Inheritance -

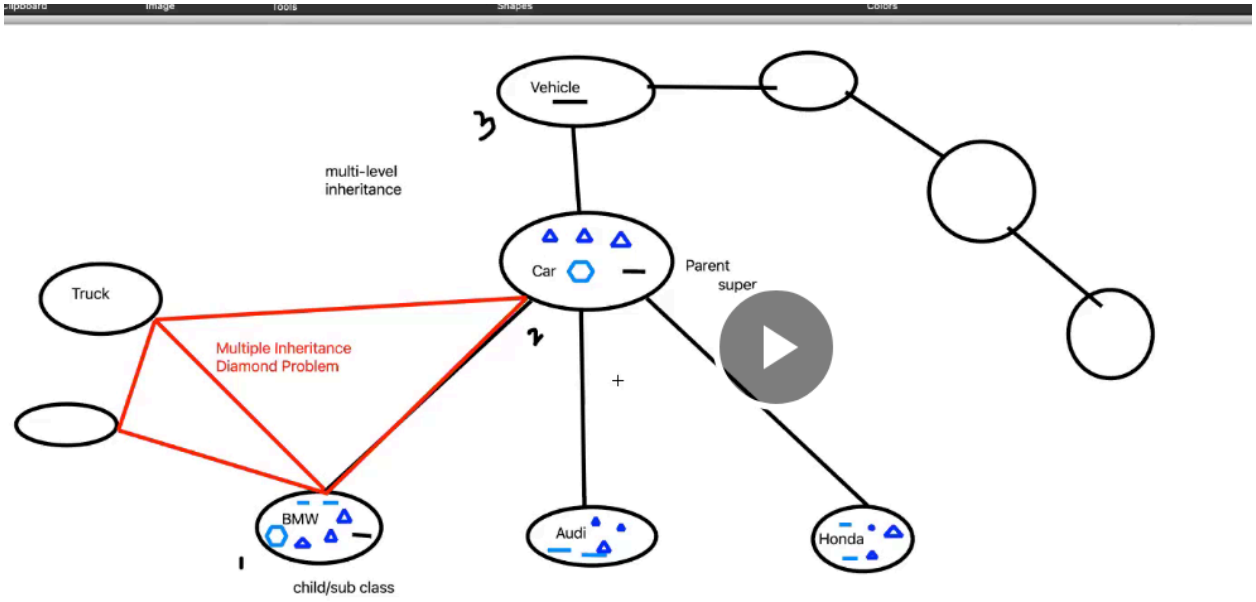
Reverse inheritance not possible in java.
Parent cannot take properties from child.

Child can take properties from parent and above parents.(grand parent).

One child cannot have two parent classes.

Multiple inheritance known as diamond problem. Not allowed in java.

Multi level inheritance is allowed. Child to parent to grand parent to grand parent etc etc.



Car class-

Car.java

```
1  package InheritanceConcept;
2
3  public class Car {
4
5
6      public void start() {
7          System.out.println("car -- start");
8      }
9
10     public void stop() {
11         System.out.println("car -- stop");
12     }
13
14     public void refuel() {
15         System.out.println("car -- refuel");
16     }
17
18
19 }
20
```

Bmw class-

```java
package InheritanceConcept;

public class BMW extends Car {

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

    public void autoParking() {
        System.out.println("BMW -- auto parking");
    }

}
```

Test-

```java
package InheritanceConcept;

public class TestCar {

    public static void main(String[] args) {

        BMW b = new BMW();
        b.start();
```

Bmw start is called.

Call parent class methods-

```java
        b.stop();
        b.refuel();
```

```
<terminated> TestCar (14) [Java Appl
car -- stop
car -- refuel
```

When you create reference of child class and object of child class, and the same method exist in parent and child class, then child class method is given preference.

