**RV COLLEGE OF ENGINEERING®**

**(Autonomous Institution Affiliated to VTU, Belagavi)**

**BENGALURU – 560 059**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

****

**BASIC OF JAVA LAB (CS115BIC)**

**I SEMESTER - B.E. (CSE)**

**INSTRUCTORS MANUAL**

**2024-25**

**Prepared by:Dr. Sindhu,Prof ManonmaniS,Apoorva.U.C Verified by: HoD, CSE**

**RV College of Engineering®, Bengaluru- 560 059**

**(Autonomous Institution Affiliated to VTU, Belagavi)**

**Department of Computer Science and Engineering**

****

**Vision**

To achieve leadership in the field of Computer Science & Engineering by strengthening fundamentals and facilitating interdisciplinary sustainable research to meet the ever growing needs of the society.

**Mission**

* To evolve continually as a centre of excellence in quality education in computers and allied fields.
* To develop state-of-the-art infrastructure and create environment capable for interdisciplinary research and skill enhancement.
* To collaborate with industries and institutions at national and international levels to enhance research in emerging areas.
* To develop professionals having social concern to become leaders in top-notch industries and/or become entrepreneurs with good ethics.

**Course Outcomes (COs)**

|  |  |
| --- | --- |
| **CO 1** | Apply features of object-oriented programming of Java to solve real world problems. |
| **CO 2** | Design and develop the real world models and entities using Java programming. |
| **CO 3** | Implement the object oriented applications using features such as Inheritance,Packages,Interfaces,Exception Handling, Multi-threaded Programming and Strings. |
| **CO 4** | Demonstrate the real world applications and programming skills to solve inter disciplinary problems effectively and exhibit team work through presentations and reports using Java programming. |

**Mapping of Course Outcomes with Program Outcomes**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CO/PO** | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| CO1 | Explore the fundamentals of Object-oriented concepts and apply features of object-oriented programming of Java to solve real world problems. | **2** | **2** | **2** | **-** | **-** | **-** | **-** | **-** | **2** | **2** | **-** | **1** |
| CO2 | Design Classes and establish relationship among Classes for various applications from problem definition. | **3** | **3** | **3** | **3** | **1** | **-** | **-** | **-** | **3** | **3** | **-** | **1** |
| CO3 | Analyze and implement reliable object-oriented applications using Java features such as Exception Handling, Multi-threaded Programming, Collection framework and Strings. | **3** | **3** | **3** | **3** | **2** | **-** | **-** | **-** | **3** | **-** | **-** | **1** |
| CO4 | Employ state-of-art methodologies for product development and testing / validation with focus on optimization and quality related aspects. | **3** | **3** | **3** | **1** | **2** | **-** | **-** | **-** | **3** | **2** | **-** | **1** |

**High-3 : Medium-2 : Low-1**

**LIST OF PROGRAMS – (Part-A)**

***Implement the following programs in Java with Java Libraries:***

|  |  |
| --- | --- |
| **Sl. No.** | **Program** |
| 1. | Create a Java class Student with the following details as variables within it.  USN,  NAME,  BRANCH,  PHONE,  PRECENTAGE.  Write a JAVA program to create n Student objects and print the USN,Name,Branch,Phone,Percentage of these objects with suitable headings. |
| 2. | Write a program to create a class Student with data ‘name, city and age’ along with method print data to display the data. Create the two objects s1,s2 to declare and access the values. |
| 3. | Write a Java program to demonstrate method Overloading by changing the number of arguments and by changing the data type. |
| 4. | Write Java Program to demonstrate Constructor Overloading by creating class Box with properties of Width, Height, Depth and methods to compute and display the volume. |
| 5. | Write a Java Program to display the employee details like employee id, age and Name using the getter and setter Methods. And also illustrate the concept of Encapsulation. |
| 6. | Write a Java Program to   1. Create a Super class called Bicycle with three methods apply brake (), Speedup () and tostring (). 2. Create a sub class called Mountain Bike () and inherit the methods from the base class. 3. Illustrate the Method overriding on tostring() method. |
| 7. | Write a java program to illustrate the following exceptions  i. Arithmetic Exception.  ii. String out of Bound Exception. |
| 8. | Write a Java program in to declare an interface called Bank, where SBI, and PNB inherits  these interfaces through class TestInterface2. Calculate the rate of interest of each bank and  print the result. |
| 9. | Write a program to make a package balance which has account class with display\_balance method in it. Import balance package in another program to access display\_balance method of account class. |
| 10. | Write a JAVA program to create five threads with different priorities. Send two threads of the highest priority to sleep state. Check the aliveness of the threads and mark which is long lasting. |

