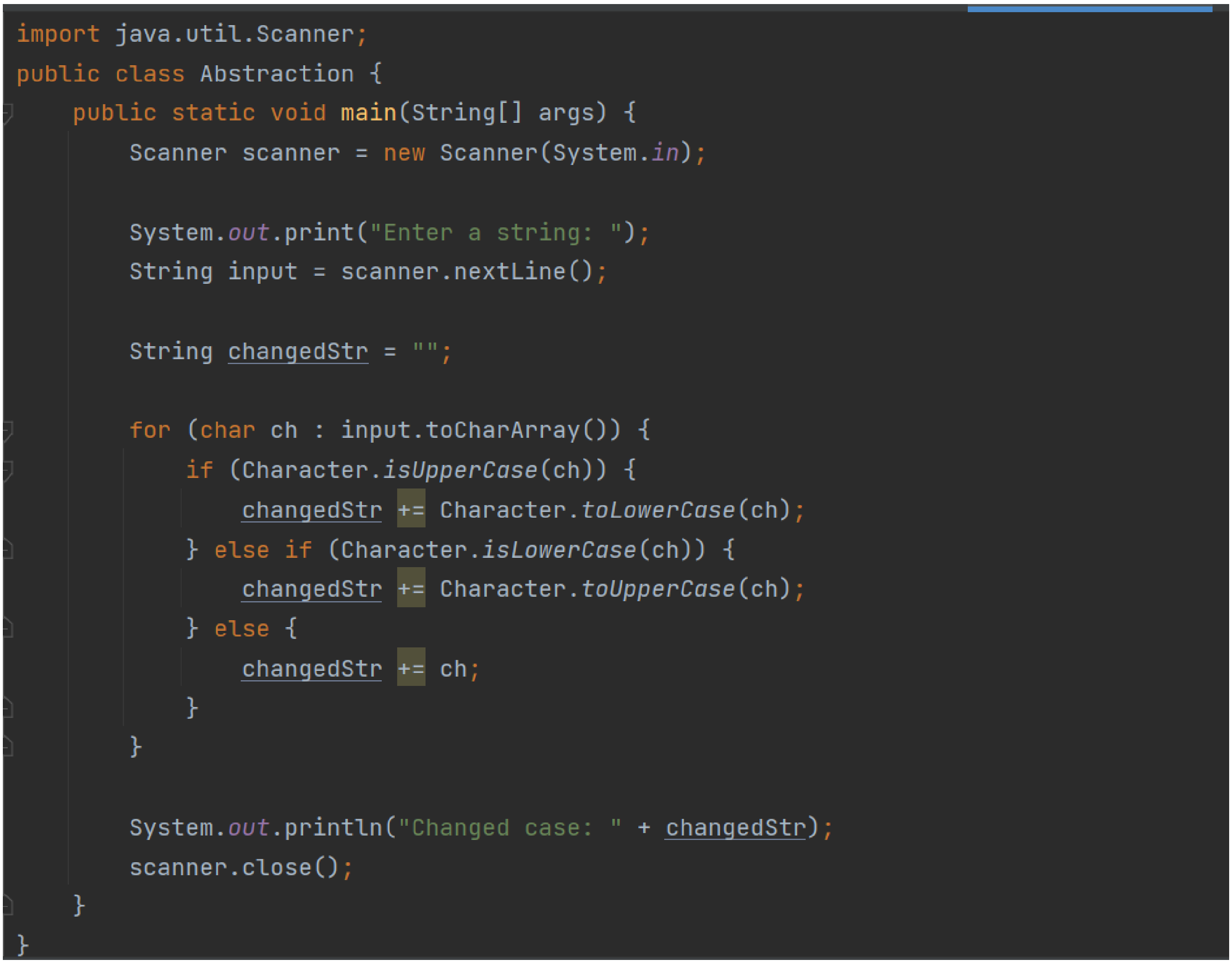
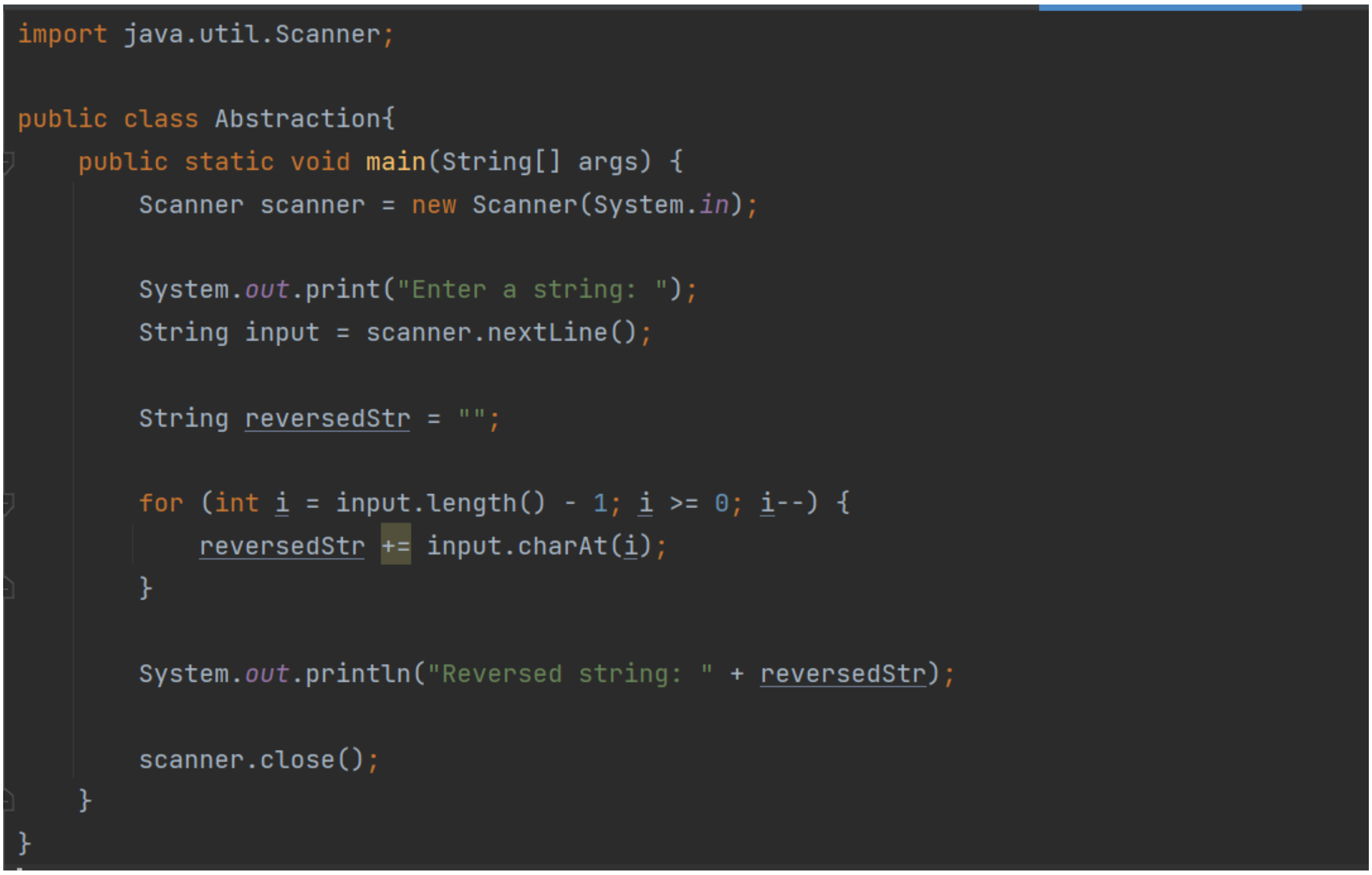
1) Write a program in java to accept a string and change the case of each letter of the string.



