**A**

**Practical File**

**On**

**Object Oriented Programming with Java**

**(BCS 452)**

**Submitted**

**For**

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

**At**

****

**<Name of your College>**

**20XX-20XX**

**Submitted to:**

<Name of Professor>

<Designation>

<place>

**Submitted by:**

<Student Name>

Roll no. :<Roll Number>

Course : B.Tech

C.S.E II Year(IV sem.)

☕︎INDEX☕︎

[1. Java Program(Command Line Argument): 3](#_cg7wvu4h4qqe)

[2. Java Simple Program(Only Main class):](#_l72mguo8fmah) 5

[3. Java Program With class and object:](#_8bmdu42po2bk) 7

[4. Java Program With Inheritance and Polymorphism:](#_xk2ngqpm21nn) 9

[5. Java Program To Implement Exception Handling:](#_o81k5xnajjez) 14

[6. Java Program To File Handling:](#_8mejznmud60h) 18

[7. Java Program To Implement Multi-Threading:](#_ly6cqm94awv7) 22

[8. Java Program Showcases the use of Packages in java.](#_kmknl4mg2pjs) 25

[9. Java Program to use lambda function with interface.](#_nzyyi67wayk3) 29

[10. Java Program to use class Base64.](#_npzuxfjquhfs) 31

# Java Program(Command Line Argument):

Write a java program to take the number from the user from the command line and check whether the number is palindrome or not.

**E.g.** if the number is 12321 then its reverse i.e. 12321 equals to the actual number.

**Source Code:**

public class PalindromeCheck {

public static void main(String[] args) {

// Check if the user has provided an argument

if (args.length != 1) {

System.out.println("Please provide one number as a command line argument.");

return;

}

try {

// Parse the argument to an integer

int number = Integer.parseInt(args[0]);

int original = number;

int reversed = 0;

// Reverse the number

while (number != 0) {

int digit = number % 10;

reversed = reversed \* 10 + digit;

number /= 10;

}

// Check if original and reversed are the same

if (original == reversed) {

System.out.println(original + " is a palindrome.");

} else {

System.out.println(original + " is not a palindrome.");

}

} catch (NumberFormatException e) {

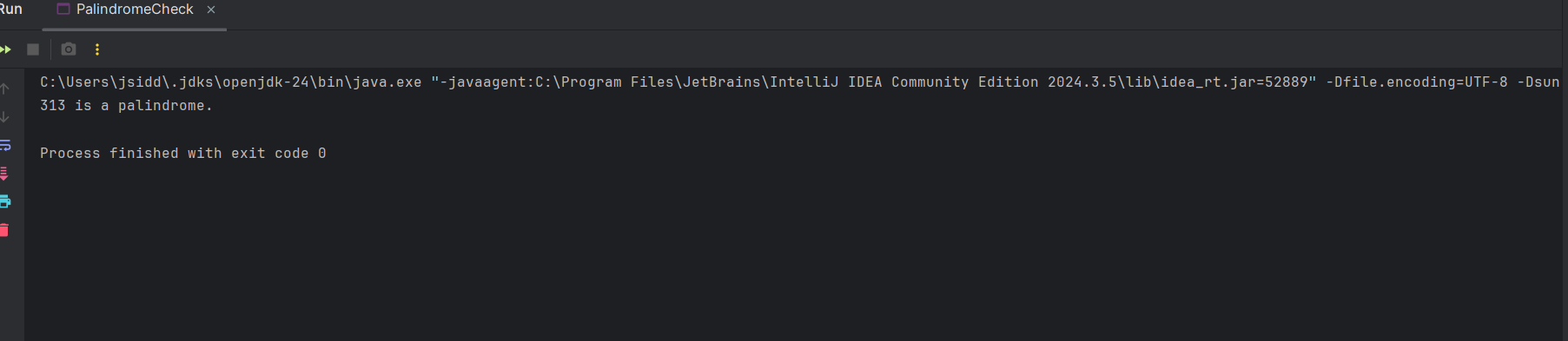
System.out.println("Invalid input. Please enter a valid integer.");

}

}

}

**Output:**



# Java Simple Program(Only Main class):

Write a Java program to check if the given number is an Armstrong or not.

