Java Closure & Lambda Expression

CSE-220

Covered Topics

- Inner Class
- Closure
- Lambda Expression
- Final Keyword
- Java Package

Inner Class

```
class A {

    class B{

            }
  }
```

Types of Inner Class

1. Member inner class
2. Static Inner class
3. Anonymous inner class

Anonymous Inner Class

It is an inner class without a name and for which only a single object is created

Syntax:

```
Test t = new Test()
{
   // data members and methods
   public void test_method()
   {
      ........
      ........
    }
};
```

Anonymous Inner Class Example

```
public class Anno_main {

    public static void main(String[] args) {
        Display D=new Display()
                {
            public void show() {
                System.out.println("CSE19");
                }
                };

                D.show();
    }
}

interface Display{
     public void show();
    }
```

Closure

- A closure is a block of code that can be referenced (and passed around) with access to the variables of the enclosing scope.
- A closure gives us access to the outer function from an inner function.

Example:

Lambda Expression

- Enable *functional Programming.*
- Readable and concise code.
- Easier to use APIs and libraries.

Lambda Expression

- *A **lambda** is just an anonymous function - a function defined with no name and doesn't belong to any class.*
- A lambda expression basically expresses instances of the ***functional interfaces***

//Syntax of lambda expression
(parameter_list) -> {function_body}

Example:
()-> System.out.println("Hello World");

(x, y) -> x + y
// Takes two arguments x and y and returns the sum of these.

Lambda expression vs method

Main Parts	Lambda	Method
Name	-	P
Parameter List	P	P
Body	P	P
Return Type	P	P

Why need Lambda Expression?

- In OOP everything is an Object
- All code blocks are associated with classes and objects.
- What to do if I need a action/logic, not the entire class?

Example:

Lambda Expression

int i=3;
float j= 45.67;

detailed_info ={

}

Lambda Expression

```
detailed_info =public void display() {
                    System.out.println("Hello World");
                     }
```

Lambda Expression

```
detailed_info = public void display() {
                    System.out.println("Hello World");
                      }
```

```
detailed_info = () {
                    System.out.println("Hello World");
                      }
```

Lambda Expression

```
detailed_info = () -> {
                    System.out.println("Hello World");
                      }
```

Final Keyword

- Restrict the user
- Define an entity that can only be assigned once

Used in:

- Variable (Create constant variable)
- Class (Prevent inheritance)
- Method (Prevent method overriding)

Final Variable

- Final variable
- Blank final variable
- Static blank final variable

Final Method

- Can be inherited but does not overridden.

Example:

Final Class

- To prevent inheritance
- Create immutable class

Example:

How to create Immutable Class?

Use the **`final`** keyword

```
public final class Student {

    final String name;
    final int roll;

    public Student(String name, int roll)
    {
        this.name = name;
        this.roll = roll;
    }
    public String getName()
    {
        return name;
    }
    public int getRegNo()
    {
        return roll;
    }

}
```

- **Class** must be declared as **final**
- Data **members** inside the class must be declared as **final**
- A parameterized constructor
- Getter method for all the variables in it
- No setter.

How to create Immutable Class?

Use the `final` keyword

```
public class Test {

    public static void main(String[] args) {
        Student s = new Student("ABC", 101);
        System.out.println(s.getName());
        System.out.println(s.getRegNo());
    }
}
```

How to create Immutable Class?

Use the `final` keyword

```
public final class Student {

    final String name;
    final int roll;

    public Student(String name, int roll)
    {
        this.name = name;
        this.roll = roll;
    }
```

The above class is immutable because:

- The instance variable of the class is final i.e. we cannot change the value of it after creating an object.
- The class is final so we cannot create the subclass.
- There is no setter methods i.e. we have no option to change the value of the instance variable.

How to create Immutable Class?

Use the `final` keyword

Further study:

https://www.javatpoint.com/final-keyword

Java Package

Package in Java is a mechanism to encapsulate a group of **classes**, **sub packages** and **interfaces.**

- Preventing naming conflicts
- Searching makes easier.
- Providing controlled access
- Packages can be considered as data encapsulation/ data hiding

Java Package

Two types of packages in java:

- Built in Package
- User defined Package

Built in Java Package

Example:

```
import java.Util.Scanner;
import java.Util.*;
```

Commonly used built in java package:

- ✓**java.lang**
- ✓**java.io**
- ✓ **java.util**
- ✓**java.applet**
- ✓**java.awt**
- ✓**java.net**

User Defined Java Package

```
import   package_name.*;
import package_name.class_name;
```

Example:

Thank You