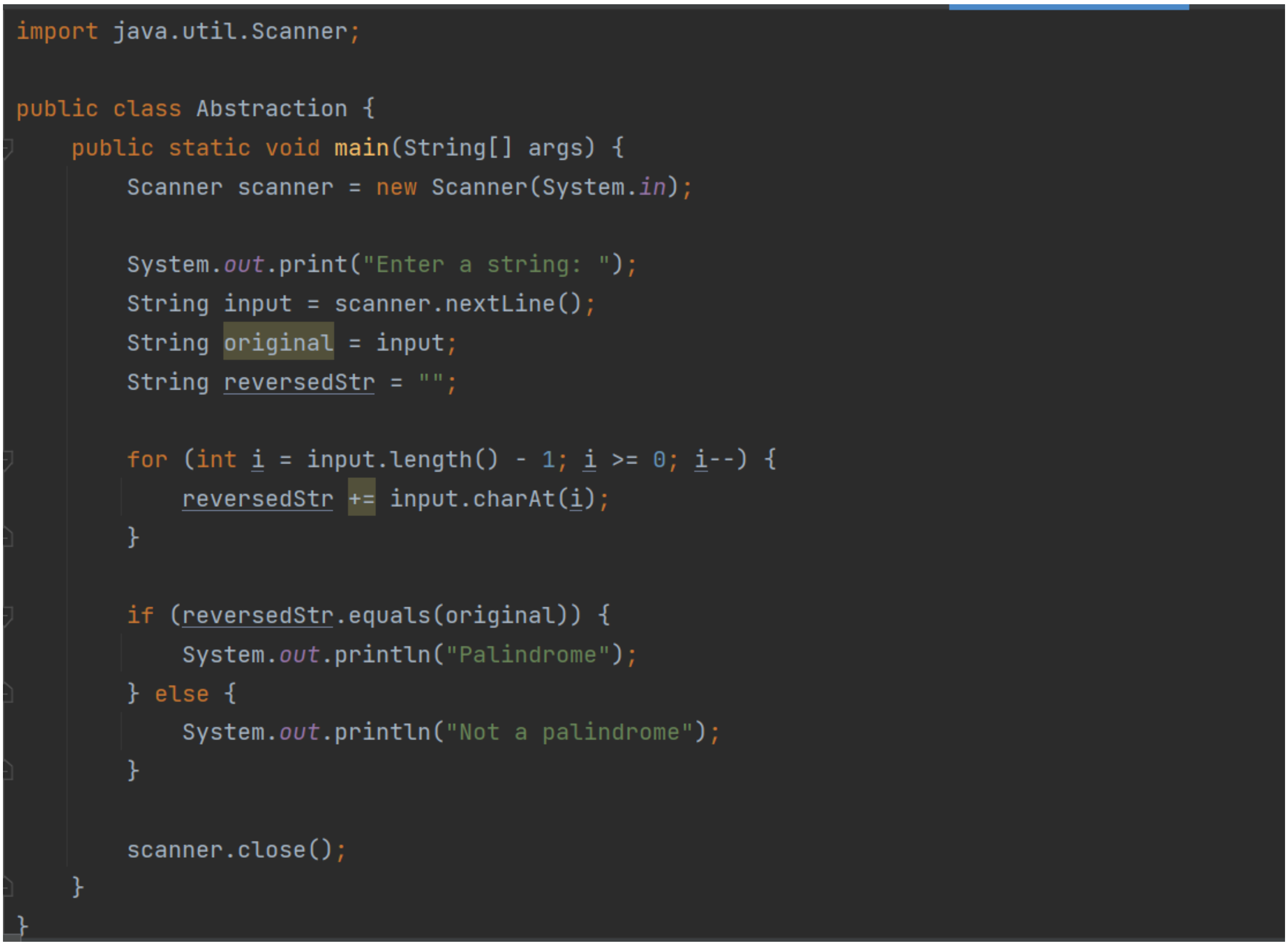
```java
import java.util.Scanner;
public class Abstraction {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        String changedStr = "";

        for (char ch : input.toCharArray()) {
            if (Character.isUpperCase(ch)) {
                changedStr += Character.toLowerCase(ch);
            } else if (Character.isLowerCase(ch)) {
                changedStr += Character.toUpperCase(ch);
            } else {
                changedStr += ch;
            }
        }

        System.out.println("Changed case: " + changedStr);
        scanner.close();
    }
}
```

2) Write a program in java to accept a string and display it after reversing.'



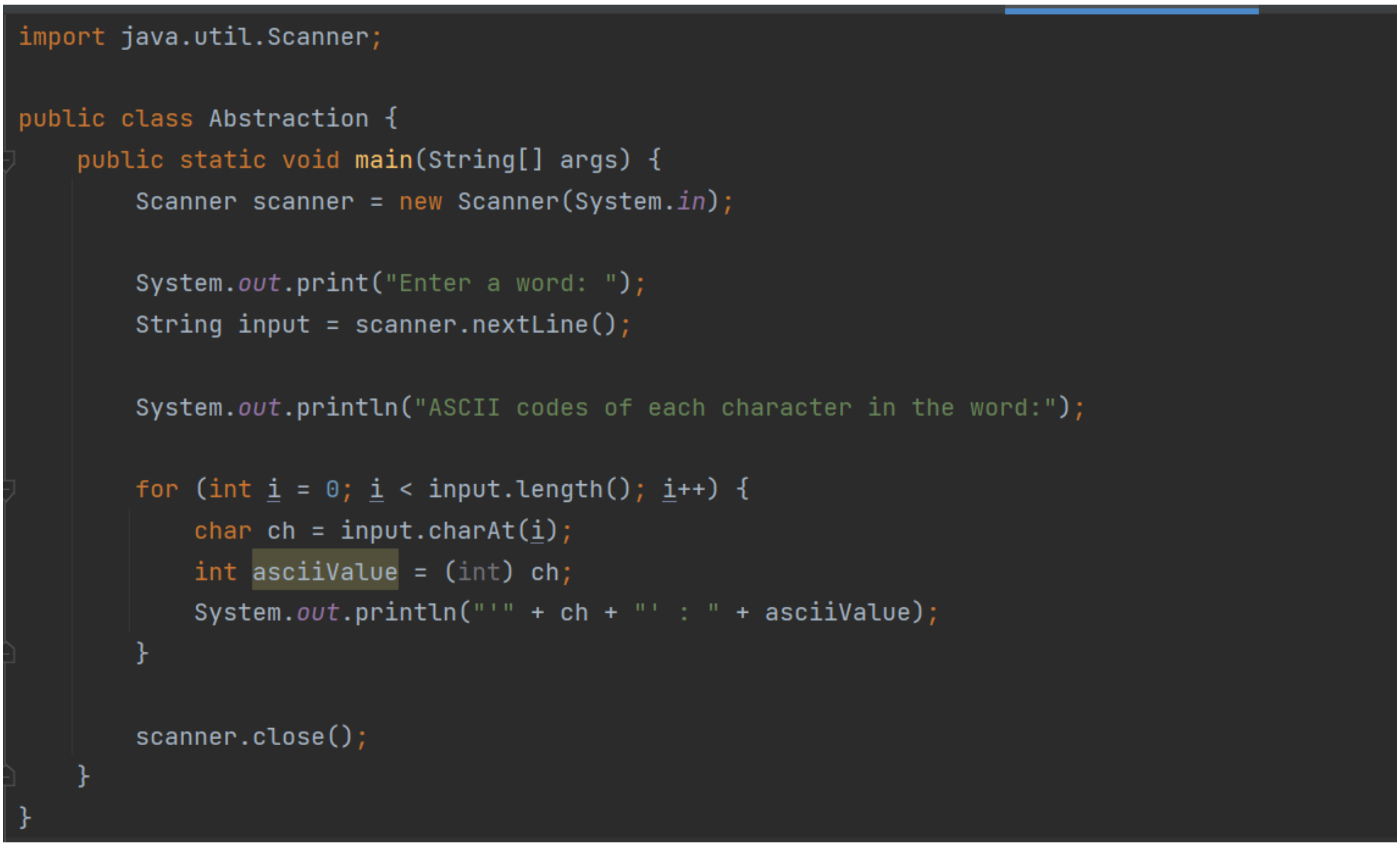
```java
import java.util.Scanner;

public class Abstraction{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        String reversedStr = "";

        for (int i = input.length() - 1; i >= 0; i--) {
            reversedStr += input.charAt(i);
        }

        System.out.println("Reversed string: " + reversedStr);

        scanner.close();
    }
}
```

3) Write a program in java to accept a word and check whether the word is palindrome or not.



