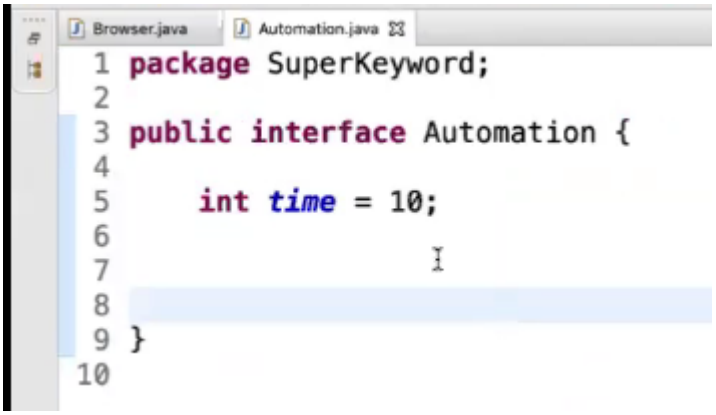
“This” keyword can be used inside getter, setter, constructor, method to initialise.

Same for the super keyword – can be used.

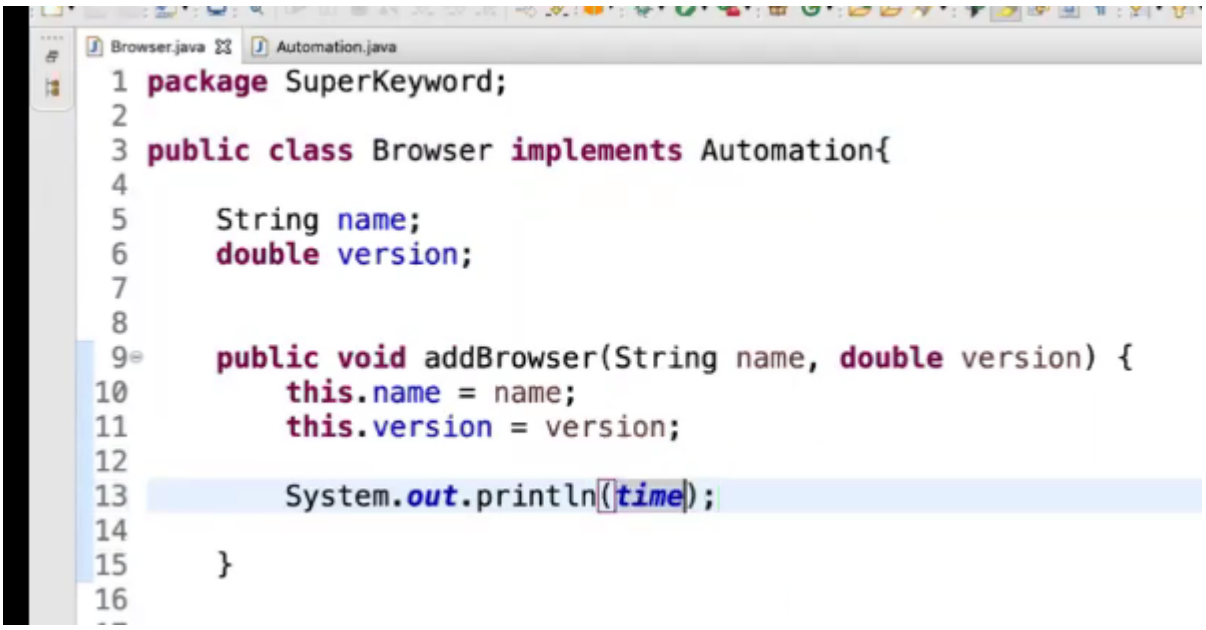
“This” cannot be used inside static methods. “This” refers to the current class object. “This” deals with objects only.



Interface-

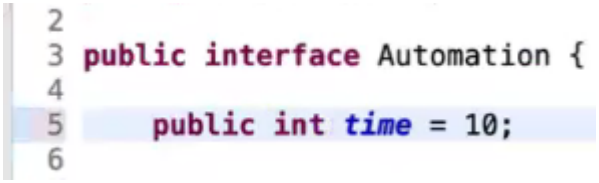


Browser-



10

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



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
```

time cannot be resolved or is not a field

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

Interface variables are static final by default.

Use class names-

```
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   }
16
```

paste browser2, automation1-

automation1.java | browser2.java

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

browser2.java

```java
package com.day21;

public class browser2 implements automation1{

    String name;
    double version;

    public void addBrowser(String name, double version) {
        this.name = name;
        this.version = version;
        System.out.println(this.name);
        System.out.println(this.version);
    }

    public static void main(String[] args) {
    browser2 br = new browser2();
        br.addBrowser("chrome", 124.56);

        int i=time;
        System.out.println(i);

