**Lab# 1: Introduction to Java**

1. **Write a program to display “Hello World” to the user.**

**public** **class** JavaIntro {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

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

}

}

**Lab# 2: Tokens, Expressions and Control Structures**

1. **Write a program to find the smallest number between 3 entered number using if else statement.**

**import** java.util.Scanner;

**public** **class** SmallestNo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.println("Enter three numbers:");

**int** a, b, c;

a = input.nextInt();

b = input.nextInt();

c = input.nextInt();

**if** (a < b && a < c) {

System.*out*.println(a + " is smallest");

} **else** **if** (b < a && b < c) {

System.*out*.println(b + " is smallest");

} **else** {

System.*out*.println(c + " is smallest");

}

input.close(); // Closing the Scanner object

}

}

1. **Write a program to find grade of a student using switch Case.**

**import** java.util.Scanner;

**public** **class** SwitchCaseGrade {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.print("Enter marks (out of 100): ");

**int** marks = input.nextInt();

**int** percentage = marks;

**char** grade;

**switch** (percentage / 10) {

**case** 9:

grade = 'A';

**break**;

**case** 8:

grade = 'B';

**break**;

**case** 7:

grade = 'C';

**break**;

**case** 6:

grade = 'D';

**break**;

**default**:

grade = 'F';

**break**;

}

System.*out*.println("Grade: " + grade);

input.close(); // Closing the Scanner object

}

}

1. **Write a program to find the Fibonacci series.**

**import** java.util.Scanner;

**public** **class** FibonacciSeries {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input=**new** Scanner(System.*in*);

System.*out*.println("Enter number of terms:");

**int** n=input.nextInt();

**int** a=0,b=1;

System.*out*.println("Fibonacci Series:");

System.*out*.print(a+" "+b+" ");

**for**(**int** i=1;i<=n;i++){

**int** c;

c=a+b;

System.*out*.print(c+" ");

a=b;

b=c;

}

}

}

1. **Write a program that demonstrates the use of break statement and continue statement.**

**import** java.util.Scanner;

**public** **class** BreakContinueDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.println("Example of break statement:");

**for** (**int** i = 1; i <= 5; i++) {

**if** (i == 3) {

System.*out*.println("Encountered break at i = 3");

System.*out*.print("After break statement:");

**break**;

}

System.*out*.print(i + " ");

}

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

System.*out*.println("Example of continue statement:");

**for** (**int** i = 1; i <= 5; i++) {

**if** (i == 3) {

System.*out*.println("Encountered continue at i = 3");

System.*out*.print("After continue statement:");

**continue**;

}

System.*out*.print(i + " ");

}

input.close();

}

}

1. **Write a program to take two numbers as input and then show the results of addition, subtraction, multiplication, division, and modulus operations using these numbers.**

**import** java.util.Scanner;

**public** **class** TwoNumOperation {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input=**new** Scanner(System.*in*);

System.*out*.println("Enter two numbers:");

**int** a=input.nextInt();

**int** b=input.nextInt();

**int** add=a+b;

**int** sub=a-b;

**int** mul=a\*b;

**float** div=a/b;

**int** mod=a%b;

System.*out*.println("Addition="+add);

System.*out*.println("Subtraction="+sub);

System.*out*.println("Multiplication="+mul);

System.*out*.println("Division="+div);

System.*out*.println("Modulus="+mod);

}

}

1. **Write a program compares two integers using each of the relational operators and displays the results.**

**import** java.util.Scanner;

**public** **class** RelationalOperator {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.print("Enter the two integers: ");

**int** num1 = input.nextInt();

**int** num2 = input.nextInt();

System.*out*.println("\nComparison Results:");

System.*out*.println(num1 + " < " + num2 + " : " + (num1 < num2));

System.*out*.println(num1 + " <= " + num2 + " : " + (num1 <= num2));

System.*out*.println(num1 + " > " + num2 + " : " + (num1 > num2));

System.*out*.println(num1 + " >= " + num2 + " : " + (num1 >= num2));

System.*out*.println(num1 + " == " + num2 + " : " + (num1 == num2));

System.*out*.println(num1 + " != " + num2 + " : " + (num1 != num2));

input.close();

}

}

1. **Write a program that demonstrates the use of all logical operators (&&, ||, and !). This program will evaluate several logical expressions and print the results.**

**public** **class** LogicalOperators {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

**boolean** a = **true**;

**boolean** b = **false**;

System.*out*.println("Logical AND (&&) Operator:");

System.*out*.println(a + " && " + b + " = " + (a && b)); // false

System.*out*.println(b + " && " + a + " = " + (b && a)); // false

System.*out*.println(a + " && " + a + " = " + (a && a)); // true

System.*out*.println(b + " && " + b + " = " + (b && b)); // false

System.*out*.println("\nLogical OR (||) Operator:");

System.*out*.println(a + " || " + b + " = " + (a || b)); // true

System.*out*.println(b + " || " + a + " = " + (b || a)); // true

System.*out*.println(a + " || " + a + " = " + (a || a)); // true

System.*out*.println(b + " || " + b + " = " + (b || b)); // false

System.*out*.println("\nLogical NOT (!) Operator:");