**Definition** :An Armstrong number is a positive integer that's equal to the sum of its digits, each raised to the power of the number of digits.

**E.g.** 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.

**Source Code:**

import java.util.Scanner;

public class ArmstrongCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input from user

System.out.print("Enter a number: ");

int number = scanner.nextInt();

scanner.close();

int originalNumber = number;

int result = 0;

int digits = 0;

// Count number of digits

int temp = number;

while (temp != 0) {

temp /= 10;

digits++;

}

// Calculate Armstrong sum

temp = number;

while (temp != 0) {

int digit = temp % 10;

result += Math.pow(digit, digits);

temp /= 10;

}

// Check if Armstrong

if (result == originalNumber) {

System.out.println(originalNumber + " is an Armstrong number.");

} else {

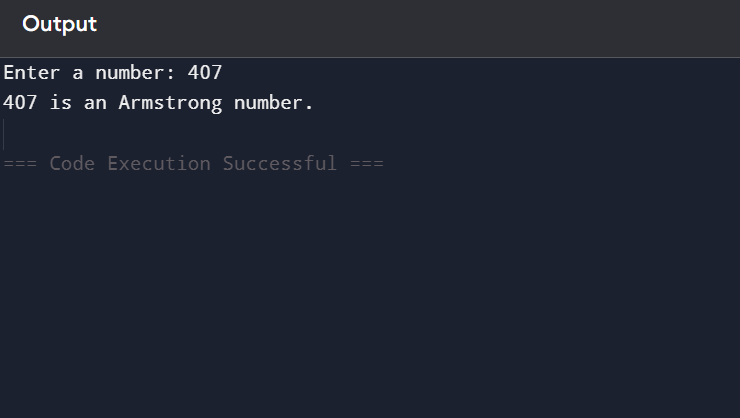
System.out.println(originalNumber + " is not an Armstrong number.");

}

}

}

**Output:**



# Java Program With class and object:

Write a Java program having a class “Person” and with attributes name, age and gender. Create the getter and setter methods for each attribute in “Person” class and display() method to display the information of the person. Use the “Person” class in the “Main” class by creating the object of the “Person” class.

**Source Code:**

// Person.java

class Person {

private String name;

private int age;

private String gender;

// Setter for name

public void setName(String name) {

this.name = name;

}

// Getter for name

public String getName() {

return name;

}

// Setter for age

public void setAge(int age) {

this.age = age;

}

// Getter for age

public int getAge() {

return age;

}

// Setter for gender

public void setGender(String gender) {

this.gender = gender;

}

// Getter for gender

public String getGender() {

return gender;

}

// Method to display information

public void display() {

System.out.println("Name : " + name);

System.out.println("Age : " + age);

System.out.println("Gender : " + gender);

}

}

// Main.java

public class Main {

public static void main(String[] args) {

// Creating object of Person class

Person person1 = new Person();

// Setting values

person1.setName("Siddhi Jain");

person1.setAge(20);

person1.setGender("Female");

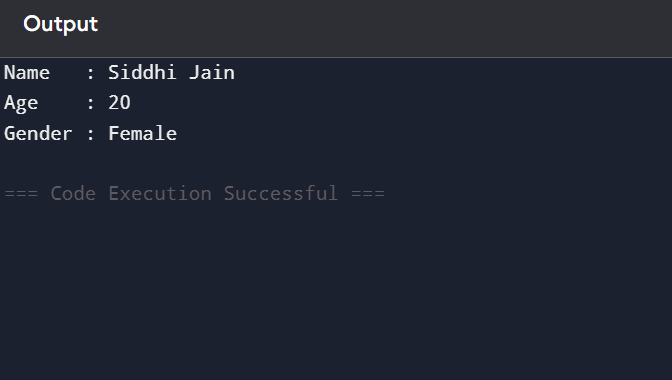
// Displaying values

person1.display();

}

}

**Output:**



# Java Program With Inheritance and Polymorphism:

Write a Java program to create a class called "Reservation"" with attributes for reservation\_id, customer\_name, and reservation\_date. Create subclasses "ResortReservation" and "RailwayReservation" that add specific attributes like room\_no, room\_type, room\_charge for hotels and seat\_no, coach\_type, ticket\_price. Implement methods as given in the chart below and can add some extra attributes and methods as per your choice.

