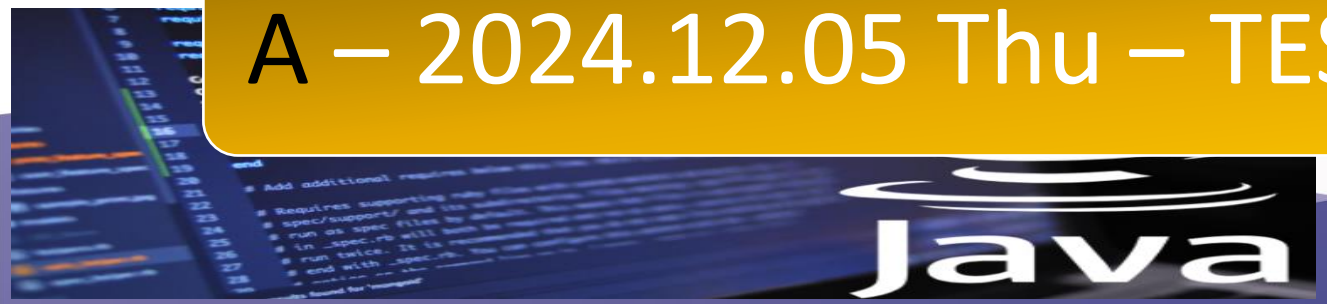
CJ Information

S – AP CS A

IC – Method Overriding & Polymorphism

HW - None

A – 2024.12.05 Thu – TEST



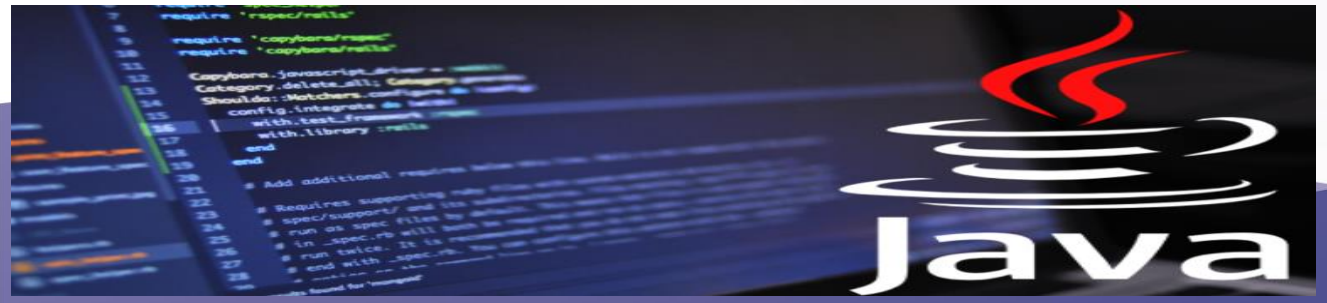
Method Overriding in Java

Outline :

- Superclass & Subclass
- Overriding methods in a Java class

What we learn today:

- How to implement method override
- Some basic rules to implement method override
- Connection to Polymorphism
- Examples



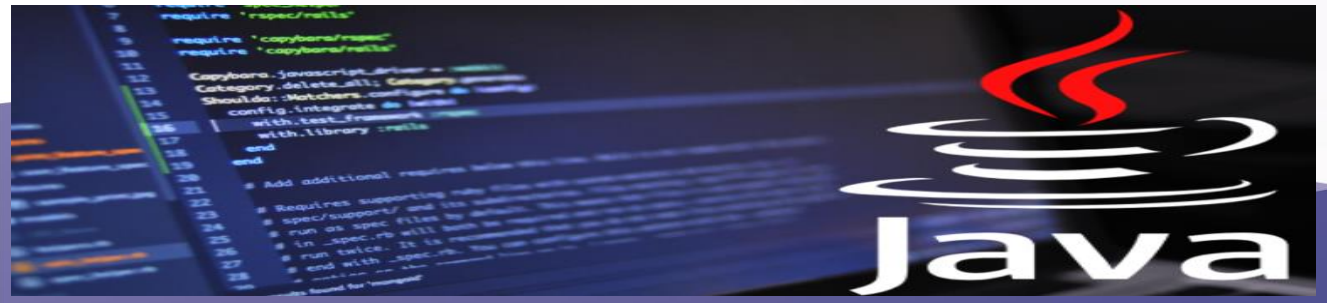
Method Overriding in Java

Overriding : *feature that allows a subclass to provide a specific implementation of a method that is already defined in the superclass*

When a method in a subclass has the

1. same name
2. same number and type of parameters
3. same return type

of a method in a superclass, then this method **overrides** the method in the superclass