Method overriding-

```
Car.java   BMW.java   TestCar.java
1  package InheritanceConcept;
2
3  public class BMW extends Car {
4
5      //Method Overridding:
6      //when we have a method in paernt class and the same method in child class:
7      //1. with the same name
8      //2. with the same number of parameters
9      //3.
10     //4. buss logic/numbers of lines in the method -- doesnt matter
11
12     @Override
13     public void start() {
14         System.out.println("BMW -- start");
15     }
16
17     @Override
18     public void autoParking() {
19         System.out.println("BMW -- auto parking");
20     }
21
22 }
23
```

We cannot write override on a method which is not overridden. Compile error.

//The method autoParking() of type bmw2 must override
//or implement a supertype method.

Note-
Override annotation is not mandatory to write. It makes the code readable.

```
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
```

Inheritance promotes reusability.

Parent class has parameter method-

```
3 public class Car {
4
5
6     public void start(int a) {
7         System.out.println("car -- start");
8     }
9
```

Child class no parameter-

```
12    @Override
13    public void start() {
14        System.out.println("BMW -- start");
15    }
16
```

Compile error - cannot override the method.

The method start() of type bmw1 must override or implement a supertype method

Sequence also matters-

Car class-

```
1 package InheritanceConcept;
2
3 public class Car {
4
5
6     public void start(int a, String b) {
7         System.out.println("car -- start");
8     }
9
```

Bmw class-

```
5     //Method Overridding:
6     //when we have a method in paernt class and the same method in child class:
7     //1. with the same name
8     //2. with the same number of parameters
9     //3. with the same sequence of parameters
10    //4. buss logic/numbers of lines in the method -- doesnt matter
11
12    @Override
13    public void start(String b, int a) {
14        System.out.println("BMW -- start");
15    }
```

//The method start(String, int) of type bmw21 must override or implement a supertype method

Parent class returns-

```java
package InheritanceConcept;

public class Car {


    public int start() {
        System.out.println("car -- start");
        return 100;
    }
```

Child class does not return-

Compile error. Not a overridden method.

```java

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

//The return type is incompatible with car3.start().

This is overriding-

Child class.

```java
    @Override
    public int start() {
        System.out.println("BMW -- start");
        return 200;
    }
```

```java
//Method Overridding:
//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
```

Test class-

```
System.out.println("-----");
Car c = new Car();
c.start();
```

Car start method.

```
c.start();
c.stop();
c.refuel();
```

```
car -- stop
car -- refuel
```

Car cannot access properties of child class. We wont get options only after dot.

Static method in car class-

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

Static method in bmw class-

```
@Override
public static void billing() {
    System.out.println("car -- billing");
}
```

Static method cannot be overridden.

// The method billing() of type bmw4 must override or implement a supertype method

Static methods can be overloaded.
Static method overloaded is called method hiding.

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

```java
    // 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
```

Paste car4, bmw4, testcar4bmw4–

car4.java

```java
package com.day17;

public class car4 {

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

}

```

bmw4.java

```java
package com.day17;

public class bmw4 extends car4 {


    //parent class has static method.
    //child class has same overridden static method.
    //not allowed.
    //error.
//  The method billing() of type bmw4 must override or implement a supertype method
//  @Override
//    public static void billing() {
//        System.out.println("car -- billing");
//    }

    //static method can be overloaded.
    //called as method hiding.
    //Static method overloaded is called method hiding.

    public static void billing() {
        System.out.println("bmw -- billing");
    }

}

```

testcar4bmw4.java

```java
package com.day17;

public class testcar4bmw4 {

    public static void main(String[] args) {

        bmw4.billing();
        car4.billing();

        bmw4 b1=new bmw4();
        b1.billing();//The static method billing() from the type bmw4 should be
        //accessed in a static way

    }

}

//bmw -- billing
//car -- billing
//bmw -- billing
```

Method hiding -

The method should be carbon copies in parent and child.

Car -

```java
    //static methods can not be overridden
    public static void billing(int a) {
        System.out.println("car -- billing");
    }
```

Bmw-

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

Here parent has parameter but child does not have so not method hiding.

Test-