System.*out*.println("!" + a + " = " + !a); // false

System.*out*.println("!" + b + " = " + !b); // true

}

}

1. **Write a program compares two integers using each of the relational operators and displays the results.**

**import** java.util.Scanner;

**public** **class** RelationalOperatorDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.print("Enter the two integers: ");

**int** num1 = input.nextInt();

**int** num2 = input.nextInt();

// Comparing integers using relational operators

System.*out*.println("\nComparison Results:");

System.*out*.println(num1 + " < " + num2 + " : " + (num1 < num2));

System.*out*.println(num1 + " <= " + num2 + " : " + (num1 <= num2));

System.*out*.println(num1 + " > " + num2 + " : " + (num1 > num2));

System.*out*.println(num1 + " >= " + num2 + " : " + (num1 >= num2));

System.*out*.println(num1 + " == " + num2 + " : " + (num1 == num2));

System.*out*.println(num1 + " != " + num2 + " : " + (num1 != num2));

input.close(); // Closing the Scanner object

}

}

1. **Write a program to perform different bitwise operations on two integers.**

**import** java.util.Scanner;

**public** **class** BitwiseOperationsDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

// Taking input for two integers

System.*out*.print("Enter the two integers: ");

**int** num1 = input.nextInt();

**int** num2 = input.nextInt();

// Bitwise AND (&) Operation

**int** resultAND = num1 & num2;

System.*out*.println("\nBitwise AND (&) Operation:");

System.*out*.println(num1 + " & " + num2 + " = " + resultAND);

// Bitwise OR (|) Operation

**int** resultOR = num1 | num2;

System.*out*.println("\nBitwise OR (|) Operation:");

System.*out*.println(num1 + " | " + num2 + " = " + resultOR);

// Bitwise XOR (^) Operation

**int** resultXOR = num1 ^ num2;

System.*out*.println("\nBitwise XOR (^) Operation:");

System.*out*.println(num1 + " ^ " + num2 + " = " + resultXOR);

// Bitwise NOT (~) Operation

**int** resultNOT1 = ~num1;

**int** resultNOT2 = ~num2;

System.*out*.println("\nBitwise NOT (~) Operation:");

System.*out*.println("~" + num1 + " = " + resultNOT1);

System.*out*.println("~" + num2 + " = " + resultNOT2);

// Bitwise Left Shift (<<) Operation

**int** resultLeftShift = num1 << 1; // Shifting num1 left by 1 bit

System.*out*.println("\nBitwise Left Shift (<<) Operation:");

System.*out*.println(num1 + " << 1 = " + resultLeftShift);

// Bitwise Right Shift (>>) Operation

**int** resultRightShift = num1 >> 1; // Shifting num1 right by 1 bit

System.*out*.println("\nBitwise Right Shift (>>) Operation:");

System.*out*.println(num1 + " >> 1 = " + resultRightShift);

// Bitwise Unsigned Right Shift (>>>) Operation

**int** resultUnsignedRightShift = num1 >>> 1; // Unsigned right shift num1 by 1 bit

System.*out*.println("\nBitwise Unsigned Right Shift (>>>) Operation:");

System.*out*.println(num1 + " >>> 1 = " + resultUnsignedRightShift);

input.close(); // Closing the Scanner object

}

}

1. **Write a program to find the smallest of three number using conditional operator.**

**import** java.util.Scanner;

**public** **class** SmallestOfThree {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.println("Enter three numbers:");

**int** a = input.nextInt();

**int** b = input.nextInt();

**int** c = input.nextInt();

**int** smallest = (a < b) ? (a < c ? a : c) : (b < c ? b : c);

System.*out*.println("Smallest number: " + smallest);

input.close(); // Closing the Scanner object

}

}

1. **Write a program to swap two numbers without using third variable.**

**import** java.util.Scanner;

**public** **class** SwapNumbers {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner input = **new** Scanner(System.*in*);

System.*out*.print("Enter the first number: ");

**int** num1 = input.nextInt();

System.*out*.print("Enter the second number: ");

**int** num2 = input.nextInt();

num1 = num1 + num2;

num2 = num1 - num2;

num1 = num1 - num2;

System.*out*.println("\nAfter swapping:");

System.*out*.println("First number: " + num1);

System.*out*.println("Second number: " + num2);

input.close();

}

}

**Lab# 3: Object Oriented Programming Concepts**

1. **Write a program to demonstrate concept of class and object.**

**public** **class** ClassAndObject {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

System.*out*.println("The name of class is ClassAndObject");

ClassAndObject obj=**new** ClassAndObject();

obj.dis();

}

**void** dis(){

System.*out*.println("This is printed with the help of object");

}

}

1. **Write a program to demonstrate concept of calling methods**

**public** **class** MethodCalling {

**public** **int** add(**int** a,**int** b){

**return** a+b;

}

**public** **int** sub(**int** a,**int** b){

**return** a-b;

}

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

MethodCalling obj=**new** MethodCalling();

//calling method

**int** sum=obj.add(20, 10);

**int** difference=obj.sub(20, 10);

System.*out*.println("Sum="+sum);

System.*out*.println("Difference="+difference);

}

}

1. **Write a program to demonstrate concept of ‘this’ keyword.**

**class** Student{

**int** id;

String name;

