**Program: -6**

**Objective**: Write a Program in java to illustrate the concept of local, instance and static variable.

**Code:**

*public* *class* variables {

*static* int staticVar = 10;

    int instanceVar;

*public* variables(int instanceVar) {

        this.instanceVar = instanceVar;

    }

*public* void demonstrateLocalVariable() {

        int localVar = 5;

        System.out.println("Local variable: " + localVar);

    }

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

        System.out.println("Static variable: " + staticVar);

        variables example = new variables(20);

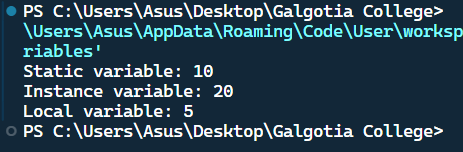
        System.out.println("Instance variable: " + example.instanceVar);

        example.demonstrateLocalVariable();

    }

}

**Output:**



**Program: -7**

**Objective**: WAP in java to implement implicit and explicit type casting.

**Code:**

*public* *class* typeCasting {

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

*// Implicit type casting*

        int i = 10;

        long l = i;

*// Explicit type casting*

        double d = 10.5;

        int j = (int) d;

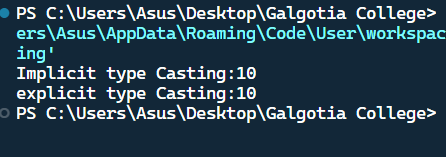
        System.out.println("Implicit type Casting:" + l);

        System.out.println("explicit type Casting:"+ j);

    }

}

**Output:**

****

**Program: -8**

**Objective**: WAP in java for implement the various operators in java;

**Code:**

*public* *class* Operators {

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

        int a = 10, b = 5;

*// Arithmetic Operators*

        System.out.println("Arithmetic Operators:");

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

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

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

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

        System.out.println("a % b = " + (a % b));

*// Relational Operators*

        System.out.println("\nRelational Operators:");

        System.out.println("a == b: " + (a == b));

        System.out.println("a != b: " + (a != b));

        System.out.println("a > b: " + (a > b));

        System.out.println("a < b: " + (a < b));

        System.out.println("a >= b: " + (a >= b));

        System.out.println("a <= b: " + (a <= b));

*// Logical Operators*

        System.out.println("\nLogical Operators:");

        System.out.println("(a > b) && (a < 15): " + ((a > b) && (a < 15)));

        System.out.println("(a > b) || (b < 2): " + ((a > b) || (b < 2)));

        System.out.println("!(a == b): " + !(a == b));

*// Bitwise Operators*

        System.out.println("\nBitwise Operators:");

        System.out.println("a & b: " + (a & b));

        System.out.println("a | b: " + (a | b));

        System.out.println("a ^ b: " + (a ^ b));

        System.out.println("~a: " + (~a));

        System.out.println("a << 2: " + (a << 2));

        System.out.println("a >> 2: " + (a >> 2));

*// Assignment Operators*

        System.out.println("\nAssignment Operators:");

        a += b;

        System.out.println("a += b: " + a);

        a -= b;

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

        a \*= b;

        System.out.println("a \*= b: " + a);

        a /= b;

        System.out.println("a /= b: " + a);

        a %= b;

        System.out.println("a %= b: " + a);

*// Unary Operators*

        System.out.println("\nUnary Operators:");

        System.out.println("++a: " + (++a));

        System.out.println("--a: " + (--a));

        System.out.println("a++: " + (a++));

        System.out.println("a--: " + (a--));

*// Ternary Operator*

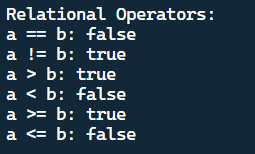
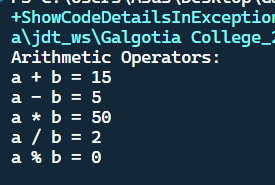
        System.out.println("\nTernary Operator:");

        System.out.println("a > b ? a : b: " + (a > b ? a : b));

    }

}

**Output:**



**Program: -9**

**Objective**: WAP in java for constructor overloading.

**Code:**

*public* *class* ConstructorOverloadingExample {

    int id;

    String name;

    int age;

*public* ConstructorOverloadingExample() {

        this.id = 0;

        this.name = "Unknown";

        this.age = 0;

    }

*public* ConstructorOverloadingExample(int id) {

        this.id = id;

        this.name = "Unknown";

        this.age = 0;

    }

*public* ConstructorOverloadingExample(int id, String name) {

        this.id = id;

        this.name = name;

        this.age = 0;