**Note:**

1. Implement method Override the display() method from the superclass Reservation.
2. Implement method overloading for
   1. changeRoom(<int>, <int>) => change Room number
   2. changeRoom(<String>, <String>) => change Room Type
   3. changeSeat(<int>, <int>) => change Seat Number
   4. changeSeat(<String>, <String>) =>change Coach Type



**Source Code:**

// Superclass

class Reservation {

int reservationId;

String customerName;

String reservationDate;

public Reservation(int reservationId, String customerName, String reservationDate) {

this.reservationId = reservationId;

this.customerName = customerName;

this.reservationDate = reservationDate;

}

// Display method (to be overridden)

public void display() {

System.out.println("Reservation ID: " + reservationId);

System.out.println("Customer Name : " + customerName);

System.out.println("Reservation Date: " + reservationDate);

}

}

// Subclass for Resort Reservation

class ResortReservation extends Reservation {

int roomNo;

String roomType;

double roomCharge;

public ResortReservation(int reservationId, String customerName, String reservationDate,

int roomNo, String roomType, double roomCharge) {

super(reservationId, customerName, reservationDate);

this.roomNo = roomNo;

this.roomType = roomType;

this.roomCharge = roomCharge;

}

// Method Overriding

@Override

public void display() {

super.display();

System.out.println("Room No : " + roomNo);

System.out.println("Room Type : " + roomType);

System.out.println("Room Charge : ₹" + roomCharge);

}

// Method Overloading for Room updates

public void changeRoom(int oldRoom, int newRoom) {

System.out.println("Changing room number from " + oldRoom + " to " + newRoom);

this.roomNo = newRoom;

}

public void changeRoom(String oldType, String newType) {

System.out.println("Changing room type from " + oldType + " to " + newType);

this.roomType = newType;

}

}

// Subclass for Railway Reservation

class RailwayReservation extends Reservation {

int seatNo;

String coachType;

double ticketPrice;

public RailwayReservation(int reservationId, String customerName, String reservationDate,

int seatNo, String coachType, double ticketPrice) {

super(reservationId, customerName, reservationDate);

this.seatNo = seatNo;

this.coachType = coachType;

this.ticketPrice = ticketPrice;

}

// Method Overriding

@Override

public void display() {

super.display();

System.out.println("Seat No : " + seatNo);

System.out.println("Coach Type : " + coachType);

System.out.println("Ticket Price : ₹" + ticketPrice);

}

// Method Overloading for Seat updates

public void changeSeat(int oldSeat, int newSeat) {

System.out.println("Changing seat number from " + oldSeat + " to " + newSeat);

this.seatNo = newSeat;

}

public void changeSeat(String oldCoach, String newCoach) {

System.out.println("Changing coach type from " + oldCoach + " to " + newCoach);

this.coachType = newCoach;

}

}

// Main class to test

public class Main {

public static void main(String[] args) {

// Resort Reservation Test

ResortReservation resort = new ResortReservation(101, "Siddhi Jain", "2025-06-11", 202, "Deluxe", 5500);

System.out.println("=== Resort Reservation ===");

resort.display();

resort.changeRoom(202, 305);

resort.changeRoom("Deluxe", "Suite");

resort.display();

System.out.println("\n=== Railway Reservation ===");

// Railway Reservation Test

RailwayReservation railway = new RailwayReservation(202, "Aman Verma", "2025-06-12", 45, "Sleeper", 1200);

railway.display();

railway.changeSeat(45, 60);