**Rubrics for Java Lab – Part-A**

**Each program is evaluated for 10 marks.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lab Write-up and Execution rubrics (Max: 6 marks)** | | | | | |
| **Sl. No** | **Criteria** | **Measuring methods** | **Excellent** | **Good** | **Poor** |
| 1 | **Understanding of problem statement.**  **(2 Marks)**  **CO1** | Observations | Student exhibits thorough understanding of program requirements and applies suitable Computer Graphics concepts for the problem.  **(1.5 - 2 M)** | Student has sufficient understanding of program requirements and applies suitable Computer Graphics concepts for the problem.  **(0.5 -1 M)** | Student does not have a clear understanding of program requirements and is unable to apply suitable Computer Graphics Concepts for the problem.  **(0 M)** |
| 2 | **Execution and Testing**  **(2 Marks)**  **CO3, CO4** | Implementation Skills | Student demonstrates the execution of the program with all possible conditions.  **(1.5 - 2 M)** | Student demonstrates the execution of the program with Few conditions.  **(0.5 -1 M)** | Student has not executed the program.  **(0 M)** |
| 3 | **Results and Documentation**  **(2 Marks)**  **CO4** | Observations | Documentation with appropriate comments and output is covered in data sheets and manual.  **(1.5 - 2 M)** | Documentation with only few comments and only few output cases is covered in data sheets and manual.  **(0.5 -1 M)** | Documentation with no comments and no output cases is covered in data sheets and manual.  **(0 M)** |
| **Viva Voce rubrics (Max: 4 marks)** | | | | | |
| 1 | **Conceptual Understanding**  **(2 Marks)**  **CO1** | Viva Voce | Explains thoroughly the Computer Graphics Concepts and Algorithms  **(1.5 - 2 M)** | Adequately explains the Computer Graphics Concepts and Algorithms  **(0.5 -1 M)** | Unable to explain the Computer Graphics Concepts and Algorithms  **(0 M)** |
| 2 | **Use of appropriate Computer Graphics APIs**  **(1 Mark)**  **CO2, CO3** | Viva Voce | Thoroughly explains the usage of appropriate API for the given problem.  **(1 M)** | Sufficiently explains the usage of appropriate API for the given problem.  **(0.5 M)** | Unable to explain the usage of appropriate API for the given problem.  **(0 M)** |
| 3 | **Communication of Concepts**  **(1 Mark)**  **CO1, CO2** | Viva Voce | Communicates the concept used in problem solving well.  **(1 M)** | Sufficiently communicates the concepts used in problem solving.  **(0.5 M)** | Unable to communicate the concepts used in problem.  **(0 M)** |

**LAB PROGRAMS**

**Part -A**

**Program 1**

1. **Write a Java program to demonstrate method Overloading by changing the number of arguments and by changing the data type.**

// Importing required classes

import java.io.\*;

// Class 1

// Helper class

class Addition {

// Method 1

// Adding two integer values

public int add(int a, int b)

{int sum = a + b;

return sum;}

// Method 2

// Adding three integer values

public int add(int a, int b, int c)

{int sum = a + b + c;

return sum;

