**Name: A.A.D.Jayangani**

**Student No: 28455**

**JAVA-Practicals**

**Practical 01**

**01)**

package com.mycompany.helloworld3;

public class HelloWorld3

{

public static void main(String[] args)

{

System.out.println("Hello World!");

}

}

**02)**

package com.mycompany.mydetails;

public class MyDetails

{

public static void main(String[] args)

{

System.out.println("My name is Duleesha Jayangani");

System.out.println("Name of my degree program is Bsc(Hons)Software Engineering ");

}

}

**03)**

package com.mycompany.loop;

public class Loop

{

public static void main(String[] args)

{

for(int a=0;a<5;a++)

{

System.out.println("Executing Loop " +a);

}

}

}

**04)**

**Break**

public class Break

{

public static void main(String[] ardgs)

{

int [] numbers = {10, 20, 30, 40, 50};

for(int x : numbers )

{

if( x == 30 )

{

break;

}

System.out.print( x );

System.out.print("\n");

}

System.out.print("I’m out of the Loop now");

}

}

**Output:**

10

20

I’m out of the Loop now

**Continue**

public class Continue

{

public static void main(String[] args)

{

int [] numbers = {10, 20, 30, 40, 50};

for(int x : numbers )

{

if( x == 30 )

{

continue;

}

System.out.print( x );

System.out.print("\n");

}

System.out.print("I’m out of the Loop now");

}

}

**Output:**

10

20

40

50

I’m out of the Loop now

**05)** public class FindGrade

{

public static void main(String[] args)

{

char grade ='A';

switch(grade)

{

case 'A' :System.out.println("Excellent!"); break;

case 'D' :System.out.println("You passed");

case 'F' :System.out.println("Better try again");break;

default :System.out.println("Invalid grade");

}

System.out.println("Your grade is " + grade);

}

}

**Output:**

Excellent!

Your grade is A

public class FindGrade

{

public static void main(String[] args)

{

char grade ='A';

switch(grade)

{

case 'A' :System.out.println("Excellent!");

case 'D' :System.out.println("You passed");

case 'F' :System.out.println("Better try again");break;

default :System.out.println("Invalid grade");

}

System.out.println("Your grade is " + grade);

}

}

**Output:**

Excellent!

You passed

Better try again

Your grade is A

public class ElseIf

{

public static void main(String[] args)

{

char grade ='A';

if(grade=='A')

{

System.out.println("Excellent!");

}

else if(grade=='D')

{

System.out.println("You passed");

}

else if(grade=='F')

{

System.out.println("Better try again");

}

else

{

System.out.println("Invalid grade");

}

System.out.println("Your grade is " + grade);

}

}

**Output**:

Excellent!

Your grade is A

**06)**

public class TestEnhanceForLoop

{

public static void main(String[] ardgs)

{

int [] numbers = {10, 20, 30, 40, 50};

for(int x:numbers)

{

System.out.print(x);

System.out.println(",");

}

System.out.print("\n");

String[]names={"James", "Larry", "Tom", "Lacy"};

for(String name:names)

{

System.out.print(name);

System.out.print(",");

}

}

}

**Output:**

10,20,30,40,50,

James,Larry,Tom,Lacy,

**Practical- 02**

1. **Part 1**

ItemObj.java(Main)

package com.mycompany.itemobj;

public class ItemObj

{

public static void main(String[] args)

{

Monster m=new Monster(10300,"Kottawa");

m.setLocation(10345);

m.setDescription("Pannipitiya");

System.out.println("Location is "+m.getLocation());

System.out.println("Description is "+m.getDescription());

}

}

Item.java(parent class)

package com.mycompany.itemobj;

public class Item

{

protected int location;

protected String description;

public Item(int location,String description)

{

this.location=location;

this.description=description;

}

public void setLocation(int a)

{

location=a;

}

public int getLocation()

{

return location;

}

public void setDescription(String b)

{

description=b;

}

public String getDescription()

{

return description;

}

}

Monster.java(child class)

package com.mycompany.itemobj;

public class Monster extends Item

{

public Monster(int location,String description)

{

super(location,description);

}

}

**02)** **Part 2**

1-super

2-private

3-packages

4- import pkg.\*

5- charAt()

6- length()

1. **Part 3**

1-attributes and behaviour

2-instance varibles

3-methods

4-Encapsulation

5-class

6-base class, sub class, extends

7-Interface

8-Package

9-Application Programming Interface

**Practical- 03**

**01)**

**EncapsulationDemo(class)**

package com.mycompany.encapstest;

public class EncapsulationDemo

{

private String empName;

private int age;

private float sal;

//constructor

public EncapsulationDemo(String empName,int age,float sal)