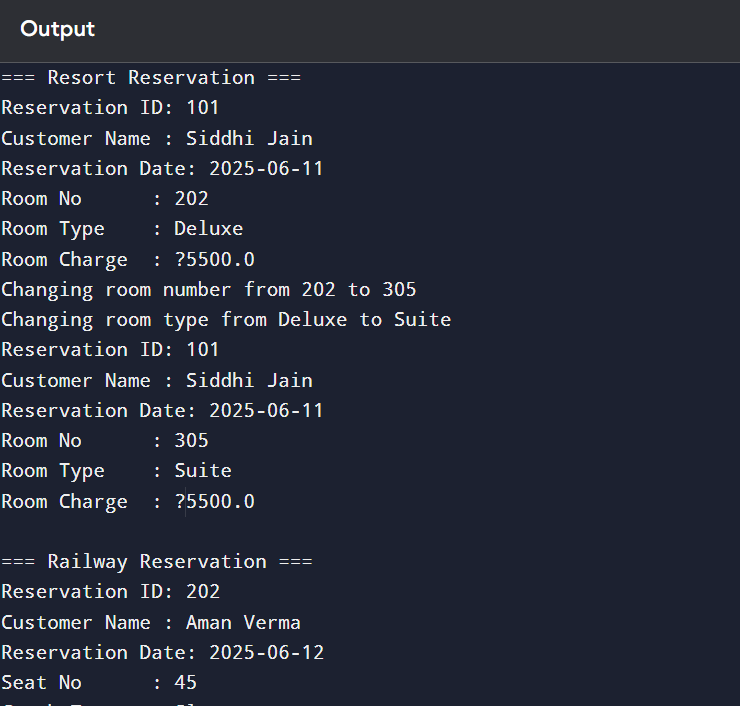
railway.changeSeat("Sleeper", "AC 3-Tier");

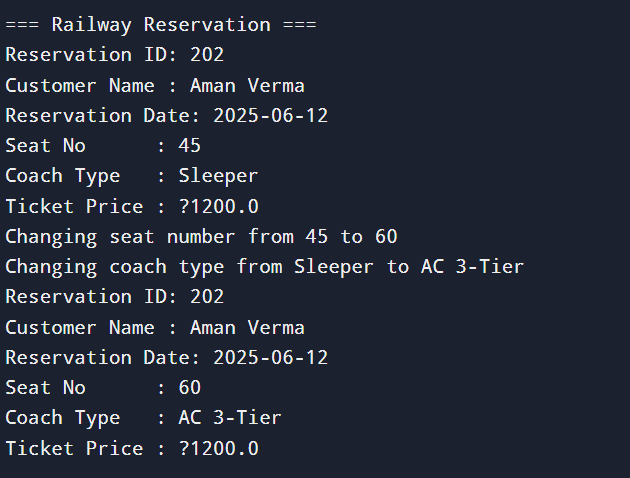
railway.display();

}

}

**Output:**





# Java Program To Implement Exception Handling:

Write a Java Program to create your own class “Registration” which is used to register the details of an User, with attributes user\_id, user\_name, password, mobile\_no, email\_id. Create the as follows:

1. Create a constructor which automatically creates the user\_id.
2. Create the methods to validate the credentials such user\_name, password, email, mobile\_no.
3. Create your own Exception and throw them when an invalid credential is encountered.
   1. InvalidEmailIdException => for an invalid email format.
   2. InvalidContactNumberException => for an invalid mobile no.
      1. Its length must be 10 and have only numbers.
   3. InvalidUserNameException => for invalid user name
      1. User name only contains uppercase, lowercase and ‘\_’ letters.
   4. InsecurePasswordException => for an easy password
      1. Password must have at least 8 letters.
      2. Password must have at least an Uppercase and special Symbol.

