**ASSIGNMENT 1:** JAVA PROGRAM FOR DECIMAL TO HEXADECIMAL CONVERSION

Given a decimal number N, convert N into an equivalent hexadecimal number i.e. convert the number with base value 10 to base value 16. The decimal number system uses 10 digits 0-9 and the Hexadecimal number system uses 0-9, A-F to represent any numeric value.

Examples of Decimal to Hexadecimal Conversion

Input : 10

Output: A

Input : 2545

Output: 9F1

**SOLUTION :**

import java.util.Scanner;

public class DecimalToHexadecimal {

// Function to convert decimal to hexadecimal

public static String decimalToHexadecimal(int decimalNumber) {

// Create a StringBuilder to store the hexadecimal result

StringBuilder hexBuilder = new StringBuilder();

// Array to store hexadecimal digits

char[] hexDigits = "0123456789ABCDEF".toCharArray();

// Edge case for 0

if (decimalNumber == 0) {

return "0";

}

// Convert decimal to hexadecimal

while (decimalNumber > 0) {

int remainder = decimalNumber % 16;

hexBuilder.insert(0, hexDigits[remainder]);

decimalNumber = decimalNumber / 16;

}

return hexBuilder.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int decimalNumber = scanner.nextInt();

// Convert decimal to hexadecimal

String hexadecimalNumber = decimalToHexadecimal(decimalNumber);

// Output the result

System.out.println("Hexadecimal equivalent: " + hexadecimalNumber);

scanner.close();

}

}

**ASSIGNMENT 2:** NUMBER-INCREASING REVERSE PYRAMID

Use the java loops to print number-increasing reverse pyramid pattern.

**SOLUTION:**

**public class NumberIncreasingReversePyramid {**

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

**int rows = 6; // Number of rows in the pyramid**

**for (int i = rows; i >= 1; i--) {**

**// Print leading spaces**

**for (int j = 1; j <= rows - i; j++) {**

**System.out.print(" "); // Adjust spaces as needed**

**}**

**// Print numbers in decreasing order**

**for (int j = i; j >= 1; j--) {**

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

**}**

**System.out.println(); // Move to the next line**

**}**

**}**

**}**

**ASSIGNMENT 3:** HOLLOW DIAMOND PYRAMID

Write a java program to print Hollow Diamond Pattern, use Java Loops to print this pyramid

**SOLUTION:**

**class Pattern34**

**{**

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

**{**

**int space=4;**

**int star=1;**

**for (int i= 1;i<=9;i++ )**

**{**

**for (int b=1; b<=space;b++)**

**{**

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

**}**

**for (int a=1;a<=star;a++)**

**{**

**if (a==1||a==star)**

**{**

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

**}**

**else**

**{**

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

**}**

**}**

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

**if (i<=4)**

**{**

**star=star+2;**

**space--;**

**}**

**else**

**{**

**star=star-2;**

**space++;**

**}**

**}**

**}**

**}**

**ASSIGNMENT 4:** CREATE A SIMPLE CLASS PERSON WITH PROPERTIES NAME AND AGE, AND A METHOD TO DISPLAY THE DETAILS

Requirement:

Create a class named Person. The class should have two properties: name (String) and age (int). Include a method displayDetails in the Person class that prints the name and age of the person. Instantiate an object of the Person class in the main method and call the displayDetail method

**SOLUTION:**

**public class Person {**

**// Properties**

**private String name;**

**private int age;**

**// Constructor**

**public Person(String name, int age) {**

**this.name = name;**

**this.age = age;**

**}**

**// Method to display details**

**public void displayDetails() {**

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

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

**}**

**// Main method to test the Person class**

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

**// Instantiate an object of the Person class**

**Person person1 = new Person("Chinnari Ponnapudi", 30);**

**// Call the displayDetails method to print the details**

**person1.displayDetails();**

**}**

**}**

ASSIGNMENT 5: WRITE A PROGRAM THAT HANDLES EXCEPTIONS BY USING TRY, CATCH, AND FINALLY BLOCKS

Requirement:

Write a Java program that prompts the user to input a number. Attempt to divide a fixed number (e.g., 10) by the user-provided number. Use try, catch, and finally blocks to handle any potential exceptions (e.g., division by zero). Print appropriate messages in the catch block for exceptions. Ensure the finally block is executed to perform any cleanup operations (e.g., closing resources)

**SOLUTION:**

import java.util.Scanner;

public class ExceptionHandlingExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

// Inside the try block, we prompt the user to enter a number (‘divisor’) to divide 10.

System.out.print("Enter a number to divide 10: ");

int divisor = scanner.nextInt();

// Attempt to divide 10 by the user-provided number

int result = 10 / divisor;

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

} catch (ArithmeticException e) {

// Catch block for handling division by zero

System.out.println("Error: Division by zero is not allowed.");

} catch (Exception e) {

// Catch block for handling any other exceptions

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

} finally {

// Finally block to ensure cleanup operations

scanner.close(); // Close the scanner to release resources

System.out.println("Scanner closed in finally block.");

}

System.out.println("End of program.");

}

}