**int** age;

Student(**int** id,String name,**int** age){

**this**.id=id;

**this**.name=name;

**this**.age=age;

}

**void** display(){

System.*out*.println("Id="+id+"\tName="+name+"\tAge="+age);

}

}

**public** **class** ThisKeyword {

**public** **static** **void** main(String[] args) {

Student obj=**new** Student(18,"Krish",21);

obj.display();

}

}

1. **Write a program to demonstrate concept of parameterize constructors.**

**class** StudentA{

**int** id;

String name;

**int** age;

StudentA(**int** id,String name,**int** age){

**this**.id=id;

**this**.name=name;

**this**.age=age;

}

**void** display(){

System.*out*.println("Id="+id+"\tName="+name+"\tAge="+age);

}

}

**public** **class** ParameterizedConstructor {

**public** **static** **void** main(String[] args) {

StudentA obj=**new** StudentA(18,"Krish",21);

StudentA obj1=**new** StudentA(4,"Binod",21);

obj.display();

obj1.display();

}

}

1. **Write a program to demonstrate concept of passing by value.**

**public** **class** PassByValue {

**public** **static** **void** main(String[] args) {

**int** a = 5;

System.*out*.println("\*\*\*Before change a = " + a);

PassByValue obj = **new** PassByValue();

**int** result = obj.Change(a);

System.*out*.println("\*\*\*After change a = " + a);

System.*out*.println("Returned value from Change method = " + result);

}

**int** Change(**int** b) {

System.*out*.println("Inside Change method before change b = " + b);

b = 20;

System.*out*.println("Inside Change method after change b = " + b);

**return** b;

}

}

1. **Write a program to demonstrate concept of method overloading.**

**public** **class** MethodOverloading {

**void** dis(){

System.*out*.println("No Parameter");

}

**void** dis(**int** a){

System.*out*.println(" Single parameter A="+a);

}

**void** dis(**int** a,**int** b){

System.*out*.println(" Double Parameter A+B="+(a+b));

}

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

MethodOverloading obj=**new** MethodOverloading();

obj.dis();

obj.dis(10);

obj.dis(10,20);

}

}

1. **Write a program to demonstrate concept of nested inner classes.**

**public** **class** OuterClass {

**private** String outerField="Outer Class Field";

**public** **class** InnerClass{

**private** String innerField="Inner class field ";

**public** **void** display(){

System.*out*.println("Outer Field:"+outerField);

System.*out*.println("Inner Field:"+innerField);

}

}

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

OuterClass outer=**new** OuterClass();

OuterClass.InnerClass inner=outer.**new** InnerClass();

inner.display();

}

}

1. **Recursive program to find the factorial of an integer.**

**import** java.util.Scanner;

**public** **class** Factorial {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner scanner=**new** Scanner(System.*in*);

System.*out*.println("Enter a positive integer:");

**int** number=scanner.nextInt();

**if**(number<0){

System.*out*.println("Please enter valid number");

}

**else**{

**int** result=*factorial*(number);

System.*out*.println("Factorial="+result);

}

scanner.close();

}

**public** **static** **int** factorial(**int** n){

**if**(n==0||n==1){

**return** 1;

}

**else**{

**return** n\**factorial*(n-1);

}

}

}

**Lab# 4: Inheritance & Packaging**

**1. Write a program to demonstrate concept of abstract class.**

**abstract** **class** Demo{

**abstract** **void** sum();

**abstract** **void** sub(**int** a,**int** b);

}

**public** **class** AbstractDemo **extends** Demo {

**void** sum(){

System.*out*.println("Sum="+(10+20));

}

**void** sub(**int** a,**int** b){

System.*out*.println("Subtraction="+(a-b));

}

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

AbstractDemo obj= **new** AbstractDemo();

obj.sum();

obj.sub(30, 20);

}

}

**2. Write a program to demonstrate concept of super keyword.**

// Base class

**class** Animal {

String name;

// Constructor of the base class

Animal(String name) {

**this**.name = name;

}

// Method of the base class

**void** display() {

System.*out*.println("I have a pet. Her name is " + name);

}

}

// Derived class

**class** Dog **extends** Animal {

String breed;

// Constructor of the derived class

Dog(String name, String breed) {

// Calling the base class constructor

**super**(name);

**this**.breed = breed;

}

// Method of the derived class

**void** display() {

// Calling the base class method

**super**.display();

System.*out*.println("She is a dog.Her breed is " + breed);

}

}

// Main class to test the program

**public** **class** Main {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Dog dog = **new** Dog("Hima", "Golden Retriever");

dog.display();

}

}

**3. Write a program to demonstrate concept of overriding methods.**

**class** ParentClass{

**void** dis(){

System.*out*.println("Hello from Parent class");

}

}

**class** ChildClass **extends** ParentClass{

**void** dis(){

**super**.dis();

System.*out*.println("Hello from Child Class");

}

}

**public** **class** MethodOverriding {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

ParentClass obj=**new** ChildClass();

obj.dis();

}

}

**4. Write a program to demonstrate concept of interface.**

**interface** A{

**void** dis();

}

**interface** B{

**void** access();

}

**public** **class** InterfaceDemo **implements** A,B {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

InterfaceDemo obj=**new** InterfaceDemo();

obj.dis();

obj.access();