    }

*public* ConstructorOverloadingExample(int id, String name, int age) {

        this.id = id;

        this.name = name;

        this.age = age;

    }

*public* void display() {

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

    }

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

*// Creating objects using different constructors*

        ConstructorOverloadingExample obj1 = new ConstructorOverloadingExample();

        ConstructorOverloadingExample obj2 = new ConstructorOverloadingExample(1);

        ConstructorOverloadingExample obj3 = new ConstructorOverloadingExample(2, "Alice");

        ConstructorOverloadingExample obj4 = new ConstructorOverloadingExample(3, "Bob", 25);

      obj1.display();

        obj2.display();

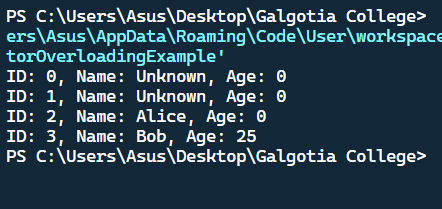
        obj3.display();

        obj4.display();

    }

}

**Output:**



**Program: -10**

**Objective**: WAP in java for method Overloading.

**Code:**

*public* *class* methodOverloading {

*public* void display() {

        System.out.println("display method ");

    }

*public* void display(int a) {

        System.out.println("the value is :" + a);

    }

*public* void display(String name) {

        System.out.println("the name is :" + name);

    }

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

        methodOverloading T = new methodOverloading();

        T.display();

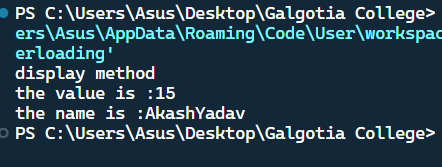
        T.display(15);

        T.display("AkashYadav");

    }

}

**Output:**

****

**Program: -11**

**Objective**: Wap in java for method Overriding.

**Code:**

*class* parent {

    void display(String name) {

        System.out.println(name);

    }

}

*class* child *extends* parent {

    @*Override*

    void display(String name) {

*// TODO Auto-generated method stub*

        super.display("Hello " + name);

    }

}

*public* *class* methodOver {

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

        parent p = new parent();

        child c = new child();

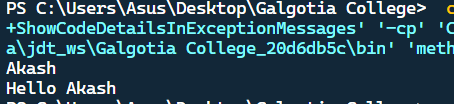
        p.display("Akash");

        c.display("Akash");

    }

}

**Output:**



**Program: -12**

**Objective**: Wap in java for run time polymorphism(up casting).

**Code:**

*class* Animal {

*public* void makeSound() {

        System.out.println("Animal makes a sound");

    }

}

*class* Dog *extends* Animal {

    @*Override*

*public* void makeSound() {

        System.out.println("Dog barks");

    }

}

*class* Cat *extends* Animal {

    @*Override*

*public* void makeSound() {

        System.out.println("Cat meows");

    }

}

*public* *class* polymo {

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

        Animal myDog = new Dog();

        Animal myCat = new Cat();

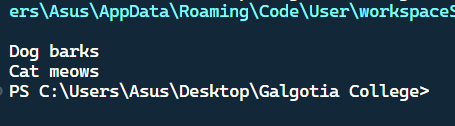
        myDog.makeSound(); *// Outputs: Dog barks*

        myCat.makeSound(); *// Outputs: Cat meows*

    }

}

**Output:**



**Program: -13**

**Objective**: Wap in java for access specifier(all four).

**Code:**

*class* AccessSpecifierDemo {

String defaultMessage = "Default Access";

*private* String privateMessage = "Private Access";

*protected* String protectedMessage = "Protected Access";

*public* String publicMessage = "Public Access";

*public* void printMessages() {

        System.out.println(defaultMessage);

        System.out.println(privateMessage);

        System.out.println(protectedMessage);

        System.out.println(publicMessage);

    }

}

*public* *class* Specifier {

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

      AccessSpecifierDemo demo =new AccessSpecifierDemo();

        demo.printMessages();

        System.out.println(demo.defaultMessage);

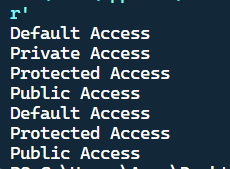
        System.out.println(demo.protectedMessage);

        System.out.println(demo.publicMessage);

    }

}

**Output:**



**Program: -14**

**Objective**: WAP in java to implement the single dimension array.

**Code:**

import *java.util.Scanner*;

*public* *class* Array {

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

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the size of array");

        int size = sc.nextInt();

        int[] array = new int[size];