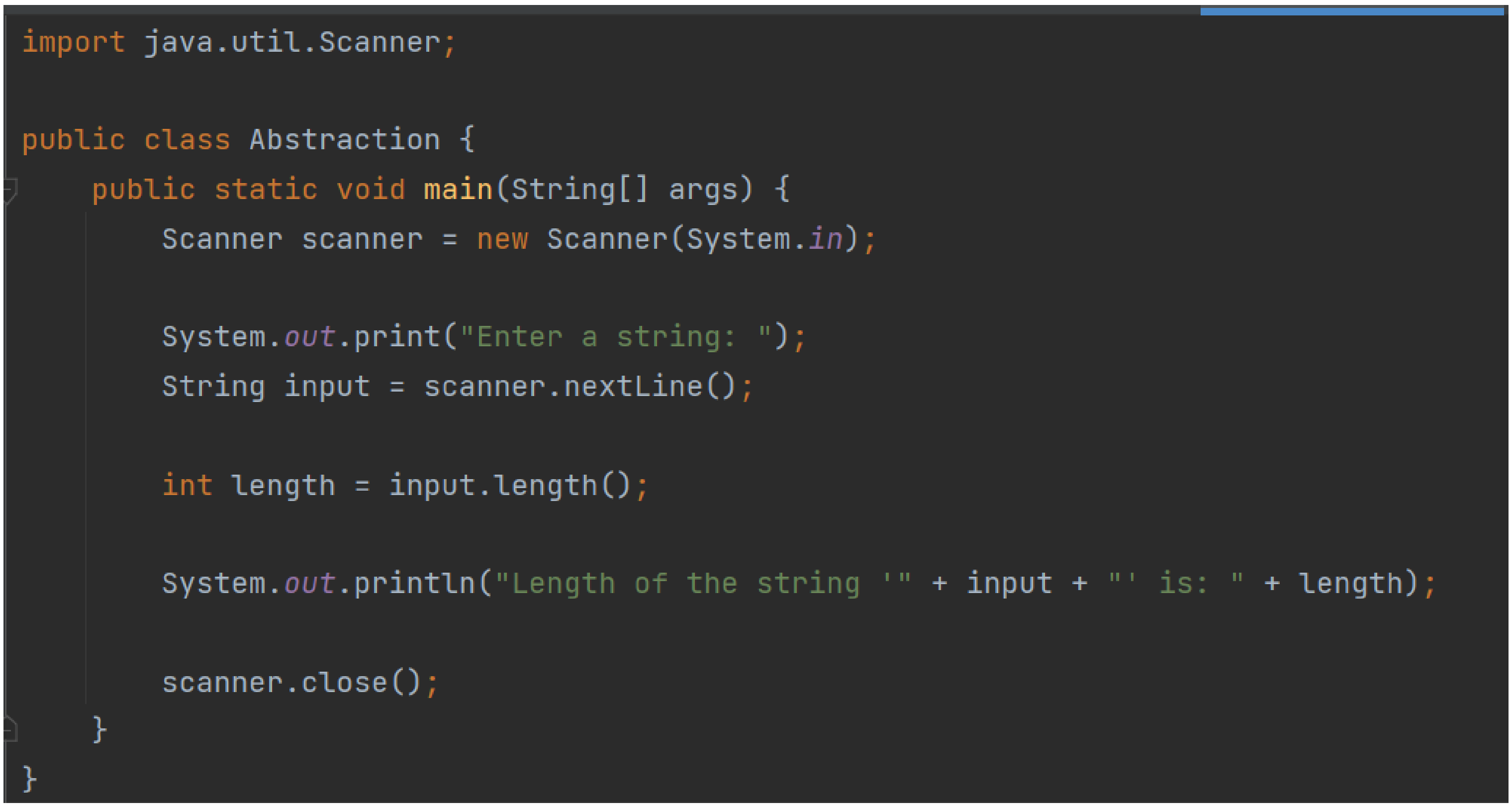
```java
import java.util.Scanner;

public class Abstraction {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        String original = input;
        String reversedStr = "";

        for (int i = input.length() - 1; i >= 0; i--) {
            reversedStr += input.charAt(i);
        }

        if (reversedStr.equals(original)) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Not a palindrome");
        }

        scanner.close();
    }
}
```

4) Write a program in java to accept a word and display the ascii code of each character of the word.



```java
import java.util.Scanner;

public class Abstraction {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a word: ");
        String input = scanner.nextLine();

        System.out.println("ASCII codes of each character in the word:");

        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            int asciiValue = (int) ch;
            System.out.println("'" + ch + "' : " + asciiValue);
        }

        scanner.close();
    }
}
```

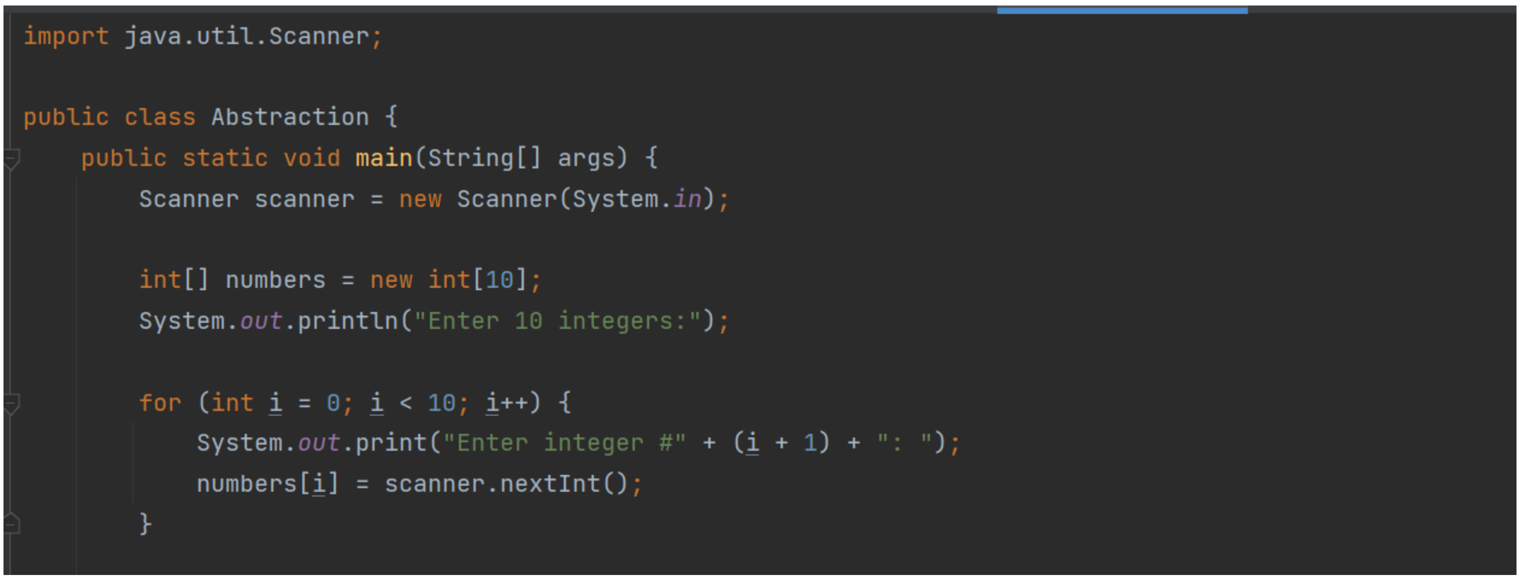
5) Write a program in java to find out the length of a given string.



```java
import java.util.Scanner;

public class Abstraction {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        int length = input.length();

        System.out.println("Length of the string '" + input + "' is: " + length);

        scanner.close();
    }
}
```

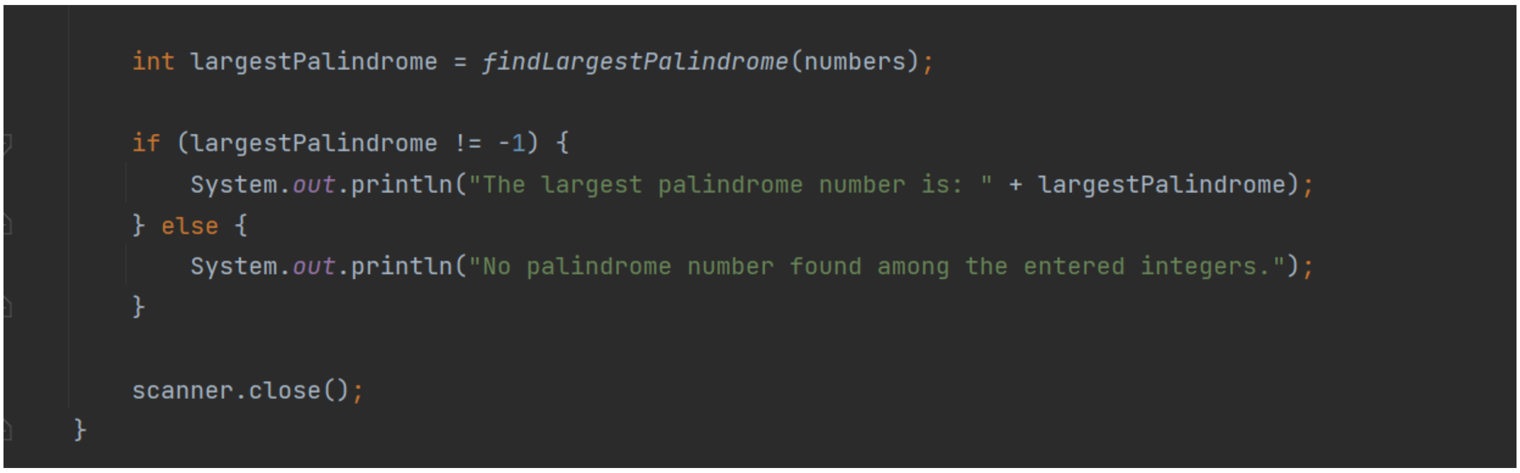
6) Create a function which accepts an integer as parameter and return true if it is a palindrome number or not. In the main() method input 10 integers and print the largest palindrome number if any.

