ADHIYAMAAN COLLEGE OF ENGINEERING
(Autonomous), Hosur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(Accredited by NBA)

322CITO4 - OBJECT ORIENTED PROGRAMMING SYSTEM

(REGULATION-2022)

STAFF INCHARGE **HOD**

Vision of the Institute

To foster ACE as a centre for nurturing and developing world class Engineers and Managers who convert global challenges into opportunities through value-based quality education.

Mission of the Institute

M 1 : To impart value-based quality education through effective teaching-learning processes

M 2 : To nurture creativity, excellence and critical thinking by applying global competency factors to contribute and excel in the rapidly growing technological world.

M 3 : To continuously develop and improve holistic and innovative personality for global Mobility.

M 4 : To make ACE a centre for excellence.

Vision of the Department

To empower young minds to become resilient professionals, instilled with ethical principles and equipped with cutting-edge technologies to meet the evolving demands of the world

Mission of the Department

M 1 : To empower individuals with a comprehensive understanding of computer engineering principles and its applications through effective teaching and learning practices.

M 2 : To cultivate excellence and critical thinking, while leveraging global competency, thus enabling significant contributions to societal challenges in the fast-paced technological landscape.

M 3 : To facilitate the students to work with modern tools and technologies to foster innovation, a zest for higher studies and to build leadership qualities by inculcating the spirit of ethical values.

Program Educational Objectives (PEOs)

PEO1 : The graduates will have sound knowledge in Mathematics, Science and Engineering concepts necessary to formulate, analyse, design and solve Engineering problems and to prepare them for higher learning, research and industry.

PEO2 : The graduates will possess innovative skills to assess and apply the rapid changes in technology and to engage in research leading to novel solutions for human, social and global competency.

PEO3 : The graduates will acquire knowledge and grab opportunities to work as teams in a multidisciplinary environment, communicate ideas effectively with diverse audiences demonstrate leadership qualities with ethical values and engage in lifelong learning.

INDEX

S.NO	DATE	NAME OF THE EXPERIMENT	SIGNATURE	PAGE. NO
1		Implement class, method and static member.		
2		Implement Inheritance (overriding).		
3		Implement packages and interfaces.		
4		Implement method and constructor overloading.		
5		Implement method and constructor overloading.		
6		Write a java program that implements a multi-threaded application		
7		Implement collection class (Arraylist,Map,Set).		
8		Write a program to perform file operations.		
9		Implement string handling functions.		
10		Develop an application using Java concepts, JavaFX controls, layouts and menus With Database Connectivity.		

EXNO:1(A) **IMPLEMENT CLASS, METHODS AND STATIC MEMBER**

DATE:

AIM:

To write a Java program using class, object and static method (Student details).

ALGORITHM:

Step 1: Start the program.

Step 2: Declare a class Student which has rollno, name, college as its element.

Step 3: The main method is created and object from Student class is allocated with suitable information of the Student.

Step 4: Then the static function is used to display the display the welcome message without creating the object.

Step 5: The print statement is executed through the main method.

Step 6: Stop the program.

PROGRAM:

```
class Student
{
int rollno;
string name;
static String college="ACE";
Student(int r, String n)
{
rollno=r;
name=n;
}
static void hello()
{
System.out.println("Welcome to Adhiyamaan college");
}
void display()
{
System.out.println(rollno+"."+name+"."+college);
}
public static void main(String args[])
{
Student s1=new Student(10,"Arun");
Student s2=new Student(20,"Kumar");
s1.display();
s2.display();
hello();
}
}
```

OUTPUT:

Welcome to Adhiyamaan college
10.Arun.ACE
20.Kumar.ACE

RESULT:

Thus, a Java program using class, object and static method (Student details) is executed.

EXNO:1(B) **IMPLEMENT CLASS, METHODS AND STATIC MEMBER**

DATE:

AIM:

To write a Java program using class, object and static method (employee details).

ALGORITHM:

Step 1: Start the program.

Step 2: Declare a class employee which has employee Id, name, phon no, address as its element.

Step 3: The main method is created and object from employee class is allocated with suitable information of the employee.

Step 4: Then the static function is used to display the output without creating the object.

Step 5: The print statement is executed through main method.

Step 6: Stop the program.

PROGRAM:

```
class Employee
{
static int id;
String name;
String address;
String phno;
static void emp()
{
System.out.println("employee id:"+id+"\n");
}
public static void main(String args[])
{
Employee emp1=new Employee();
emp1.id=1689;
emp1.phno="9999988888";
emp1.address="Hosur";
emp1.name="John";
emp1.emp();
System.out.println("employee.name:"+ emp1.name+"\n");
System.out.println("employee.phno:"+ emp1.phno+"\n");
System.out.println ("employee.address:"+ emp1.address+"\n");
}
}
```

OUTPUT:

employee id:1689
employee.name:John
employee.phno:9999988888
employee.address:Hosur

RESULT:

To write a Java program using class, object and static method (employee details) executed.

EX NO:2 **IMPLEMENT INHERITANCE (OVERRIDING).**

DATE:

AIM:

To write Java program for finding area of shapes using Inheritance.

ALGORITHM:

Step 1: Start the program

Step 2: Create a superclass for inherit the property to subclass.

Step 3: Declare different methods to find
the area of different shapes.

Step 4: To get user input import java scanner package and create a object for it

Step 5: Using switch statement get the input from the user for different shapes you want.

Step 6: By passing the given parameters to the methods to print the area of shapes.

Step 7: Stop the program.

PROGRAM:

```
import java.util.Scanner;
class Area{
int length;
int breadth;
int height;
int base;
int radius;
void rectangle ()
{
System.out.println(length*breadth );
}
void triangle ()
{
System.out.println(base*height*0.5);
}
void circle ()
{
System.out.println(3.142*radius*radius);
}
static class Output extends Area{
void start(){
Scanner input=new Scanner(System.in);
Output op=new Output();
System.out.println("Find area:\n 1.rectangle\n2.triangle\n3.circle");
int shape= input.nextInt();
switch (shape)
{
case 1:
System.out.println("enter length:");
int l= input.nextInt();
op.length=l;
System.out.println("enter breadth:");
int b= input.nextInt();
op.breadth=b;
op.rectangle();
break;
case 2:
System.out.println("enter base:");
int ba= input.nextInt();
op.base=ba;
System.out.println("enter base:");
int h=input.nextInt();
op.height=h;
op.triangle();
break;
case 3:
System.out.println("enter radius:");
```

```
int r= input.nextInt();
op.radius=r;
op.circle();
break;
default:
System.out.println("invalid input");
break;
}
}
}
public static void main(String args[])
{
Output op=new Output();
op.start();
}
}
```

OUTPUT:

```
Find area:
1.rectangle
2.triangle
3.circle
3
enter radius:10
314.2
```

RESULT:

Thus, a Java program for finding area of shapes using Inheritance is executed.

EX NO:3 **IMPLEMENT PACKAGES AND INTERFACES.**

DATE:

AIM:

To create a Java program that demonstrates how to create packages, interfaces, and implement interfaces.

ALGORITHM:

Step 1: Start the program

Step 2: Create a package named com.example

Step 3: Create an Interface named MyInterface

Step 4: Create a class that implements the MyInterface interface

Step 5: Create a main class to test the package and interface implementation

Step 6: All these files should be in the same directory structure as the package.

Compile and run these files using the following commands:javac com/example/MyPackageExample.java

javac com/example/MyInterface.java

javac com/example/MyImplementation.java

javac com/example/Main.java

java com.example.Main

Step 7: Stop the program

PROGRAM:

```
// File: MyPackageExample.java
package com.example;

public class MyPackageExample {
public void display() {
System.out.println("This is a class inside the com.example package.");
}
}

// File: MyInterface.java
package com.example;

public interface MyInterface {
void myMethod();
}

// File: MyImplementation.java
package com.example;

public class MyImplementation implements MyInterface {
@Override
public void myMethod() {
System.out.println("Implementation of the myMethod() from the interface.");
}
}

// File: Main.java
package com.example;

public class Main {
public static void main(String[] args) {
MyPackageExample packageExample = new MyPackageExample();
packageExample.display();

MyImplementation implementation = new MyImplementation();
implementation.myMethod();
}
}
```

OUTPUT:

This is a class inside the com.example package.
Implementation of the myMethod() from the interface.

