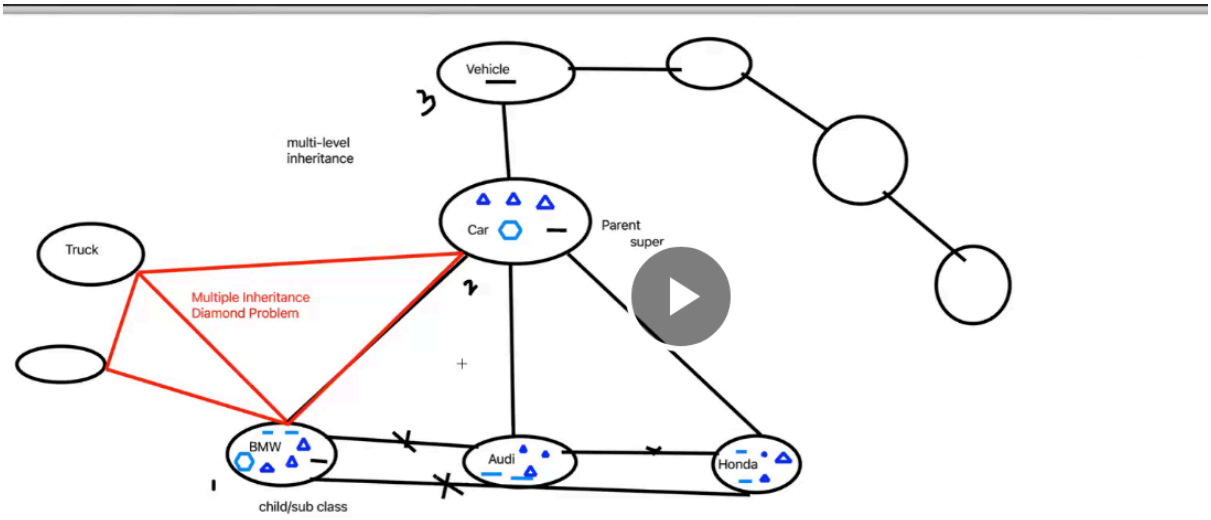


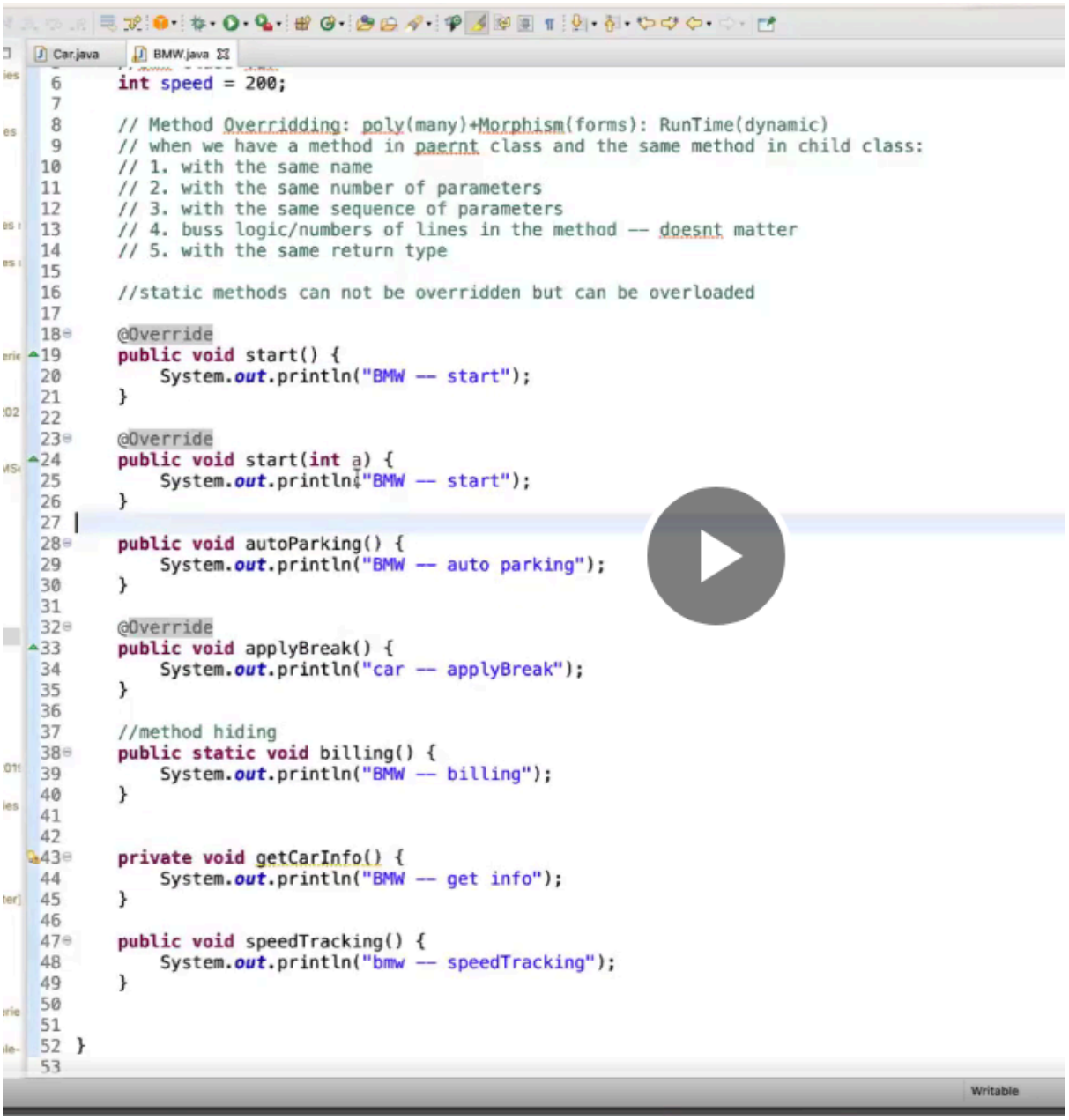

```
6   int speed = 200;
7
8   // Method Overridding: poly(many)+Morphism(forms): RunTime(dynamic)
9   // when we have a method in paernt class and the same method in child class:
10  // 1. with the same name
11  // 2. with the same number of parameters
12  // 3. with the same sequence of parameters
13  // 4. buss logic/numbers of lines in the method -- doesnt matter
14  // 5. with the same return type
15
16  //static methods can not be overridden but can be overloaded
17
18  @Override
```

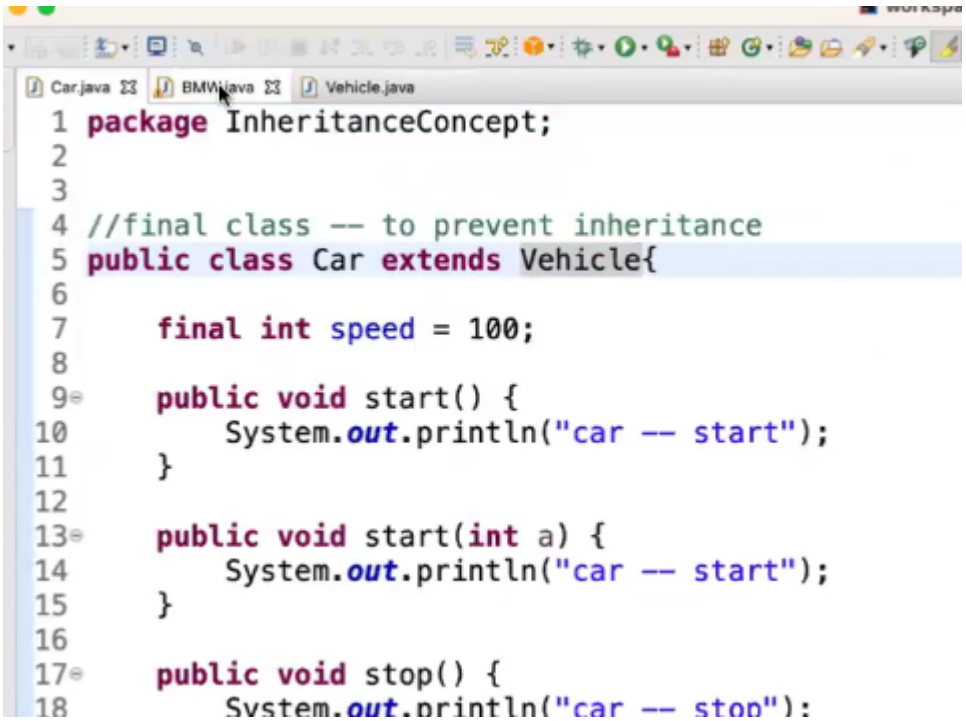
Final, static, private methods cannot be overridden.