// 10 numbers from usser



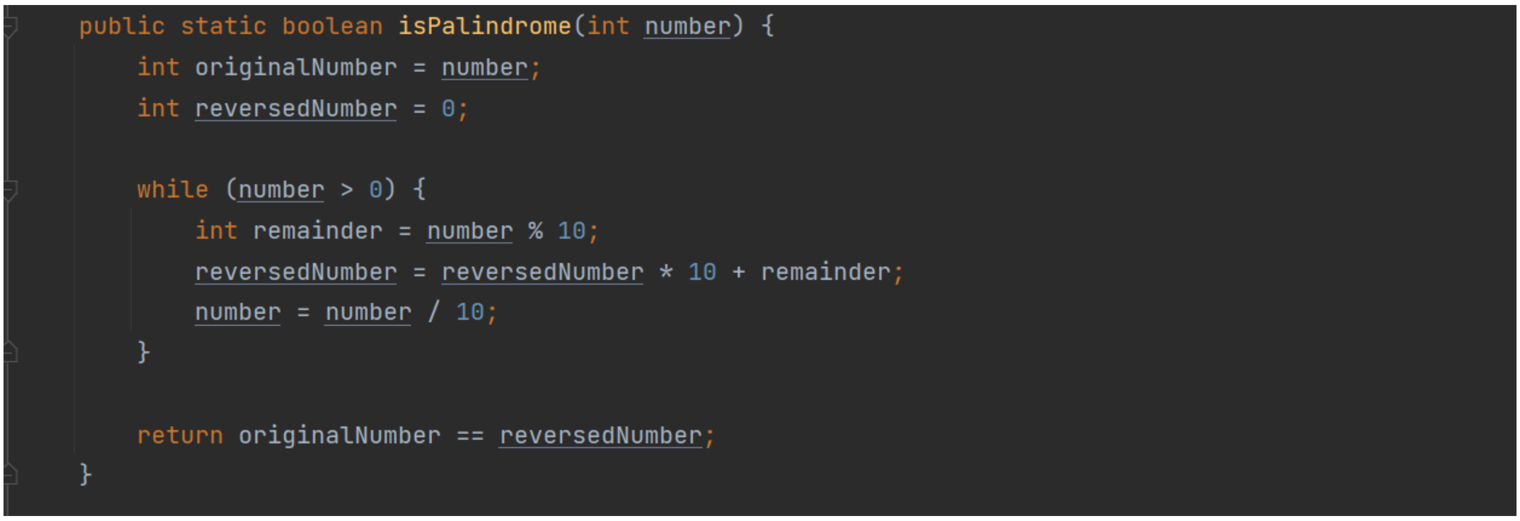
```java
import java.util.Scanner;

public class Abstraction {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int[] numbers = new int[10];
        System.out.println("Enter 10 integers:");

        for (int i = 0; i < 10; i++) {
            System.out.print("Enter integer #" + (i + 1) + ": ");
            numbers[i] = scanner.nextInt();
        }
```

//Print the result



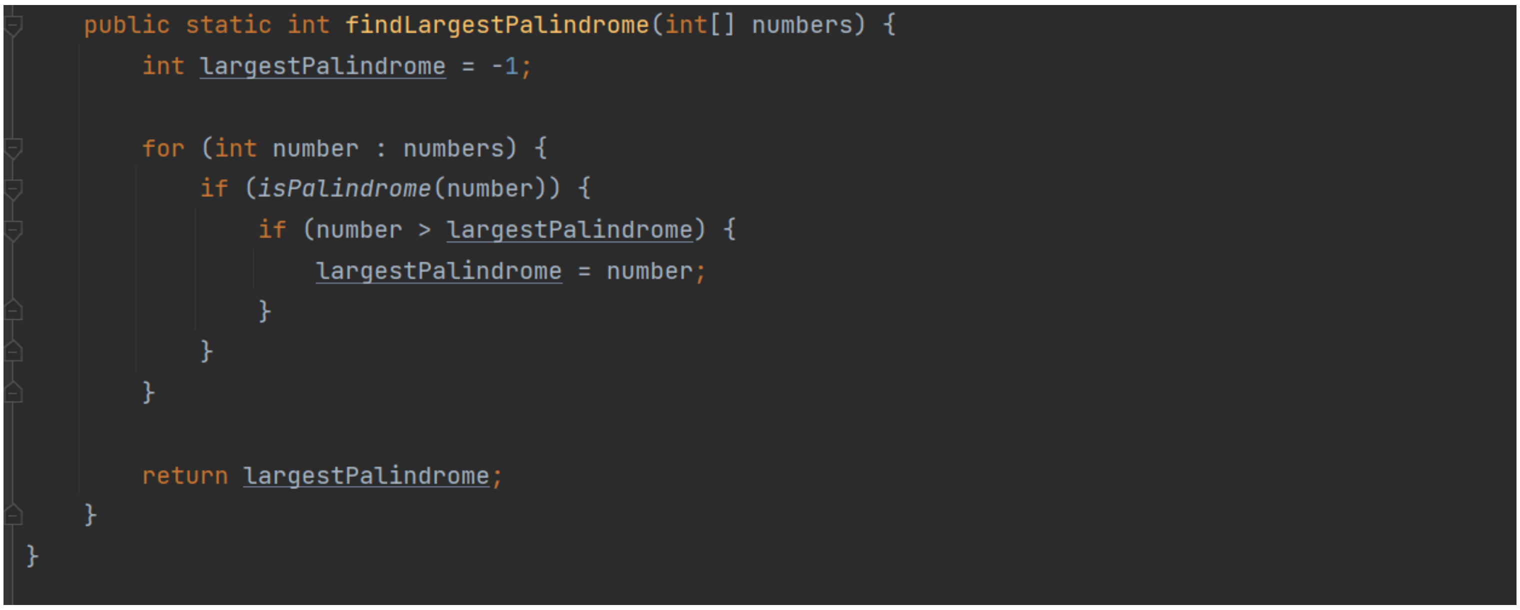
```java
        int largestPalindrome = findLargestPalindrome(numbers);

        if (largestPalindrome != -1) {
            System.out.println("The largest palindrome number is: " + largestPalindrome);
        } else {
            System.out.println("No palindrome number found among the entered integers.");
        }

        scanner.close();
    }
```

//Function to check if palindrome or not



```java
    public static boolean isPalindrome(int number) {
        int originalNumber = number;
        int reversedNumber = 0;

        while (number > 0) {
            int remainder = number % 10;
            reversedNumber = reversedNumber * 10 + remainder;
            number = number / 10;
        }

        return originalNumber == reversedNumber;
    }
```

// Function to find the largest palindrome number in an array of integers



```
    public static int findLargestPalindrome(int[] numbers) {
        int largestPalindrome = -1;

        for (int number : numbers) {
            if (isPalindrome(number)) {
                if (number > largestPalindrome) {
                    largestPalindrome = number;
                }
            }
        }

        return largestPalindrome;
    }
}
```