RESULT:

Thus, a Java program that demonstrates how to create packages, interfaces, and implement interfaces is executed.

EX NO:4 **IMPLEMENT METHOD AND CONSTRUCTOR OVERLOADING**

DATE:

AIM:

To create a java program using method and constructor overloading.

ALGORITHM:

Step 1: Start.

Step 2: Create a constructor using a class name "student “with different parameters, which showcases constructor overloading.

Step 3: Create the methods with same methods name and different parameters which demonstrates method overloading.

Step 4: By calling method and constructor in the main method print the result.

Step 5: Stop.

PROGRAM:

```
import java.util.Scanner;

class student {
int roll_no;
String student_name;
String branch;
String department="Information technology";
static String year="2022-2026";
String phno;
String college;
String city;

//CONSTRUCTOR OVERLOADING

student()
{
Scanner scan=new Scanner(System.in);
System.out.print("Enter the roll_no:");
int a =scan.nextInt();
System.out.println("THE INFORMATION OF THE STUDENT .....");
System.out.print("The roll no of the student:"+a);
}

student(String name,String branch_name)
{
student_name=name;
branch=branch_name;
}

//METHOD OVERLOADING

public void display(String phno)
{
System.out.println("Mobile no:"+phno);
}

public void display(String college,String city)
{

System.out.println("College name:"+college);
System.out.println("Name of the city:"+city);
}

public static void main(String[] args)
{
student s1 = new student();
student s = new student("sona", "B-tect");
```

```
System.out.println("The name of the student is:" + s.student_name);
s.display("7010588669");
System.out.println("The branch of the student is:" + s.branch);
System.out.println("The name of the department:" + s1.department);
System.out.println("year:" + year);
s.display("Adhiyamaan","Hosur");
}
}
```

OUTPUT:

```
Enter the roll_no:49
THE INFORMATION OF THE STUDENT .....
The roll no of the student:49The name of the student is:sona
Mobile no:7010588669
The branch of the student is:B-tect
The name of the department:Information technology
year:2022-2026
College name:Adhiyamaan
Name of the city:Hosur
```

RESULT:

Thus, the java program using method and constructor overloading is executed successfully

EX NO:5 **IMPLEMENT EXCEPTION HANDLING AND CREATION OF USER DEFINED EXCEPTIONS.**

DATE:

AIM:

To create a Java program to demonstrate the exception handling and creation of user defined exceptions.

ALGORITHM:

Step 1: We define a user-defined exception class Custom Exception.

Step 2: The Example class has a method perform Operation that throws Custom Exception for negative values.

Step 3: In the main method, we test perform Operation with a negative value.

Step 4: Catching Custom Exception, we print the error message.

Step 5: The finally block ensures the program always prints "Exception handling example completed."

PROGRAM:

```
class CustomException extends Exception {
public CustomException(String message) {
super(message);
}
}

// Sample class with a method that throws a custom exception
class Example {
public void performOperation(int value) throws CustomException {
if (value < 0) {
throw new CustomException("Value cannot be negative");
} else {
System.out.println("Operation performed successfully with value: " + value);
}
}
}
public class ExceptionHandlingDemo {
public static void main(String[] args) {
try {
Example example = new Example();
int inputValue = -3;
example.performOperation(inputValue);
} catch (CustomException ce) {
System.err.println("Custom Exception Caught: " + ce.getMessage());
} finally {
System.out.println("Exception handling example completed.");
}
}
}
```

OUTPUT:

Custom Exception Caught: Value cannot be negative
Exception handling example completed.

RESULT:

Thus, a Java program to demonstrate the exception handling and creation of user defined exceptions is executed.

EX NO:6 **IMPLEMENT MULTI-THREADED APPLICATION.**

DATE:

AIM:

To write the program for multithreading applications in Java.

ALGORITHM:

Step 1: Start the program

Step 2: We create a class that extends the java.lang.Thread class.

Step 3: This class overrides the run() method available in the Thread class.

Step 4: A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread.

Step 5: Start() invokes the run() method on the Thread object.

Step 6: Stop the program

PROGRAM:

```
/ Java code for thread creation by extending
// the Thread class
class MultithreadingDemo extends Thread {
public void run()
{
try {
// Displaying the thread that is running
System.out.println(
"Thread " + Thread.currentThread().getId()
+ " is running");
}
catch (Exception e) {
// Throwing an exception
System.out.println("Exception is caught");
}
}
}

// Main Class
public class Multithread {
public static void main(String[] args)
{
int n = 8; // Number of threads
for (int i = 0; i < n; i++) {
MultithreadingDemo object
= new MultithreadingDemo();
object.start();
}
}
}
```

OUTPUT:

```
Thread 15 is running
Thread 14 is running
Thread 16 is running
Thread 12 is running
Thread 11 is running
Thread 13 is running
Thread 18 is running
Thread 17 is running
```

RESULT:

Thus, the program for multithreading applications in Java is executed.

EX NO: **IMPLEMENT COLLECTION CLASS (ARRAY, LIST, MAP, SET).**

DATE:

AIM:

To create a Java program that demonstrates the usage of Array List, Map, and Set.

ALGORITHM:

Step 1: Start the program
Step 2: Create an Array List, a Map (HashMap), and a Set (HashSet).
Step 3: Add elements to the Array List.
Step 4: Add key-value pairs to the Map.
Step 5: Add elements to the Set.
Step 6: Output the elements of the Array List using a for-each loop.
Step 7: Output the elements of the Map using a for-each loop with Map. Entry.
Step 8: Output the elements of the Set using a for-each loop.
Step 9: Stop the program

PROGRAM:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class CollectionExample {
public static void main(String[] args) {
// Array List Example
Array List<String> arrayList = new Array List<>();
arrayList.add("Apple");
arrayList.add("Banana");
arrayList.add("Orange");

// Map Example
Map<String, Integer> map = new HashMap<>();
map.put("One", 1);
map.put("Two", 2);
map.put("Three", 3);

// Set Example
Set<String> set = new HashSet<>();
set.add("Red");
set.add("Green");
set.add("Blue");

// Output Array List elements
System.out.println("Array List Elements:");
for (String fruit : arrayList) {
System.out.println(fruit);
}

// Output Map elements
System.out.println("\nMap Elements:");
for (Map.Entry<String, Integer> entry : map.entrySet()) {
System.out.println(entry.getKey() + ": " + entry.getValue());
}

// Output Set elements
System.out.println("\nSet Elements:");
for (String color : set) {
System.out.println(color);
}
}
}
```

OUTPUT:

Array List Elements:
Apple
Banana
Orange

Map Elements:
One: 1
Two: 2
Three: 3

Set Elements:
Red
Green
Blue

RESULT:

Thus, a Java program that demonstrates the usage of Array List, Map, and Set is executed.

EXNO:8(A) **IMPLEMENT FILE OPERATIONS (READING A FILE)**

DATE:

AIM:

To write a java program for reading a file using file operation.

ALGORITHM:

Step 1: Start

Step 2: Import statements related to file operations for performing reding operations.

Step 3: Create a class and main method and use try-catch block for exceptional handling.

Step 4: Using input statement read the file.

Step 5: Create a while loop to check whether the file is has content to read, if true then read the content and close the file using file operation.

Step 6: If not go to the catch block and print the custom message.

Step 7: Stop.

PROGRAM:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
class file {
public static void main(String args[]) {
try {

File f = new File("filename.txt");
Scanner read = new Scanner(f);
while (read.hasNextLine()) {
System.out.println(read.nextLine());
}
read.close();
}
catch(FileNotFoundException exception)
{
System.out.println("ERROR");
}
}
}
```

OUTPUT:

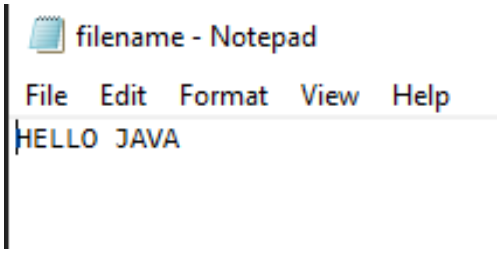
```
D:\>d:

D:\>cd/

D:\>javac read.java

D:\>java read.java
HELLO JAVA

D:\>
```



RESULT:

Thus, a java program for reading a file using file operation is executed.

EXNO:8(B) **IMPLEMENTATION OF WRITING A FILE**

DATE:

AIM:

To write a java program for writing a file using file operation.

ALGORITHM:

Step 1: Start
Step 2: Import statements related to file operations for performing writing operations.
Step 3: Create a class and main method and use try-catch block for exceptional handling.
Step 4: Call the File Writer class and pass the content to that function which we want to write.
Step 5: Close the file fw using close () method.
Step 6: Stop.

PROGRAM:

```
import java.io.File;
import java.io.FileWriter;
import java.util.Scanner;
class file {
public static void main(String args[]) {
try {

File f = new File("filename.txt");
FileWriter fw=new FileWriter(f);
fw.write("HELLO JAVA CODE");
fw.close();
}
catch(Exception E)
{
System.out.println("ERROR");
}
}
}
```

OUTPUT:

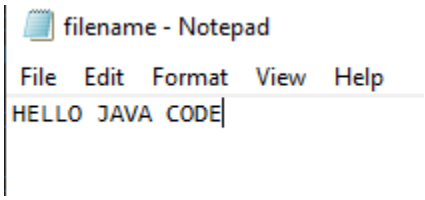
```
D:\>d:

D:\>cd/

D:\>javac file.java

D:\>java file.java

D:\>
```



RESULT:

Thus, a java program for writing a file using file operation is executed.

EXNO:9 **IMPLEMENT STRING HANDLING FUNCTIONS.**

DATE:

AIM:

To create a java program for handling string and its functions.

ALGORITHM:

Step 1: Length of the String. Find and print the length of the input string.

Step 2: Character at Index. Retrieve and print the character at index 7.

Step 3: Substring Extraction. Extract and print the substring from index 13 to 22.

Step 4: String Concatenation. Concatenate "Hello, " and "World!" and print the result.

Step 5: String Comparison. Compare two strings ("apple" and "banana") and print the result.

Step 6: String Transformation. Convert the input string to uppercase and print it.

Step 7: Stop the program.

PROGRAM:

```
public class StringHandlingExample {
public static void main(String[] args) {
// Sample input string
String text = "Hello, World! This is a Java String Handling Example.";

// 1. Length of the String
int length = text.length();
System.out.println("1. Length of the String: " + length);

// 2. Character at a specific index
char characterAtIndex = text.charAt(7); // Index 7 is 'W'
System.out.println("2. Character at index 7: " + characterAtIndex);

// 3. Substring from index 13 to 22
String substring = text.substring(13, 23); // "This is a"
System.out.println("3. Substring from index 13 to 22: " + substring);

// 4. Concatenation
String string1 = "Hello, ";
String string2 = "World!";
String concatenatedString = string1 + string2;
System.out.println("4. Concatenation: " + concatenatedString);

// 5. String comparison
String str1 = "apple";
String str2 = "banana";
int comparisonResult = str1.compareTo(str2);
System.out.println("5. String Comparison (Lexicographically): " + comparisonResult);

// 6. String to Upper Case
String upperCaseText = text.toUpperCase();
System.out.println("6. Uppercase: " + upperCaseText);

// 7. String to Lower Case
String lowerCaseText = text.toLowerCase();
System.out.println("7. Lowercase: " + lowerCaseText);

// 8. Checking if a string contains a substring
boolean containsSubstring = text.contains("Java");
System.out.println("8. Contains 'Java': " + containsSubstring);

// 9. Replacing a substring
String replacedText = text.replace("World", "Universe");
System.out.println("9. Replace 'World' with 'Universe': " + replacedText);
```

```
// 10. Splitting a string into an array
String[] words = text.split(" ");
System.out.println("10. Splitting into words (space-separated):");
for (String word : words) {
System.out.println(word);
}
}
}
```