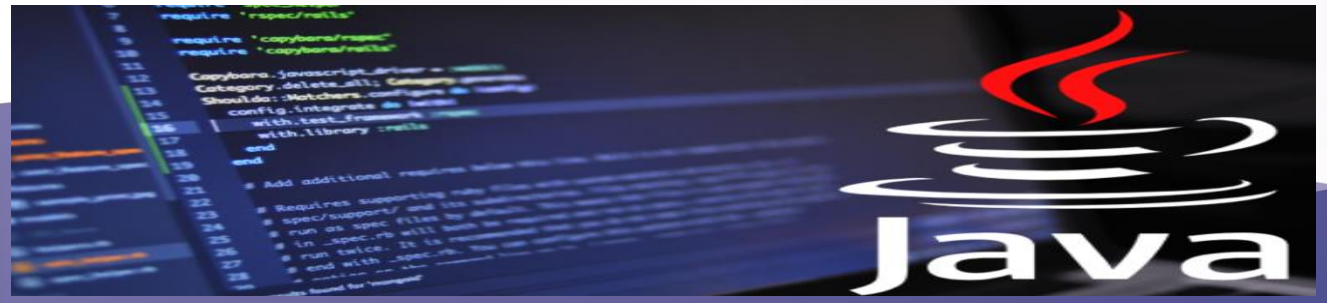
Method Overriding in Java

Overriding Rules:

To override a method in a subclass, it must have

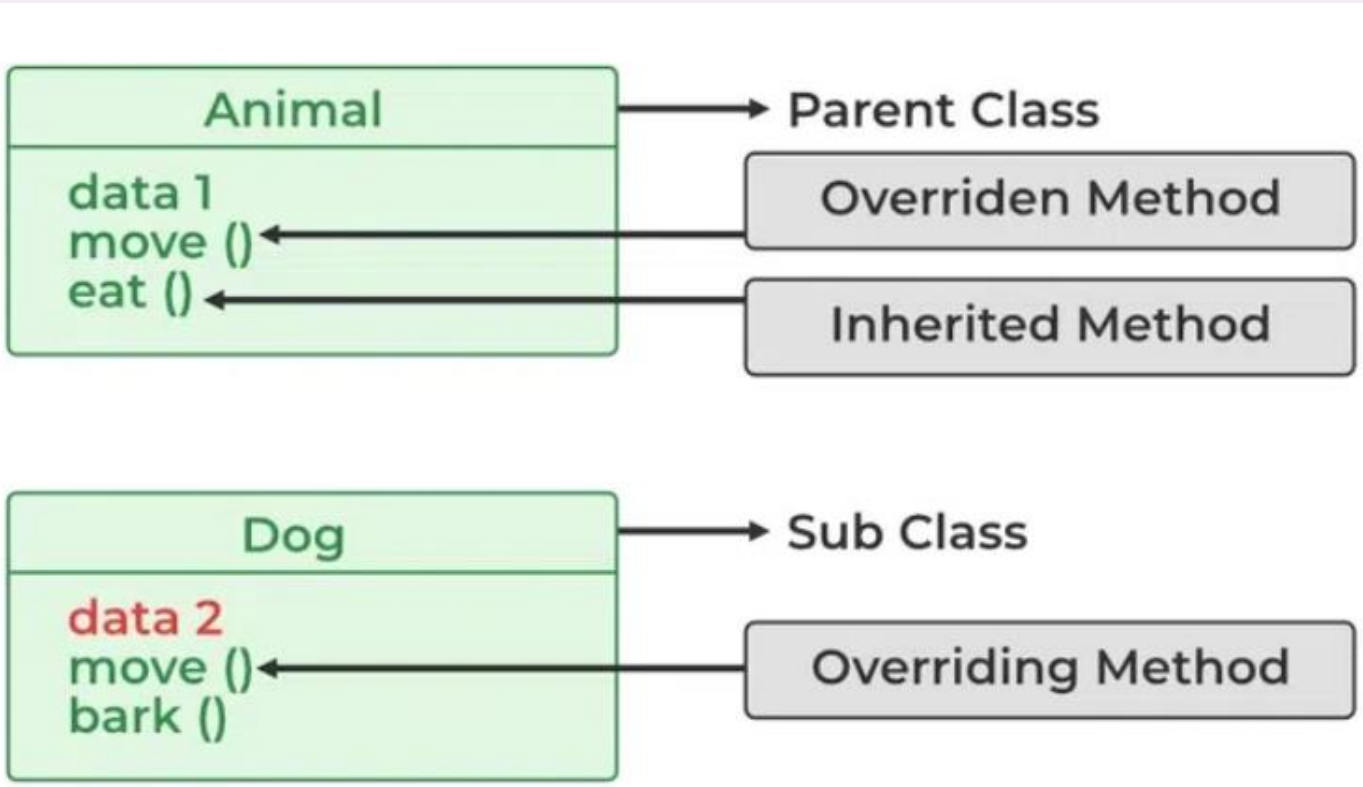
1. same name , same number and type of parameters, same return type