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

        //interface variables can be directly accessed using interface name.
        //they are public static.
        int i2=automation1.time;
        System.out.println(i2);
```

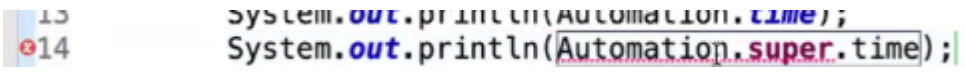
```java
        int i2=automation1.time;
        System.out.println(i2);

        //can we try changing interface variable value.
        //can access interface variable using child class name also.
        int i3=browser2.time;
        System.out.println(i3);

        //interface variables are also final.
        //cannot change.
        //The final field automation1.time cannot be assigned
        automation1.time=100;
        //The final field automation1.time cannot be assigned
        browser2.time=34554;

        //No enclosing instance of the type automation1 is accessible in scope
        //not allowed.
        System.out.println(automation1.super.time);
```

```
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-

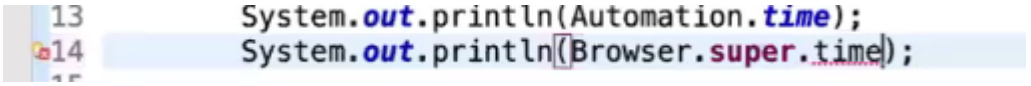
//No enclosing instance of the type automation1 is accessible in scope



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



Added default method in interface-

Browser.java *Automation.java

```java
package SuperKeyword;

public interface Automation {

    public int time = 10;


    default void billing() {
        System.out.println("automation billing");
    }


```

Browser-

```java

    public void addBrowser(String name, double version) {
        this.name = name;
        this.version = version;

        System.out.println(Automation.time);

        billing();

```

paste automation2, browser3-

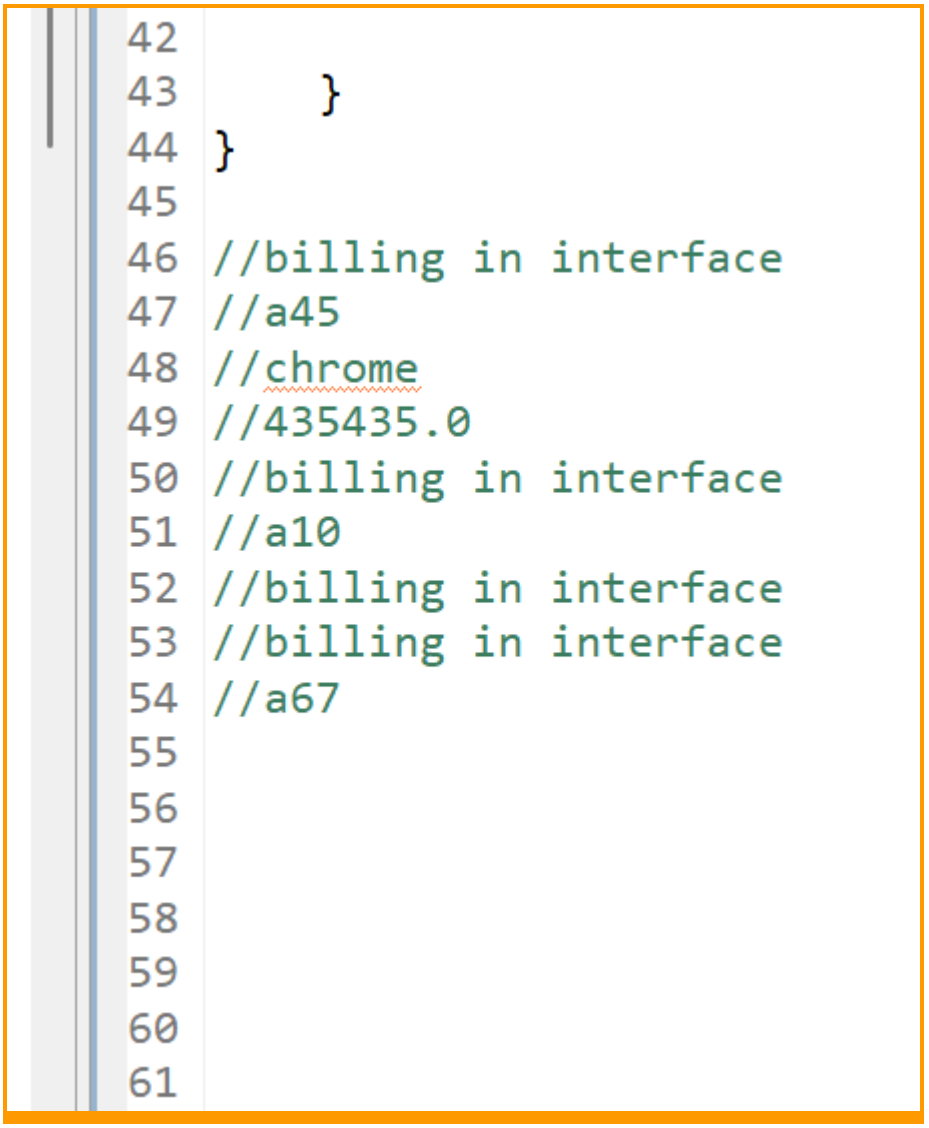
browser3.java automation2.java

```java
package com.day21;

public interface automation2 {

    int time=10;

    default String billing(int a) {
        System.out.println("billing in interface");
        return "a"+a; //i learnt another way to return one string.
    }

}