{

this.empName=empName;

this.age=age;

this.sal=sal;

}

//Getter and Setter methods

public void setEmpName(String en)

{

empName=en;

}

public String getEmpName()

{

return empName;

}

public void setAge(int a)

{

age=a;

}

public int getage()

{

return age;

}

public void setsal(float s)

{

sal=s;

}

public float getsal()

{

return sal;

}

}

**EncapsTest(main)**

package com.mycompany.encapstest;

public class EncapsTest

{

public static void main(String[] args)

{

EncapsulationDemo obj = new EncapsulationDemo("Mario",23,20000.00f);

obj.setEmpName("Mario");

obj.setAge(23);

obj.setsal(20000.00f);

System.out.println("Employee name is " +obj.getEmpName());

System.out.println("Age is " +obj.getage());

System.out.println("Salary is " +obj.getsal());

}

}

**02)**

**EncapsulationDemo1(class)**

package com.mycompany.encapstest1;

public class EncapsulationDemo1

{

private String empName;

private float bsal,bonous,newsal;

//constructor

public EncapsulationDemo1(String empName,float bsal,float bonous)

{

this.empName=empName;

this.bsal=bsal;

this.bonous=bonous;

}

//Getter and Setter methods

public void setEmpName(String en)

{

empName=en;

}

public String getEmpName()

{

return empName;

}

public void setbsal(float s)

{

bsal=s;

}

public float getbsal()

{

return bsal;

}

public void setbonous(float b)

{

bonous=b;

}

public float getbonous()

{

return bonous;

}

public void setBonousAmount()

{

newsal=bonous+bsal;

System.out.println("Bonous Amount is " +newsal);

}

}

**EncapsTest1(main)**

package com.mycompany.encapstest1;

public class EncapsTest1

{

public static void main(String[] args)

{

{

EncapsulationDemo1 obj = new EncapsulationDemo1("Mario",5000.00f,1000.00f);

obj.setEmpName("Mario");

obj.setbsal(5000.00f);

obj.setbonous(1000.00f);

System.out.println("Employee name is " +obj.getEmpName());

System.out.println("Salary is " +obj.getbsal());

System.out.println("Bonous is " +obj.getbonous());

obj.setBonousAmount();

}

}

}

**Practical- 04**

01)

**Employee(class)**

package com.mycompany.testclass;

public class Employee

{

private int empID;

private String empName, empDesignation;

public void setEmpId(int empID)

{

this.empID=empID;

}

public int getEmpId()

{

return empID;

}

public void setName(String empName)

{

this.empName=empName;

}

public String getName()

{

return empName;

}

public void setDesignation(String empDesignation)

{

this.empDesignation=empDesignation;

}

public String getDesignation()

{

return empDesignation;

}

}

**TestClass(main)**

package com.mycompany.testclass;

public class TestClass

{

public static void main(String[] args)

{

Employee e1=new Employee();

e1.setName("Mr.Bogdan");

e1.setEmpId(001);

e1.setDesignation("Abc");

System.out.println("1st Employee Name is " +e1.getName());

System.out.println("1st Employee ID is " +e1.getEmpId());

System.out.println("1st Employee Designation is " +e1.getDesignation());

System.out.println("\n");

Employee e2=new Employee();

e2.setName("Ms.Bird");

e2.setEmpId(002);

e2.setDesignation("Efg");

System.out.println("2nd Employee Name is " +e2.getName());

System.out.println("2nd Employee ID is " +e2.getEmpId());

System.out.println("2nd Employee Designation is " +e2.getDesignation());

}

}

02)

9

6

03)

**Person (class)**

package com.mycompany.testclass4;

public class Person

{

private String name;

private int id;

// Constructor

public Person(String name, int id)

{

this.name = name;

this.id = id;

}

// Getters and Setters

public String getName()

{

return name;

}

public int getID()

{

return id;

}

public void setName(String name)

{

this.name = name;

}

public void setID(int id)

{

this.id = id;

}

}

**Student(class)**

package com.mycompany.testclass4;

public class Student extends Person

{

private String course;

// Constructor

public Student(String name, int id, String course)

{

super(name, id);

this.course = course;

}

// Getter and Setter for course

public String getCourse()

{

return course;

}

public void setCourse(String course)

{

this.course = course;

}

}

**Lecturer(class)**

package com.mycompany.testclass4;

public class Lecturer extends Person

{

private String programme;

// Constructor

public Lecturer(String name, int id, String programme) {

super(name, id);

this.programme = programme;

}

// Getter and Setter for programme

public String getProgramme() {

return programme;

}

public void setProgramme(String programme) {

this.programme = programme;