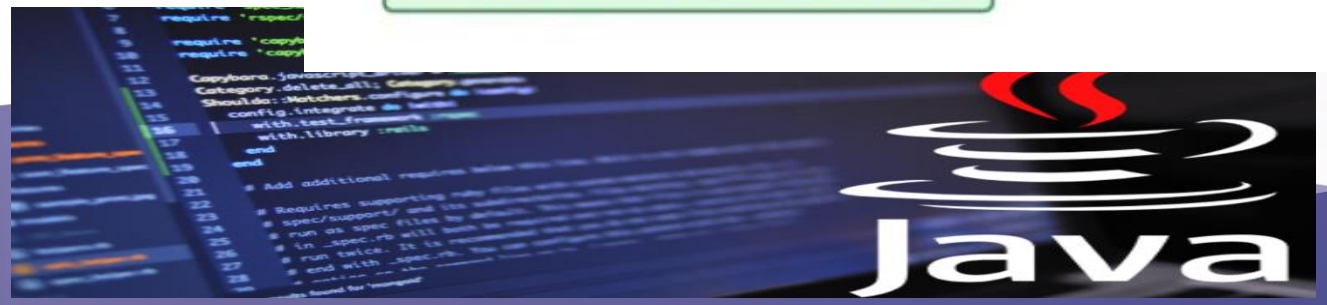
2. The access qualifier must be equal or more accessible than the method to override



