RegNo: 23MCA1030 **Name**: Vinayak Kumar Singh

Java Programming Lab (PMCA502P)

Exercise 5F Dynamic method Disptach (Run Time polymorphism)

1.Write a Java program that demonstrates dynamic method dispatch. Define a superclass `Animal` with a method `makeSound()` that prints a generic sound. Create subclasses `Dog`, `Cat`, and `Bird` inheriting from `Animal`. Override the `makeSound()` method in each subclass to print the specific sound of each animal. Finally, create an array of `Animal` objects containing instances of `Dog`, `Cat`, and `Bird`, and iterate through the array to call the `makeSound()` method for each object.

Expected Output:

Woof!
Meow!
Chirp chirp!

Instructions:

Define the `Animal` class with a method `makeSound()` that prints a generic sound.
Define subclasses `Dog`, `Cat`, and `Bird` inheriting from `Animal`.
Override the `makeSound()` method in each subclass to print the specific sound of each animal.
In the `main` method, create an array of `Animal` objects containing instances of `Dog`, `Cat`, and `Bird`.

Iterate through the array and call the `makeSound()` method for each object, demonstrating dynamic method dispatch.

—----------

Modify the previous program to include parameters in the `makeSound()` method of the `Animal` class. Each subclass (`Dog`, `Cat`, `Bird`) should override the `makeSound()` method with its own specific sound and accept any necessary parameters. Demonstrate dynamic method dispatch by calling the `makeSound()` method with different parameters for each object in the array.

—----------

Code:

Part 1: Basic Dynamic Method Dispatch

```
// Define base class called Animal with method to make a sound
class Animal {
    public void makeSound() {
        System.out.println("Generic animal sound");
    }
}
// Define Dog subclass that inherits from Animal
class Dog extends Animal {
    @Override
    public void makeSound() {
```

```
        // Override makeSound method to print the specific sound of a dog
        System.out.println("Woof!");
    }
}
// Define Cat subclass that inherits from Animal
class Cat extends Animal {
    @Override
    public void makeSound() {
        // Override makeSound method to print the specific sound of a cat
        System.out.println("Meow!");
    }
}
// Define Bird subclass that inherits from Animal
class Bird extends Animal {
    @Override
    public void makeSound() {
        // Override makeSound method to print the specific sound of a bird
        System.out.println("Chirp chirp!");
    }
}
// Main class demonstrate dynamic method dispatch
public class Main {
    public static void main(String[] args) {
```

```
        System.out.println("Vinayak Kumar Singh 23MCA1030");
        // Create array of Animal objects, but store objects of different
subclasses
        Animal[] animals = { new Dog(), new Cat(), new Bird() };
        // Loop through array and call the makeSound method for each
animal
        for (Animal animal : animals) {
            animal.makeSound();
        }
    }
}
```

Output:

2.Dynamic Method Dispatch with Parameters

```
// Define base class called Animal with method to make sound with a
parameter
class Animal {
    public void makeSound(String sound) {
      System.out.println(sound);
    }
```

```
 }
 // Define Dog subclass that inherits from Animal
 class Dog extends Animal {
  @Override
  public void makeSound(String sound) {
    // Prepend specific dog sound to the parameter and call the parent's
makeSound
    super.makeSound("Woof! " + sound);
  }
 }
 // Define Cat subclass that inherits from Animal
 class Cat extends Animal {
  @Override
  public void makeSound(String sound) {
    // Prepend the specific cat sound to the parameter and call the
parent's makeSound
    super.makeSound("Meow! " + sound);
  }
 }
 // Define Bird subclass that inherits from Animal
 class Bird extends Animal {
  @Override
  public void makeSound(String sound) {
```

```
    // Prepend the specific bird sound to the parameter and call the
parent's makeSound
    super.makeSound("Chirp chirp! " + sound);
  }
}
// Main class demonstrate dynamic method dispatch with parameters
public class Main {
  public static void main(String[] args) {
    // Create array of Animal objects and store objects of different
subclasses
    Animal[] animals = {new Dog(), new Cat(), new Bird()};
    // Loop through the array and call the makeSound method with the
same parameter for each animal
    for (Animal animal : animals) {
      animal.makeSound("Additional sound");
    }
  }
}
```

Output: