Lab 03

Course Code: CSE-124

Course Title**:** OOP Lab

Submitted To: Golam Daiyan

Submitted By:

Name: Abdur Rahim

ID: **231001422E**

Department: ECSE

Semester: 3DCSE (EVENING)

Submitted Date: 17/11/2023

Assignment Name: Java Packages / APIJava InheritanceJava PolymorphismJava Inner ClassesJava AbstractionJava InterfaceJava EnumsJava User InputJava DateJava ArrayListJava LinkedListJava HashMapJava HashSetJava IteratorJava Wrapper ClassesJava ExceptionsJava RegExJava ThreadsJava Lambda

Java Packages / API:

Definition:

Packages in Java are used to organize classes into namespaces. They help in avoiding naming conflicts and provide a modular structure to the code. The Java API (Application Programming Interface) is a collection of classes and methods that provide pre-built functionality for common tasks.

Java Inheritance:

Definition:

Inheritance in Java is a mechanism where a class (subclass or derived class) inherits properties and behaviors from another class (superclass or base class). This promotes code reuse and establishes a relationship between classes.

Java Polymorphism:

Definition:

Polymorphism in Java allows a single entity (such as a method or object) to take on multiple forms. There are two types of polymorphism in Java: compile-time polymorphism (achieved through method overloading) and runtime polymorphism (achieved through method overriding).

Java Inner Classes:

Definition:

Inner classes in Java are classes defined inside another class. They have access to the enclosing class's members and can be used for encapsulation and organization. Inner classes can be static or non-static, and they are often used in event handling.

Java Abstraction:

Definition:
Abstraction in Java is a process of hiding the implementation details and showing only the essential features of an object. Abstract classes and interfaces are used to achieve abstraction. Abstract classes may have abstract methods (methods without a body), and interfaces define a contract for implementing classes.

Java Interface:

Definition:
An interface in Java is a collection of abstract methods. It provides a way to achieve full abstraction and multiple inheritance. Classes implement interfaces by providing concrete implementations for the abstract methods declared in the interface.

Java Enums:

Definition:
Enums (enumerations) in Java are a special type used to define a collection of constants. Enumerations are often used to represent a fixed set of values, like days of the week or colors.

Java User Input:

Example:

```
import java.util.Scanner;


public class UserInputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a number: ");
        int number = scanner.nextInt();
        System.out.println("You entered: " + number);
    }
```

```
}
```

Java Date:

Example:

```
import java.util.Date;


public class DateExample {
    public static void main(String[] args) {
        Date currentDate = new Date();
        System.out.println("Current Date and Time: " + currentDate);
    }
}
```

Java ArrayList:

Example:

```
import java.util.ArrayList;
public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        System.out.println("Names: " + names);
    }
}
```

Java LinkedList:

Example:

```
import java.util.LinkedList;

public class LinkedListExample {
    public static void main(String[] args) {
        LinkedList<Integer> numbers = new LinkedList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        System.out.println("Numbers: " + numbers);
    }
}
```

Java HashMap:

Example:

```
import java.util.HashMap;

public class HashMapExample {
    public static void main(String[] args) {
        HashMap<String, Integer> ages = new HashMap<>();
        ages.put("Alice", 25);
```

```
        ages.put("Bob", 30);
        ages.put("Charlie", 35);
        System.out.println("Ages: " + ages);
    }
}
```

Java HashSet:

Example:

```
import java.util.HashSet;

public class HashSetExample {
    public static void main(String[] args) {
        HashSet<String> uniqueNames = new HashSet<>();
        uniqueNames.add("Alice");
        uniqueNames.add("Bob");
        uniqueNames.add("Charlie");
        System.out.println("Unique Names: " + uniqueNames);
    }
}
```

Java Iterator:

Example:

```
import java.util.ArrayList;
import java.util.Iterator;
```

```
public class IteratorExample {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<>();
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");

        Iterator<String> iterator = colors.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Java Wrapper Classes:

Definition:
Wrapper classes in Java are used to convert primitive data types into objects. They provide methods to work with the values of primitive types as objects.

Java Exceptions:

Definition:
Exceptions in Java represent errors or exceptional events during the execution of a program. Exception handling is done using try, catch, and finally blocks.

Java RegEx (Regular Expressions):

Definition:
Regular expressions in Java provide a powerful way to search, match, and manipulate strings. The `java.util.regex` package is used for working with regular expressions.

Java Threads:

Definition:

Threads in Java are lightweight processes that execute independently. Multithreading is a way to execute multiple threads concurrently. The `Thread` class or the `Runnable` interface is used for creating and managing threads.

Java Lambda:

Definition:

Lambda expressions in Java provide a concise way to express instances of single-method interfaces (functional interfaces). They are used to enable functional programming features in Java.