```

browser3.java

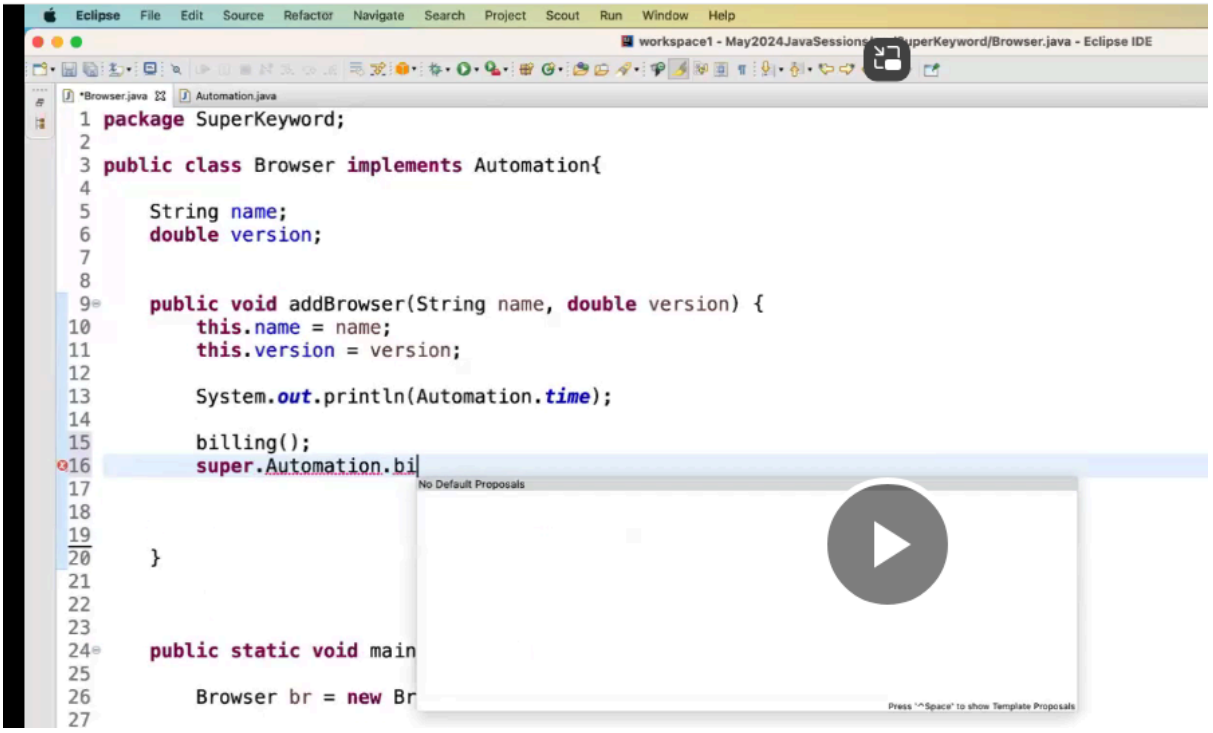
```java
package com.day21;

public class browser3 implements automation2{

    String name;
    double version;

    public void addBrowser(String name, double version) {
        this.name = name;
        this.version = version;
        System.out.println(this.name);
        System.out.println(this.version);

        String s1=  billing(10);
        System.out.println(s1);

//      super.//no options after super to call parent.

        automation2.super.billing(25);

        String s11=automation2.super.billing(67);
        System.out.println(s11);


        //No enclosing instance of the type automation2 is accessible in scope
//      int i1=automation2.super.time;
    }
```

```java
//      int i1=automation2.super.time;
    }

    public static void main(String[] args) {

        browser3 b1=new browser3();
        String s2=b1.billing(45);
        System.out.println(s2);
//      super.//no options after super.
        b1.addBrowser("chrome", 435435);

        //Cannot use super in a static context
//      automation2.super.billing(10);

        //No enclosing instance of the type automation2 is accessible in scope
//      int i1=automation2.super.time;

    }
```