for static, private method the error is –//The method billing() of type bmw1 must override or implement a supertype method

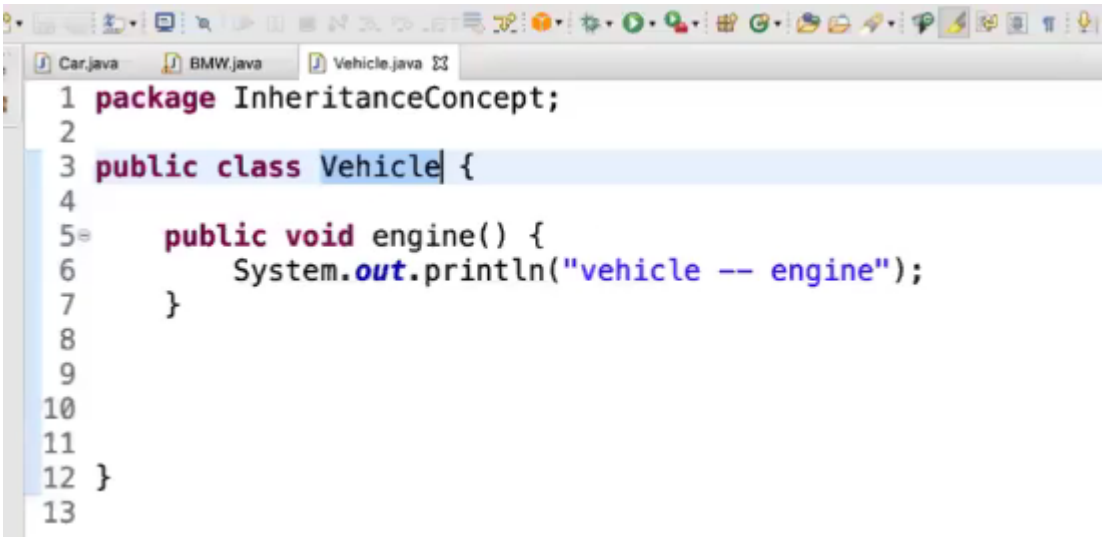
for final method the error is - Cannot override the final method from car1

2

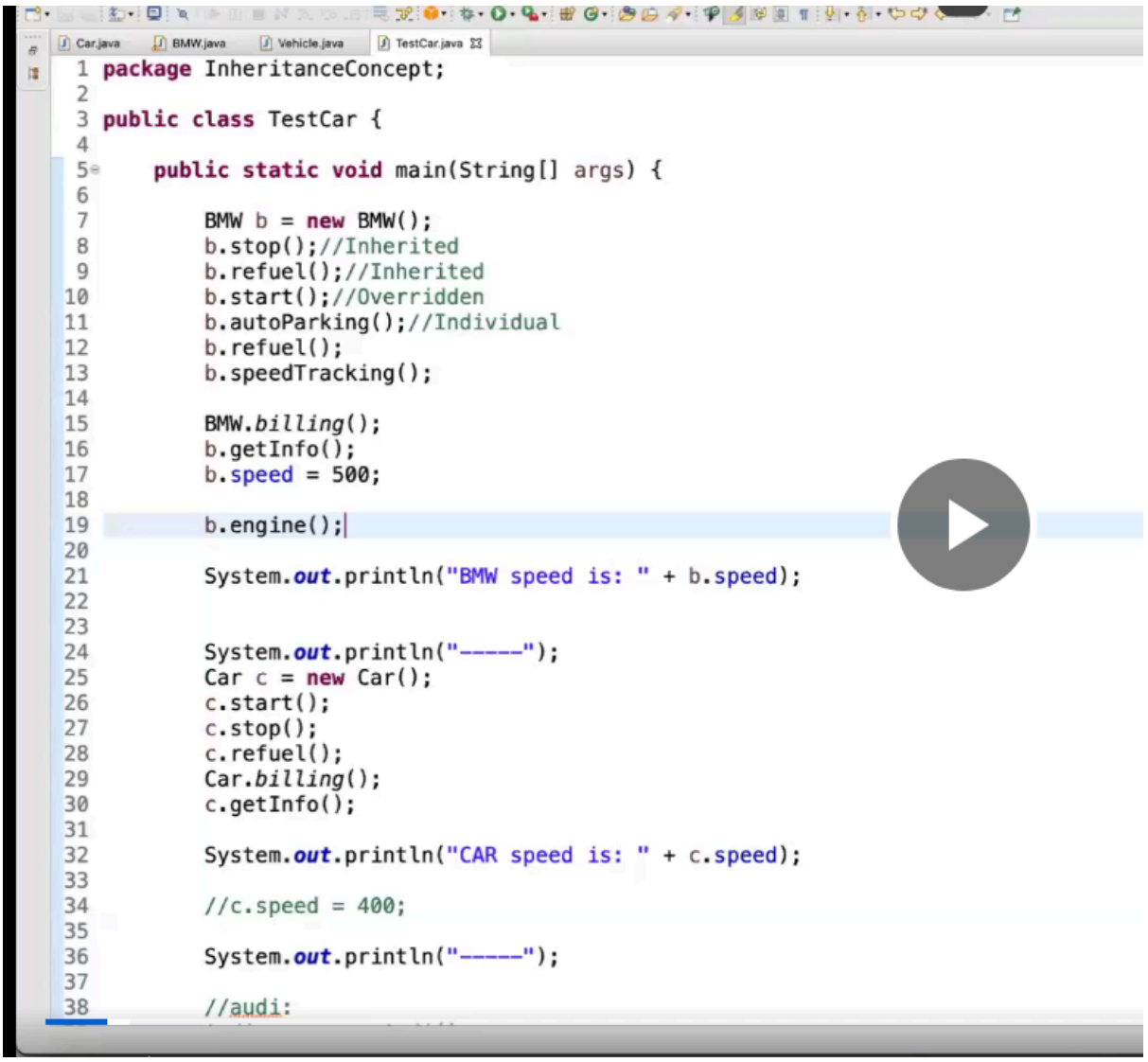

Car.java
BMW.java
int speed = 200;
// Method Overridding: poly(many)+Morphism(forms): RunTime(dynamic)
// when we have a method in paernt class and the same method in child class:
// 1. with the same name
// 2. with the same number of parameters
// 3. with the same sequence of parameters
// 4. buss logic/numbers of lines in the method -- doesnt matter
// 5. with the same return type
//static methods can not be overridden but can be overloaded
@Override
public void start() {
System.out.println("BMW -- start");
}
@Override
public void start(int a) {
System.out.println("BMW -- start");
}
public void autoParking() {
System.out.println("BMW -- auto parking");
}
@Override
public void applyBreak() {
System.out.println("car -- applyBreak");
}
//method hiding
public static void billing() {
System.out.println("BMW -- billing");
}
private void getCarInfo() {
System.out.println("BMW -- get info");
}
public void speedTracking() {
System.out.println("bmw -- speedTracking");
}
}
Writable