}

//Method 3

public double add (double a , double b)

{

double sum = a + b;

return sum;

}

}

// Class 2

// Main class

class GFG {

// Main driver method

public static void main(String[] args)

{// Creating object of above class inside main()

// method

Addition ob = new Addition();

// Calling method to add 3 numbers

int sum1 = ob.add(1, 2);

// Printing sum of 2 numbers

System.out.println("sum of the two integer value :"+ sum1);

// Calling method to add 3 numbers

int sum2 = ob.add(1, 2, 3);

// Printing sum of 3 numbers

System.out.println("sum of the three integer value :" + sum2);

// Printing the sum of two numbers with different data type

double sum3 =ob.add(1, 2.4);

System.out.println (" sum is :" +sum3);

}

}

**2. Write Java Program to demonstrate Constructor Overloading by creating class Box with properties of Width, Height, Depth and methods to compute and display the volume.**

class Box

{

double width, height, depth;

// constructor with dimensions

Box(double w, double h, doule d)

{

width = w;

height = h;

depth = d;

}

// constructor used when no dimensions

Box()

{

width = height = depth = 0;

}

// constructor used when cube is created

Box(double len)

{

width = height = depth = len;

}

// compute and return volume

double volume()

{

return width \* height \* depth;

}

}

public class Test

{

public static void main(String args[])

{

// create boxes using the various

// constructors

Box mybox1 = new Box(5, 10, 15);

Box mybox2 = new Box();

Box mycube = new Box(7);

double vol;

// get volume of first box

vol = mybox1.volume();

System.out.println(" Volume of mybox1 is " + vol);

// get volume of second box

vol = mybox2.volume();

System.out.println(" Volume of mybox2 is " + vol);

// get volume of cube

vol = mycube.volume();

System.out.println(" Volume of mycube is " + vol);

}

}

**3. Write a Java Program to display the employee details like employee id, age and Name using the getter and setter Methods. And also illustrate the concept of Encapsulation.**

class Employee {

// private field

private int age;

private int id;

private String name;

// getter method

public int getAge() {

return age;

}

public int getid() {

return id;

}

public String getname() {

return name;

}

// setter method

public void setAge(int age) {

this.age = age;

}

public void setid(int id) {

this.id = id;

}

public void setname(String name ) {

this.name = name;

}

}

class Main {

public static void main(String[] args) {

// create an object of Person

Employee p1 = new Employee();

//Employee p2 = new Employee();

// change age using setter

p1.setAge(24);

p1.setid(1234);

p1.setname("sindhu");

// access age using getter

System.out.println("My age is " + p1.getAge());

System.out.println("My id is " + p1.getid());

System.out.println("My name is " + p1.getname());

}

}

**4. Write a Java Program to**

1. **Create a Super class called Bicycle with three methods apply brake (), Speedup () and tostring ().**
2. **Create a sub class called Mountain Bike () and inherit the methods from the base class.**
3. **Illustrate the Method overriding on tostring() method.**

// base class

class Bicycle {

public int gear;

public int speed;

public Bicycle(int gear, int speed)

{

this.gear = gear;

this.speed = speed;

}

public void applyBrake(int decrement)

{

speed -= decrement;

}

public void speedUp(int increment)

{

speed += increment;

}

// toString() method to print info of Bicycle

public String toString()

{

return ("No of gears are " + gear + "\n"

+ "speed of bicycle is " + speed);

}

}

// derived class

class MountainBike extends Bicycle {

public int seatHeight;

// the MountainBike subclass has one constructor

public MountainBike(int gear, int speed,

int startHeight)

{

// invoking base-class(Bicycle) constructor

super(gear, speed);

seatHeight = startHeight;

}

// the MountainBike subclass adds one more method

public void setHeight(int newValue)

{

seatHeight = newValue;

}

// overriding toString() method

// of Bicycle to print more info

@Override public String toString()