**Source Code:**

import java.util.regex.\*;

import java.util.Scanner;

// Custom Exception Classes

class InvalidEmailIdException extends Exception {

public InvalidEmailIdException(String message) {

super(message);

}

}

class InvalidContactNumberException extends Exception {

public InvalidContactNumberException(String message) {

super(message);

}

}

class InvalidUserNameException extends Exception {

public InvalidUserNameException(String message) {

super(message);

}

}

class InsecurePasswordException extends Exception {

public InsecurePasswordException(String message) {

super(message);

}

}

// Registration Class

class Registration {

private static int idCounter = 1000;

private final int user\_id;

private String user\_name;

private String password;

private String mobile\_no;

private String email\_id;

// Constructor

public Registration(String user\_name, String password, String mobile\_no, String email\_id)

throws InvalidUserNameException, InsecurePasswordException,

InvalidContactNumberException, InvalidEmailIdException {

this.user\_id = idCounter++; // auto increment

setUserName(user\_name);

setPassword(password);

setMobileNo(mobile\_no);

setEmailId(email\_id);

}

public void setUserName(String user\_name) throws InvalidUserNameException {

if (!user\_name.matches("[a-zA-Z\_]+")) {

throw new InvalidUserNameException("Username must contain only letters and underscores.");

}

this.user\_name = user\_name;

}

public void setPassword(String password) throws InsecurePasswordException {

if (password.length() < 8 ||

!password.matches(".\*[A-Z].\*") ||

!password.matches(".\*[^a-zA-Z0-9].\*")) {

throw new InsecurePasswordException("Password must be at least 8 characters long, contain an uppercase letter, and a special symbol.");

}

this.password = password;

}

public void setMobileNo(String mobile\_no) throws InvalidContactNumberException {

if (!mobile\_no.matches("\\d{10}")) {

throw new InvalidContactNumberException("Mobile number must be 10 digits only.");

}

this.mobile\_no = mobile\_no;

}

public void setEmailId(String email\_id) throws InvalidEmailIdException {

String emailRegex = "^[\\w.-]+@[\\w.-]+\\.\\w{2,}$";

if (!email\_id.matches(emailRegex)) {

throw new InvalidEmailIdException("Invalid email format.");

}

this.email\_id = email\_id;

}

// Display method

public void display() {

System.out.println("User ID : " + user\_id);

System.out.println("User Name : " + user\_name);

System.out.println("Mobile No : " + mobile\_no);

System.out.println("Email ID : " + email\_id);

}

}

// Main class to test

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

try {

System.out.print("Enter Username: ");

String uname = sc.nextLine();

System.out.print("Enter Password: ");

String pass = sc.nextLine();

System.out.print("Enter Mobile No: ");

String mobile = sc.nextLine();

System.out.print("Enter Email ID: ");

String email = sc.nextLine();

// Creating user object

Registration user = new Registration(uname, pass, mobile, email);

System.out.println("\n Registration Successful!");

user.display();

} catch (InvalidUserNameException | InsecurePasswordException |

InvalidContactNumberException | InvalidEmailIdException e) {

System.out.println("\n Registration Failed: " + e.getMessage());

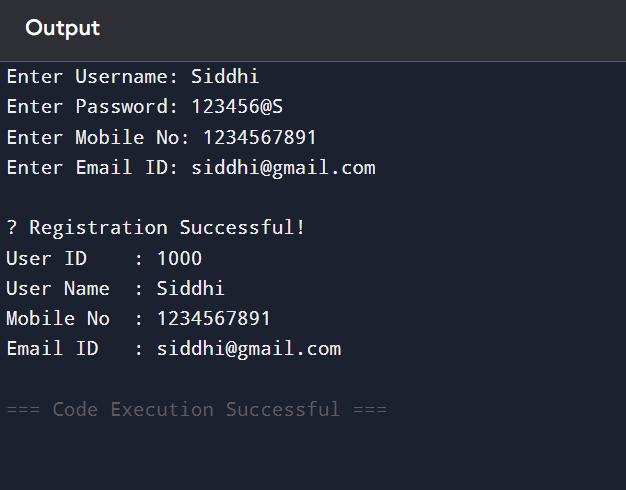
}

sc.close();

}

}

Output:



# Java Program To File Handling:

Write a menu oriented Java program to perform all 4 CRUD{ create, read, update, delete} operations on a basic text file. Make the program modular and user friendly.