Method Overriding in Java

Example



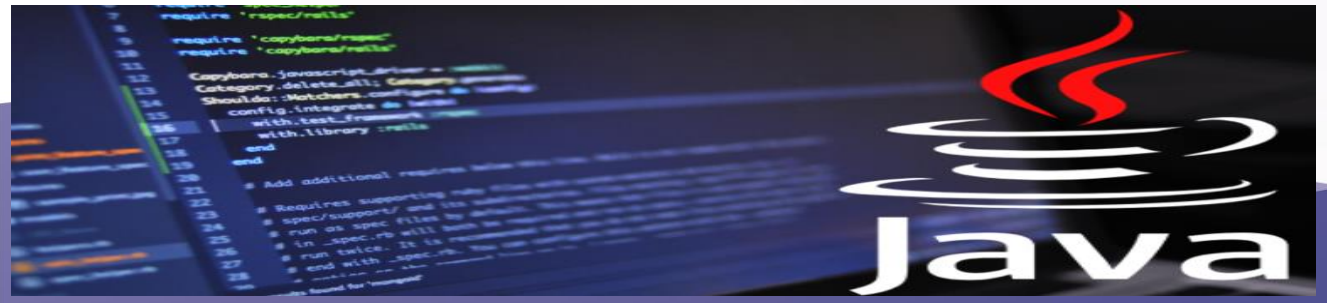


Method Overriding in Java

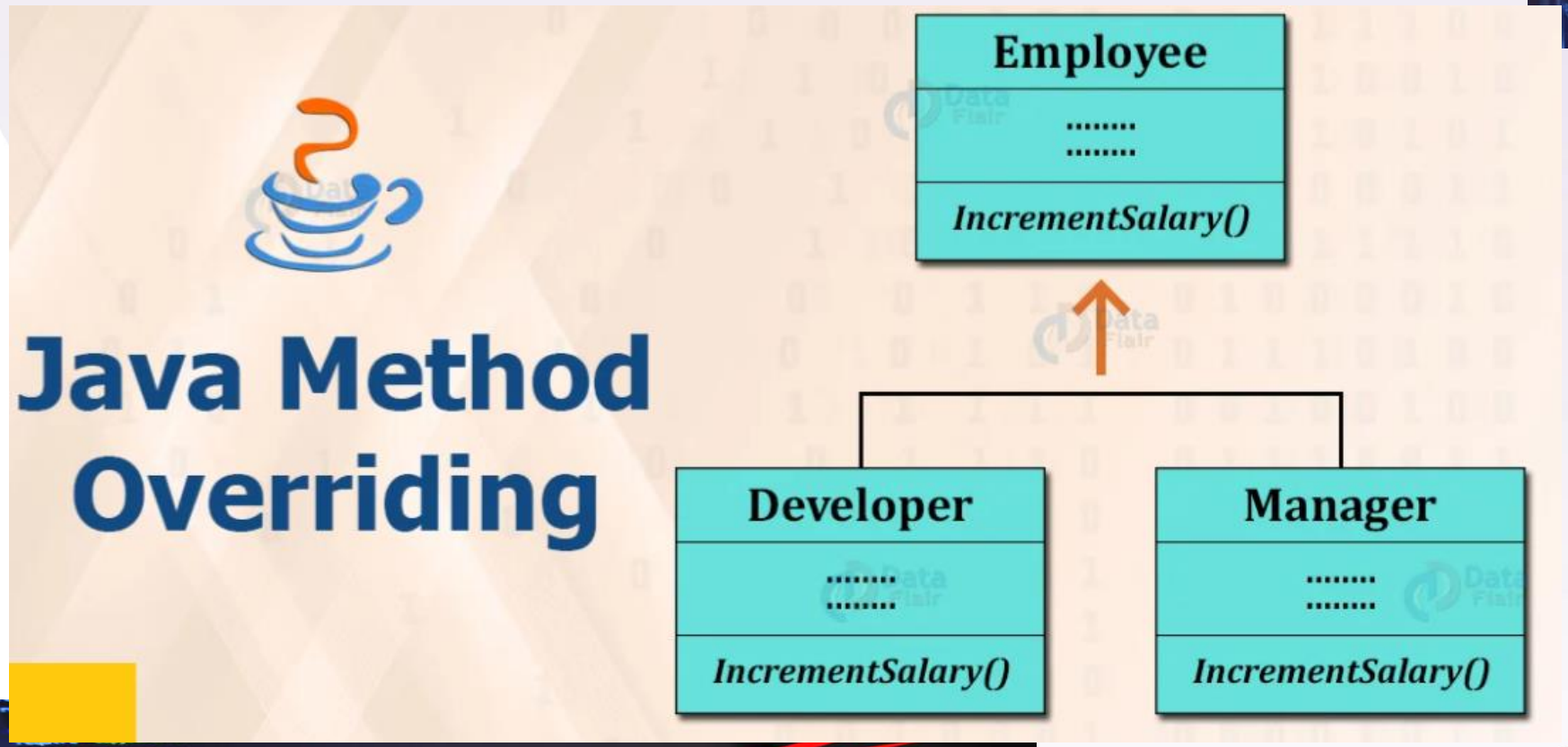
Q : Why Override a method?

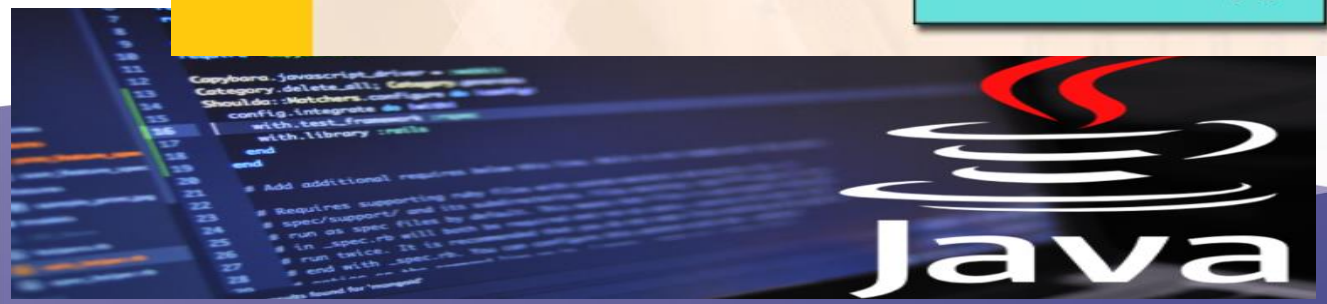
A : it is useful as some times the subclass needs to provide its own implementation of a method that is already present in the superclass

Usually we say that the subclass provides a "specialization" of the same method presents in the superclass, that means, it provide a different or more detailed implementation of such a method.