        System.out.println("Enter the Element of array");

        for (int i = 0; i < size; i++) {

            array[i] = sc.nextInt();

        }

        System.out.println("The element of array is");

        for (int num : array) {

            System.out.print(num+" ");

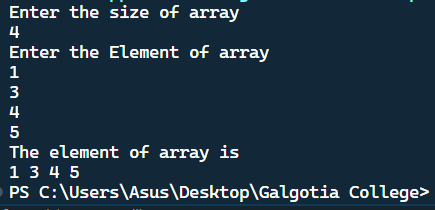
        }

        sc.close();

    }

}

**Output:**



**Program: -15**

**Objective**: WAP in java to copy the element from one array to another array.

**Code:**

*public* *class* ArrayCopyDemo {

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

        int[] sourceArray = {1, 2, 3, 4, 5};

        int[] destinationArray = new int[sourceArray.length];

        for (int i = 0; i < sourceArray.length; i++) {

            destinationArray[i] = sourceArray[i];

        }

        System.out.println("Source Array: ");

        for (int element : sourceArray) {

            System.out.print(element + " ");

        }

        System.out.println("\nDestination Array: ");

        for (int element : destinationArray) {

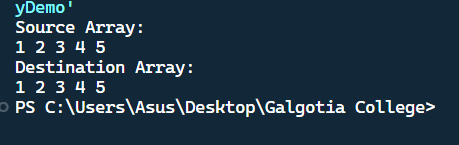
            System.out.print(element + " ");

        }

    }

}

**Output:**



**Program: -16**

**Objective**: WAP in java to perform the addition and multiplication in 2-d array.

**Code:**

*public* *class* TwoDimensionArray {

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

*// Define and initialize a 2D array*

        int[][] array = {  {1, 2, 3}, {4, 5, 6},  {7, 8, 9}   };

        int sum = 0;

        for (int i = 0; i < array.length; i++) {

            for (int j = 0; j < array[i].length; j++) {

                sum += array[i][j];

            }

        }

        int product = 1;

        for (int i = 0; i < array.length; i++) {

            for (int j = 0; j < array[i].length; j++) {

                product \*= array[i][j];

            }

        }

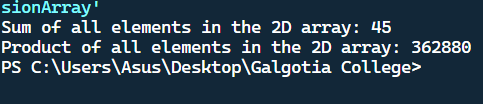
        System.out.println("Sum of all elements in the 2D array: " + sum);

        System.out.println("Product of all elements in the 2D array: " + product);

    }

}

**Output:**



**Program: -17**

**Objective**: WAP in java to show the simple inheritence.

**Code:**

*class* Animal {

    void eat() {

        System.out.println("eating...");

    }

}

*class* Dog *extends* Animal {

    void bark() {

        System.out.println("barking...");

    }

}

*public* *class* inheritence {

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

        Dog d = new Dog();

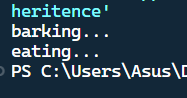
        d.bark();

        d.eat();

    }

}

**Output:**



**Program: -18**

**Objective**: WAP in java to Final keyword.

**Code:**

*public* *class* Final {

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

*// Define a constant variable PI*

*final* double PI = 3.14159;

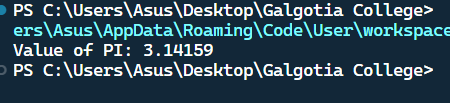
*// Print the value of PI*

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

    }

}

**Output:**



**Program: -19**

**Objective**: WAP in java for super keyword.

**Code:**

*class* Anim {

    String color = "white";

}

*class* dog *extends* Anim {

    String color = "black";

    void printColor() {

        System.out.println(color);*// prints color of Dog class*

        System.out.println(super.color);*// prints color of Animal class*

    }

}

*public* *class* Super {

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

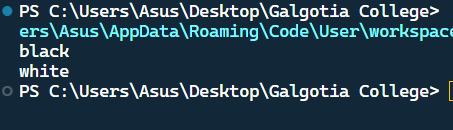
        dog d = new dog();

        d.printColor();

    }

}

**Output:**



**Program: -20**

**Objective**: WAP in java chaining constructor.

**Code:**

*class* Example {

*// Default constructor*

*public* Example() {

        this("Default message"); *// Call the parameterized constructor*

        System.out.println("Default constructor called");

    }

*// Parameterized constructor*

*public* Example(String message) {

        System.out.println("Parameterized constructor called with message: " + message);