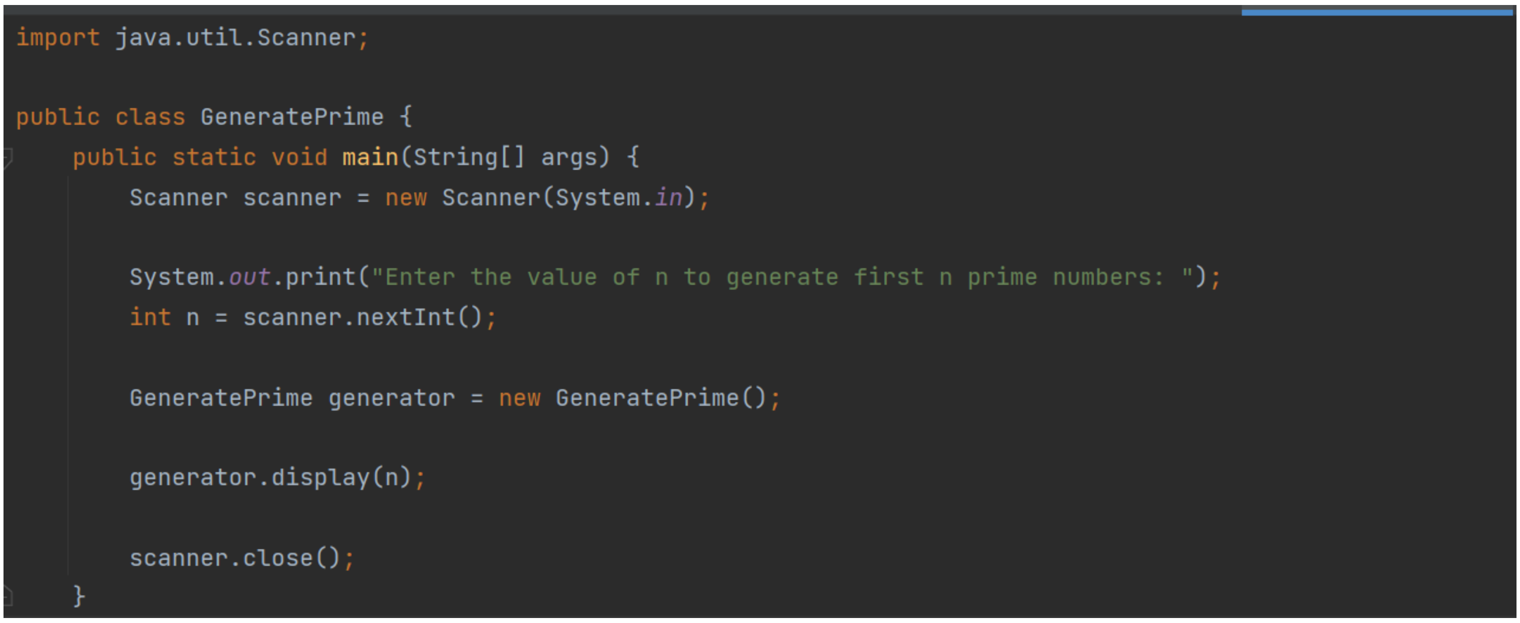
7) Create a class called generateprime which will be used to generate n number of primenumbers.

The class should have the following methods:

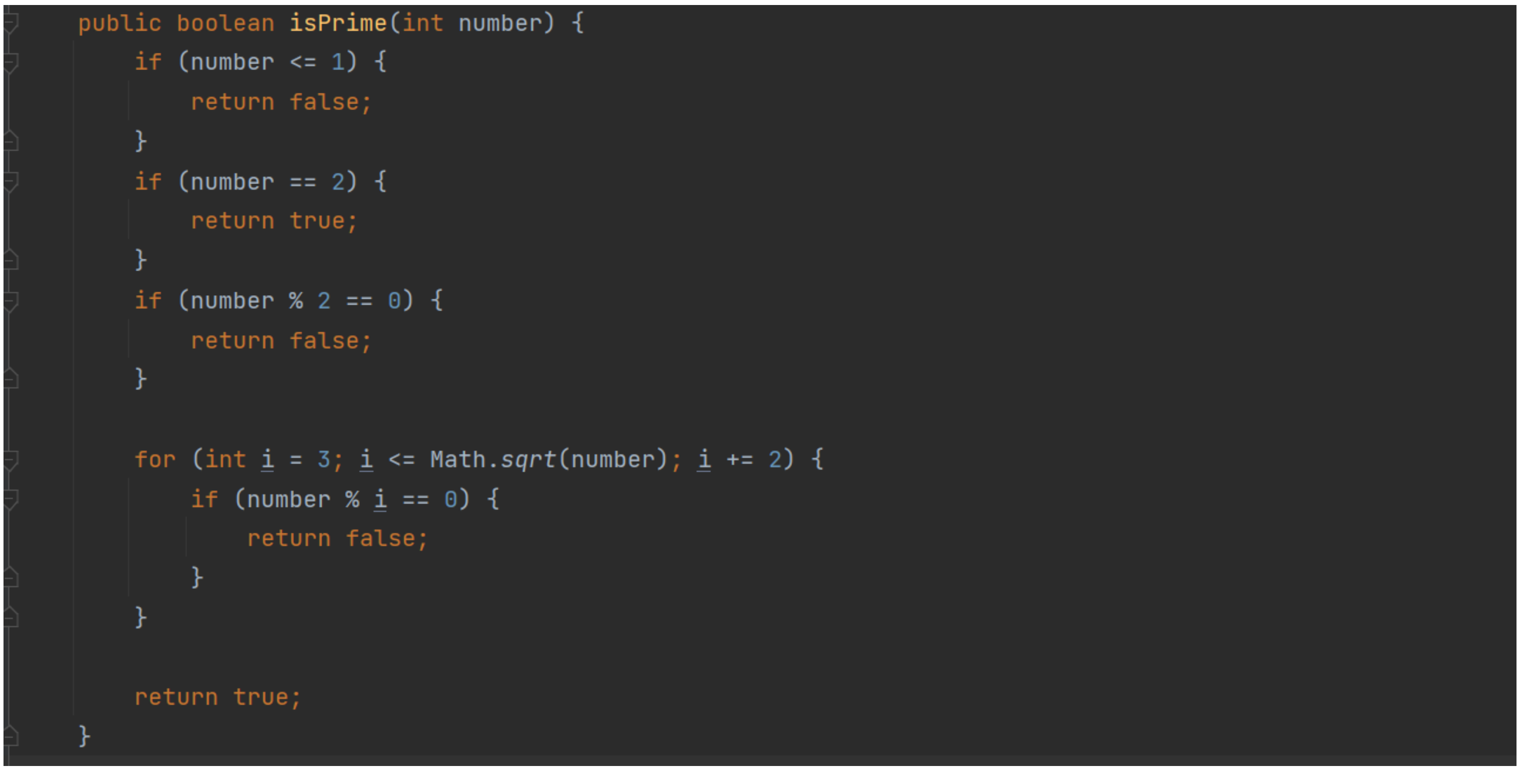
(i) Method called isprime() which accepts an integer as a parameter and return true if it is a

Prime number otherwise return false.

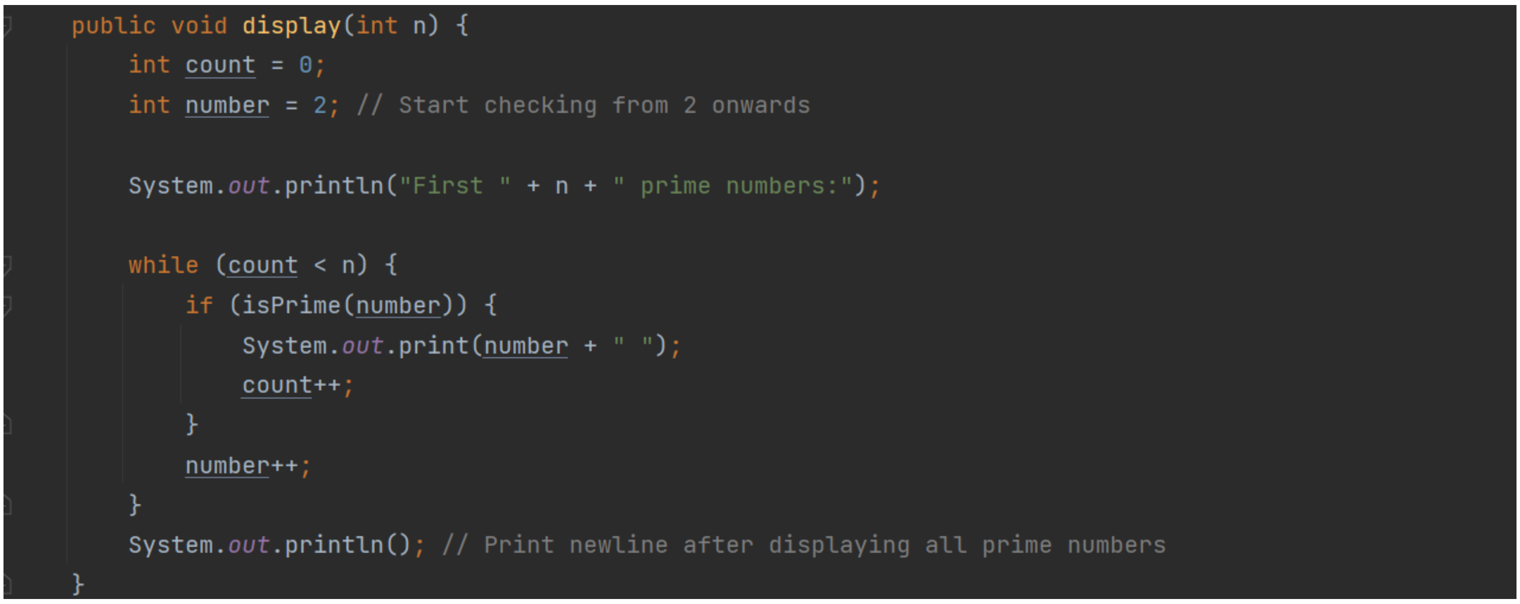
(ii) Method called display() which accepts an integer n as scanner input and display the first n prime number by calling the above function.



```java
import java.util.Scanner;

public class GeneratePrime {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the value of n to generate first n prime numbers: ");
        int n = scanner.nextInt();

        GeneratePrime generator = new GeneratePrime();

        generator.display(n);

        scanner.close();
    }
```