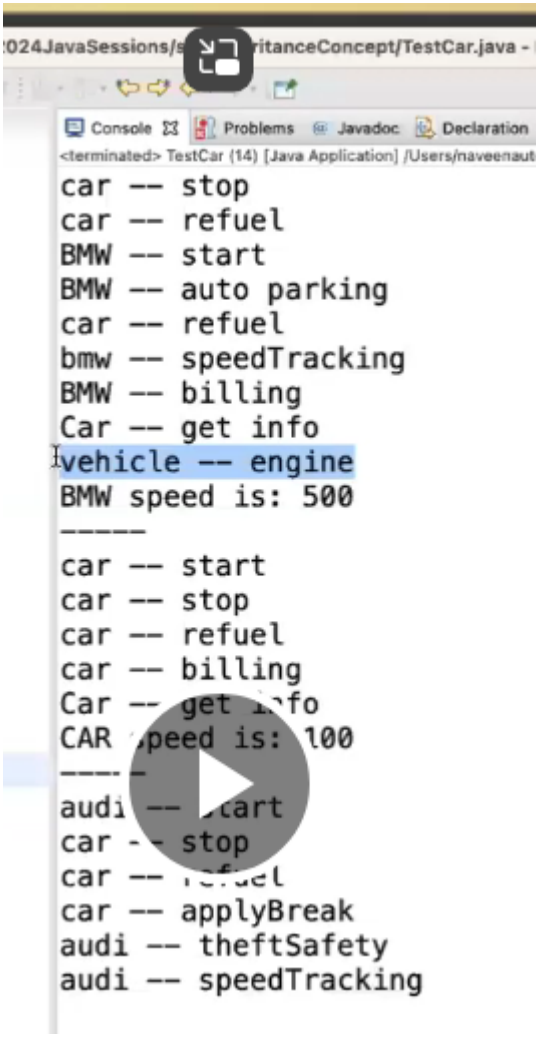


```
Car.java  BMW.java  Vehicle.java
 1 package InheritanceConcept;
 2
 3
 4 //final class -- to prevent inheritance
 5 public class Car extends Vehicle{
 6
 7     final int speed = 100;
 8
 9     public void start() {
10         System.out.println("car -- start");
11     }
12
13     public void start(int a) {
14         System.out.println("car -- start");
15     }
16
17     public void stop() {
18         System.out.println("car -- stop");
```



```
Car.java  BMW.java  Vehicle.java
 1 package InheritanceConcept;
 2
 3 public class Vehicle {
 4
 5     public void engine() {
 6         System.out.println("vehicle -- engine");
 7     }
 8
 9
10
11
12 }
13
```


Car.java
BMW.java
Vehicle.java
TestCar.java
1 package InheritanceConcept;
2
3 public class TestCar {
4
5 public static void main(String[] args) {
6
7 BMW b = new BMW();
8 b.stop();//Inherited
9 b.refuel();//Inherited
10 b.start();//Overridden
11 b.autoParking();//Individual
12 b.refuel();
13 b.speedTracking();
14
15 BMW.billing();
16 b.getInfo();
17 b.speed = 500;
18
19 b.engine();
20
21 System.out.println("BMW speed is: " + b.speed);
22
23
24 System.out.println("------");
25 Car c = new Car();
26 c.start();
27 c.stop();
28 c.refuel();
29 Car.billing();
30 c.getInfo();
31
32 System.out.println("CAR speed is: " + c.speed);
33
34 //c.speed = 400;
35
36 System.out.println("------");
37
38 //audi:

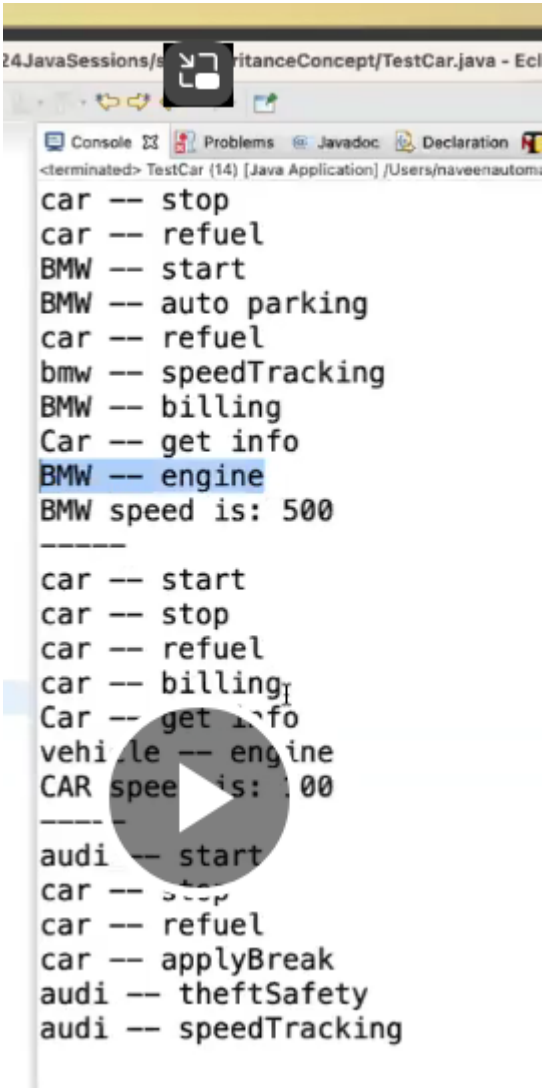



```
27          c.stop();
28          c.refuel();
29          Car.billing();
30          c.getInfo();
31          c.engine();
32
```

Bmw override-

```
36
37      @Override
38      public void engine() {
39          System.out.println("BMW -- engine");
40      }
41
```

```
17          b.speed = 500;
18
19          b.engine();
20
```



paste car2, vehicle2, bmw2, testfor2-

vehicle2.java

```java
package com.day18;

public class vehicle2 {

    public void startengine() {
        System.out.println("vehicle start engine method");
    }

    public int stopengine(String name) {
        System.out.println("vehicle start engine method");
        return 10;
    }

     public static void billing1() {
            System.out.println("vehicle static method");
        }

     private void getvehicledevice() {
         System.out.println("vehicle private method.");
     }

     public final int getvehiclename(String name) {
         System.out.println("final vehicle method");
         return 23324;
     }

}

```

car2.java

```java
package com.day18;

public class car2 extends vehicle2 {

    int speed =100;

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

     public void start(int a) {
            System.out.println("car -- start with parameter");
        }

     public void applyBreak() {
            System.out.println("car -- applyBreak");
        }

     //static method.
     public static void billing() {
            System.out.println("car -- billing");
        }

     //private method.
     private void getCarInfo() {
            System.out.println("car -- get info");
        }
            System.out.println("car -- get info");
        }

     public final int getdetails(String name) {
         System.out.println("car ---- get details method");
         return 98;
     }


}

```

bmw2.java

```java
package com.day18;

public class bmw2 extends car2{

    @Override
     public void start() {
            System.out.println("bmw -- start");
        }

    @Override
     public void start(int a) {
            System.out.println("bmw -- start with parameter");
        }

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

    @Override
    public void applyBreak() {
        System.out.println("bmw -- applyBreak");
    }

    public void getspeedfrombmw() {
        System.out.println("bmw only method");
    }
```

```java
        System.out.println("bmw only method");
    }

    public int getnamefrombmw(String name) {
        System.out.println("bmw only method");
        return 878;
    }

    public static int getcarlocation(String name) {
        System.out.println("bmw static method car location");
        return 78;
    }

}
```

testfor2.java

```java
package com.day18;

public class testfor2 {

    public static void main(String[] args) {

        bmw2 b1=new bmw2();
        //b1 can access all public, static, final of itself, parent and grandparent.

        car2 c1=new car2();
        //c1 can access all public, static, final of itself, parent and grandparent.
//      c1.

        vehicle2 v1=new vehicle2();

        //v1 can access all public, static, final of itself, parent and grandparent.

        System.out.println("bmw can access its own static method");
        int carlocation=bmw2.getcarlocation("tiger");
        System.out.println(carlocation);

        System.out.println("bmw can access its parent static method");
        bmw2.billing();

        System.out.println("bmw can access its grand parent static method");
        bmw2.billing1();

        //similarly for the car
        System.out.println("car can access its own static method");
        car2.billing();

        System.out.println("car can access its parent static method");
        car2.billing1();

        //similarly for vehicle
        System.out.println("vehicle can access its own static method");
        vehicle2.billing1();
    }

}
```

