Lab Number:	08
Student Name:	SHIVAM SHARMA
Roll No :	E-19

Title : Method overriding in java inheritance

Learning Objective:

- Ability to execute a simple method overriding in the java inheritance Java program with and without any inputs to the program.

Learning Outcome:

- Understanding method overriding in java inheritancejava programming

Theory:

Method Overriding in Java

1. Understanding the problem without method overriding
2. Can we override the static method
3. Method overloading vs. method overriding

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Usage of Java Method Overriding

- o Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
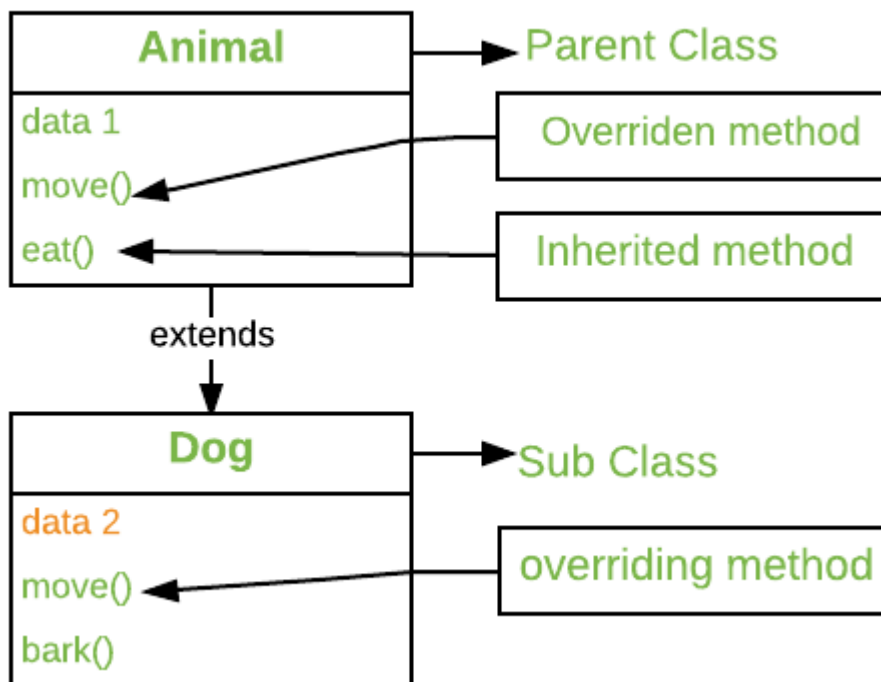- o Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

Faculty: Ms. Deepali Kayande

1. The method must have the same name as in the parent class

2. The method must have the same parameter as in the parent class

3. There must be an IS-A relationship (inheritance).

In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to *override* the method in the super-class.

Method overriding is one of the way by which java achieve Run Time Polymorphism.The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed. In other words, *it is the type of the object being referred to* (not the type of the reference variable) that determines which version of an overridden method will be executed.

Faculty: Ms. Deepali Kayande

Algorithm 1:

Step 1 : start
Step 2 :create class animal
Step 3 : method the superclass dog inherits animal
Step 4 : class dog extend animal
 Step 5 :Overriding the eat method
Step 6 : new method in subclass create an object of the subclass
Step 7 : call the eat () method
Step 8 : End

INPUT:

```java
class Animal {


  // method in the superclass
  public void eat() {
    System.out.println("I can eat");
  }
}


// Dog inherits Animal
class Dog extends Animal {


  // overriding the eat() method
  @Override
  public void eat() {
    System.out.println("I eat dog food");
  }


  // new method in subclass
  public void bark() {
    System.out.println("I can bark");
  }
```
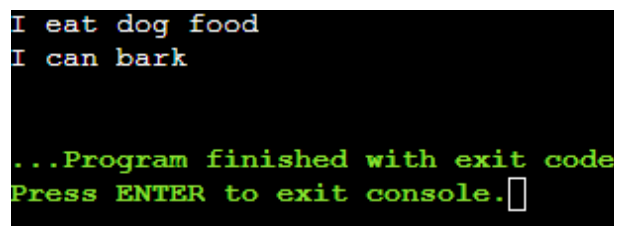
```java
}

class Main {
 public static void main(String[] args) {

   // create an object of the subclass
   Dog labrador = new Dog();

   // call the eat() method
   labrador.eat();
   labrador.bark();
 }
}
```

Algorithm 1:

OUTPUT:


```
I eat dog food
I can bark

...Program finished with exit code
Press ENTER to exit console.
```

Step 1 : start
Step 2 : create class animal
Step 3 : method in superclass dog inherits animal class

Step 4 : class dog extends animals
Step 5 : overriding the eat () method
Step 6 : call method of superclass
Step 7 : create an object of the subclass
Step 8 : call the eat () method
Step 9 : End

INPUT:

```java
class Animal {


  // method in the superclass
  public void eat() {
    System.out.println("I can eat");
  }
}


// Dog inherits Animal
class Dog extends Animal {


  // overriding the eat() method
  @Override
  public void eat() {


    // call method of superclass
    super.eat();
    System.out.println("I eat dog food");
  }


  // new method in subclass
  public void bark() {
```
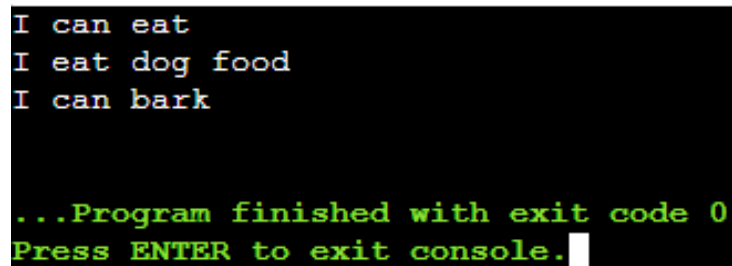
```java
 System.out.println("I can bark");
 }
}


class Main {
 public static void main(String[] args) {


  // create an object of the subclass
  Dog labrador = new Dog();


  // call the eat() method
  labrador.eat();
  labrador.bark();
 }
}
```

OUTPUT:

```
I can eat
I eat dog food
I can bark


...Program finished with exit code 0
Press ENTER to exit console.
```

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

Faculty: Ms. Deepali Kayande