**Source Code:**

import java.io.\*;

import java.util.Scanner;

public class FileCRUD {

static final String FILE\_NAME = "data.txt";

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

public static void main(String[] args) {

int choice;

do {

System.out.println("\n========= FILE CRUD MENU =========");

System.out.println("1. Create (Write to file)");

System.out.println("2. Read (Display file contents)");

System.out.println("3. Update (Modify file contents)");

System.out.println("4. Delete (Clear file contents)");

System.out.println("5. Exit");

System.out.print("Enter your choice (1-5): ");

choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

createData();

break;

case 2:

readData();

break;

case 3:

updateData();

break;

case 4:

deleteData();

break;

case 5:

System.out.println( “Exiting the program...");

break;

default:

System.out.println(" Invalid choice. Please try again.");

}

} while (choice != 5);

}

// Create or Append data

public static void createData() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME, true))) {

System.out.print("Enter data to write: ");

String data = scanner.nextLine();

writer.write(data);

writer.newLine();

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

} catch (IOException e) {

System.out.println(" Error writing to file: " + e.getMessage());

}

}

// Read file contents

public static void readData() {

File file = new File(FILE\_NAME);

if (!file.exists()) {

System.out.println(" File does not exist. Nothing to read.");

return;

}

System.out.println("\n File Contents:");

try (BufferedReader reader = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

int lineNo = 1;

while ((line = reader.readLine()) != null) {

System.out.println(lineNo++ + ": " + line);

}

} catch (IOException e) {

System.out.println(" Error reading file: " + e.getMessage());

}

}

// Update file contents (replace full content)

public static void updateData() {

System.out.println(" This will overwrite all content in the file.");

System.out.print("Are you sure? (yes/no): ");

String confirm = scanner.nextLine();

if (!confirm.equalsIgnoreCase("yes")) {

System.out.println("Update cancelled.");

return;

}

try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME))) {

System.out.print("Enter new content to update: ");

String data = scanner.nextLine();

writer.write(data);

writer.newLine();

System.out.println(" File content updated successfully.");

} catch (IOException e) {

System.out.println(" Error updating file: " + e.getMessage());

}

}

// Delete file contents (clear the file)

public static void deleteData() {

try (PrintWriter writer = new PrintWriter(FILE\_NAME)) {

writer.print(""); // clears the file

System.out.println(" File content deleted successfully.");

} catch (FileNotFoundException e) {

System.out.println(" File not found: " + e.getMessage());

}

}

}

**Output:**



# Java Program To Implement Multi-Threading:

Create a Mathematics test in java to test the speed and accuracy of the player in solving the arithmetic expressions with a time limit of 2 mins.

* 1. The score is displayed at the end of the test.
  2. The arithmetic expressions must be generated randomly.

****

**Source Code:**

import java.util.Random;

import java.util.Scanner;

public class MathSpeedTest {

static final int TIME\_LIMIT = 2 \* 60 \* 1000;

// 2 minutes in milliseconds

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Random rand = new Random();

int score = 0;

long startTime = System.currentTimeMillis();

System.out.println("Welcome to the Mathematics Speed & Accuracy Test!");

System.out.println("Solve as many questions as you can in 2 minutes.");

System.out.println("Press Enter to begin...");

scanner.nextLine();

while (System.currentTimeMillis() - startTime < TIME\_LIMIT) {

int num1 = rand.nextInt(100) + 1; // 1 to 100

int num2 = rand.nextInt(100) + 1;

char operator = getRandomOperator(rand);

int correctAnswer = calculate(num1, num2, operator);

System.out.printf("What is %d %c %d? ", num1, operator, num2);

int userAnswer;

try {

userAnswer = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Invalid input! Skipping...");

continue;

}

if (userAnswer == correctAnswer) {

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

score++;

} else {

System.out.println(" Wrong! Correct answer: " + correctAnswer);

}

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

}

System.out.println("\n Time's up!");

System.out.println("Your final score is: " + score);

}

public static char getRandomOperator(Random rand) {

char[] operators = {'+', '-', '\*', '/'};

return operators[rand.nextInt(operators.length)];