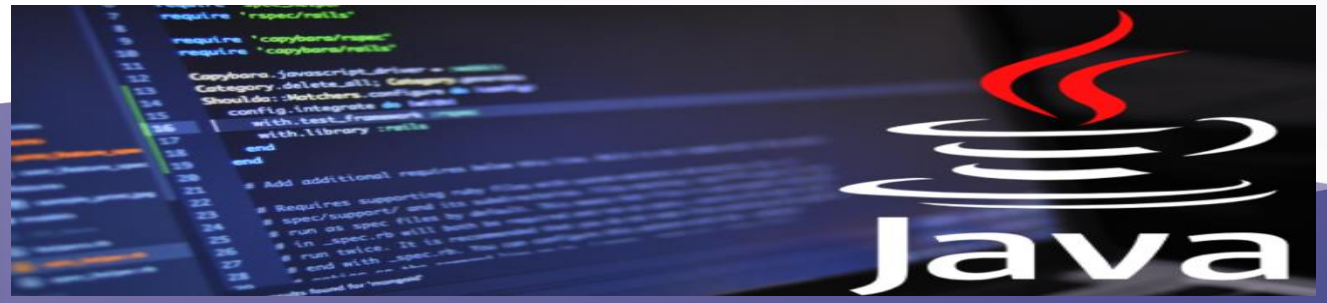
Method Overriding in Java





Method Overriding in Java

Lets take a simple example to understand this. We have two classes: A child class Boy and a parent class Human. The `Boy` class extends `Human` class. Both the classes have a common method `void eat()`. Boy class is giving its own implementation to the `eat()` method or in other words it is overriding the `eat()` method.



BASIS INTERNATIONAL™ SCHOOLS · CHINA
BASIS BILINGUAL™ SCHOOLS · CHINA

Method Overriding in Java


```java
class Human{
   //Overridden method
   public void eat()
   {
      System.out.println("Human is eating");
   }
}
class Boy extends Human{
   //Overriding method
   public void eat(){
      System.out.println("Boy is eating");
   }
   public static void main( String args[]) {
      Boy obj = new Boy();
      //This will call the child class version of eat()
      obj.eat();
   }
}
```

BASIS INTERNATIONAL™ SCHOOLS · CHINA
BASIS BILINGUAL™ SCHOOLS · CHINA

Method Overriding in Java


```java
// Parent Class
class Animal {
    void sound() {
        System.out.println("Animals make some sound");
    }
}

// Subclass 1
class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

// Subclass 2
class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}
```

```java
// Main Class
public class Main {
    public static void main(String[] args) {
        Animal myAnimal; // Reference of the parent type

        myAnimal = new Dog(); // Dog object
        myAnimal.sound(); // Calls Dog's sound()

        myAnimal = new Cat(); // Cat object
        myAnimal.sound(); // Calls Cat's sound()
    }
}
```

Polymorphism in Java

DEF: Polymorphism is based on two Greek words:

1. Poly ➔ many
2. Morphism ➔ forms

And it is the ability of an object to take on many forms depending on the contest

To better understand what Polymorphism is, let's have a look at the following example:

Take the example of a class Human and an object of this class named "Aaron". Aaron is a father and has two children.

He lives with his family in Delhi and works for DataFlair. Now, imagine what Aaron is to his wife. Yes, he is her husband.

Similarly, he is a father to his two children, a neighbor to his neighbors and a coworker to anyone who works with him at DataFlair.

This is polymorphism which means multiple forms.

Polymorphism in Java

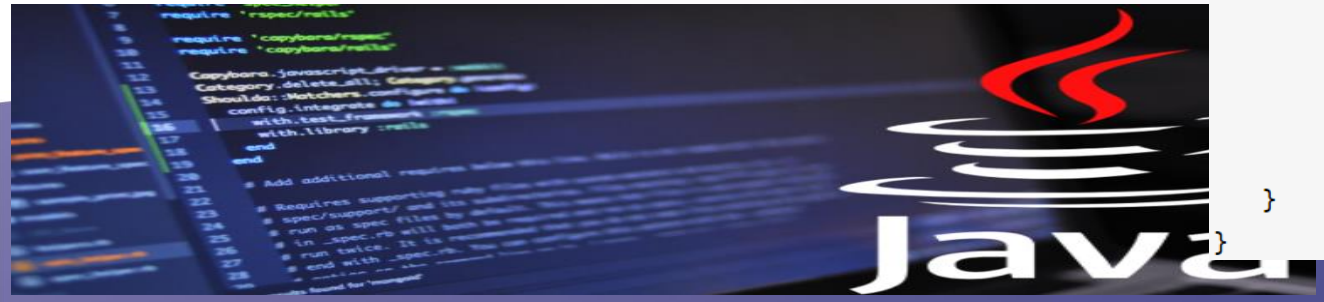
To understand this new concept, have a look at the following Java code:

```
class ABC{
   //Overridden method
   public void disp()
   {
        System.out.println("disp() method of parent class");
   }
}
class Demo extends ABC{
   //Overriding method
   public void disp(){
        System.out.println("disp() method of Child class");
   }
   public void newMethod(){
        System.out.println("new method of child class");
   }
   public static void main( String args[]) {
        /* When Parent class reference refers to the parent class object
         * then in this case overridden method (the method of parent class)
         *  is called.
         */
        ABC obj = new ABC();
        obj.disp();

        /* When parent class reference refers to the child class object
         * then the overriding method (method of child class) is called.
         * This is called dynamic method dispatch and runtime polymorphism
         */
        ABC obj2 = new Demo();
        obj2.disp();
   }
}
```

