10.4 - Abstract Class and Anonymous Inner Class

Introduction:

In Java, an **anonymous inner class** allows you to create an instance of a class with new behaviour without having to explicitly define a subclass. This is particularly useful when you need to modify or extend the functionality of an existing class only once, saving the effort of defining a new class.

On the other hand, an **abstract class** is a class marked with the abstract keyword, which means:

- It may contain both **abstract methods** (without implementations) and **concrete methods** (with implementations).
- You **cannot create an object** of an abstract class directly.

But can we combine the use of abstract classes with anonymous inner classes? The answer is yes! Let's explore this combination with an example.

Example: Combining Abstract Class and Anonymous Inner Class

```java
abstract class A {
    // Abstract method
    public abstract void show();
}

public class Demo {
    public static void main(String[] args) {
        A obj = new A() {
            // Implementation for abstract method in anonymous inner class
            @Override
            public void show() {
                System.out.println("In new show");
            }
        };
        obj.show(); // Calling the overridden method
    }
}
```

Output:

```
In new show
```

Explanation:

In this example:

- A is an **abstract class** that contains an abstract method show().
- Even though you cannot create an object of A directly because it is abstract, we are able to provide an implementation of the show() method using an **anonymous inner class**.
- The anonymous inner class gives us the ability to **override the abstract method** and provide a new implementation.

This demonstrates that:

- **Anonymous inner classes** can implement abstract methods, even for abstract classes.
- You can combine both concepts to dynamically implement abstract methods without having to define a separate subclass.

Additionally, this approach works for **multiple abstract methods** in an abstract class, allowing you to implement several methods within an anonymous inner class.

Key Points to Remember:

- **Abstract classes** can’t be instantiated directly, but **anonymous inner classes** allow you to create an instance that implements abstract methods.
- This combination is particularly useful when you need a **one-time implementation** of an abstract class without creating a named subclass.
- The implementation of abstract methods inside an anonymous inner class is done inline and is **not reusable** elsewhere, making it ideal for short, specific tasks.