    }

}

*public* *class* chainingConstructor {

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

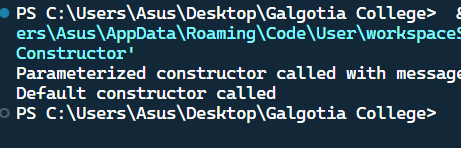
*// Create an object of the Example class*

        Example example = new Example();

    }

}

**Output:**



**Program: -21**

**Objective**: WAP in java for abstract method and abstract class.

**Code:**

*abstract* *class* Shape {

*abstract* void draw();

}

*class* Circle *extends* Shape {

    void draw() {

        System.out.println("Drawing Circle");

    }

}

*class* Rectangle *extends* Shape {

    void draw() {

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

    }

}

*public* *class* abstractClass {

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

        Shape circle = new Circle();

        Shape rectangle = new Rectangle();

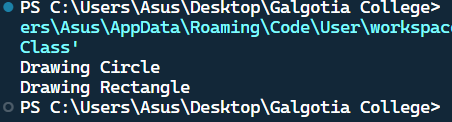
        circle.draw();

        rectangle.draw();

    }

}

**Output:**



**Program: -22**

**Objective**: WAP in java for interface.

**Code:**

*interface* *Drawable* {

    void draw();

}

*class* Circle *implements* *Drawable* {

*public* void draw() {

        System.out.println("Drawing Circle");

    }

}

*public* *class* interf {

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

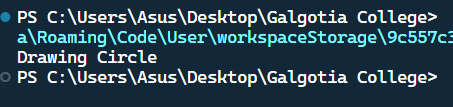
        Circle cr = new Circle();

        cr.draw();

    }

}

**Output:**



**Program: -23**

**Objective**: WAP in java multiple inheritence.

**Code:**

Java does not support multiple inheritance directly with classes, but it does support multiple inheritance with interfaces. Here’s a concise example demonstrating multiple inheritance using interfaces in Java:

*interface* *A* { void methodA(); }

*interface* *B* { void methodB(); }

*class* C *implements* *A*, *B* {

*public* void methodA() { System.out.println("Method A"); }

*public* void methodB() { System.out.println("Method B"); }

}

*public* *class* multiple {

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

        C obj = new C();

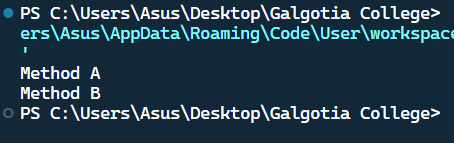
        obj.methodA();

        obj.methodB();

    }

}

**Output:**



**Program: -24**

**Objective**: WAP in java for object Cloning(shallow and deep copy).

**Code:**

*class* Address *implements* *Cloneable* {

    String city;

    Address(String city) { this.city = city; }

*protected* Object clone() *throws* CloneNotSupportedException { return super.clone(); }

}

*class* Person *implements* *Cloneable* {

    String name;

    Address address;

    Person(String name, Address address) { this.name = name; this.address = address; }

*protected* Object clone() *throws* CloneNotSupportedException {

        Person cloned = (Person) super.clone();

        cloned.address = (Address) address.clone();

        return cloned;

    }

}

*public* *class* objectclone {

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

        Address addr = new Address("New York");

        Person p1 = new Person("John", addr);

        Person p2 = (Person) p1.clone();

        p2.address.city = "San Francisco";

        System.out.println(p1.address.city);

    }

}

**Output: **

**Program: -25**

**Objective**: WAP in java for inner class(all types).

**Code:**

*public* *class* OuterClass {

*private* String outerField = "Outer field";

*private* *static* String staticOuterField = "Static outer field";

*class* MemberInnerClass {

        void display() {

            System.out.println("Accessing outer class field from Member Inner Class: " + outerField);

        }

    }

*static* *class* StaticNestedClass {

        void display() {

            System.out.println("Accessing outer class static field from Static Nested Class: " + staticOuterField);

        }

    }

    void methodWithLocalInnerClass() {

*class* LocalInnerClass {

            void display() {

                System.out.println("Accessing outer class field from Local Inner Class: " + outerField);

            }

        }

        LocalInnerClass localInner = new LocalInnerClass();

        localInner.display();

    }

    void methodWithAnonymousInnerClass() {

*Greeting* greeting = new Greeting() {

            @*Override*

*public* void greet() {

                System.out.println("Hello from the Anonymous Inner Class!");

            }

        };

        greeting.greet();

    }

*interface* *Greeting* {

        void greet();

    }

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

        OuterClass outer = new OuterClass();

        MemberInnerClass memberInner = outer.new MemberInnerClass();

        memberInner.display();

        StaticNestedClass staticNested = new StaticNestedClass();

        staticNested.display();