}

**public** **void** dis(){

System.*out*.println("This is message of interface A");

}

**public** **void** access(){

System.*out*.println("This is message of interface B");

}

}

**5. Write a program to demonstrate concept of final class.**

**final** **class** Faculty{

**void** dis(){

System.*out*.println("Faculty cannot be subclassed");

}

}

**public** **class** FinalClass {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Faculty f=**new** Faculty();

f.dis();

}

}

**6. Write a program to demonstrate concept of package.**

**package** com.example.animal;

**public** **class** Dog {

**public** **void** makesound(){

System.*out*.println("DogS Bark");

}

}

**package** com.example.animal;

**public** **class** Cat {

**public** **void** makesound(){

System.*out*.println("Cats Meow");

}

}

**import** com.example.animal.Cat;

**import** com.example.animal.Dog;

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Dog dog=**new** Dog();

Cat cat=**new** Cat();

dog.makesound();

cat.makesound();

}

}

**Lab# 5: Handling Error/Exception**

**1. Write a program to demonstrate concept of exception.**

**public** **class** ExceptionDemo {

**public** **static** **void** main(String[] args) {

**try**{

System.*out*.println("Krish");

**int** result=*divide*(10,0);

System.*out*.println("Division="+result);

}

**catch** (ArithmeticException e){

System.*out*.println("Caught an exception\t"+e.getMessage());

}

}

**public** **static** **int** divide(**int** a,**int** b){

**return** a/b;

}

}

**2. Write a program to demonstrate concept of user defined exception.**

**class** InvalidAgeException **extends** Exception {

**public** InvalidAgeException(String message) {

**super**(message);

}

}

**public** **class** UserDefinedExceptionDemo {

**public** **static** **void** main(String[] args) {

**try** {

System.*out*.println("Krish");

*validateAge*(15);

} **catch** (InvalidAgeException e) {

System.*out*.println("Caught an exception: " + e.getMessage());

}

**try** {

*validateAge*(20);

System.*out*.println("Age is valid (20).");

} **catch** (InvalidAgeException e) {

System.*out*.println("Caught an exception: " + e.getMessage());

}

}

**public** **static** **void** validateAge(**int** age) **throws** InvalidAgeException {

**if** (age < 18) {

**throw** **new** InvalidAgeException("Age is less than 18");

}

}

}

**3. Write a program to demonstrate concept of throw and throws.**

**public** **class** ThrowAndThrows {

**void** div(**int** a,**int** b)**throws** ArithmeticException{

**if**(b==0){

**throw** **new** ArithmeticException();

}

**else**{

**int** d=a/b;

System.*out*.println("Division="+d);

}

}

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

ThrowAndThrows t=**new** ThrowAndThrows();

**try**{

t.div(20,0);

}

**catch**(Exception e){

System.*out*.println("The value of second integer is 0");

}

}

}

**Lab# 6: Handling Strings**

**1. Write a program to demonstrate concept of string buffer.**

**public** **class** StringBufferDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

StringBuffer sb = **new** StringBuffer();

sb.append("Hello");

sb.append(", ");

sb.append("world");

sb.append("!");

System.*out*.println(sb);

System.*out*.println(sb.reverse());

}

}

**2. Write a program to find the string is palindrome or not.**

**import** java.util.Scanner;

**public** **class** PalindromeCheck {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Scanner scan=**new** Scanner(System.*in*);

System.*out*.println("Enter the string");

String c=scan.next();

StringBuffer aa=**new** StringBuffer(c);

aa.reverse();

**if**(c.equals(aa.toString())){

System.*out*.println("Palindrome");

}

**else**

System.*out*.println("Not Palindrome");

scan.close();

}

}

**Lab# 7: Threads**

**1. Write a program to demonstrate concept of threads.**

**class** MyThread **extends** Thread {

**private** String message;

**public** MyThread(String message) {

**this**.message = message;

}

**public** **void** run() {

**for** (**int** i = 0; i < 5; i++) {

System.*out*.println(message + " - " + i);

**try** {

Thread.*sleep*(500); // Sleep for 500 milliseconds

} **catch** (InterruptedException e) {

System.*out*.println(e);

}

}

}

}

**public** **class** ThreadDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

MyThread thread1 = **new** MyThread("Thread 1");

MyThread thread2 = **new** MyThread("Thread 2");

thread1.start();

thread2.start();

}

}

**2. Write a program to demonstrate concept of runnable interface.**

**class** MyRunnable **implements** Runnable {

**private** String message;

**public** MyRunnable(String message) {

**this**.message = message;

}

**public** **void** run() {

**for** (**int** i = 0; i < 5; i++) {

System.*out*.println(message + " - " + i);

**try** {

Thread.*sleep*(500); // Sleep for 500 milliseconds

} **catch** (InterruptedException e) {

System.*out*.println(e);

}

}

}

}

**public** **class** RunnableDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

MyRunnable myRunnable1 = **new** MyRunnable("Runnable 1");

MyRunnable myRunnable2 = **new** MyRunnable("Runnable 2");

Thread thread1 = **new** Thread(myRunnable1);

