Lab 29

Explain encapsulation in java. Explain access modifiers.

Encapsulation is one of the four fundamental concepts of object-oriented programming (OOP), along with inheritance, polymorphism, and abstraction. It is a mechanism that restricts direct access to some of an object's components (attributes or methods), encapsulating the internal state of an object and protecting it from unauthorized access and modification. Encapsulation is achieved by using access modifiers.

Access Modifiers in Java:

Access modifiers are keywords in Java that define the visibility or accessibility of classes, methods, and variables. There are four main types of access modifiers in Java:

1. Public (`public`): The member is accessible from any other class.

2. Private (`private`): The member is accessible only within its own class.

3. Protected (`protected`): The member is accessible within its own class, its subclasses, and classes in the same package.

4. Default (no modifier): Also known as package-private, the member is accessible only within classes in the same package. If no access modifier is specified, it defaults to package-private.

Encapsulation Using Access Modifiers:

Encapsulation is achieved in Java by using access modifiers to control the visibility of classes, methods, and variables.

1. Private Access Modifier:

The `private` access modifier is used to restrict access to members (variables and methods) to within the same class. This means that these members can only be accessed or modified from within the class in which they are declared.

public class Student {

private String name; // Private variable

public void setName(String newName) {

name = newName;

}

public String getName() {

return name;

}

}

In this example, the `name` variable is declared as `private`, so it cannot be directly accessed or modified from outside the `Student` class. We provide public methods (`setName` and `getName`) to access and modify the `name` variable in a controlled manner.

2. Public Access Modifier:

The `public` access modifier allows members to be accessed from any class, whether it's in the same package or a different one.

public class Circle {

public double radius; // Public variable

public double calculateArea() { // Public method

return Math.PI \* radius \* radius;

}

}

In this example, both the `radius` variable and the `calculateArea` method are declared as `public`, so they can be accessed from anywhere in the code.

3. Protected and Default Access Modifiers:

Protected members can be accessed within the same class, its subclasses, and classes in the same package. Default members (no modifier) can be accessed only within classes in the same package.

package mypackage;

public class Animal {

protected String name; // Protected variable

String color; // Default (package-private) variable

}

In this example, the `name` variable is `protected`, making it accessible to subclasses and classes within the same package. The `color` variable is not explicitly marked with an access modifier, making it package-private and accessible within classes in the `mypackage` package.

Benefits of Encapsulation:

- Control: It provides control over the access to class members, allowing you to specify which parts of your class can be accessed and modified.

- Flexibility: You can change the internal implementation of a class without affecting the code that uses the class, as long as the public interface remains the same.

- Security: Encapsulation helps protect the integrity of an object's state by preventing unauthorized access and modification.

Encapsulation and access modifiers are essential principles in Java programming that contribute to code organization, maintainability, and security. They allow you to build robust and maintainable software systems.