}

public static int calculate(int a, int b, char op) {

switch (op) {

case '+': return a + b;

case '-': return a - b;

case '\*': return a \* b;

case '/': return b == 0 ? 0 : a / b; // Prevent division by zero

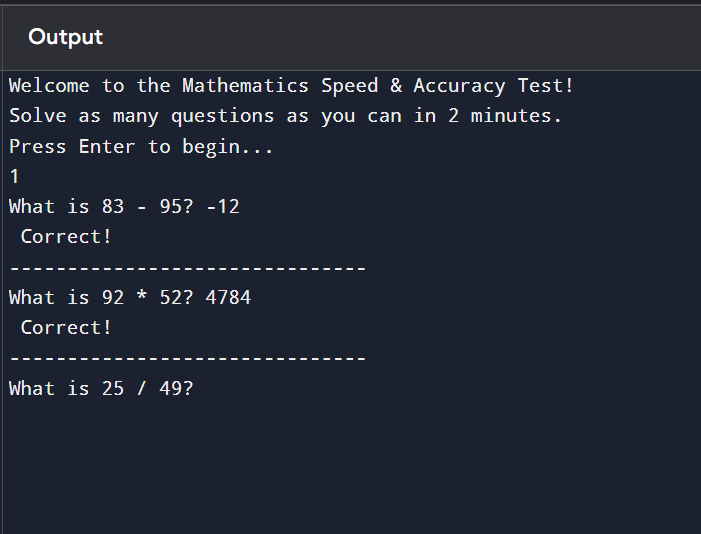
default: return 0;

}

}

}

**Output:**



# Java Program Showcases the use of Packages in java.

Create a package name “authenticate” in java with the “ Auth” class .

* 1. The class “Auth” having the attribute user\_id, user\_name, password and is\_login
  2. Having methods:
     1. login(user\_name, password) => to login the user.
     2. logout(user\_name) => to logout the user.
     3. isUser(user\_name) => to check user\_name is present
     4. isLogin(user\_name) => to check the user is login or not
     5. changePassword(password, new\_password) => to change the user password
  3. use this in a Source file with class “Shopping”.
  4. Package tree Structure.

**.**

├── Shopping.java

└── authenticate

└── Auth.java

**Source Code:**

import java.util.HashMap;

public class Shopping {

public static void main(String[] args) {

Auth auth = new Auth();

auth.login("john", "pass123");

System.out.println("Is John logged in? " + auth.isLogin("john"));

auth.login("bob", "wrongpass");

auth.changePassword("alice", "newAlicePass");

auth.login("alice", "newAlicePass");

auth.logout("john");

System.out.println("Is 'mike' a user? " + auth.isUser("mike"));

}

}

public class Auth {

private class User {

String userId;

String userName;

String password;

boolean isLogin;

User(String userId, String userName, String password) {

this.userId = userId;

this.userName = userName;

this.password = password;

this.isLogin = false;

}

}

private HashMap<String, User> users = new HashMap<>();

public Auth() {

users.put("john", new User("101", "john", "pass123"));

users.put("alice", new User("102", "alice", "alice123"));

users.put("bob", new User("103", "bob", "bob123"));

}

public boolean login(String userName, String password) {

if (users.containsKey(userName)) {

User user = users.get(userName);

if (user.password.equals(password)) {

user.isLogin = true;

System.out.println( userName + " logged in successfully.");

return true;

} else {

System.out.println(" Incorrect password.");

return false;

}

} else {

System.out.println(" User not found.");

return false;

}

}

public void logout(String userName) {

if (users.containsKey(userName)) {

users.get(userName).isLogin = false;

System.out.println( userName + " logged out successfully.");

} else {

System.out.println(" User not found.");

}

}

public boolean isUser(String userName) {

return users.containsKey(userName);

}

public boolean isLogin(String userName) {

return users.containsKey(userName) && users.get(userName).isLogin;

}

public boolean changePassword(String userName, String newPassword) {

if (users.containsKey(userName)) {

users.get(userName).password = newPassword;

System.out.println(" Password changed successfully for " + userName);

return true;

} else {

System.out.println("User not found.");