Output:

```
disp() method of parent class
disp() method of Child class
```



Method Overriding in Java

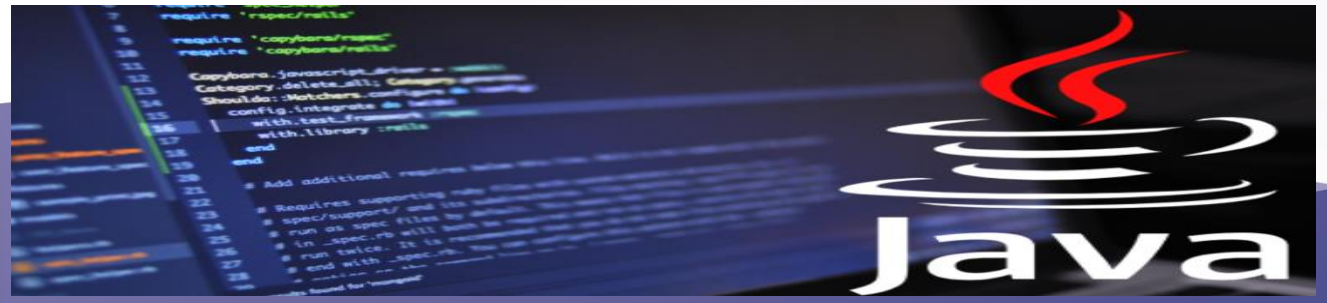
Note: In dynamic method dispatch the object can call the overriding methods of child class and all the non-overridden methods of base class but it cannot call the methods which are newly declared in the child class. In the above example the object `obj2` is calling the `disp()`. However if you try to call the `newMethod()` method (which has been newly declared in Demo class) using obj2 then you would give compilation error with the following message:

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem: The method xyz() is undefined for the type ABC
```

Method Overriding in Java

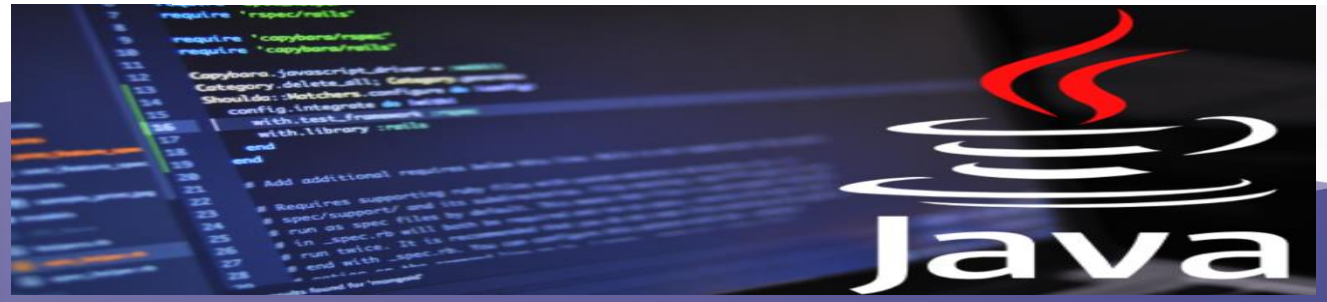
Rules of method overriding in Java

1. Argument list: The argument list of overriding method (method of child class) must match the Overridden method(the method of parent class). The data types of the arguments and their sequence should exactly match.
2. **Access Modifier** of the overriding method (method of subclass) cannot be more restrictive than the overridden method of parent class. For e.g. if the Access Modifier of parent class method is public then the overriding method (child class method) cannot have private, protected and default Access modifier,because all of these three access modifiers are more restrictive than public.



Method Overriding in Java

3. private, static and final methods cannot be overridden as they are local to the class. However static methods can be re-declared in the sub class, in this case the sub-class method would act differently and will have nothing to do with the same static method of parent class.
4. Overriding method (method of child class) can throw **unchecked exceptions**, regardless of whether the overridden method(method of parent class) throws any exception or not. However the overriding method should not throw **checked exceptions** that are new or broader than the ones declared by the overridden method. We will discuss this in detail with example in the upcoming tutorial.
5. Binding of overridden methods happen at runtime which is known as **dynamic binding**.
6. If a class is extending an **abstract class** or implementing an **interface** then it has to override all the abstract methods unless the class itself is a abstract class.



Method Overriding in Java

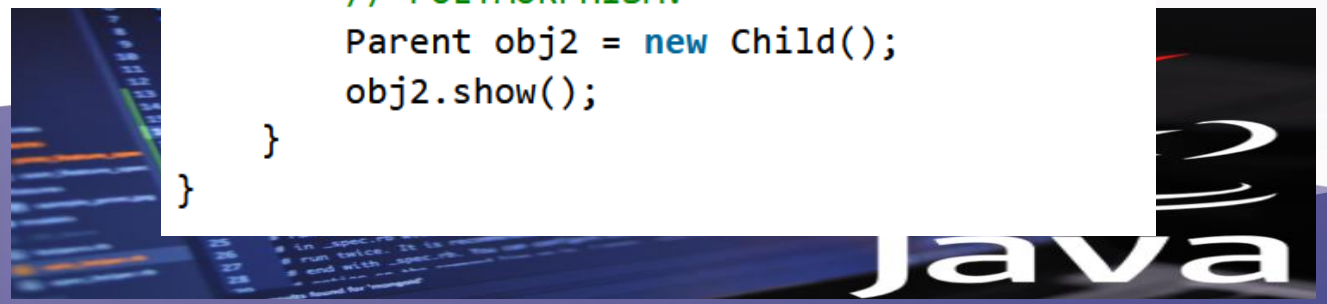
IMPORTANT : it is the type of the object created at run-time that determines what method will be executed .