```
        BMW.billing();
```

Bmw billing is called.
Call car billing-

```
        Car.billing();
```

Comment out billing in bmw-

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

Test-

```
        BMW.billing();
```

Car billing is called.

Note- 49.00
~~If parent class has static method but not in child classes, then both parent and child class can call it using class name.~~

Paste car6, bmw6-

car6.java bmw6.java

```java
package com.day17;

public class car6 {

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

}

```

bmw6.java

```java
package com.day17;

public class bmw6 extends car6 {

    //This instance method cannot override the static method from car4
//    public void billing() {
//        System.out.println("bmw -- billing");
//    }

    //The method billing() of type bmw6 must override or implement a supertype method
    @Override
    public void billing() {
        System.out.println("bmw -- billing");
    }

}

```

```java
    //static methods can not be overridden but can be overloaded
```

Private method in car -

```java
    private void getCarInfo() {
        System.out.println("Car -- get info");
    }
```

Override in bmw-

```
34
35    @Override
36    private void getCarInfo() {
37        System.out.println("BMW -- get info");
38    }
39
```

Cannot override private method.

Paste car7 bmw7-

car7.java

```java
package com.day17;

public class car7 {

    private void billing() {
        System.out.println("car -- billing");
    }

    private static int getnumber(int a) {
        System.out.println("return the car number");
        return 10;
    }

}

```

bmw7.java

```java
package com.day17;

public class bmw7 extends car7 {

    //Cannot override private method.


    //The method billing() of type bmw7 must override or implement a supertype method
    @Override
    private void billing() {
        System.out.println("car -- billing");
    }

    //The method getnumber(int) of type bmw7 must override or implement a supertype method
    @Override
    private static int getnumber(int a) {
        System.out.println("return the car number");
        return 10;
    }

}

```

Private methods can be overloaded.

Paste car8 bmw8 testcar8bmw8 -

car8.java | bmw8.java | testcar8bmw8.java

```java
package com.day17;

public class car8 {

    private void billing() {
        System.out.println("car -- billing");
    }

    private static int getnumber(int a) {
        System.out.println("return the car number");
        return 10;
    }

}

```

bmw8.java | testcar8bmw8.java

```java
package com.day17;

public class bmw8 extends car8 {

    //private methods can be overloaded.


    //The method billing() from the type bmw8 is never used locally
    private void billing() {
        System.out.println("car -- billing");
    }

//The method getnumber(int) from the type bmw8 is never used locally
    private static int getnumber(int a) {
        System.out.println("return the car number");
        return 10;
    }

}

```

testcar8bmw8.java

```java
package com.day17;

public class testcar8bmw8 {

    public static void main(String[] args) {

        bmw8 b1=new bmw8();
//      b1.//dont see any static methods here from parent or child.
        bmw.//dont see any static methods here from parent or child.

    }

}
```

To avoid overriding a method-

Use the final keyword.

Car class.

```java
    //final methods can not be overridden
    public final void refuel() {
        System.out.println("car -- refuel");
    }
```

Child class-

```java
    }

    public final void refuel() {
        System.out.println("car -- refuel");
    }
```

//Cannot override the final method from car9

Paste car9 bmw 9–

car9.java | bmw9.java

```java
package com.day17;

public class car9 {

     // final methods cannot be overridden
    public final void refuel() {
        System.out.println("car -- refuel");
    }

}

```

bmw9.java

```java
package com.day17;

public class bmw9 extends car9{

    //Cannot override the final method from car9
    //final method cannot be overridden.
////@Override
//  public final void refuel() {
//        System.out.println("car -- refuel");
//    }

    //Cannot override the final method from car9
    public final void refuel() {
        System.out.println("car -- refuel");
    }

}

```

Test -

Bmw can call final method but no changes can be done.