Thread thread2 = **new** Thread(myRunnable2);

thread1.start();

thread2.start();

}

}

**3. Write a program to demonstrate concept of synchronization.**

**class** Counter {

**private** **int** count = 0;

// Synchronized method to increment the counter

**public** **synchronized** **void** increment() {

count++;

}

// Method to get the current count value

**public** **int** getCount() {

**return** count;

}

}

**class** MyRunnableA **implements** Runnable {

**private** Counter counter;

**public** MyRunnableA(Counter counter) {

**this**.counter = counter;

}

@Override

**public** **void** run() {

**for** (**int** i = 0; i < 1000; i++) {

counter.increment();

}

}

}

**public** **class** SynchronizationDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

Counter counter = **new** Counter();

Thread thread1 = **new** Thread(**new** MyRunnableA(counter));

Thread thread2 = **new** Thread(**new** MyRunnableA(counter));

thread1.start();

thread2.start();

**try** {

thread1.join();

thread2.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

// Print the final count value

System.*out*.println("Final count: " + counter.getCount());

}

}

**4. Write a program to demonstrate concept of deadlock.**

**public** **class** Deadlock{

String str1="Krish";

String str2="Nirjala";

Thread thread1=**new** Thread(){

**public** **void** run(){

**synchronized** (str1) {

**try**{

System.*out*.println(Thread.*currentThread*().getName()+"Locked:"+str1);

Thread.*sleep*(5000);

}

**catch**(Exception e){

e.printStackTrace();

}

**synchronized** (str2){

System.*out*.println(Thread.*currentThread*().getName()+"Locked:"+str2);

}

}

}

};

Thread thread2=**new** Thread(){

**public** **void** run(){

**synchronized** (str2) {

System.*out*.println(Thread.*currentThread*().getName()+"Locked:"+str2);

**synchronized** (str1){

System.*out*.println(Thread.*currentThread*().getName()+"Locked:"+str1);

}

}

}

};

**public** **static** **void** main(String[]args){

System.*out*.println("Krish");

Deadlock obj=**new** Deadlock();

obj.thread2.start();

obj.thread1.start();

}

}

**Lab# 8: I/O and Streams**

**1. Write a program to demonstrate concept of reading and writing files.**

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** FileReadWriteDemo {

**public** **static** **void** main(String[]args){

**try**{

FileWriter fwriter=**new** FileWriter("D:\\Java\\example.txt");

String s="My name is Krish Paudel";

fwriter.write(s);

fwriter.close();

FileReader fread=**new** FileReader("D:\\Java\\example.txt");

**int** i;

**while**((i=fread.read())!=-1){

System.*out*.println((**char**)i);

}

fread.close();

}

**catch**(IOException e){

System.*out*.println("Error="+e.getMessage());

}

}

}

**2. Write a program to demonstrate concept of Serialization.**

**import** java.io.FileOutputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.Serializable;

**class** SerializationEg **implements** Serializable{

**int** id;

String name;

SerializationEg(**int** id,String name){

**this**.id=id;

**this**.name=name;

}

}

**public** **class** SerializationDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

**try**{

SerializationEg obj=**new** SerializationEg(100,"Krish");

FileOutputStream fos=**new** FileOutputStream("D:\\Java\\serializable.txt");

ObjectOutputStream out=**new** ObjectOutputStream(fos);

out.writeObject(obj);

out.flush();

out.close();

System.*out*.println("Success");

}

**catch** (Exception e){

System.*out*.println(e.getMessage());

}

}

}

**3. Write a program to demonstrate concept of Deserialization.**

**import** java.io.FileInputStream;

**import** java.io.ObjectInputStream;

**public** **class** DeserializationDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

String filename="D:\\Java\\serializable.txt";

**try**{

FileInputStream fis=**new** FileInputStream(filename);

ObjectInputStream ois=**new** ObjectInputStream(fis);

SerializationEg obj=(SerializationEg )ois.readObject();

System.*out*.println("ID="+obj.id);

System.*out*.println("Name="+obj.name);

}

**catch** (Exception e){

System.*out*.println("Error="+e.getMessage());

}

}

}

**Lab# 9: Understanding Core Packages**

**1. Write a program to demonstrate concept of wrapper class.**

**public** **class** WrapperClassDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

**int** intValue = 10;

**double** doubleValue = 20.5;

**char** charValue = 'A';

Integer integerWrapper = Integer.*valueOf*(intValue);

Double doubleWrapper = Double.*valueOf*(doubleValue);

Character charWrapper = Character.*valueOf*(charValue);

**int** intValueUnboxed = integerWrapper.intValue();

**double** doubleValueUnboxed = doubleWrapper.doubleValue();

**char** charValueUnboxed = charWrapper.charValue();

System.*out*.println("Original values:");

System.*out*.println("int: " + intValue + ", double: " + doubleValue + ", char: " + charValue);

System.*out*.println("\nWrapper class values:");

System.*out*.println("Integer: " + integerWrapper + ", Double: " + doubleWrapper + ", Character: " + charWrapper);

System.*out*.println("\nUnboxed values:");

System.*out*.println("int: " + intValueUnboxed + ", double: " + doubleValueUnboxed + ", char: " + charValueUnboxed);