```
class Main {
    public static void main(String[] args)
    {
        // If a Parent type reference refers
        // to a Parent object, then Parent's
        // show is called
        Parent obj1 = new Parent();
        obj1.show();

        // If a Parent type reference refers
        // to a Child object Child's show()
        // is called. This is called RUN TIME
        // POLYMORPHISM.
        Parent obj2 = new Child();
        obj2.show();
    }
}
```

Output

```
Parent's show()
Child's show()
```



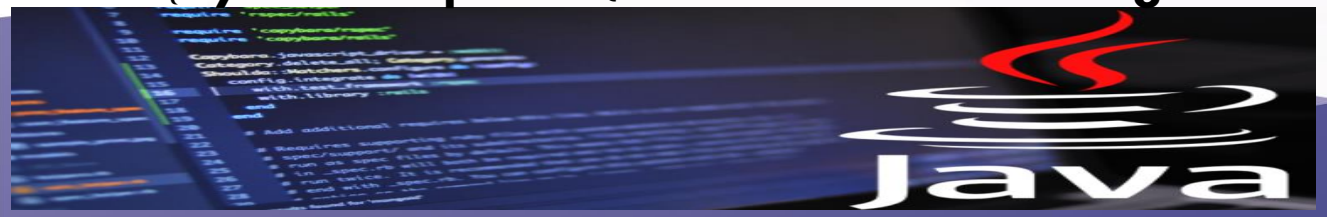
Method Overriding in Java

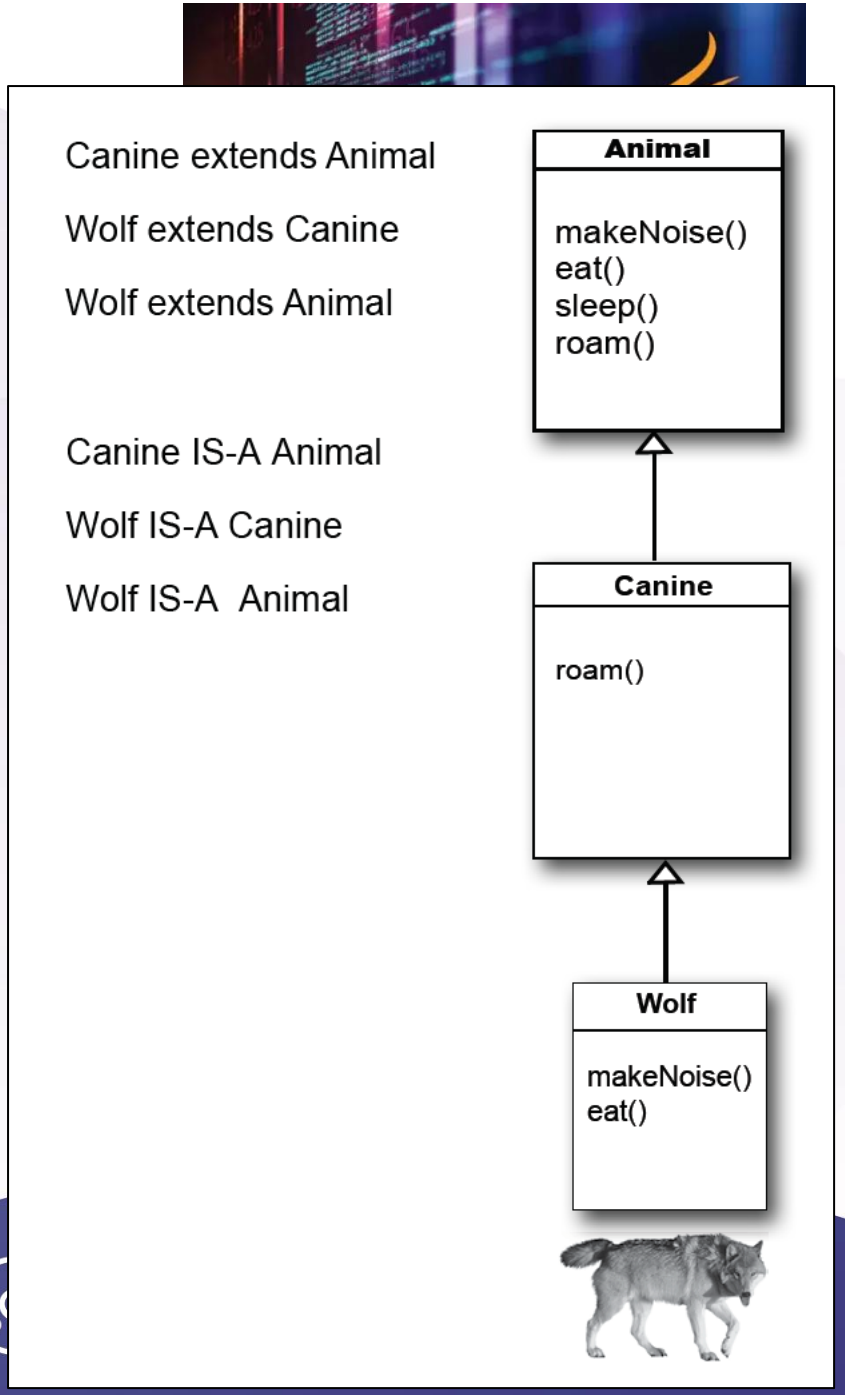
Exercise

1. Implement in Eclipse the classes and methods represented in the picture
2. For methods in Animal class implement the following sentence :
   makeNoise ()
   { System.out.println ("Animal class....makeNoise method");