```
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();
```

Class can be final-

```
2
3 public final class Car {
4
```

Final class cannot be parent.

Child class will throw error.

//The type bmw11 cannot subclass the final class car11

```
Car.java   BMW.java   TestCar.java
1 package InheritanceConcept;
2
3 public class BMW extends Car {
```

```
4 //final class -- to prevent inheritance
5 public final class Car {
```

```
15     // final methods can not be overridden -- prevent method overridding
16     public final void refuel() {
```

Car class-

Encapsulate to call the private method.

```java
    // private methods can not be overridden
    private void getCarInfo() {
        System.out.println("Car -- get info");
    }

    public void getInfo() {
        getCarInfo();
    }
```

Test class-

```java
        b.getInfo();
```

```
Car -- get info
-----
```

Even car object can call -

```java
        System.out.println("-----");
        Car c = new Car();
        c.start();
        c.stop();
        c.refuel();
        Car.billing();
        c.getInfo();
```

Variables in parent class-

```java
//final class -- to prevent inheritance
public class Car {

    int speed = 100;
```

Test-

Bmw can access speed also.

```
17
18        System.out.println("BMW speed is: " + b.speed);
```

Bmw speed is 100

Bmw also has its own speed-

```
Car.java   *BMW.java   TestCar.java
1 package InheritanceConcept;
2
3 public class BMW extends Car {
4
5     int speed = 200;
```

Test-

```
17
18        System.out.println("BMW speed is: " + b.speed);
```

Bmw speed is 200

Create object of car class and print speed-

```
28
29        System.out.println("CAR speed is: " + c.speed);
30
```

Car speed is 100.

Paste car13 bmw13 testcar13bmw13-

car13.java

```java
package com.day17;

public class car13 {

    public int speed =10;
    private String name = "opel";


}

```

bmw13.java

```java
package com.day17;

public class bmw13 extends car13 {

    public int speed =100;
    private String name = "austria";

}

```

testcar13bmw13.java

```java
package com.day17;

public class testcar13bmw13 {

    public static void main(String[] args) {

        bmw13 b1=new bmw13();
        int speed = b1.speed;
        //private variables cant be accessed without encapsulation.
        System.out.println(speed);

        b1.speed=2323434;
        speed = b1.speed;
        System.out.println(speed);

        car13 c1=new car13();
        int speed2 = c1.speed;
        System.out.println(speed2);

        c1.speed=234324;
        speed2 = c1.speed;
        System.out.println(speed2);
    }

}
```

```java

}

//100
//2323434
//10
//234324

```

Variable overriding is not present.

//The annotation @Override is disallowed for this location

Not override annotation for variables-

```
3 public class BMW extends Car {
4
5     @Override
6     int speed = 200;
```

Lets declare variable as final-

```
Car.java   BMW.java   TestCar.java
1 package InheritanceConcept;
2
3
4 //final class -- to prevent inheritance
5 public class Car {
6
7     final int speed = 100;
```

Bmw-

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

Hence proved no variable overriding.
Else we would have got error for final variable.

Test-

```
31        c.speed = 400;
32
33
34    }
35
```

The final field Car.speed cannot be assigned
1 quick fix available:
Remove 'final' modifier of 'speed'

Car also cannot change final.

/The final field car15.speed cannot be assigned

Bmw can change variable because it is not final-

```
17
18          b.speed = 500;
```

Paste car16, bmw16, testcar16bmw16 -

car16.java | bmw16.java | testcar16bmw16.java

```
1  package com.day17;
2
3  public class car16 {
4
5      final public int speed =10;
6      private String name = "opel";
7
8
9  }
10
```

bmw16.java | testcar16bmw16.java

```
1  package com.day17;
2
3  public class bmw16 extends car16 {
4
5
6      private String name = "austria";
7
8  }
9
```

testcar16bmw16.java

```java
package com.day17;

public class testcar16bmw16 {

    public static void main(String[] args) {

        //child class has no same variable.
        //car class has final speed.
        //bmw can change the final speed.
        //car cannot change its final speed.