}

}

**2. Write a program to demonstrate concept of Vector.**

**import** java.util.Iterator;

**import** java.util.Vector;

**public** **class** VectorDemo {

**public** **static** **void** main(String[] args) {

Vector vector = **new** Vector();

System.*out*.println("Initial capacity of vector = " + vector.capacity());

vector.add("Krish");

vector.add("Nirjala");

vector.add(1, "Binod");

vector.add("Shraddha");

System.*out*.println("Size of vector = " + vector.size());

Iterator it = vector.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

}

}

**3. Write a program to demonstrate concept of Dictionary.**

**import** java.util.Dictionary;

**import** java.util.Hashtable;

**public** **class** DictionaryDemo {

**public** **static** **void** main(String[] args) {

Dictionary<Integer, String> d=**new** Hashtable<Integer, String>();

d.put(30, "Krish");

d.put(24, "Binod");

d.put(3, "Nirjala");

d.put(7, "Shraddha");

System.*out*.println("Size="+d.size());

System.*out*.println(d);

System.*out*.println("After removal:");

d.remove(30);

System.*out*.println("Size="+d.size());

System.*out*.println(d);

System.*out*.println("Is dictionary empty? "+d.isEmpty());

}

}

**4. Write a program to demonstrate concept of Hashtable.**

**import** java.util.Hashtable;

**public** **class** HashtableDemo {

**public** **static** **void** main(String[] args) {

Hashtable h=**new** Hashtable();

h.put(1, "Krish");

h.put(2, "Nirjala");

h.put(3, "Shraddha");

h.put("A", "Binod");

System.*out*.println(h);

System.*out*.println("Value at key 1="+h.get(1));

}

}

**5. Write a program to demonstrate concept of Random Number Generation.**

**import** java.util.Random;

**import** java.util.concurrent.ThreadLocalRandom;

**public** **class** RandomDemo {

**public** **static** **void** main(String[] args) {

Random r=**new** Random();

**int** gen=r.nextInt(10);

**int** gen2=r.nextInt(10);

System.*out*.println(gen);

System.*out*.println(gen2);

System.*out*.println(Math.*random*());

**int** ran=ThreadLocalRandom.*current*().nextInt();

System.*out*.println(ran);

}

}

**Lab# 10: Holding Collection of Data**

**1. Write a program to demonstrate concept of ArrayList.**

**import** java.util.ArrayList;

**public** **class** ArrayListDemo {

**public** **static** **void** main(String[] args) {

ArrayList a=**new** ArrayList();

a.add("Krish");

a.add("Nirjala");

a.add("Shraddha");

a.add("Binod");

a.add(100);

System.*out*.println("ArrayList elements:"+a);

System.*out*.println("Element at index 0:"+a.get(0));

a.remove("Binod");

System.*out*.println("New array="+a);

}

}

**2. Write a program to demonstrate concept of Linked.**

**import** java.util.LinkedList;

**public** **class** LinkedListDemo {

**public** **static** **void** main(String[] args) {

LinkedList<String> ll=**new** LinkedList<String>();

ll.add("Krish");

ll.add("Binod");

ll.add("Puja");

ll.add("Pramish");

System.*out*.println("LinkedList elements:"+ll);

System.*out*.println("Element at index 0:"+ll.get(0));

ll.addFirst("Mohan");

System.*out*.println("New LinkedList after adding element in first place:"+ll);

ll.remove("Krish");

System.*out*.println("New LinkedList after removing Krish:"+ll);

}

}

**3. Write a program to demonstrate concept of List interface.**

**import** java.util.List;

**import** java.util.Stack;

**public** **class** StackDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

List stack=**new** Stack();

stack.add("Java");

stack.add("SAD");

stack.add("Web");

System.*out*.println("Elements of List:"+stack);

((Stack) stack).push("DSA");

((Stack)stack).push("Statistics");

System.*out*.println("Elements of List after pushing :"+stack);

System.*out*.println("Popped element:"+((Stack)stack).pop());

System.*out*.println("Elements after popping an element:"+stack);

System.*out*.println("Is the Stack empty? " + ((Stack<String>) stack).empty());

}

}

**4. Write a program to demonstrate concept of Hashset.**

**import** java.util.HashSet;

**public** **class** HashSetDemo {

**public** **static** **void** main(String[] args) {

HashSet<String> h=**new** HashSet<String>();

h.add("Krish");

h.add("Puja");

h.add("Mohan");

h.add("Pramish");

System.*out*.println("Elements of hashset:"+h);

**boolean** isAdded=h.add("Krish");

System.*out*.println("Elements of hashset:"+h);

System.*out*.println("Was 'Krish' added again? "+isAdded);

h.clear();

System.*out*.println("Elements after clearing the hashset:"+h);

}

}

**5. Write a program to demonstrate concept of Comparator.**

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**class** StudentAA{

String name;

**float** salary;

StudentAA(String name,**float** salary){

**this**.name=name;

**this**.salary=salary;

}

}

**class** Comparatorsalary **implements** Comparator<StudentAA>{

**public** **int** compare(StudentAA obj1,StudentAA obj2){

**if** (obj1.salary<obj2.salary)

**return** 1;

**return** -1;

}

}

**public** **class** ComparatorDemo {

**public** **static** **void** main(String[] args) {

System.*out*.println("Krish");