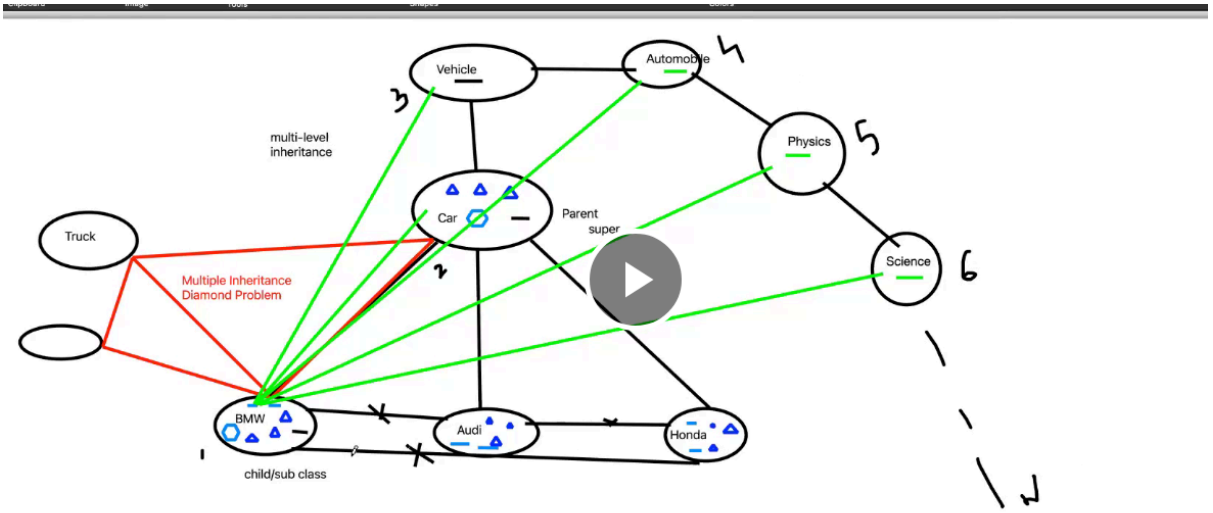
```java
}

//bmw can access its own static method
//bmw static method car location
//78
//bmw can access its parent static method
//car -- billing
//bmw can access its grand parent static method
//vehicle static method
//car can access its own static method
//car -- billing
```

Parents cannot take from child-

you wont get option only after dot.

```
49
50          Vehicle vh = new Vehicle();
51          vh.engine();
52
```

No limit on number of levels-




```
54
55          //child class object can be referred by parent class reference variable: Top Casting/Up Casting
56          Car c1 = new BMW();
57          c1.start();
```

```
-----
BMW -- start
```

Bmw start method is called not car class.

Car cannot access bmw individual method like auto parking.

```
60          c1.auto
61
```

This will work-

```
58          c1.stop();
59          c1.refuel();
60
```

```
-----
BMW -- start
car -- stop
car -- refuel
```

```
55      //child class object can be referred by parent class reference variable: Top Casting/Up Casting
56      //can access all the overridden and inherited methods only
57      //but can not access individual methods -- ref type check will be failed
```

Down casting-

Not allowed in java. Error.

```
63      //Down casting:
64      //parent class object can be referred by child class reference variable: Down Casting
65
66      BMW b1 = new Car();
```

Every car is not bmw so error.

Type mismatch: cannot convert from car3 to bmw3

```
65
66          BMW b1 = (BMW) new Car();
```

Cast it. Now error comes at run time. Class cast exception. run the code for it to come.

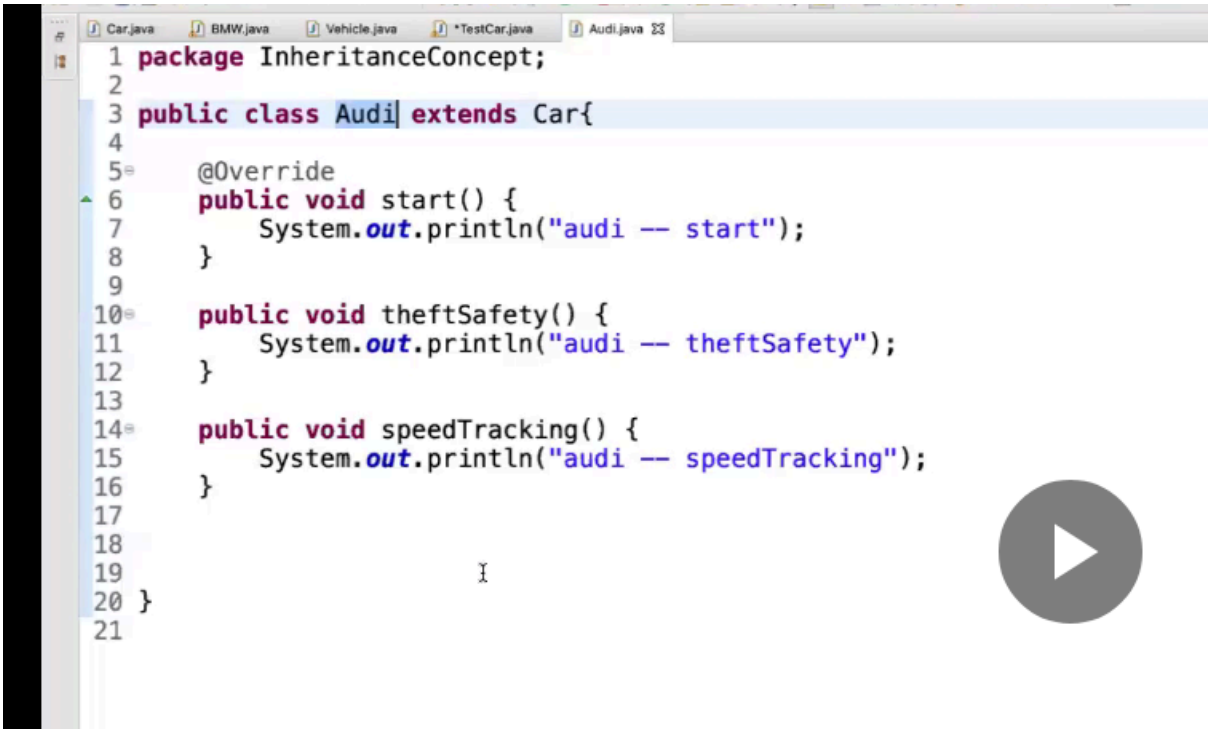
```
Exception in thread "main" java.lang.ClassCastException: class Inheritance
        at InheritanceConcept.TestCar.main(TestCar.java:66)
```

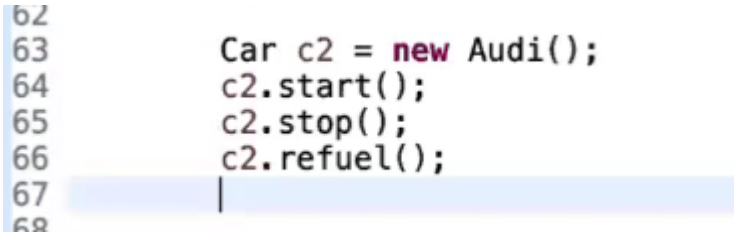
Is a relationship-

```
54
55      //child class object can be referred by parent class reference variable: Top Casting/Up Casting
56      //can access all the overridden and inherited methods only
57      //but can not access individual methods -- ref type check will be failed
58      Car c1 = new BMW();//IS_A relationship
```

In top casting and in inheritance.