{

return (super.toString() + "\nseat height is "

+ seatHeight);

}

}

// driver class

public class Test {

public static void main(String args[])

{

MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}}

**5. Write a java program to illustrate the following exceptions**

**i. Arithmetic Exception.**

**ii. String out of Bound Exception.**

// Java program to demonstrate ArithmeticException

class ArithmeticException\_Demo

{

public static void main(String args[])

{

try {

int a = 10, b = 0;

int c = a/b; // cannot divide by zero

System.out.println ("Result = " + c);

}

catch(ArithmeticException e) {

System.out.println ("Can't divide a number by 0");

}

}

}

// Java program to demonstrate StringIndexOutOfBoundsException

class StringIndexOutOfBound\_Demo

{

public static void main(String args[])

{

try {

String a = "R V College of Engineering ";

char c = a.charAt(28); // accessing 25th element

System.out.println(c);

}

catch(StringIndexOutOfBoundsException e) {

System.out.println("StringIndexOutOfBoundsException");

}

}

}

**6. Write a Java program in to declare an interface called Bank, where SBI, and PNB inherits these interfaces through class TestInterface2. Calculate the rate of interest of each bank and print the result.**

interface Bank{

float rateOfInterest();

}

class SBI implements Bank{

public float rate Of Interest(){return 9.15f;}

}

class PNB implements Bank{

public float rate Of Interest(){return 9.7f;}

}

class TestInterface2{

public static void main(String[] args){

Bank b=new SBI();

System.out.println("ROI: "+b.rateOfInterest());

Bank b1 =new PNB ();

System.out.println("ROI: "+b1.rateOfInterest());

}

}

1. **Write a program to make a package balance which has account class with display\_balance method in it. Import balance package in another program to access display\_balance method of account class.**

**Account.java**

package Balance;

import java.util.Scanner;

public class Account {

int curBalance, amt;

public Account() {

curBalance = 500;

}

void deposit() {

Scanner s = new Scanner(System.in);

System.out.println("Enter the amount :");

amt = s.nextInt();

curBalance += amt;

System.out.println("Current balance is :" + curBalance);

}

void withdraw() {

Scanner s = new Scanner(System.in);

System.out.println("Enter the amount :");

amt = s.nextInt();

try {

if ((curBalance - amt) < 500)

throw new LessBalanceException(amt);

curBalance -= amt;

System.out.println("\nBalance left :" + curBalance);

} catch (LessBalanceException e) {

System.out.println(e);

}

}

void display\_balance() {

System.out.println("Balance in your a/c :" + curBalance);

}

}

class LessBalanceException extends Exception {

int amt;

LessBalanceException(int x) {

System.out.println("Balance is less :" + amt);

}

}

**MainProgram.java**

package Balance;

import java.util.Scanner;

public class MainProgram {

public static void main(String[] args) {

int ch;

Scanner s = new Scanner(System.in);

Account a = new Account();

while (true) {

System.out.println("1:Deposit\t2:Withdraw\t3:Balance\t4:Exit\n");

System.out.println("Enter your choice:");

ch = s.nextInt();

switch (ch) {

case 1:

a.deposit();

break;

case 2:

a.withdraw();

break;

case 3:

a.display\_balance();

break;

case 4:

return;

default:

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

return;

}

}

}

}

**8.Write a JAVA program to create five threads with different priorities. Send two threads of the highest priority to sleep state. Check the aliveness of the threads and mark which is long lasting.**

class ThreadClass implements Runnable

{

long click=0;

Thread t;

private volatile boolean running =true;

public ThreadClass(int p)

{

t=new Thread(this);

t.setPriority(p);

}

public void run()

{

while(running)

{

click++;

}

}

public void stop()

{

running =false;

}

public void start()

{

t.start();

}

}

public class Demo {

public static void main(String args[])

{

Thread.currentThread().setPriority(Thread.MAX\_PRIORITY);