ArrayList<StudentAA> s=**new** ArrayList<StudentAA>();

s.add(**new** StudentAA("Prami",5000));

s.add(**new** StudentAA("Poo",4000));

s.add(**new** StudentAA("Raha",10000));

Comparator<StudentAA> com=**new** Comparatorsalary();

Collections.*sort*(s,com);

**for**(StudentAA ss:s){

System.*out*.println(ss.name+"-->"+ss.salary);

}

}

}

**Lab# 11: Java Applications**

**1) Write a program to create Swing application that receive two numbers through a JTextFeilds and display the multiplication two numbers in a JTextField when the OK button is pressed and when exit button is pressed the program will terminate.**

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**public** **class** MultiplicationApp {

// Declare text fields as instance variables

**private** **static** JTextField *textField1*;

**private** **static** JTextField *textField2*;

**private** **static** JTextField *textFieldResult*;

**public** **static** **void** main(String[] args) {

// Create the frame

JFrame frame = **new** JFrame("Multiplication Application");

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

frame.setSize(400, 200);

frame.setLayout(**new** GridLayout(5, 2));

// Create labels

JLabel label1 = **new** JLabel("Number 1:");

JLabel label2 = **new** JLabel("Number 2:");

JLabel labelResult = **new** JLabel("Result:");

// Initialize text fields

*textField1* = **new** JTextField();

*textField2* = **new** JTextField();

*textFieldResult* = **new** JTextField();

*textFieldResult*.setEditable(**false**);

// Create buttons

JButton buttonOk = **new** JButton("OK");

JButton buttonExit = **new** JButton("Exit");

// Add action listener for OK button

buttonOk.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

**try** {

// Get numbers from text fields

**double** num1 = Double.*parseDouble*(*textField1*.getText());

**double** num2 = Double.*parseDouble*(*textField2*.getText());

// Calculate multiplication

**double** result = num1 \* num2;

// Display result in result text field

*textFieldResult*.setText(String.*valueOf*(result));

} **catch** (NumberFormatException ex) {

// Display error message in result text field

*textFieldResult*.setText("Invalid input");

}

}

});

// Add action listener for Exit button

buttonExit.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

System.*exit*(0);

}

});

// Add components to frame

frame.add(label1);

frame.add(*textField1*);

frame.add(label2);

frame.add(*textField2*);

frame.add(labelResult);

frame.add(*textFieldResult*);

frame.add(buttonOk);

frame.add(buttonExit);

// Set the frame to be visible

frame.setVisible(**true**);

}

}

**2) Write a program to demonstrate concept of layout manager.**

**import** javax.swing.\*;

**import** java.awt.\*;

**public** **class** LayoutManagerDemo {

**public** **static** **void** main(String[] args) {

// Create the main frame

JFrame frame = **new** JFrame("Layout Manager Demo");

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

frame.setSize(600, 400);

frame.setLayout(**new** BorderLayout());

// BorderLayout Panel

JPanel borderPanel = **new** JPanel(**new** BorderLayout());

borderPanel.setBorder(BorderFactory.*createTitledBorder*("BorderLayout"));

borderPanel.add(**new** JButton("North"), BorderLayout.*NORTH*);

borderPanel.add(**new** JButton("South"), BorderLayout.*SOUTH*);

borderPanel.add(**new** JButton("East"), BorderLayout.*EAST*);

borderPanel.add(**new** JButton("West"), BorderLayout.*WEST*);

borderPanel.add(**new** JButton("Center"), BorderLayout.*CENTER*);

// FlowLayout Panel

JPanel flowPanel = **new** JPanel(**new** FlowLayout());

flowPanel.setBorder(BorderFactory.*createTitledBorder*("FlowLayout"));

flowPanel.add(**new** JButton("Button 1"));

flowPanel.add(**new** JButton("Button 2"));

flowPanel.add(**new** JButton("Button 3"));

flowPanel.add(**new** JButton("Button 4"));

flowPanel.add(**new** JButton("Button 5"));

// GridLayout Panel

JPanel gridPanel = **new** JPanel(**new** GridLayout(2, 3));

gridPanel.setBorder(BorderFactory.*createTitledBorder*("GridLayout"));

gridPanel.add(**new** JButton("1"));

gridPanel.add(**new** JButton("2"));

gridPanel.add(**new** JButton("3"));

gridPanel.add(**new** JButton("4"));

gridPanel.add(**new** JButton("5"));

gridPanel.add(**new** JButton("6"));

// BoxLayout Panel

JPanel boxPanel = **new** JPanel();

boxPanel.setBorder(BorderFactory.*createTitledBorder*("BoxLayout"));

boxPanel.setLayout(**new** BoxLayout(boxPanel, BoxLayout.*Y\_AXIS*));

boxPanel.add(**new** JButton("Button A"));

boxPanel.add(Box.*createVerticalStrut*(10));

boxPanel.add(**new** JButton("Button B"));

boxPanel.add(Box.*createVerticalStrut*(10));

boxPanel.add(**new** JButton("Button C"));

boxPanel.add(Box.*createVerticalStrut*(10));

boxPanel.add(**new** JButton("Button D"));

