- What is protected access modifier?
  - Scope: Visible within the same package and subclasses in other packages.

Commonly used in inheritance to provide controlled access to child classes.

```
package package01;

public class ProtectedClass {

    protected String protectedVariable = "I am protected";

    protected void display(){
        System.out.println(protectedVariable);
    }
}



package package02;

import package01.ProtectedClass;

public class TestProtected extends ProtectedClass {
    public static void main(String[] args) {
        TestProtected obj = new TestProtected();
        obj.display();  // Accessible through inheritance
    }
}
```

What is default access modifier(No keyword)

Scope: Restricted to the same package. Also “package-private” access.

Modifiers	Same Class	Same Package	Sub Class(Different Package)	Other packages
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

Abstraction:

Hides the implementation details from user and exposes only the essential functionalities.

Example: A car's interface for driving hides the internal structure/mechanism of its engine.

Inheritance:

It enables a child class to inherit properties and methods from parent class.

```
package inheritanceExample;

public class Vehicle {
    String brand = "Ford";
}
```

```
package inheritanceExample;

public class Car extends Vehicle{
    String model = "Mustang";
}


package inheritanceExample;

public class InheritanceEx {
    public static void main(String[] args) {
        Car myCar = new Car();
        System.out.println(myCar.brand + " " + myCar.model);
    }
```

```
}
```

Polymorphism:

Allows methods or objects to behave differently based on the context.

1. Compile Time Polymorphism (Method Overloading)

Multiple methods with the same name but different parameters.

```
package polymorphismExample;

public class Calculator {

    int add(int a, int b){
        return a+b;
    }

    double add(double a, double b){
        return a+b;
    }
}
```

2. Runtime Polymorphism(Method Overriding)

```
package runtimeExample;

public class Animal {
    void sound(){
        System.out.println("Animal Makes a sound");
    }
}


package runtimeExample;

public class Dog extends Animal{
    @Override
    void sound() {
        System.out.println("Dog Barks!!!");
    }
}
```

Benefits of OOPs:

1. Reusability
2. Modularity
3. Maintainability
4. Scalability

Inheritance:

It is the process by which one class acquires the properties(fields) and functionalities(methods) of another class.
It allows the concept of reusability.

Types of inheritance:

1. Single Inheritance
   - A child class inherits from a single parent class.

```java
package singleInheritanceExample;

public class Parent {

    void showMessage(){
        System.out.println("Parent Class");
    }
}
```

```java
package singleInheritanceExample;

public class Child extends Parent{
    void display(){
        System.out.println("Child Class");
    }
}
```

```java
package singleInheritanceExample;

public class Main {
    public static void main(String[] args) {
        Child obj = new Child();

        obj.display();
        obj.showMessage();
    }
}
```

o/p:
Child Class
Parent Class

Explanation: A child class inherits showMessage method from parent class, allowing access to both parent and child methods.

2. Multilevel Inheritance

A class is derived from another derived class(forms a chain of inheritance)
TASK:
class GrandParent-> create method message()
class Parent -> extends Grandparent -> method showMessage()
class Child -> extends Parent -> method display()
Main Class -> main method -> Create child class object and call methods

3. Hierarchical Inheritance:

Multiple child classes inherit from a single parent class.