OUTPUT:

1. Length of the String: 51
2. Character at index 7: W
3. Substring from index 13 to 22: This is a
4. Concatenation: Hello, World!
5. String Comparison (Lexicographically): -1
6. Uppercase: HELLO, WORLD! THIS IS A JAVA STRING HANDLING EXAMPLE.

RESULT:

Thus, a java program for handling string and its functions is executed.

EXNO:10 **DEVELOP AN APPLICATION USING JAVA CONCEPTS, JAVAFX COTROLS, LAYOUTS AND MEANS WITH DATABASE CONNECTIVITY.**

DATE:

AIM:

To Develop an application using Java Concepts, JavaFX controls, layouts and menus with Database Connectivity

ALGORITHM:

1. Create labels for "Email" and "Password".
2. Generate text fields for email and password input
3. Create "Submit" and "Clear" buttons.
4. Initialize a Grid Pane layout to organize the UI elements.
5. Set size, padding, and gaps between elements.
6. Align content to the center within the grid.
7. Apply styles to buttons, labels, and the grid pane using CSS.
8. Create a Scene object containing the grid pane.
9. Set title and scene for the Stage.
10. Show the Stage to present the login interface to users.

PROGRAM:

```java
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.geometry.Insets;
import javafx.geometry.Pos;

import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Text;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class LoginPage extends Application {
   @Override
   public void start(Stage stage) {
      //creating label email
      Text text1 = new Text("Email");

      //creating label password
      Text text2 = new Text("Password");

      //Creating Text Filed for email
      TextField textField1 = new TextField();

      //Creating Text Filed for password
      PasswordField textField2 = new PasswordField();

      //Creating Buttons
      Button button1 = new Button("Submit");
      Button button2 = new Button("Clear");

      //Creating a Grid Pane
      GridPane gridPane = new GridPane();

      //Setting size for the pane
      gridPane.setMinSize(400, 200);

      //Setting the padding
      gridPane.setPadding(new Insets(10, 10, 10, 10));

      //Setting the vertical and horizontal gaps between the columns
      gridPane.setVgap(5);
      gridPane.setHgap(5);

      //Setting the Grid alignment
      gridPane.setAlignment(Pos.CENTER);

      //Arranging all the nodes in the grid
      gridPane.add(text1, 0, 0);
      gridPane.add(textField1, 1, 0);
      gridPane.add(text2, 0, 1);
      gridPane.add(textField2, 1, 1);
      gridPane.add(button1, 0, 2);
      gridPane.add(button2, 1, 2);

      //Styling nodes
      button1.setStyle("-fx-background-color: darkslateblue; -fx-text-fill: white;");
      button2.setStyle("-fx-background-color: darkslateblue; -fx-text-fill: white;");

      text1.setStyle("-fx-font: normal bold 20px 'serif' ");
      text2.setStyle("-fx-font: normal bold 20px 'serif' ");
      gridPane.setStyle("-fx-background-color: BEIGE;");

      //Creating a scene object
      Scene scene = new Scene(gridPane);

      //Setting title to the Stage
      stage.setTitle("CSS Example");

      //Adding scene to the stage
      stage.setScene(scene);

      //Displaying the contents of the stage
      stage.show();
   }
   public static void main(String args[]){
      launch(args);
   }
}
```

OUTPUT:

RESULT:

To develop an application using Java Concepts, JavaFX controls, layouts and menus with Database Connectivity has been executed successfully.