// Add panels to the main frame using BorderLayout

frame.add(borderPanel, BorderLayout.*NORTH*);

frame.add(flowPanel, BorderLayout.*WEST*);

frame.add(gridPanel, BorderLayout.*CENTER*);

frame.add(boxPanel, BorderLayout.*EAST*);

// Set the frame to be visible

frame.setVisible(**true**);

}

}

**3) Write a program to change background color of a frame using action event.**

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**public** **class** BackgroundColorChange {

**public** **static** **void** main(String[] args) {

// Create the main frame

JFrame frame = **new** JFrame("Background Color Change Krish");

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

frame.setSize(400, 300);

// Create a panel to hold the content

**final** JPanel panel = **new** JPanel(); // Declare panel as final

// Create a button to change the background color

JButton button = **new** JButton("Change Color");

button.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// Generate a random color

Color randomColor = **new** Color(

(**int**) (Math.*random*() \* 256), // Red component

(**int**) (Math.*random*() \* 256), // Green component

(**int**) (Math.*random*() \* 256)); // Blue component

// Set the background color of the panel

panel.setBackground(randomColor);

}

});

// Add the button to the frame

frame.add(button, BorderLayout.*NORTH*);

// Add a panel to the center to display the color

panel.setBackground(Color.*WHITE*); // Initial background color

frame.add(panel, BorderLayout.*CENTER*);

// Set the frame to be visible

frame.setVisible(**true**);

}

}

**4) Write a program to create a table using Jtable class and then add it to the Jframe container.**

**import** javax.swing.\*;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**public** **class** JTableExample {

**public** **static** **void** main(String[] args) {

// Create a JFrame

JFrame frame = **new** JFrame("JTable Example");

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

frame.setSize(600, 400);

// Create data for the table

Object[][] data = {

{"Pramish", 18, "Male"},

{"Krish", 20, "Female"},

{"Mohan", 45, "Male"},

{"Puja", 40, "Female"}

};

// Create column names

String[] columnNames = {"Name", "Age", "Gender"};

// Create a DefaultTableModel

DefaultTableModel model = **new** DefaultTableModel(data, columnNames);

// Create a JTable with the model

JTable table = **new** JTable(model);

// Set table preferences (optional)

table.setPreferredScrollableViewportSize(**new** Dimension(500, 300));

table.setFillsViewportHeight(**true**);

// Create a JScrollPane and add table to it

JScrollPane scrollPane = **new** JScrollPane(table);

// Add JScrollPane to the JFrame

frame.add(scrollPane, BorderLayout.*CENTER*);

// Set JFrame visibility

frame.setVisible(**true**);

}

}

**Lab# 12: Introduction to Java Applets**

1. **Write a program to demonstrate concept of lifecycle of applet.**

**import** java.applet.Applet;

**import** java.awt.\*;

**public** **class** AppletLifecycleDemo **extends** Applet {

// Initialization method

**public** **void** init() {

System.*out*.println("Init method called");

}

// Starting method

**public** **void** start() {

System.*out*.println("Start method called");

}

// Painting method

**public** **void** paint(Graphics g) {

System.*out*.println("Paint method called");

g.drawString("Applet Lifecycle Demo", 20, 20);

}

// Stopping method

**public** **void** stop() {

System.*out*.println("Stop method called");

}

// Destruction method

**public** **void** destroy() {

System.*out*.println("Destroy method called");

}

}

**Html:**

<!DOCTYPE html>

<html>

<head>

<title>Applet Lifecycle Demo</title>

</head>

<body>

<applet code="AppletLifecycleDemo.class" width="300" height="200">

Your browser does not support Java applets.

</applet>

</body>

</html>

**Lab# 13: Database Programming using JDBC**

1. **Write a program to display student details from student** **table.**

**import** java.sql.\*;

**public** **class** DisplayStudentDetails {

// JDBC URL, username and password of MySQL server

**private** **static** **final** String *JDBC\_URL* = "jdbc:mysql://localhost:3306/students\_db";

**private** **static** **final** String *JDBC\_USER* = "root";

**private** **static** **final** String *JDBC\_PASSWORD* = "admin";

**public** **static** **void** main(String[] args) {

Connection conn = **null**;

Statement stmt = **null**;

ResultSet rs = **null**;

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

System.*out*.println("Connecting to database...");

conn = DriverManager.*getConnection*(*JDBC\_URL*, *JDBC\_USER*, *JDBC\_PASSWORD*);

System.*out*.println("Creating statement...");

stmt = conn.createStatement();

String sql = "SELECT id, name, age, grade FROM students";

rs = stmt.executeQuery(sql);

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

**while** (rs.next()) {

**int** id = rs.getInt("id");

String name = rs.getString("name");

**int** age = rs.getInt("age");

String grade = rs.getString("grade");

System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age + ", Grade: " + grade);

}

} **catch** (SQLException se) {

se.printStackTrace();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**try** {

**if** (rs != **null**) rs.close();

**if** (stmt != **null**) stmt.close();

**if** (conn != **null**) conn.close();

} **catch** (SQLException se) {

se.printStackTrace();

}

}

System.*out*.println("Goodbye!");

}

}