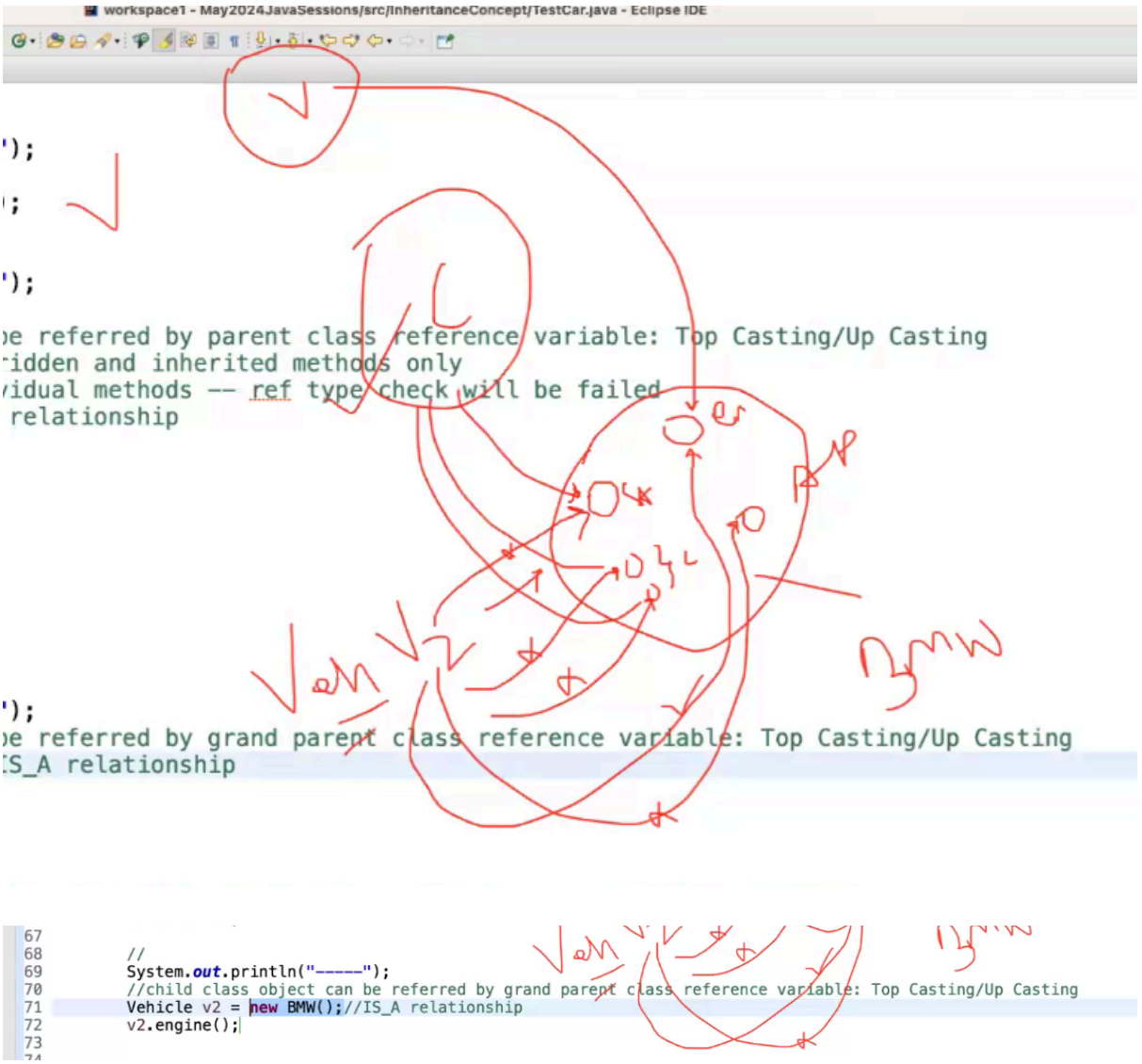
Audi car-



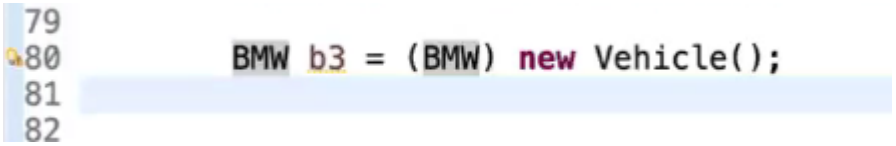


Note-

Parents can access the methods which are overridden from them because they also have the same methods..



Downcast to grandparent-

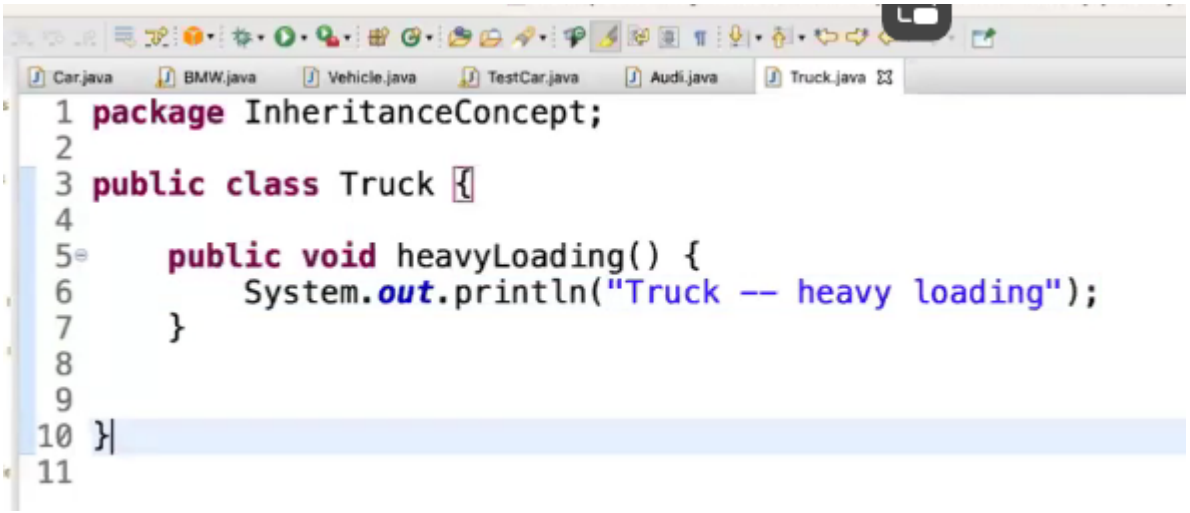


Class cast exception. we get this after running the code.//after running we get:

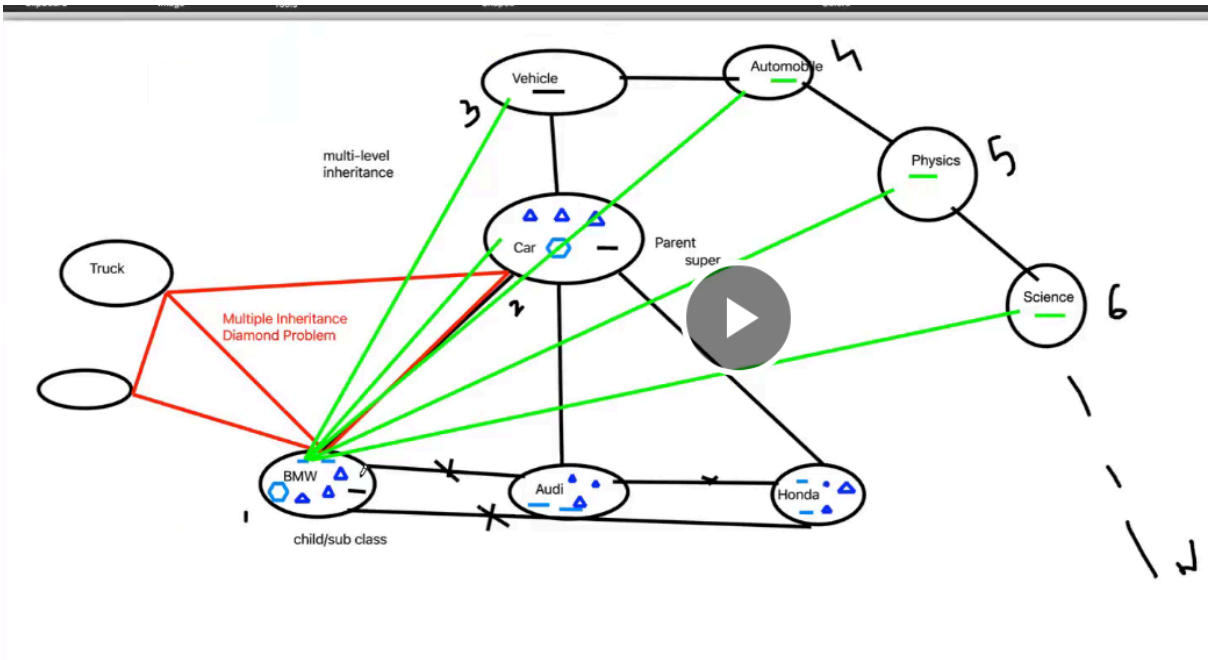
```
//Exception in thread "main" java.lang.ClassCastException:
//class com.day18.vehicle5 cannot be cast to class
com.day18.bmw5
//(com.day18.vehicle5 and com.day18.bmw5 are in unnamed
module of loader 'app')
```

//at com.day18.testfor5.main(testfor5.java:11)