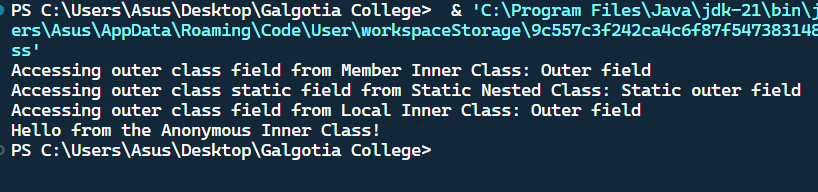
        outer.methodWithLocalInnerClass();

        outer.methodWithAnonymousInnerClass();

    }

}

**Output:**



**Program: -26**

**Objective**: WAP in java to create user defined package.

**Code:**

**MyPackage**

**|->Package.java-**

package *MyPackage*; *//user defined package*

*public* *class* Package {

*public* void displayMessage() {

        System.out.println("Hello Java World Package Imported Successfully.....");

    }

}

**Program 26-**

import *MyPackage.Package*; *//user defined package*

*public* *class* p26 {

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

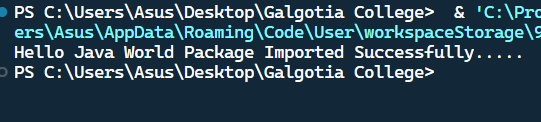
        Package myClass = new Package();

        myClass.displayMessage();

    }

}

**Output:**

****

**Program: -27**

**Objective**: WAP in java for exception handling by using try catch, finally.

**Code:**

*public* *class* P27{

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

        int numerator = 60;

        int denominator = 30;

        int result = 0;

 try {

result = numerator / denominator;

System.out.println("Result: " + result);

        } catch (ArithmeticException e) {

         System.out.println("Exception caught: " + e.getMessage());

        } finally {

           System.out.println("Finally block executed.");

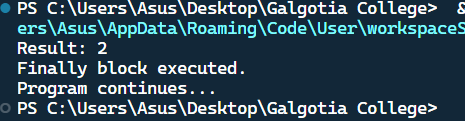
        }

System.out.println("Program continues...");

    }

}

**Output:**

****

**Program: -28**

**Objective**: WAP in java for throw and thows exception.

**Code:**

*class* InvalidAgeException *extends* Exception {

*public* InvalidAgeException(String message) {

        super(message); }

}

*public* *class* P28 {

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

        if (age < 18) {

throw new InvalidAgeException("Age is not valid, must be 18 or older.");

        } else {

            System.out.println("Age is valid."); }

    }

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

int[] testAges = {15, 20, 17, 21};

System.out.println("Ages are:- \n"+ testAges[0]+"\n"+testAges[1]+"\n"+testAges[2]+"\n"+testAges[3]+"\nResults:-");

  for (int age : testAges) {

try { validateAge(age);

            } catch (InvalidAgeException e) {

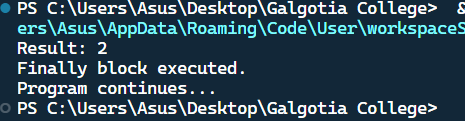
     System.out.println("Exception caught: " + e.getMessage());

            }

        }

    }

}

**Output: **

**Program: -29**

**Objective**: WAP in java for throw your own exception.

**Code:**

*class* MyCustomException *extends* Exception {

*public* MyCustomException(String message) {

        super(message);

    }

}

*public* *class* P29 {

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

        try {

            validateAge(15);

        } catch (MyCustomException e) {

            System.out.println("Exception caught: " + e.getMessage());

        }

    }

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

        if (age < 18) {

            throw new MyCustomException("Age is less than 18, access denied.");

        } else {

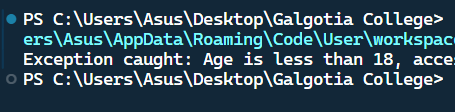
            System.out.println("Access granted.");

        }

    }

}

**Output:**

****

**Program: -30**

**Objective**: WAP in java to reading and writing in file System using byte stream.

**Code:**

import *java.io.FileOutputStream*;

import *java.io.FileInputStream*;

import *java.io.IOException*;

*public* *class* P30 {

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

        String data = "Hello, this is a test message!";

        byte[] dataBytes = data.getBytes();

        try (FileOutputStream fos = new FileOutputStream("output.txt")) {

            fos.write(dataBytes);

            System.out.println("Data written successfully.");

        } catch (IOException e) {

            System.out.println("An error occurred during writing to the file: " + e.getMessage());