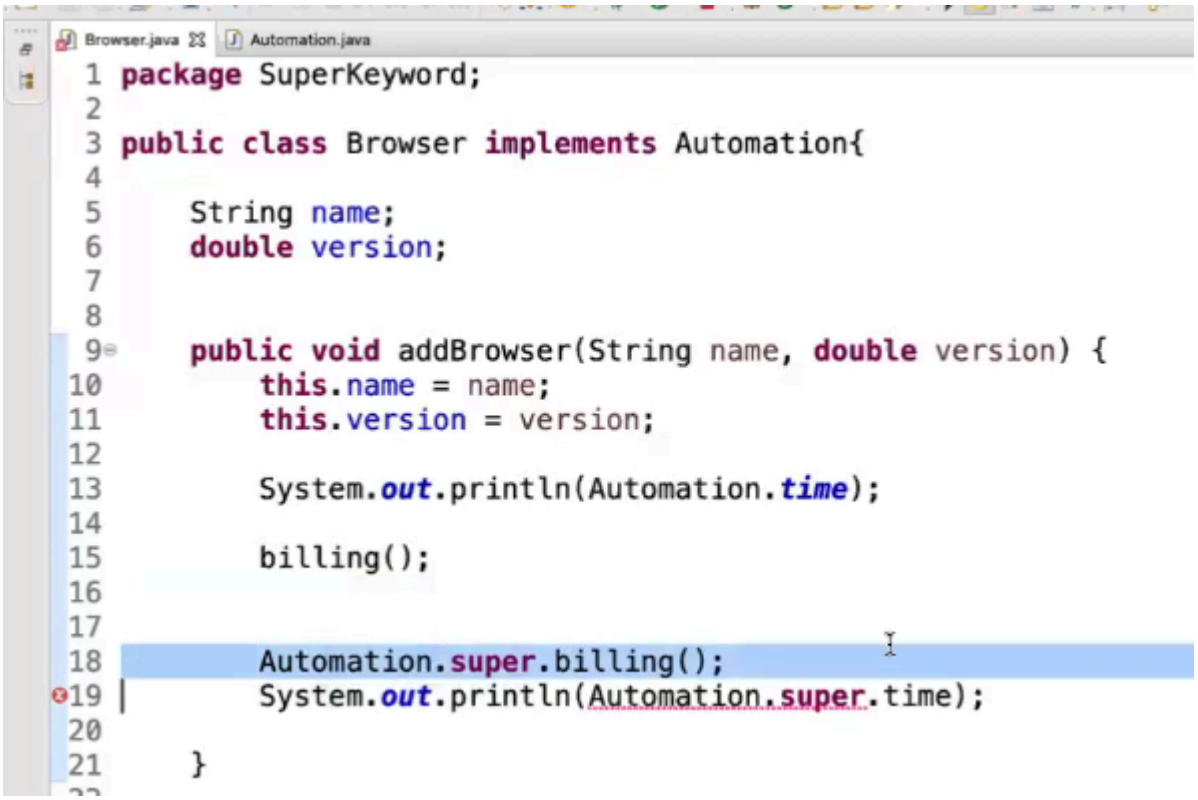


```
42
43      }
44  }
45
46  //billing in interface
47  //a45
48  //chrome
49  //435435.0
50  //billing in interface
51  //a10
52  //billing in interface
53  //billing in interface
54  //a67
55
56
57
58
59
60
61
```