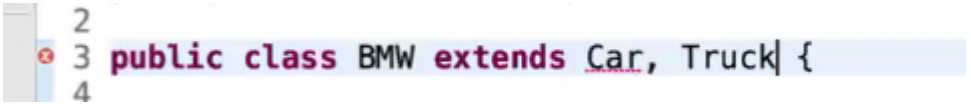
Truck class-



Multiple inheritance not allowed-



Not allowed error. //Syntax error on token(s), misplaced construct(s)



```
58
59    public void bmwLoading() {
60        Truck tr = new Truck();//BMW class is having Truck class object and able to access its methods:Composition
61        tr.heavyLoading();
62    }
```

```
19            b.engine();
20            b.bmwLoading();
```

```
BMW -- engine
Truck -- heavy loading
BMW speed is: 500
```

paste car6, vehicle6, bmw6, truck6, testfor6-

car6.java | bmw6.java | testfor6.java | vehicle6.java | truck6.java

JavaDataTypes/src/com/day18/car6.java

```java
public class vehicle6 {

    public void start() {
        System.out.println("Vehicle is starting...");
    }

    public void stop() {
        System.out.println("Vehicle is stopping...");
    }

    public void fuelUp() {
        System.out.println("Vehicle is fueling up...");
    }

}
```

car6.java | bmw6.java | testfor6.java | truck6.java

```java
package com.day18;

public class car6 extends vehicle6 {

    @Override
    public void start() {
        System.out.println("====method from vehicle class====");
        System.out.println("Car is starting with a key or button...");
        System.out.println("====method from vehicle class====");
    }

    @Override
    public void stop() {
    System.out.println("====method from vehicle class====");
        System.out.println("Car is stopping with brakes...");
        System.out.println("====method from vehicle class====");
    }

    public void openTrunk() {
        System.out.println("Car trunk is now open.");
    }

}
```

bmw6.java | testfor6.java | truck6.java

```java
package com.day18;

//Syntax error on token(s), misplaced construct(s)
//public class bmw6 extends car6, truck6{

public class bmw6 extends car6{

    @Override
    public void start() {
        System.out.println("====method from car class====");
        System.out.println("BMW is starting with push-button and advanced system...");
        System.out.println("====method from car class====");
    }

    @Override
    public void fuelUp() {
    System.out.println("====method from vehicle class====");
        System.out.println("BMW requires premium fuel.");
        System.out.println("====method from vehicle class====");
    }

    public void activateSportMode() {
        System.out.println("BMW is now in Sport Mode!");
    }
```

```java
    public void activateSportMode() {
        System.out.println("BMW is now in Sport Mode!");
    }

    //how to call sibling class methods and variables.
    public void callsibling() {
    truck6 t1=new truck6();
    int v1= t1.truck;
    System.out.println(v1);
    t1.loadCargo();
    t1.loadCargo("karan");
    }

}
```

testfor6.java truck6.java

```java
package com.day18;

public class truck6 {

    int truck=100;

    public void loadCargo() {
        System.out.println("Truck is loading cargo...");
    }

    public void loadCargo(String name) {
        System.out.println("Truck is loading cargo..." + " " + name);
    }

}
```

testfor6.java

```java
package com.day18;

public class testfor6 {

    public static void main(String[] args) {

        bmw6 b1=new bmw6();
        b1.callsibling();
    }

}

//100
//Truck is loading cargo...
//Truck is loading cargo... karan
```

Add variable-

```java
public class Vehicle {

    int price = 50;
```

```
        System.out.println("BMW price is : " + b.price);
```

```
BMW price is : 50
-----
```

Add same variable in child-

```
3
4 //final class -- to prevent inheritance
5 public class Car extends Vehicle{
6
7     final int speed = 100;
8     int price = 70;
9
```

```
22          System.out.println("BMW speed is: " + b.speed);
23          System.out.println("BMW price is : " + b.price);
24
```

```
BMW price is : 70
-----
```

Recent most value is captured. Also the immediate parent is given preference.

Bmw uses same variable and increased price-

Car.java | BMW.java | Vehicle.java | TestCar.java | Audi.java

```
1 package InheritanceConcept;
2
3 public class BMW extends Car {
4
5     //bmw class var
6     int speed = 200;
7     int price = 500;
8
```

```
22          System.out.println("BMW speed is: " + b.speed);
23          System.out.println("BMW price is : " + b.price);
```

```
BMW speed is: 500
BMW price is : 500
-----
```

Bmw value is taken.

Car price-

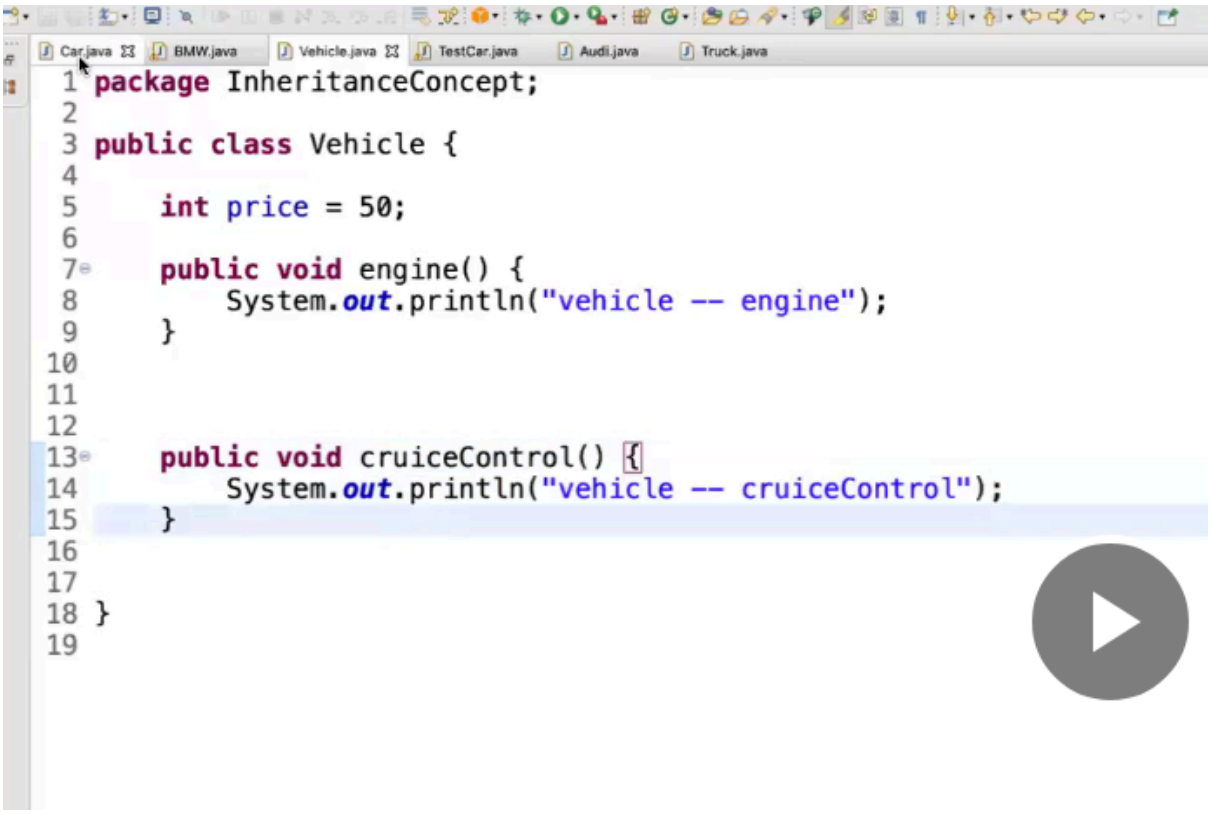
```
35          System.out.println("CAR speed is: " + c.speed);
36          System.out.println("CAR price is : " + c.price);
37
```

70.

Note-

First own class is checked, then one level up, if not there then one more level up.

One more method-




```
17          b.speed = 500;
18
19          b.engine();
20          b.bmwLoading();
21          b.cruiceControl();
22
```

Cruise control from vehicle printed.

Car overrides-

```
45
46      @Override
47      public void cruiceControl() {
48          System.out.println("car -- cruiceControl");
49      }
50
```

```
20          b.bmwLoading();
21          b.cruiceControl();
```

Car cruise control is called.