        }

        try (FileInputStream fis = new FileInputStream("output.txt")) {

            int byteData;

            while ((byteData = fis.read()) != -1) {

                System.out.print((char) byteData);

            }

            System.out.println(); *// For a new line after file content*

        } catch (IOException e) {

            System.out.println("An error occurred during reading from the file: " + e.getMessage());

        }

    }

} **Output: **

**Program: -31**

**Objective**: WAP in java reading writing in file system using character stream

**Code:**

import *java.io.FileWriter*;

import *java.io.FileReader*;

import *java.io.IOException*;

*public* *class* P31 {

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

    String data = "Hello, this is a test message!";

        try (FileWriter writer = new FileWriter("example.txt")) {

            writer.write(data);

            System.out.println("Data has been written to the file.");

        } catch (IOException e) {

            System.err.println("An IOException occurred during writing: " + e.getMessage());

        }

        try (FileReader reader = new FileReader("example.txt")) {

            int character;

            System.out.print("Data read from the file: ");

            while ((character = reader.read()) != -1) {

                System.out.print((char) character);

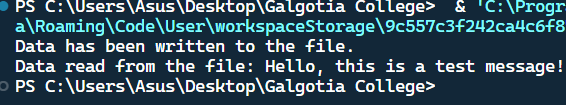
            }

        } catch (IOException e) {

            System.err.println("An IOException occurred during reading: " + e.getMessage());

        }

    }

} **Output: **

**Program: -32**

**Objective**: WAP in java to reading writing through console class.

**Code:**

import *java.io.Console*;

*public* *class* P32 {

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

*// Obtain the Console instance*

        Console console = System.console();

        if (console == null) {

            System.out.println("No console available");

            return;

        }

*// Read a line from the console*

        String name = console.readLine("Enter your name: ");

*// Read a password from the console (input won't be displayed)*

        char[] passwordArray = console.readPassword("Enter your password: ");

        String password = new String(passwordArray);

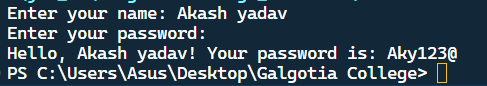
*// Display the read information*

        console.printf("Hello, %s! Your password is: %s%n", name, password);

    }

}

**Output:**

****

**Program: -33**

**Objective**: WAP how to create thread using thread class.

**Code:**

*class* MyThread *extends* Thread {

*public* void run() {

*// Code to execute in the new thread*

        for (int i = 0; i < 5; i++) {

            System.out.println("Thread: " + i);

            try {

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

            } catch (InterruptedException e) {

                System.out.println(e);

            }

        }}}

*public* *class* P33 {

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

*// Create an instance of MyThread*

        MyThread thread = new MyThread();

*// Start the thread*

        thread.start();

*// Main thread work*

        for (int i = 0; i < 5; i++) {

            System.out.println("Main: " + i);

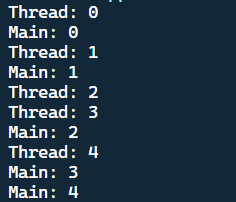
            try {

                Thread.sleep(1000); *// Sleep for 1000 milliseconds*

            } catch (InterruptedException e) {

                System.out.println(e);

            }

****        }

    }

}

**Output:**

**Program: -34**

**Objective**: WAP how to create thread using runnable interface.

**Code:**

*class* MyRunnable *implements* *Runnable* {

    @*Override*

*public* void run() {

        for (int i = 0; i < 5; i++) {

            System.out.println(Thread.currentThread().getName() + " - " + i);

            try {

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

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

}

*public* *class* P34 {

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

        MyRunnable myRunnable = new MyRunnable();

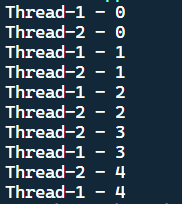
        Thread thread1 = new Thread(myRunnable, "Thread-1");

        Thread thread2 = new Thread(myRunnable, "Thread-2");

        thread1.start();

        thread2.start();

    }}

****

**Output:**

**Program: -35**

**Objective**: WAP how to implementing multithreading.

**Code:**

*class* MyThread *extends* Thread {

*public* void run() {

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

            System.out.println(Thread.currentThread().getName() + " - Value: " + i);

            try {

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

            } catch (InterruptedException e) {

                System.out.println(e);

            }

        }

    }

}

*public* *class* P35 {

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

        MyThread t1 = new MyThread();

        MyThread t2 = new MyThread();

        t1.setName("Thread 1");