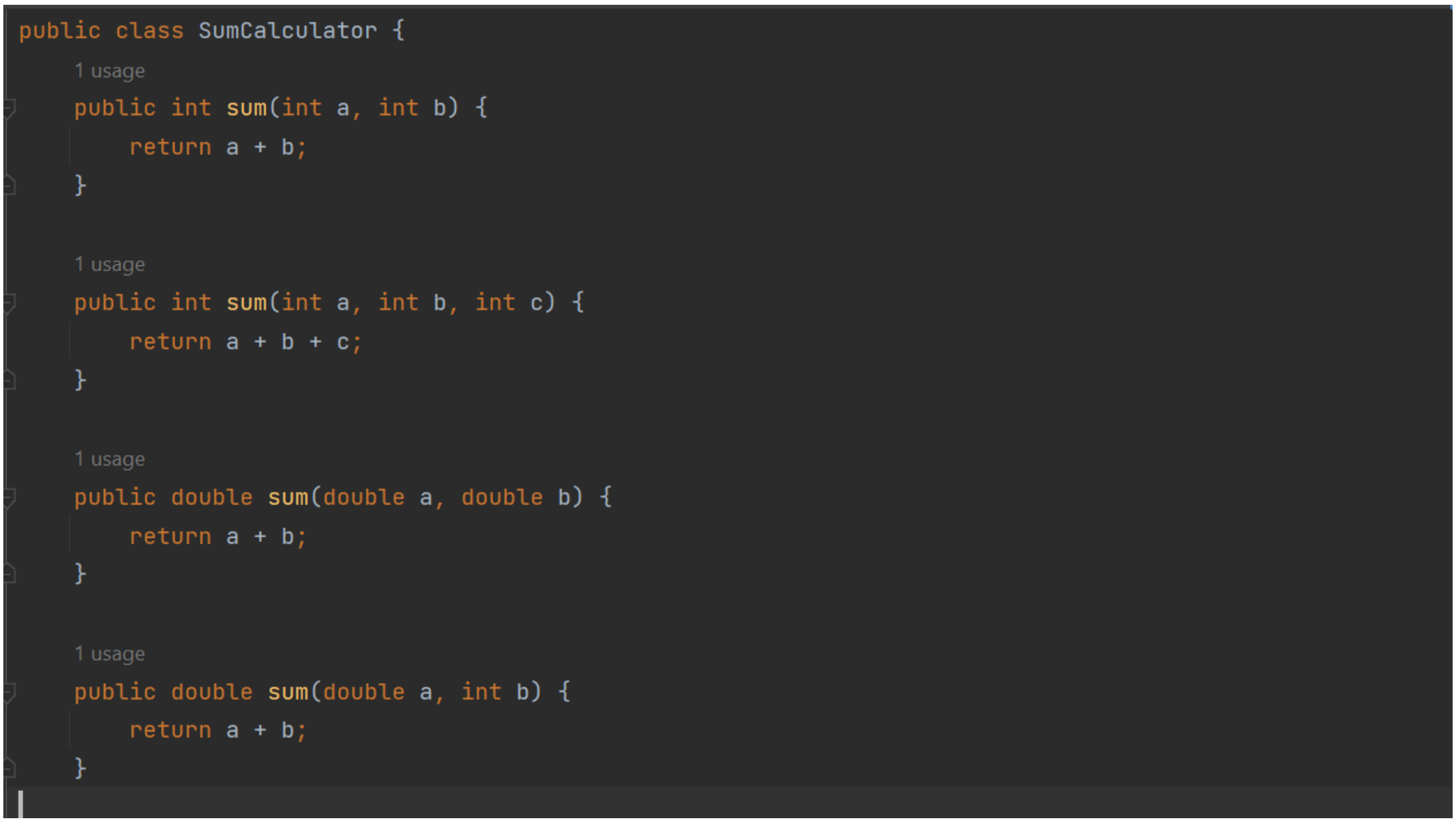
```java
    public boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        if (number == 2) {
            return true;
        }
        if (number % 2 == 0) {
            return false;
        }

        for (int i = 3; i <= Math.sqrt(number); i += 2) {
            if (number % i == 0) {
                return false;
            }
        }

        return true;
    }
```



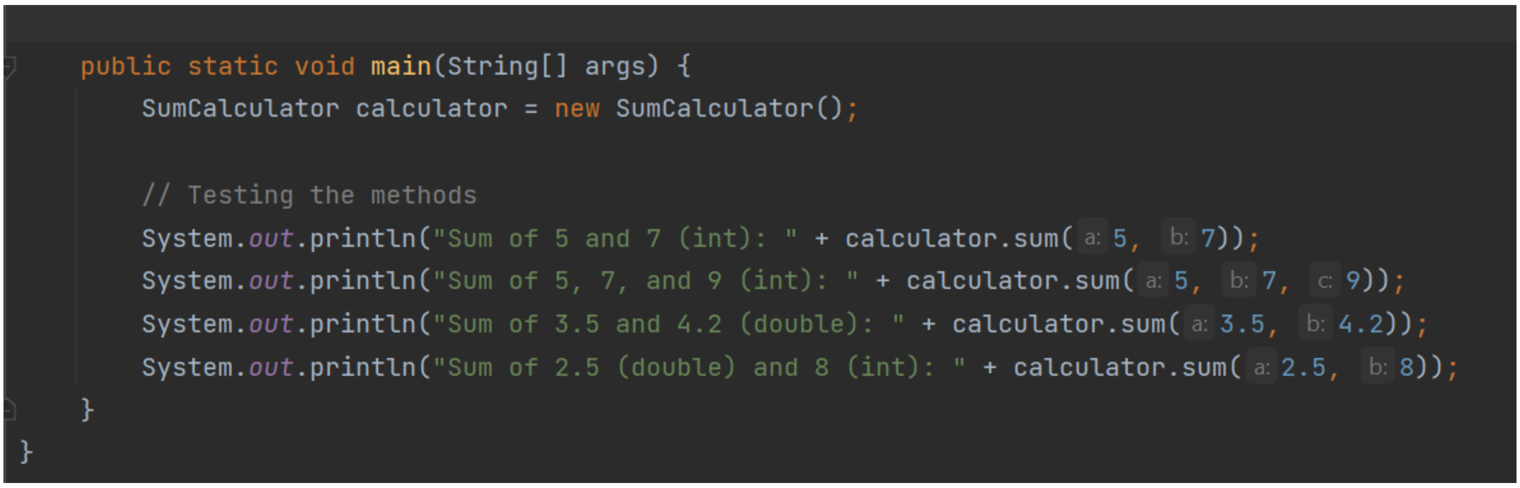
```java
public void display(int n) {
    int count = 0;
    int number = 2; // Start checking from 2 onwards

    System.out.println("First " + n + " prime numbers:");

    while (count < n) {
        if (isPrime(number)) {
            System.out.print(number + " ");
            count++;
        }
        number++;
    }
    System.out.println(); // Print newline after displaying all prime numbers
}
```

8) Using overloading technique, write methods to:

- Accept two int type data as parameters and return their sum.
- Accept three int type data as parameters and return their sum.
- Accept two double type data as parameter and return their sum.
- Accept a double type and int type as parameter and return their sum.

