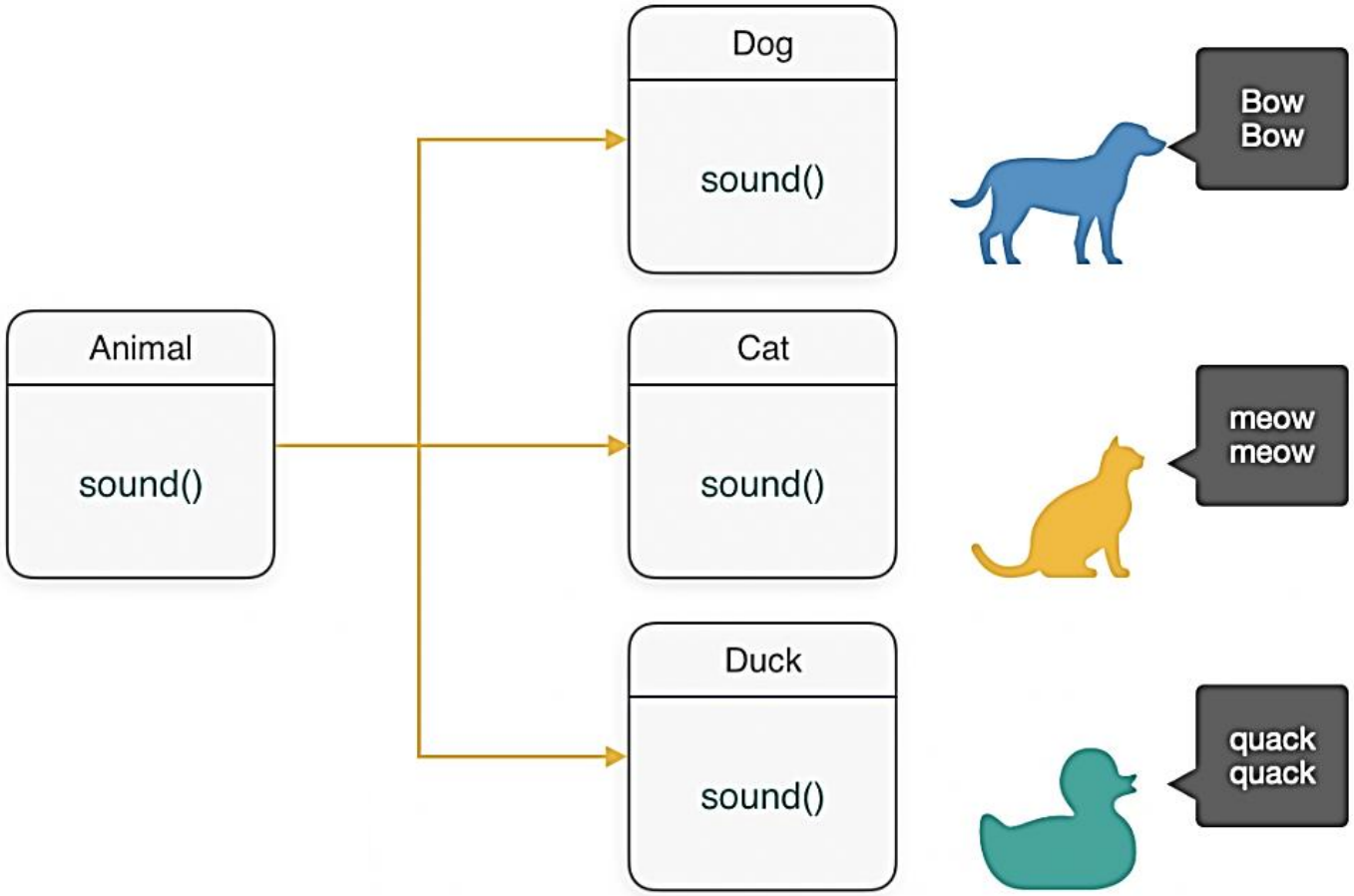
# POLYMORPHISM

## (METHOD OVERRIDING)



IN THE NAME OF ALLAH

بسم الله الرحمن الرحيم

THE GRACIOUS, THE MERCIFUL.

# GETS - GETTING EDUCATION WITH TECHNOLOGICAL SYSTEM

Instructor: Sir A.Rehman Ali Brohi

# LECTURE: 9
# Polymorphism in java

# What is Polymorphism?

**Poly** **Morphism**
**(many)** **(forms)**

**Shapes: circle, square, triangle etc.**
**Sound: bark, roar etc**
**Water: solid, liquid, gas**

polymorphism in Java allows us to perform the same action in many different ways

# Types of Polymorphism

1. Compile-time/Static Polymorphism

Method Overloading

Compiler handle it

1. Run-Time/Dynamic Polymorphism

Method Overriding

JVM handle it

In this exercise we will practice with method overriding

# CONDITIONS

## Method Overloading

1. Same Class
2. Same Method
3. Different Arguments
   - -No of Arg
   - -Seg of Arg
   - -Types of Arg

## Method Overriding

1. Same Method
2. Different Class
3. Same Arguments
   - -No of Arg
   - -Seg of Arg
   - -Types of Arg
4. Inheritance (IS-A)

Program to create same-method but different-in-class having same no-of-arg and seg-of-arg and types-of-arg with inheritance is known as method overriding

```java
public class Test {
    void show(){
        System.out.println("show method 1");
    }
}
class Test2 extends Test{
    void show(){
        System.out.println("show method 2");
    }
    public static void main(String[] args){
        Test ob1 = new Test();
        ob1.show();

        Test2 ob2 = new Test2();
        ob2.show();
    }
}
```

```
"C:\Users\Rehman Ali\.jdks\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Fi
show method 1
show method 2

Process finished with exit code 0
```