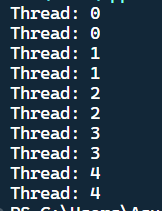
        t2.setName("Thread 2");

        t1.start();

        t2.start();

    }

}

****

**Output:**

**Program: -36**

**Objective**: WAP how to achieve synchronization using thread.

**Code:**

*class* Counter {

*private* int count = 0;

*public* *synchronized* void increment() {count++;  }

*public* int getCount() {  return count;}

}

*class* CounterThread *extends* Thread {

*private* Counter counter;

*public* CounterThread(Counter counter) { this.counter = counter;}

    @*Override*

*public* void run() {

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

            counter.increment();

        } } }

*public* *class* P36 {

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

        Counter counter = new Counter();

        CounterThread thread1 = new CounterThread(counter);

        CounterThread thread2 = new CounterThread(counter);

        CounterThread thread3 = new CounterThread(counter);

        thread1.start();

        thread2.start();

        thread3.start();

        try {

            thread1.join();

            thread2.join();

            thread3.join();

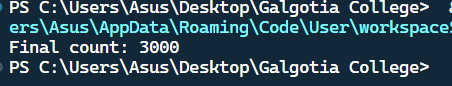
        } catch (InterruptedException e) {

            e.printStackTrace();

        }

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

 }}

**Output: **

**Program: -37**

**Objective**: WAP how to implement the concept of priorities in threads.

**Code:**

*class* CustomThread *extends* Thread {

*public* CustomThread(String name) {

        super(name);

    }

@*Override*

*public* void run() {

        for (int i = 0; i < 5; i++) {

            System.out.println(getName() + " with priority " + getPriority() + " is running. Iteration: " + i);

            try {

*// Sleep for a while to simulate some work and make the output readable*

                Thread.sleep(100);

            } catch (InterruptedException e) {

                System.out.println(e);

            }

        }

    }

}

*public* *class* P37 {

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

        CustomThread thread1 = new CustomThread("Thread-1");

        CustomThread thread2 = new CustomThread("Thread-2");

        CustomThread thread3 = new CustomThread("Thread-3");

*// Set priorities*

        thread1.setPriority(Thread.MIN\_PRIORITY); *// Priority 1*

        thread2.setPriority(Thread.NORM\_PRIORITY); *// Priority 5*

        thread3.setPriority(Thread.MAX\_PRIORITY); *// Priority 10*

*// Start threads*

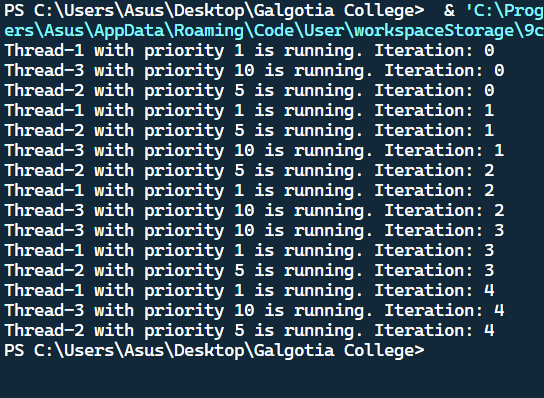
        thread1.start();

        thread2.start();

        thread3.start();

    }}

**Output:**

****

**Program: -38**

**Objective**: WAP how to implement the concept of generic programming.

**Code:**

*public* *class* P38<T> {

*private* T data;

*public* P38(T data) {

        this.data = data;

    }

*public* T getData() {

        return data;

    }

*public* void setData(T data) {

        this.data = data;

    }

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

*// Create a P38 with Integer type*

        P38<Integer> intExample = new P38<>(10);

        System.out.println("Data from Integer example: " + intExample.getData());

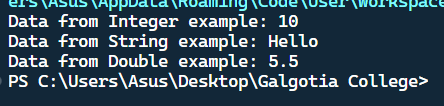
        P38<String> stringExample = new P38<>("Hello");

        System.out.println("Data from String example: " + stringExample.getData());

        P38<Double> doubleExample = new P38<>(5.5);

        System.out.println("Data from Double example: " + doubleExample.getData());

    }

} **Output: **

**Program: -39**

**Objective**: WAP in java how to implement the concept of event handling.

**Code:**

import *javax.swing.\**;

import *java.awt.\**;

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

*public* *class* P39 {

*private* JFrame frame;

*private* JButton button;

*public* P39() {

*// Create the frame*

        frame = new JFrame("Event Handling Example");

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

        frame.setSize(300, 200);

*// Create the button*

        button = new JButton("Click Me");

        button.addActionListener(new ButtonClickListener());