ThreadClass hi1=new ThreadClass(Thread.NORM\_PRIORITY + 2);

ThreadClass hi2=new ThreadClass(Thread.NORM\_PRIORITY -2);

ThreadClass hi3=new ThreadClass(Thread.NORM\_PRIORITY + 3);

ThreadClass hi4=new ThreadClass(Thread.NORM\_PRIORITY - 3);

ThreadClass hi5=new ThreadClass(Thread.NORM\_PRIORITY +4);

hi1.start();

hi2.start();

hi3.start();

hi4.start();

hi5.start();

System.out.println("thread one is alive:" +hi1.t.isAlive());

System.out.println("thread two is alive:" +hi2.t.isAlive());

System.out.println("thread three is alive:" +hi3.t.isAlive());

System.out.println("thread four is alive:" +hi4.t.isAlive());

System.out.println("thread four is alive:" +hi5.t.isAlive());

try

{ hi5.t.sleep(1000);

hi3.t.sleep(1000);

}

catch(InterruptedException e){

System.out.println("main thread interrupted");

}

hi1.stop();

hi2.stop();

hi3.stop();

hi4.stop();

hi5.stop();

try

{

System.out.println("waiting for threads to finish");

hi1.t.join();

hi2.t.join();

hi3.t.join();

hi4.t.join();

hi5.t.join();

}

catch(InterruptedException e)

{

System.out.println("main thread interrupted");

}

System.out.println("priority of thread1:" +hi1.t.getPriority());

System.out.println("priority of thread2:" +hi2.t.getPriority());

System.out.println("priority of thread3:" +hi3.t.getPriority());

System.out.println("priority of thread4:" +hi4.t.getPriority());

System.out.println("priority of thread5:" +hi5.t.getPriority());

System.out.println("thread one is alive:" +hi1.t.isAlive());

System.out.println("thread two is alive:" +hi2.t.isAlive());

System.out.println("thread three is alive:" +hi3.t.isAlive());

System.out.println("thread four is alive:" +hi4.t.isAlive());

System.out.println("thread five is alive:" +hi5.t.isAlive());

System.out.println("main thread exiting");

}

}

**PART B**

***Open-Ended learning*** is to be demonstrated by Case study of any real world problems and solved using Java tools (development of a small application or mini project using Java tools).

**Case study of any real world problems and Implementation using Java Libraries.**

* Students to explore the real world programs and solve it at the primary, elementary and secondary levels.
* Demonstrate the Case-Study implemented.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirements Specification, Execution & Report (Max: 10 marks)** | | | | | |
| **Sl no** | **Criteria** | **Measuring methods** | **Excellent** | **Good** | **Poor**  **Score** |
| 1 | **Problem Formulation and Requirement Specification**  **(2 Marks)**  **CO1, CO2** | Observations  VR Concepts | Student exhibits thorough understanding of project requirements and applies VR Concepts.  **(1.5M - 2M)** | Student has sufficient understanding of project requirements and applies VR concepts  **(0.5M - 1M)** | Student does not have clear understanding of project requirements and is unable to apply VR concepts **(0M)** |
| 2 | **Execution/Demonstration**  **(6 Marks)**  **CO3,CO4** | Observations  VR Tools usage | Student demonstrates the execution of the case study using VR tools in the project with optimized code with all the necessary conditions and test cases handled.  **(3.5M - 6M)** | Student demonstrates the execution of the case study using VR tools in the project without optimization of the code and handles only few test cases.  **(0.5M –3M)** | Student has not demonstrated VR in the project.  **(0 M)** |
| 3 | **Results and Documentation, Report**  **(2 Marks)**  **CO4** | Observations | Documentation with appropriate comments and different possible output cases is covered in report  **(1.5M - 2M)** | Documentation with only few comments and only few output cases is covered in report  **(0.5M –1M)** | Documentation with no comments and no output cases is covered in report.  **(0 M)** |