Dynamic Method Dispatch in Java - Full Notes with Clear Code Formatting

1. Introduction
Dynamic Method Dispatch, also known as Runtime Polymorphism, is a feature in Java that allows a program to decide at runtime which version of an overridden method to execute. It is a key concept in Object-Oriented Programming (OOP) that supports flexibility and extensibility.

Example Code:

```java
// Superclass
class Animal {
    void sound() {
        System.out.println("Animal makes a 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");
    }
}

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

        // Superclass reference
        Animal ref;

        // ref refers to a Dog object
        ref = new Dog();
        ref.sound(); // Output: Dog barks

        // ref refers to a Cat object
        ref = new Cat();
        ref.sound(); // Output: Cat meows
    }
}
```

Explanation:
1. The superclass **Animal** defines the method **sound()**. 2. The subclasses **Dog** and **Cat** override this method with their own implementations. 3. A reference of type **Animal** can point to objects of either **Dog** or **Cat**. 4. The method to execute is determined at runtime based on the actual object. 5. This demonstrates **Runtime Polymorphism** in Java.