*// Add the button to the frame*

        frame.getContentPane().add(button, BorderLayout.CENTER);

*// Display the frame*

        frame.setVisible(true);

   }

*// ActionListener implementation for handling button clicks*

*private* *class* ButtonClickListener *implements* *ActionListener* {

*public* void actionPerformed(ActionEvent e) {

*// When the button is clicked, show a message dialog*

            JOptionPane.showMessageDialog(frame, "Button Clicked!");

        }

    }

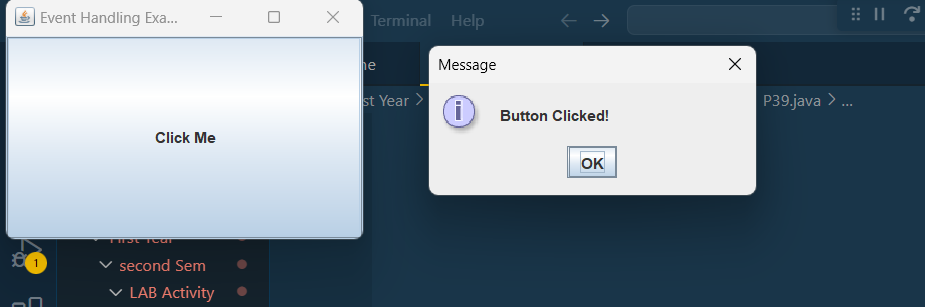
*public* *static* void main(String[] args) {

*// Create an instance of the P39 class*

        P39 eventHandler = new P39();

    }

}

**Output: **

**Program: -40.**

**Objective**: WAP how to create a simple student registration application using various swing.

**Code:**

import *javax.swing.\**;

import *java.awt.\**;

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

*public* *class* P40 {

*private* JFrame frame;

*private* JLabel nameLabel, ageLabel, genderLabel;

*private* JTextField nameField, ageField;

*private* JRadioButton maleRadioButton, femaleRadioButton;

*private* JButton registerButton;

*public* P40() {

*// Create the frame*

        frame = new JFrame("Student Registration");

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

        frame.setSize(300, 200);

        frame.setLayout(new GridLayout(4,2)); *// Create and add components*

        nameLabel = new JLabel("Name:");

        frame.add(nameLabel);

        nameField = new JTextField();

        frame.add(nameField);

        ageLabel = new JLabel("Age:");

       frame.add(ageLabel);

        ageField = new JTextField();

        frame.add(ageField);

      genderLabel = new JLabel("Gender:");

        frame.add(genderLabel);

       maleRadioButton = new JRadioButton("Male");

        femaleRadioButton = new JRadioButton("Female");

        ButtonGroup genderGroup = new ButtonGroup();

        genderGroup.add(maleRadioButton);

        genderGroup.add(femaleRadioButton);

        JPanel genderPanel = new JPanel();

        genderPanel.add(maleRadioButton);

        genderPanel.add(femaleRadioButton);

        frame.add(genderPanel);

        registerButton = new JButton("Register");

        registerButton.addActionListener(new ButtonClickListener());

        frame.add(registerButton);

*// Display the frame*

        frame.setVisible(true);}

*private* *class* ButtonClickListener *implements* *ActionListener* {

*public* void actionPerformed(ActionEvent e) {

*// Get the entered values*

            String name = nameField.getText();

            int age = Integer.parseInt(ageField.getText());

            String gender = maleRadioButton.isSelected() ? "Male" : "Female";

            String message = "Name: " + name + "\nAge: " + age + "\nGender: " + gender;

            JOptionPane.showMessageDialog(frame, message, "Registration Details",            JOptionPane.INFORMATION\_MESSAGE);

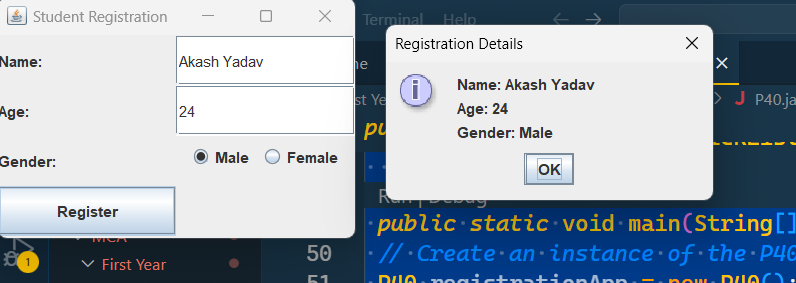
        }}

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

*// Create an instance of the P40 class*

 P40 registrationApp = new P40();

    }}

**Output: **