# Program with same No-of-Arguments

Main.java | Test.java

C:\Users\Rehman Ali\IdeaProjects\B
.idea
out
src
com.company
A.java
Test.java
B.iml
ernal Libraries
atches and Consoles

```java
public class Test {
    void show(int a){
        System.out.println("show method 1");
    }
}
class Test2 extends Test{
    void show(int a){
        System.out.println("show method 2");
    }
    public static void main(String[] args){
        Test ob1 = new Test();
        ob1.show( a: 10);

        Test2 ob2 = new Test2();
        ob2.show( a: 20);
    }
}
```

Test2

```
"C:\Users\Rehman Ali\.jdks\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Progra
show method 1
show method 2

Process finished with exit code 0
```

# Program with same Seg-of-Arguments

```java
public class Test {
    void show(String name, int age){
        System.out.println("show method 1");
    }
}
class Test2 extends Test{
    void show(String name, int age){
        System.out.println("show method 2");
    }
    public static void main(String[] args){
        Test ob1 = new Test();
        ob1.show( name: "rehman", age: 27);

        Test2 ob2 = new Test2();
        ob2.show( name: "Essa", age: 25);
    }
}
```

```
"C:\Users\Rehman Ali\.jdks\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Progr
show method 1
show method 2

Process finished with exit code 0
```

# Program with same Type-of-Arguments

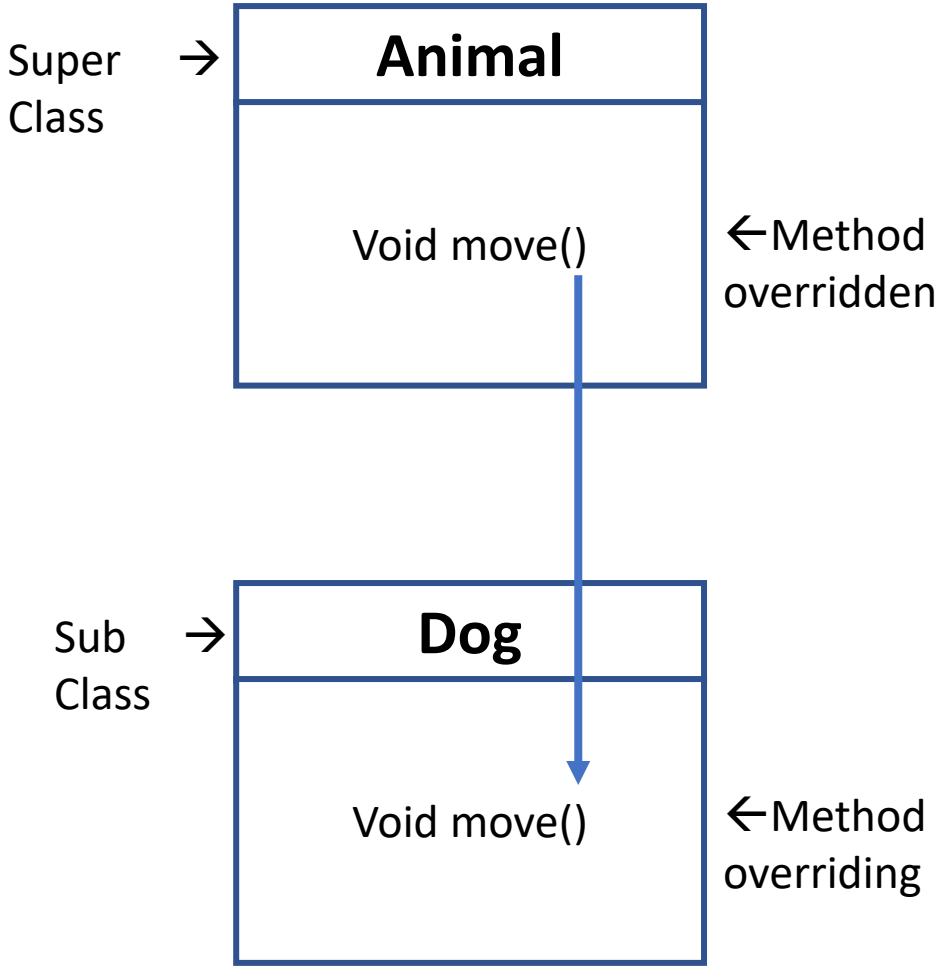
```java
public class Test {
    void show(int age, String name){
        System.out.println("show method 1");
    }
}
class Test2 extends Test{
    void show(int age, String name){
        System.out.println("show method 2");
    }
    public static void main(String[] args){
        Test ob1 = new Test();
        ob1.show( age: 27,  name: "rehman");

        Test2 ob2 = new Test2();
        ob2.show( age: 25,  name: "Essa");
    }
}
```

```
"C:\Users\Rehman Ali\.jdks\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Pr
show method 1
show method 2

Process finished with exit code 0
```

# Use of **Method Overriding**




***Method Overriding*** *allows a subclass child class to provide a* ***specific implementation*** *of a method that is already provided by one of its super-classes or parent classes.*

*The* ***implementation in the subclass overrides (replaces) the implementation*** *in the superclass by providing a method that has same, same parameters and same return type as the method in the parent class.*

## IMPORTANT NOTE (useful interview questions)

1. Can we achieve method overriding by changing return type of method only.
2. We will study more about exception handling in method overriding