Program to Print 268 on Separate Lines

Name of the Program

Print Digits on Separate Lines using While Loop and Modulus Operator

Explanation

This program prints each digit of the number `268` on separate lines using a while loop, the modulus operator (`%`), and division (`/`). The modulus operator is used to extract the last digit of the number, and division is used to remove the last digit. The digits are printed directly as they are extracted.

Steps Involved:

1. **Initialize the Number**: The program starts with the given number.
2. **Extract and Print Digits**: Using a while loop, the program extracts each digit using the modulus operator and prints it.
3. **Remove the Last Digit**: The program removes the last digit by dividing the number by 10.

Code

```
public class PrintDigits {
    public static void main(String[] args) {
        // The number to be printed
        int number = 268;

        // Print each digit on a new line
        while (number != 0) {
            int digit = number % 10;  // Extract the last digit
            System.out.println(digit);  // Print the digit
            number /= 10;  // Remove the last digit
        }
    }
}
```

Name of the Program

Palindrome Check using While Loop and Modulus Operator

Explanation

This program checks if a number is a palindrome. A number is a palindrome if it reads the same backward as forward. The program uses a while loop to reverse the digits of the number and then compares the reversed number with the original number.

Steps Involved:

1. **Initialize the Number**: Start with the given number.
2. **Reverse the Number**: Use a while loop to reverse the digits of the number.
3. **Compare the Original and Reversed Numbers**: If they are the same, the number is a palindrome.

Code

```
import java.util.Scanner;

public class PalindromeCheck {
    public static void main(String[] args) {

        // Create a Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);


        // Ask the user to enter the number to check
        System.out.print("Enter a number to check if it is a palindrome: ");
        int number = scanner.nextInt();


        // Store the original number
        int originalNumber = number;


        // Variable to store the reversed number
        int reversedNumber = 0;


        // Reverse the number
```

```
        while (number != 0) {
            int digit = number % 10;  // Extract the last digit
            reversedNumber = reversedNumber * 10 + digit;  // Build the reversed number
            number /= 10;  // Remove the last digit
        }


        // Compare the original and reversed numbers
        if (originalNumber == reversedNumber) {
            System.out.println(originalNumber + " is a palindrome.");
        } else {
            System.out.println(originalNumber + " is not a palindrome.");
        }

        scanner.close();
    }
}
```

Name of the Program

To check if a number is a Spy number

Explanation

A Spy number is a number where the sum of its digits equals the product of its digits. We will use a while loop, the modulus operator (`%`), and division (`/`) to achieve this.

Steps Involved:

1. **Initialize the Number**: Start with the given number.
2. **Initialize Sum and Product**: Set initial values for the sum and product of the digits.
3. **Extract Digits and Update Sum and Product**: Use a while loop to extract each digit and update the sum and product.
4. **Compare Sum and Product**: If they are equal, the number is a Spy number.

Code

```
import java.util.Scanner;
```

```
public class SpyNumberCheck {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        // Ask the user to enter the number to check
        System.out.print("Enter a number to check if it is a Spy number: ");
        int number = scanner.nextInt();

        // Store the original number
        int originalNumber = number;

        // Initialize sum and product of the digits
        int sum = 0;
        int product = 1;

        // Extract each digit and update sum and product
        while (number != 0) {
            int digit = number % 10;  // Extract the last digit
            sum += digit;             // Add the digit to the sum
            product *= digit;         // Multiply the digit to the product
            number /= 10;             // Remove the last digit
        }

        // Compare the sum and product
        if (sum == product) {
            System.out.println(originalNumber + " is a Spy number.");
        } else {
            System.out.println(originalNumber + " is not a Spy number.");
        }

       // in the case of special number(sum of sum of digits and product of digits equals the
number itself eg:59 => 5 + 9 + 5*9)

/*
            if (sum + product == originalNumber) {
            System.out.println(originalNumber + " is a Special number.");
        } else {
            System.out.println(originalNumber + " is not a Special number.");
        }
*/
```

```
 // in the case of niven/harshad number(a number which is divisible by the sum of digits of the number eg:156 => 1 + 5 + 6 = 12 and 156 is completely divisible by 12)

/*
        if (originalNumber % sum == 0) {
            System.out.println(originalNumber + " is a Harshad number.");
        } else {
            System.out.println(originalNumber + " is not a Harshad number.");
        }
*/

        scanner.close();
    }
}
```

Name of the Program

To check if a number is a Duck number

Explanation

A Duck number is a positive number that has at least one zero in it, but it cannot start with zero.

Steps Involved:

1. Initialize the Number: Start with the given number.
2. Initialize Zero Count: Set the initial count for zeros to zero.
3. Extract Digits and Count Zeros: Use a while loop to extract each digit and count the zeros.
4. Check Conditions: Ensure the number does not start with zero and has at least one zero to be a Duck number.

Code

```
import java.util.Scanner;

public class DuckNumberCheck {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the user
```

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

        // Ask the user to enter the number to check
        System.out.print("Enter a number to check if it is a Duck number: ");
        int number = scanner.nextInt();

        // Initialize sum and count of zeros
        int sum = 0;
        int zeroCount = 0;

        // Store the original number
        int originalNumber = number;

          // Extract each digit and count zeros
           while (number != 0) {
               int digit = number % 10;  // Extract the last digit
               if (digit == 0) {
                   zeroCount++;  // Increment zero count
               }
               number /= 10;  // Remove the last digit
           }

           // Check the zero count
           if (zeroCount > 0) {
               System.out.println(originalNumber + " is a Duck number.");
           } else {
               System.out.println(originalNumber + " is not a Duck number.");
           }


        scanner.close();
    }
}
```

Name of the Program

Neon Number Check using While Loop and Modulus Operator

Explanation

This program checks if a number is a Neon number. A Neon number is a number where the sum of the digits of the square of the number is equal to the number itself. The program uses a while loop to extract each digit of the squared number and calculate the sum.

Steps Involved:

1. Initialize the Number: Start with the given number.
2. Calculate the Square: Compute the square of the number.
3. Initialize Sum: Set the initial sum of the digits to zero.
4. Extract Digits and Calculate Sum: Use a while loop to extract each digit and update the sum.
5. Check the Sum: If the sum of the digits equals the original number, it is a Neon number.

Code

```
import java.util.Scanner;

public class NeonNumberCheck {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        // Ask the user to enter the number to check
        System.out.print("Enter a number to check if it is a Neon number: ");
        int number = scanner.nextInt();

        // Calculate the square of the number
        int square = number * number;

        // Initialize sum of the digits of the square
        int sum = 0;

        // Extract each digit of the square and calculate the sum
        while (square != 0) {
            int digit = square % 10;  // Extract the last digit
            sum += digit;             // Add the digit to the sum
            square /= 10;             // Remove the last digit
        }

        // Check if the sum is equal to the original number
        if (sum == number) {
            System.out.println(number + " is a Neon number.");
        } else {
```

```
            System.out.println(number + " is not a Neon number.");
        }

        scanner.close();
    }
}
```