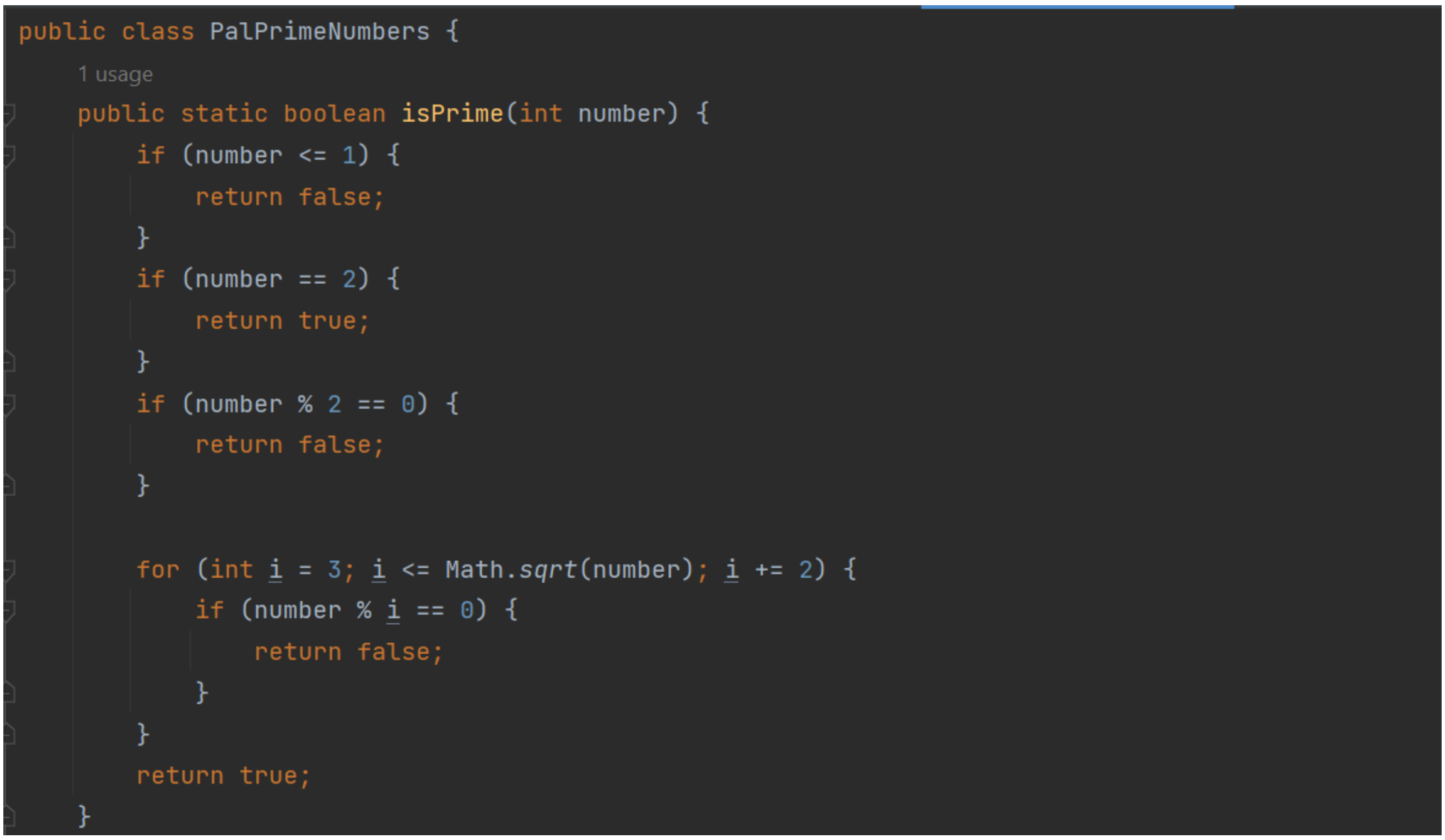


```java
public class SumCalculator {
    1 usage
    public int sum(int a, int b) {
        return a + b;
    }

    1 usage
    public int sum(int a, int b, int c) {
        return a + b + c;
    }

    1 usage
    public double sum(double a, double b) {
        return a + b;
    }

    1 usage
    public double sum(double a, int b) {
        return a + b;
    }
```

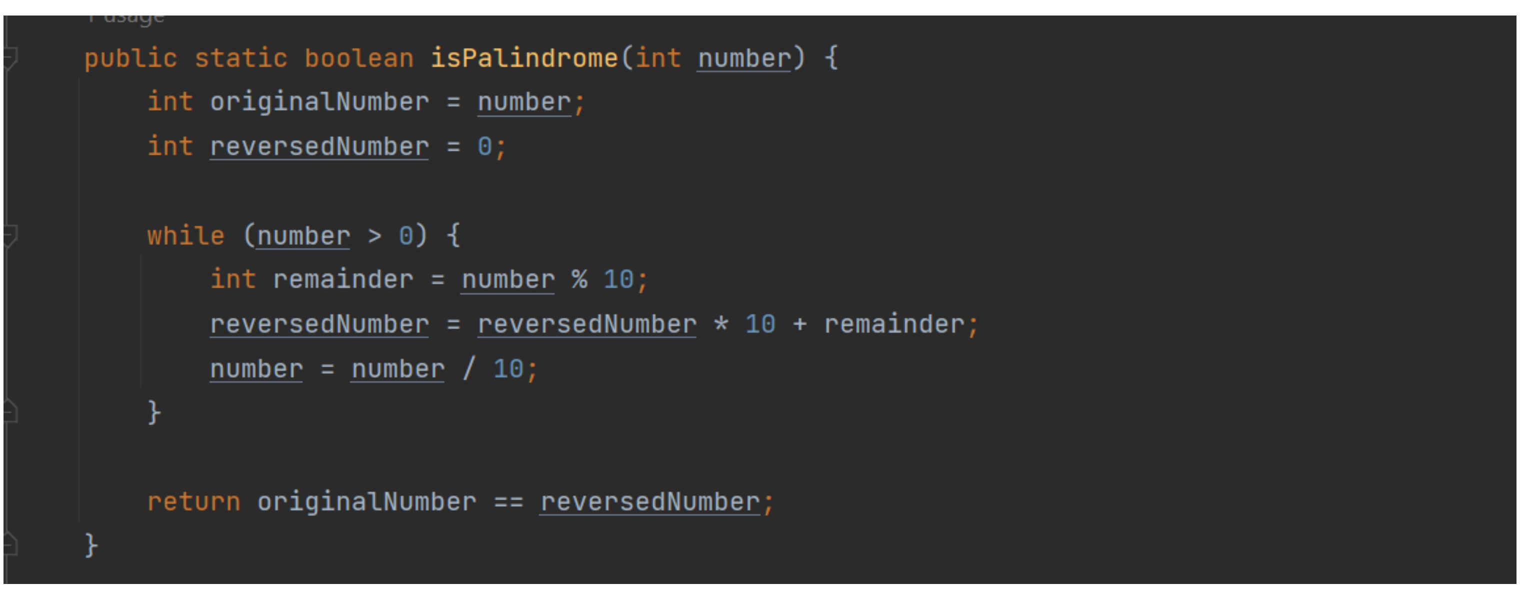


```java
    public static void main(String[] args) {
        SumCalculator calculator = new SumCalculator();

        // Testing the methods
        System.out.println("Sum of 5 and 7 (int): " + calculator.sum( a: 5,  b: 7));
        System.out.println("Sum of 5, 7, and 9 (int): " + calculator.sum( a: 5,  b: 7,  c: 9));
        System.out.println("Sum of 3.5 and 4.2 (double): " + calculator.sum( a: 3.5,  b: 4.2));
        System.out.println("Sum of 2.5 (double) and 8 (int): " + calculator.sum( a: 2.5,  b: 8));
    }
}
```