Bmw cruise control-

```
66      @Override
67      public void cruiceControl() {
68          System.out.println("BMW -- cruiceControl");
69      }
70
```

```
17          b.speed = 500;
18
19          b.engine();
20          b.bmwLoading();
21          b.cruiceControl();
```

Bmw cruise control.

```
34          c.engine();
35          c.cruiceControl();
```

Car cruise control.

method chaining–

```
28      }
29
30      public void autoParking() {
31          System.out.println("BMW -- auto parking");
32          stop();//Inheritance
33      }
34
35      @Override
```

```
29
30      public void autoParking() {
31          System.out.println("BMW -- auto parking");
32          applyBreak();//method chaining
33          stop();//Inheritance
34      }
```

paste vehicle14, car14, bmw14, testfor14 -

vehicle14.java | car14.java | bmw14.java | testfor14.java

```java
package com.day18;

public class vehicle14 {

    String type = "Generic Vehicle";

    //create one simple method here.
    public void startengine() {
        System.out.println("vehicle start engine method");
    }

    //call one method with different name and call one method from bmw14 and one method from car14.
    public void karanmethod1() {
        bmw14 b1=new bmw14();
        b1.method5();
        car14 c1=new car14();
        c1.karanmethod();
    }

}
```

car14.java | bmw14.java | testfor14.java

```java
package com.day18;

public class car14 extends vehicle14 {

    String type = "Car CLASs";

    //override the same method from vehicle11.
    @Override
    public void startengine() {
        System.out.println("car start engine method");
    }

    //create one method and call the bmw14 method here.
    public void karanmethod() {
        bmw14 b1=new bmw14();
        b1.method5();
    }

}
```

bmw14.java | testfor14.java

```java
package com.day18;

public class bmw14 extends car14{

    //we can chain any method with anything.
    //overridden methods can also be chained.

    String type = "bmw CLASs";

    //override method from car12.
    @Override
    public void startengine() {
        System.out.println("bmw start engine method");
        method1();
        method2();
        method3();
        method4();
    }

    //create another four methods.

    public void method1() {
        System.out.println("method1");
    }
    public void method2() {
        System.out.println("method2");
    }
    public static void method3() {
        System.out.println("method3");
    }
    public final void method4() {
        System.out.println("method4");
    }

    //create one method which calls all the above methods inside it.
    public void method5() {
        method1();
        method2();
        method3();
        method4();
        startengine();
    }

}
```

testfor14.java

```java
package com.day18;

public class testfor14 {

    public static void main(String[] args) {

        //create object of bmw14.
        bmw14 b1=new bmw14();
        b1.method5();
        b1.startengine();

        b1.karanmethod();
        b1.karanmethod1();

    }

}

//method1
//method2
//method3
//method4
//bmw start engine method
//method1
//method2
//method3
//method4
//bmw start engine method
//method1
//method2
```

```
//method1
//method2
//method3
//method4
//method1
//method2
//method3
//method4
//bmw start engine method
//method1
//method2
//method3
//method4
//method1
//method2
//method3
//method4
//bmw start engine method
//method1
//method2
//method3
//method4
//method1
//method2
//method3
//method4
//bmw start engine method
```

```
54 //method4
55 //bmw start engine method
56 //method1
57 //method2
58 //method3
59 //method4
60
61
62
```

Final and static methods can be inherited but not overridden.

paste car15, vehicle15.

car15.java vehicle15.java

```java
package com.day18;

public class vehicle15 {

    final String type = "Generic Vehicle";

    static int i=100;

    //create two final methods one with return and other no return.
    public final int getvehiclename(String name) {
        System.out.println("final vehicle method");
        return 23324;
    }

    public final void startengine() {
        System.out.println("vehicle start engine method");
    }

    //create two static methods one with return and other no return.
    public static int getvehiclename1(String name) {
        System.out.println("final vehicle method");
        return 23324;
    }

    public static void startengine1() {
        System.out.println("vehicle start engine method");
    }
}

```

car15.java

```java
package com.day18;

public class car15 extends vehicle15 {

final String type = "Generic Vehicle";

    static int i=100;

    //override the final and static variables from vehicle15.
    //Cannot override the final method from vehicle15
//  @Override
//  public final void startengine() {
//      System.out.println("car start engine method");
//  }

    //The method startengine1() of type car15 must override or implement a supertype method
    @Override
    public static void startengine1() {
        System.out.println("car start engine method");
    }

}

```

paste vehicle16, car16.

car16.java | vehicle16.java

```java
package com.day18;

public class vehicle16 {

    final String type = "Generic Vehicle";

    static int i=100;

    //create two final methods one with return and other no return.
    public final int getvehiclename(String name) {
        System.out.println("final vehicle method");
        return 23324;
    }

    public final void startengine() {
        System.out.println("vehicle start engine method");
    }

    //create two static methods one with return and other no return.
    public static int getvehiclename1(String name) {
        System.out.println("final vehicle method");
        return 23324;
    }

    public static void startengine1() {
        System.out.println("vehicle start engine method");
    }
}

```

car16.java

```java
package com.day18;

public class car16 extends vehicle15 {

final String type = "Generic Vehicle";

    static int i=100;

    //even without overload annotation the same error for final methods.

//Cannot override the final method from vehicle15
    public final void startengine() {
        System.out.println("car start engine method");
    }

//method hiding.
    public static void startengine1() {
        System.out.println("car start engine method");
    }

}

```

paste vehicle17, car17, testfor17-

vehicle17.java

```java
package com.day18;

public class vehicle17 {

    final String type = "Generic Vehicle";

    static int i=100;

    //create two final methods one with return and other no return.
    public final int getvehiclename(String name) {
        System.out.println("final vehicle method");
        return 23324;
    }

    public final void startengine() {
        System.out.println("vehicle start engine method");
    }

    //create two static methods one with return and other no return.
    public static int getvehiclename1(String name) {
        System.out.println("final vehicle method");
        return 23324;
    }

    public static void startengine1() {
        System.out.println("vehicle start engine method");
    }
}

```

car17.java

```java
package com.day18;

public class car17 extends vehicle15 {

final String type = "++car Generic Vehicle";

    static int i=2100;

//method hiding.
    public static void startengine1() {
        System.out.println("car start engine method");
    }

}

```

testfor17.java

```java
package com.day18;

public class testfor17 {

    public static void main(String[] args) {

        car17 c1=new car17();
        c1.startengine1();
        String s1=c1.type;
        System.out.println(s1);
        int v1=c1.i;
        System.out.println(v1);
    }

}

//car start engine method
//++car Generic Vehicle
//2100

```

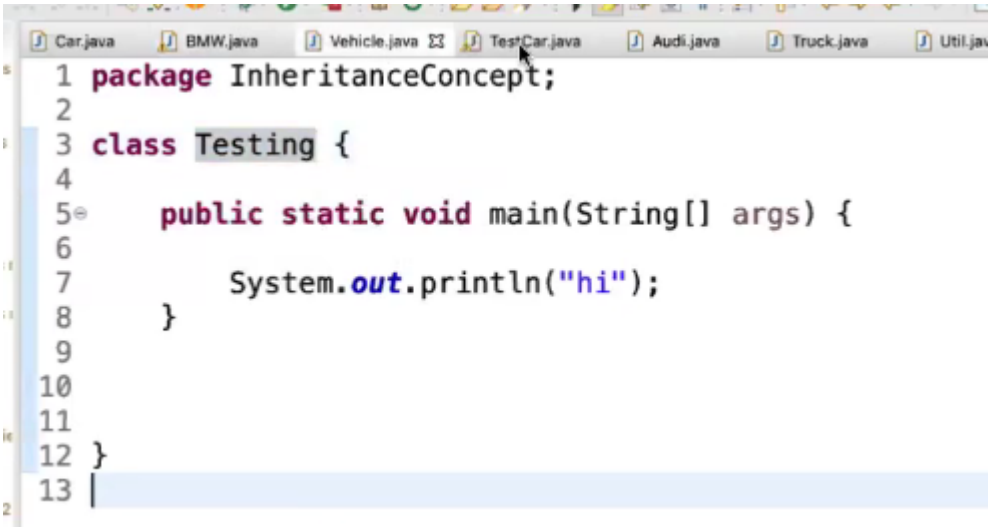
Calling final methods-

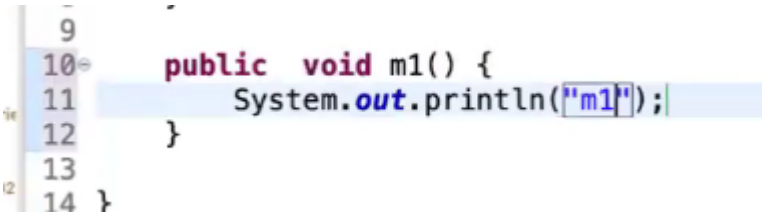
```java
        Audi au = new Audi();
        au.speedTracking();
```

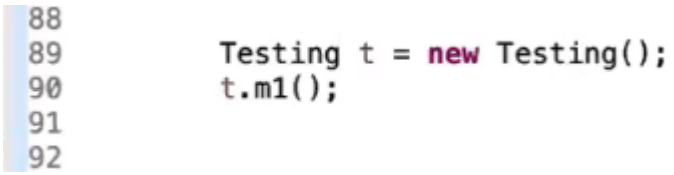
Audi speed tracking.