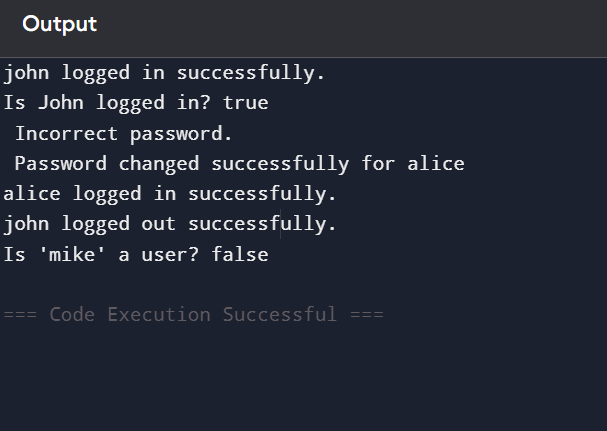
return false;

}

}

}

**Output:**

****

# Java Program to use lambda function with interface.

Write a java program to create an interface with the name “Shape” having an abstract method name area(). Use the interface “Shape” in the main class using lambda Expression by overriding the method area(). Create the objects with lambda expression for the following:

| Object name | Formula use { double area = } |
| --- | --- |
| circle | Math.PI \* radius \* radius |
| square | side \* side |
| triangle | 1/2 \* base \* height |

Note: Do not use inheritance to override the area() method.

**Source Code:**

@FunctionalInterface

interface Shape {

void area();

}

public class Main {

public static void main(String[] args) {

// Circle (radius = 5)

Shape circle = () -> {

double radius = 5;

double area = Math.PI \* radius \* radius;

System.out.println("Area of Circle: " + area);

};

circle.area();

// Square (side = 6)

Shape square = () -> {

int side = 6;

double area = side \* side;

System.out.println("Area of Square: " + area);

};

square.area();

// Triangle (base = 8, height = 4)

Shape triangle = () -> {

double base = 8;

double height = 4;

double area = 0.5 \* base \* height;

System.out.println("Area of Triangle: " + area);

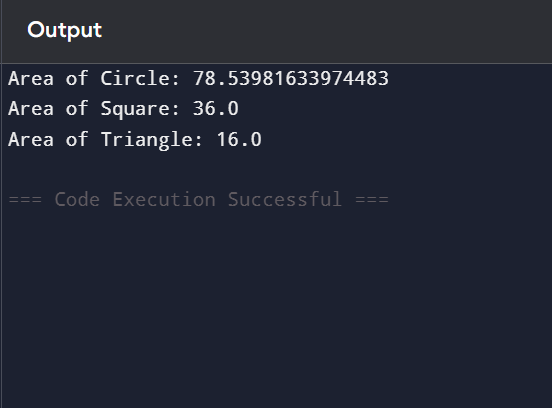
};

triangle.area();

}

}

**Output:**



# Java Program to use class Base64.

Write a menu oriented program to take choice from the user to encode or decode the messages using Base64 class in java.util package.

**Source Code:**

import java.util.Base64;

import java.util.Scanner;

public class Base64EncoderDecoder {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("\n====== Base64 Menu ======");

System.out.println("1. Encode Message");

System.out.println("2. Decode Message");

System.out.println("3. Exit");

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

choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

System.out.print("Enter message to encode: ");

String messageToEncode = scanner.nextLine();

String encoded = Base64.getEncoder().encodeToString(messageToEncode.getBytes());

System.out.println("Encoded Message: " + encoded);

break;

case 2:

System.out.print("Enter Base64 string to decode: ");

String messageToDecode = scanner.nextLine();

try {

byte[] decodedBytes = Base64.getDecoder().decode(messageToDecode);

String decoded = new String(decodedBytes);

System.out.println("Decoded Message: " + decoded);

} catch (IllegalArgumentException e) {

System.out.println(" Invalid Base64 input.");

}

break;

case 3:

System.out.println("Exiting program...");

break;

default:

System.out.println(" Invalid choice. Please try again.");

}

} while (choice != 3);

scanner.close();

}

}

**Output:**