}

}

**TestClass4(main)**

package com.mycompany.testclass4;

public class TestClass4

{

public static void main(String[] args)

{

Student student = new Student("John Doe", 1001, "Computer Science");

Lecturer lecturer = new Lecturer("Dr. Smith", 2001, "Software Engineering");

System.out.println("Student Details:");

System.out.println("Name: " + student.getName());

System.out.println("ID: " + student.getID());

System.out.println("Course: " + student.getCourse());

System.out.println("\nLecturer Details:");

System.out.println("Name: " + lecturer.getName());

System.out.println("ID: " + lecturer.getID());

System.out.println("Programme: " + lecturer.getProgramme());

}

}

04)

true

true

true

**Practical- 05**

01)

1- There is no difference between declaring an interface variable with or without public static final keywords. All interface variables are implicitly treated as public static final.

public interface MyFirstInterface

{

int x = 10; // Equivalent to: public static final int x = 10;

}

2-There is no difference between declaring an abstract method with or without the abstract keyword. All interface methods are implicitly treated as abstract.

public interface MyFirstInterface

{

void display(); // Equivalent to: abstract void display();

}

3-It is not possible to change the value of x inside the class implementing the interface. Interface variables are implicitly public static final, making them constants, and their values cannot be changed once assigned.

public interface MyFirstInterface

{

int x = 10;

void display();

}

public class InterfaceImplemented implements MyFirstInterface

{

@Override

public void display()

{

// Attempt to change the value of x (Will cause a compilation error)

x = 20;

System.out.println("Value of x: " + x);

}

}

Attempting to change the value of x in the InterfaceImplemented class will cause a compilation error with the message: "error: cannot assign a value to final variable x". This is because x is declared as final in the interface and cannot be modified in implementing classes.

02)

**TestSpeakers(Main)**

package com.mycompany.testspeakers;

public class TestSpeakers

{

public static void main(String[] args)

{

Speaker politician = new Politician();

Speaker priest = new Priest();

Speaker lecturer = new Lecturer();

System.out.println("Politician's speech:");

politician.speak();

System.out.println("\nPriest's speech:");

priest.speak();

System.out.println("\nLecturer's speech:");

lecturer.speak();

}

}

**Speaker(Interface)**

package com.mycompany.testspeakers;

interface Speaker

{

void speak();

}

**Politician (class)**

package com.mycompany.testspeakers;

class Politician implements Speaker

{

@Override

public void speak()

{

System.out.println("Politician: I promise to serve the people.");

}

}

**Priest (class)**

package com.mycompany.testspeakers;

class Priest implements Speaker

{

@Override

public void speak()

{

System.out.println("Priest: Let us pray for peace and harmony.");

}

}

**Lecturer (class)**

package com.mycompany.testspeakers;

class Lecturer implements Speaker

{

@Override

public void speak()

{

System.out.println("Lecturer: Today's lecture is on Java programming.");

}

}

03)

final class Student

{

final int marks = 100;

final void display()

{

System.out.println("Marks: " + marks);

}

}

class Undergraduate extends Student

{

// Compilation error: Cannot extend a final class.

}

The code will result in a compilation error because the Undergraduate class cannot extend the Student class due to the final class declaration

04)

**Main2(main)**

package com.mycompany.main2;

public class Main2

{

public static void main(String[] args)

{

// Creating a Rectangle object

double rectangleLength = 5.0;

double rectangleWidth = 3.0;

Shape rectangle = new Rectangle(rectangleLength, rectangleWidth);

// Creating a Circle object

double circleRadius = 4.0;

Shape circle = new Circle(circleRadius);

// Displaying the area of the Rectangle and Circle

System.out.println("Rectangle:");

rectangle.display();

System.out.println("\nCircle:");

circle.display();

}

}

**Shape(class)**

package com.mycompany.main2;

// Shape class (Superclass)

abstract class Shape

{

// Abstract method to calculate the area

public abstract double calculateArea();

// Non-abstract method to display shape information

public void display()

{

System.out.println("Area: " + calculateArea());

}

}

**Rectangle(class)**

package com.mycompany.main2;

// Rectangle class (Subclass)

public class Rectangle extends Shape

{

private double length;

private double width;

// Constructor for Rectangle

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

// Implementing the calculateArea() method for Rectangle

@Override

public double calculateArea()

{

return length \* width;

}

}

**Circle (class)**

package com.mycompany.main2;

public class Circle extends Shape

{

private double radius;

// Constructor for Circle

public Circle(double radius)

{

this.radius = radius;

}

// Implementing the calculateArea() method for Circle

@Override

public double calculateArea()

{

return Math.PI \* radius \* radius;

}

}