Cannot use super to access the method of interface.-



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



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-

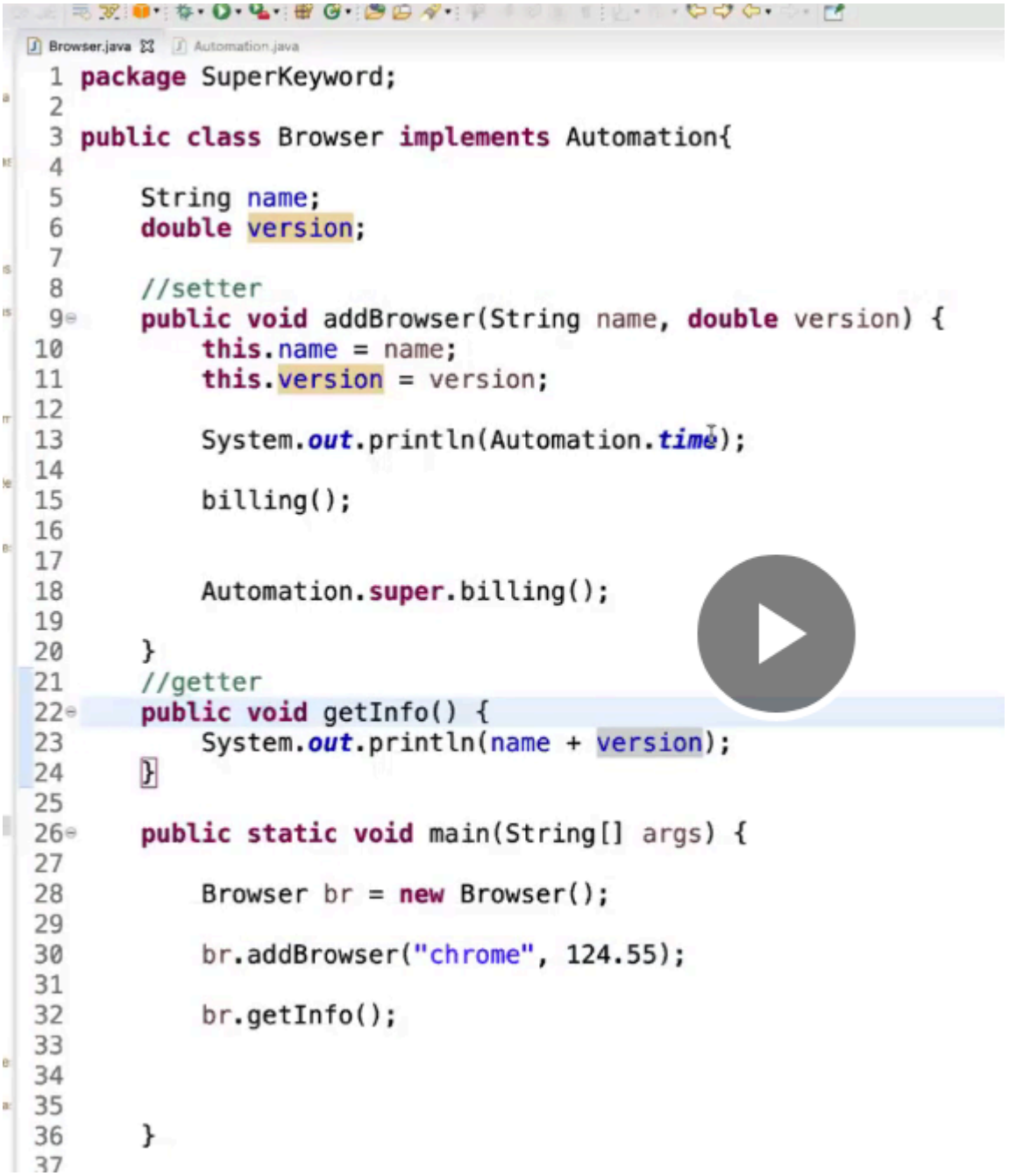
automation3.java

```java
package com.day21;

public interface automation3 {

    int time=10;

     static String billing(int a) {
        System.out.println("billing in interface");
        return "a"+a; //i learnt another way to return one string.
    }

}

```

browser4.java

```java
package com.day21;

public class browser4 implements automation3{

    String name;
    double version;

    public void addBrowser(String name, double version) {
        this.name = name;
        this.version = version;
        System.out.println(this.name);
        System.out.println(this.version);

        //This static method of interface automation3 can only be accessed as automation3.billing
//      automation3.super.billing(25);
        automation3.billing(23432432);

        //This static method of interface automation3 can only be accessed as automation3.billing
//      String s11=automation3.super.billing(67);
        String s11=automation3.billing(67);
        System.out.println(s11);

        //No enclosing instance of the type automation3 is accessible in scope
//      int i1=automation3.super.time;
        int time2 = automation3.time;
        System.out.println("time from interface " + time2);
    }
```

```
    }

    public static void main(String[] args) {

        browser4 b1=new browser4();
        b1.addBrowser("chrome", 435435);
    }
}

//chrome
//435435.0
//billing in interface
//billing in interface
//a67
//time from interface 10
```



Browser.java | Automation.java

```
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