   }
   sleep()
   {System.out.println ("Animal class ...sleeping");}

3. For method in **Wolf** class implement the following sentence:
   makeNoise ()
   {System.out.println ("Wolf class....makeNoise method");

4. For method in **Canine** class implement the following:
   roam()
   {System.out.println ("Canine class....walking around")}





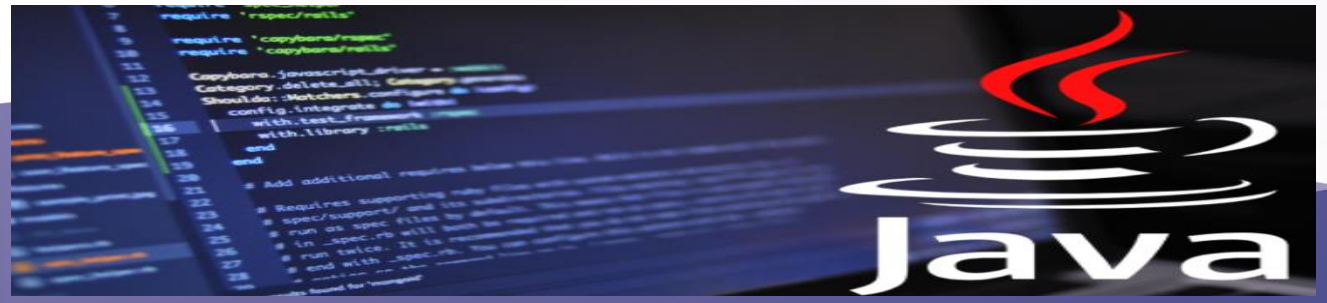
Method Overriding in Java

Exercise

4. Create a class named **TestingInheritance** with main() method. Inside it implement the following instruction:

```
Animal obj = new Wolf();
obj.makeNoise() ;
obj.eat();
obj.roam();
obj.sleep();
```

5. What are the results on your screen??
6. Can you explain these results?



Canine extends Animal

Wolf extends Canine

Wolf extends Animal

Canine IS-A Animal

Wolf IS-A Canine

Wolf IS-A Animal