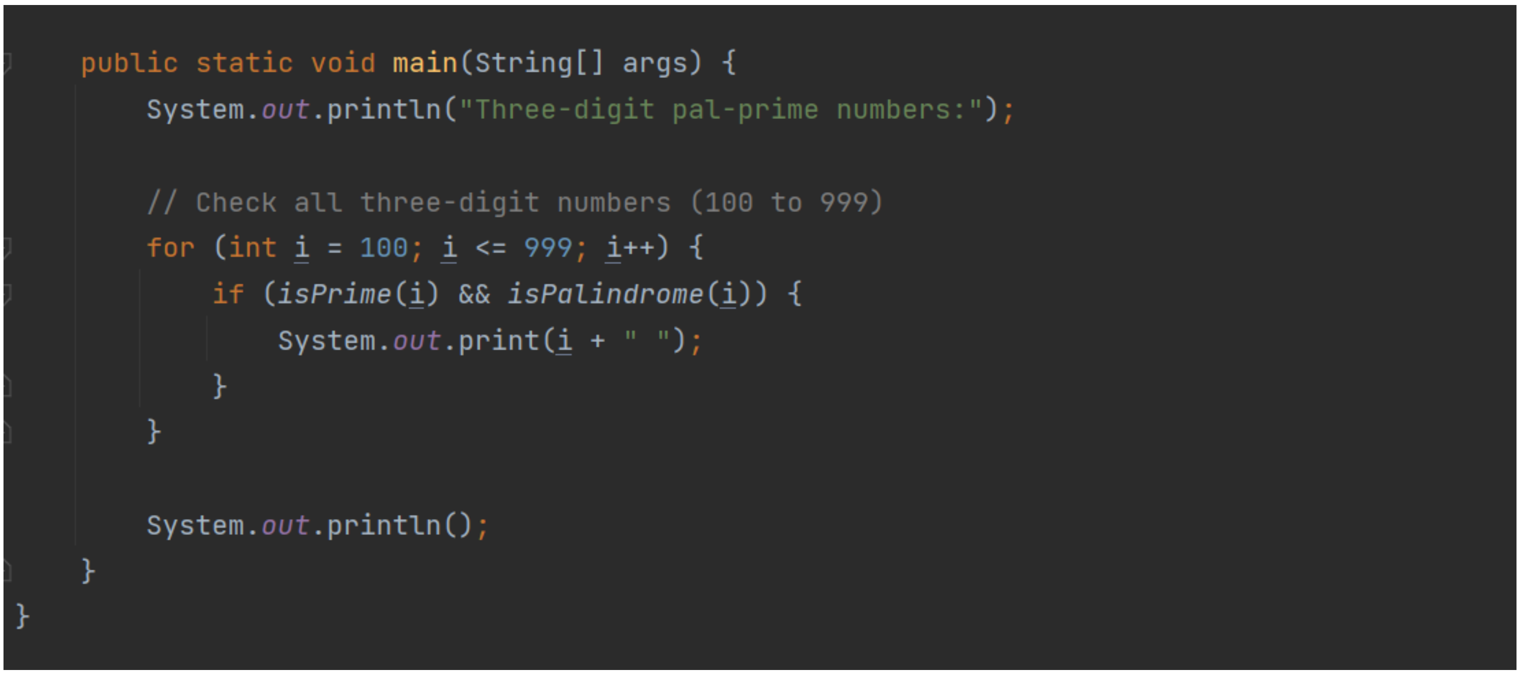
9) Create a function which accepts an integer as parameter and return true if it is a prime number Otherwise return false. Create another function which accepts an integer as parameter and Return true if it is palindrome otherwise return false. In the main() method display all three Digit pal-prime number. Palprime numbers are such numbers which are both palindrome as Well as prime numbers.



```java
public class PalPrimeNumbers {
    1 usage
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        if (number == 2) {
            return true;
        }
        if (number % 2 == 0) {
            return false;
        }

        for (int i = 3; i <= Math.sqrt(number); i += 2) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }
```

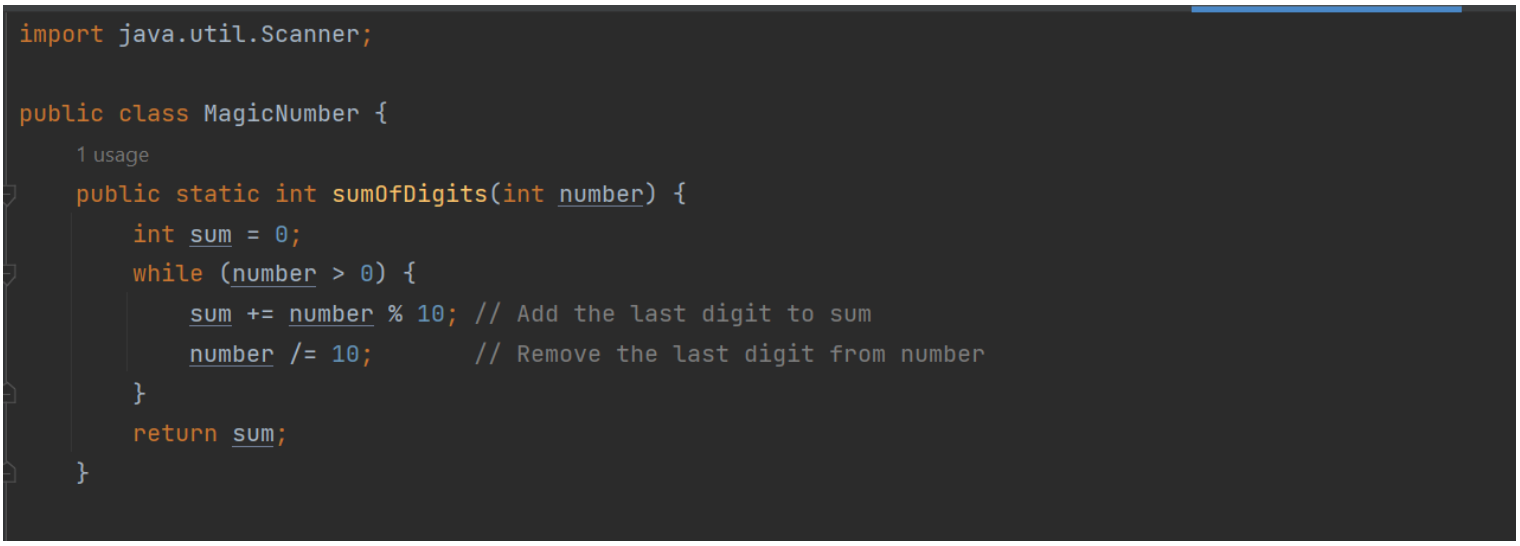


```java
    public static boolean isPalindrome(int number) {
        int originalNumber = number;
        int reversedNumber = 0;

        while (number > 0) {
            int remainder = number % 10;
            reversedNumber = reversedNumber * 10 + remainder;
            number = number / 10;
        }

        return originalNumber == reversedNumber;
    }
```

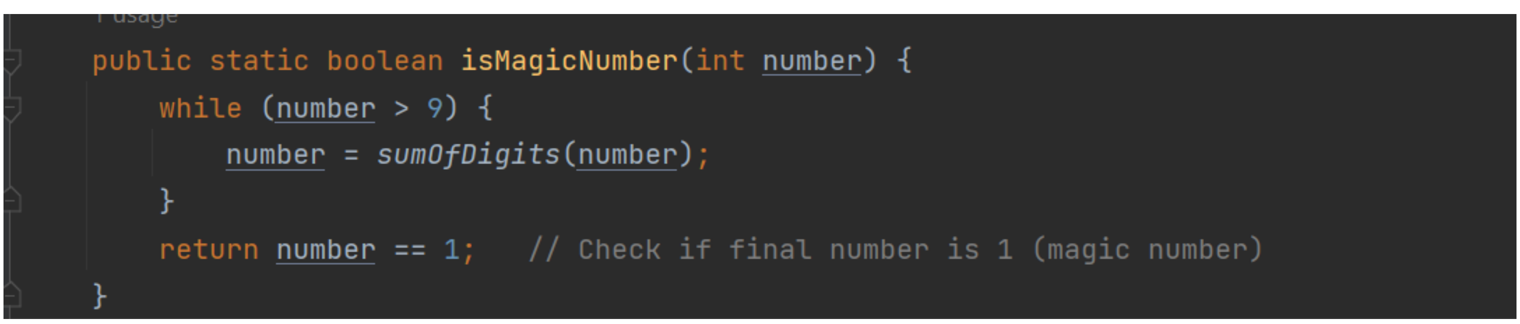


```
    public static void main(String[] args) {
        System.out.println("Three-digit pal-prime numbers:");

        // Check all three-digit numbers (100 to 999)
        for (int i = 100; i <= 999; i++) {
            if (isPrime(i) && isPalindrome(i)) {
                System.out.print(i + " ");
            }
        }

        System.out.println();
    }
}
```

10) Create a function which accepts an integer as parameter and return the sum of its digits. Create another function which accepts an integer as parameter and return true if it is magic number otherwise return false. In the main input an integer and check whether it is a magic number or not. If you iterate the process of summing the decimal digits of a number and if this process terminates in 1, then the original number is called a magic number.



```java
import java.util.Scanner;

public class MagicNumber {
    1 usage
    public static int sumOfDigits(int number) {
        int sum = 0;
        while (number > 0) {
            sum += number % 10; // Add the last digit to sum
            number /= 10;       // Remove the last digit from number
        }
        return sum;
    }
```



```java
    public static boolean isMagicNumber(int number) {
        while (number > 9) {
            number = sumOfDigits(number);
        }
        return number == 1;    // Check if final number is 1 (magic number)
    }
```