        bmw15 b1=new bmw15();
        int speed = b1.speed;
        System.out.println(speed);

        b1.speed=324324;
        speed = b1.speed;
        System.out.println(speed);

        car15 c1=new car15();
        int speed2 = c1.speed;
        System.out.println(speed2);

//      c1.speed=234324; //The final field car15.speed cannot be assigned
//      speed2 = c1.speed;
//      System.out.println(speed2);
    }
```

```java
//      System.out.println(speed2);
    }

}

//100
//324324
//10
```

Writable | Smart Insert | 1 : 1 : 0

Sibling inheritance not allowed.

Audi-

```
Car.java   BMW.java   TestCar.java   *Audi.java

1  package InheritanceConcept;
2
3  public class Audi extends Car{
4
5      @Override
6      public void start() {
7          System.out.println("audi -- start");
8      }
9
10     public void theftSafety() {
11         System.out.println("audi -- theftSafety");
12     }
```

Test-

```
36          System.out.println("-----");
37
38          //audi:
39          Audi au = new Audi();
40          au.start();
41
```

Audi start called.

```
41          au.stop();
42          au.refuel();
43          au.applyBreak();
44          au.theftSafety();
```

```
car -- stop
car -- refuel
car -- applyBreak
audi -- theftSafety
```

Audi cannot get any method of its sibling. After dot you wont get anything.

Paste car17, bmw17, audi17, car17bmw17audi17-

car17.java | bmw17.java | audi17.java | testcar17bmw17audi17.java

```java
package com.day17;

public class car17 {

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

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

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

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

}

```

bmw17.java | audi17.java | testcar17bmw17audi17.java

```java
package com.day17;

public class bmw17 extends car17 {

    public void antilock() {
        System.out.println("bmw anti lock method");
    }

}

```

audi17.java | testcar17bmw17audi17.java

```java
package com.day17;

public class audi17 extends car17 {

    @Override
    public void start() {
        System.out.println("Car start method from audi");
    }

    public void theftsafety() {
        System.out.println("audi theft safety method");
    }


    //The method antilock() of type audi17 must override or implement a supertype method
//  @Override
//  public void antilock() {
//      System.out.println("bmw anti lock method");
//  }

}

```

testcar17bmw17audi17.java

```java
package com.day17;

public class testcar17bmw17audi17 {

    public static void main(String[] args) {

        audi17 a1=new audi17();
        a1.start();
        a1.applybreak();
        a1.refuel();
        a1.stop();
        a1.theftsafety();
//      a1.
    }

}

//Car start method from audi
//car break method
//car refuel method
//Car stop method
//audi theft safety method
```

Classes can have public, final and abstract, attached to them, else error.

Paste access 1 to 6 -

access1.java | access2.java | access3.java | access4.java | access5.java | access6.java

```java
package com.day17;

//The class access1 can be either abstract or final, not both
public final abstract class access1 {

}

```

access2.java | access3.java | access4.java | acces

```java
package com.day17;

//valid class name.
public  abstract class access2 {

}

```

access3.java | access4.java | access5.java | access6.

```java
package com.day17;

//valid class name.
public  abstract class access3 {

}

```

access4.java | access5.java | access6.java

```java
package com.day17;

//Illegal modifier for the class access4; only public, abstract & final are permitted
protected  abstract class access4 {

}

```

access5.java | access6.java

```java
package com.day17;

//Syntax error on token "default", delete this token
default  abstract class access5 {

}

```

access6.java

```java
package com.day17;

//valid class name.
   class access6 {

     public static void main(String[] args) {

         System.out.println("hello world"); //hello world
     }

}

```

Car class-

```java

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

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

Bmw-

As we have used same method but with different parameters. Also used the override annotation. Overloading + overriding at the same time.

```
16      //static methods can not be overridden but can b
17
18      @Override
19      public void start() {
20          System.out.println("BMW -- start");
21      }
22
23      @Override
24      public void start(int a) {
25          System.out.println("BMW -- start");
26      }
```