**File Name:** Java_Inheritance_Basic_Concepts.pdf

# Basic Concepts of Inheritance in Java

## 1. Introduction

Inheritance is one of the core concepts of Object-Oriented Programming (OOP). It allows a class (child/subclass) to acquire the properties and behaviors of another class (parent/superclass). This helps in:

- **Code reuse**
- **Easier maintenance**
- **Hierarchical modeling** of real-world entities

---

## 2. Key Terms

- **Parent Class / Superclass:** The class whose properties and methods are inherited.
- **Child Class / Subclass:** The class that inherits properties and methods from the parent class.
- `extends` **keyword:** Used in Java to establish an inheritance relationship.

---

## 3. Why Use Inheritance?

1. **Code Reusability:** Avoid rewriting the same code in multiple classes.
2. **Method Overriding:** Child class can modify behavior of parent class methods.
3. **Polymorphism Support:** Allows dynamic method invocation using parent reference.
4. **Hierarchical Modeling:** Represents real-world relationships (e.g., Vehicle → Car → ElectricCar).

---

## 4. Java Syntax for Inheritance

```
// Parent Class
class Animal {
    String name;

    void eat() {
        System.out.println(name + " eats food");
    }
}

// Child Class
class Dog extends Animal {
    void bark() {
```

```
        System.out.println(name + " barks");
    }
}

// Main Class
public class TestInheritance {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.name = "Tommy";
        d.eat();   // Parent class method
        d.bark();  // Child class method
    }
}
```

**Output:**

```
Tommy eats food
Tommy barks
```

---

## 5. Explanation

1. **Animal class** is the **parent class** with a method `eat()` and a variable `name`.
2. **Dog class** is the **child class** that **extends** `Animal`.
3. By extending, `Dog` automatically has access to `Animal`'s methods and variables.
4. We can also add new methods to the child class (`bark()` in this case).
5. The object `d` of class `Dog` can call **both parent and child methods**.

---

## 6. Access Modifiers and Inheritance

| Modifier | Access by Subclass? |
|---|---|
| public | Yes |
| protected | Yes (same package & subclass) |
| private | No |

```
class Animal {
    private int age;       // Not inherited
    protected String type; // Inherited
}
```

## 7. Points to Remember

1. A child class inherits all **non-private fields and methods** of parent class.
2. **Constructors are not inherited**, but parent constructor is called when child object is created.
3. A child class can **add new methods or variables** in addition to inherited ones.
4. **Single inheritance** is the simplest form: one child inherits from one parent.