Using composition we can access child class method in parent and grand parents methods. We create objects.

Default class-



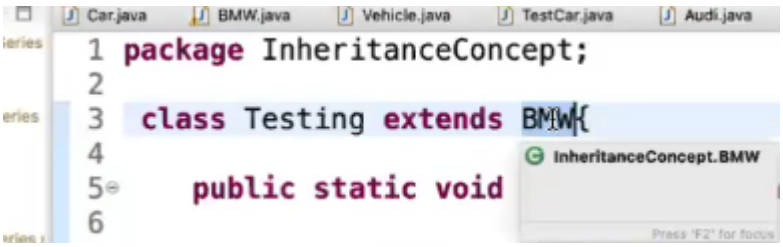




Works fine.

M1 is the output.

Default class can extend another class –



paste default2, bmw14-

default2.java bmw14.java

```java
package com.day18;

//default class can also extend another class.
 class default2 extends bmw14 {

    public static void main(String[] args) {

        default2 d1=new default2();
        d1.method5();
        d1.startengine();
        String name=d1.type;
        System.out.println(name);
        d1.karanmethod();
        d1.karanmethod1();
        d1.method1();
        d1.method2();
        d1.method4();
        d1.startengine();
        d1.method3();//The static method method3() from the type bmw14 should be
        //accessed in a static way

        default2.method3();//default can access static method by the class name.
        bmw14.method3(); //bmw can call its static method.


}

 }
```

```
27
28  }
29
30
31 // method1
32 // method2
33 // method3
34 // method4
35 // bmw start engine method
36 // method1
37 // method2
38 // method3
39 // method4
40 // bmw start engine method
41 // method1
42 // method2
43 // method3
44 // method4
45 // bmw CLASs
46 // method1
47 // method2
48 // method3
49 // method4
50 // bmw start engine method
51 // method1
52 // method2
53 // method3
54 // method4
```

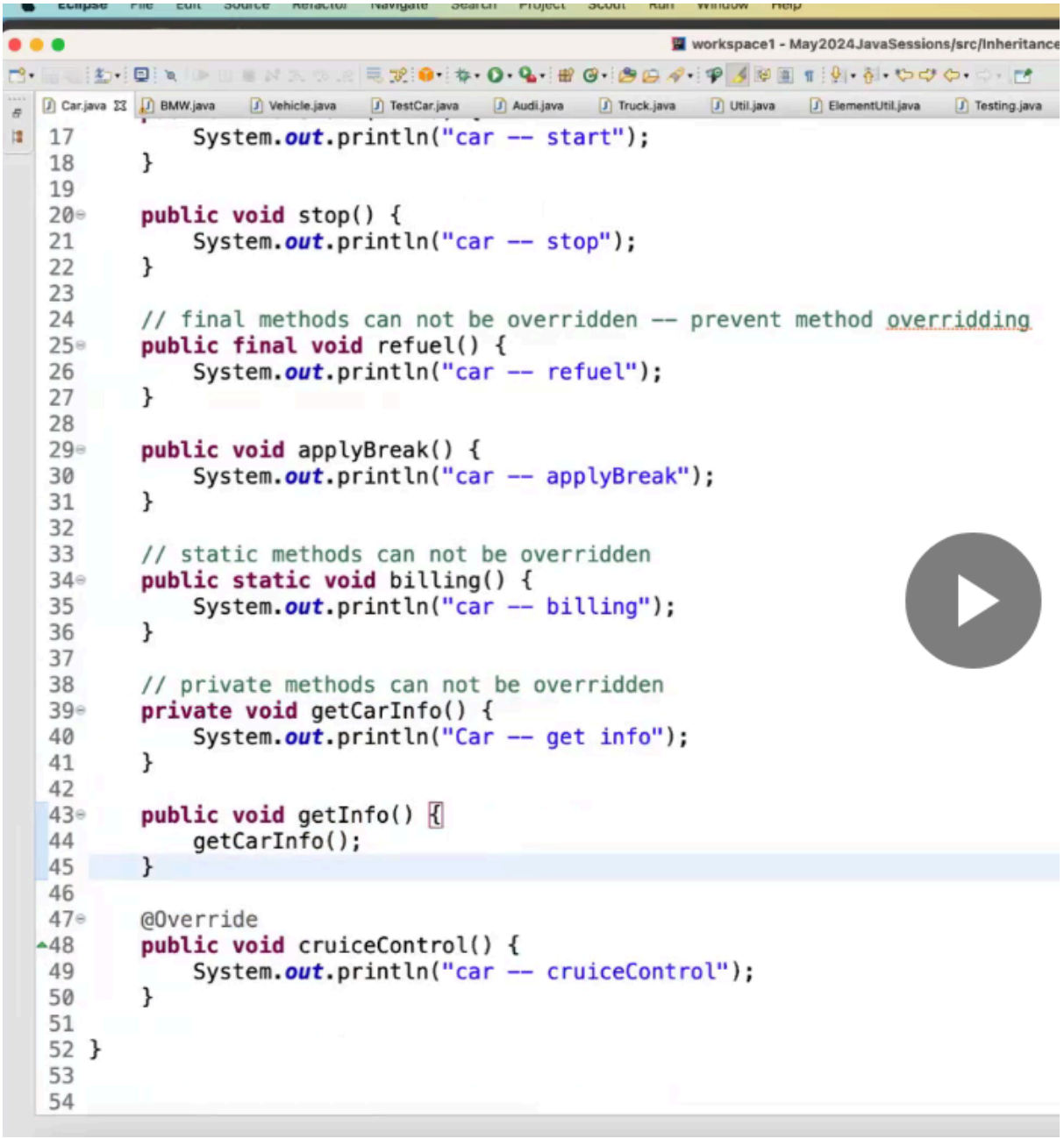
```
53 // method3
54 // method4
55 // method1
56 // method2
57 // method3
58 // method4
59 // bmw start engine method
60 // method1
61 // method2
62 // method3
63 // method4
64 // method1
65 // method2
66 // method3
67 // method4
68 // bmw start engine method
69 // method1
70 // method2
71 // method3
72 // method4
73 // method1
74 // method2
75 // method4
76 // bmw start engine method
77 // method1
```

```
76 // bmw start engine method
77 // method1
78 // method2
79 // method3
80 // method4
81 // method3
82 // method3
83 // method3
84
85
86
87
```

bmw14.java

```java
package com.day18;

public class bmw14 extends car14{

    //we can chain any method with anything.
    //overridden methods can also be chained.

    String type = "bmw CLASs";

    //override method from car12.
    @Override
    public void startengine() {
        System.out.println("bmw start engine method");
        method1();
        method2();
        method3();
        method4();
    }

    //create another four methods.

    public void method1() {
        System.out.println("method1");
    }
    public void method2() {
        System.out.println("method2");
    }
    public static void method3() {
        System.out.println("method3");
    }
    public final void method4() {
        System.out.println("method4");
    }

    //create one method which calls all the above methods inside it.
    public void method5() {
        method1();
        method2();
        method3();
        method4();
        startengine();
    }

}
```

Note-


workspace1 - May2024JavaSessions/src/Inheritance
Car.java
BMW.java
Vehicle.java
TestCar.java
Audi.java
Truck.java
Util.java
ElementUtil.java
Testing.java
System.out.println("car -- start");
}
public void stop() {
System.out.println("car -- stop");
}
// final methods can not be overridden -- prevent method overridding
public final void refuel() {
System.out.println("car -- refuel");
}
public void applyBreak() {
System.out.println("car -- applyBreak");
}
// static methods can not be overridden
public static void billing() {
System.out.println("car -- billing");
}
// private methods can not be overridden
private void getCarInfo() {
System.out.println("Car -- get info");
}
public void getInfo() {
getCarInfo();
}
@Override
public void cruiceControl() {
System.out.println("car -- cruiceControl");
}
}