1. For a given input of two strings, return a Boolean TRUE if the two strings are anagrams.

```
import java.util.Arrays;

public class AnagramChecker {

    public static boolean areAnagrams(String str1, String str2) {
        // Remove spaces and convert to lowercase (optional)
        str1 = str1.replaceAll("\\s", "").toLowerCase();
        str2 = str2.replaceAll("\\s", "").toLowerCase();

        // Convert strings to char arrays and sort them
        char[] charArray1 = str1.toCharArray();
        char[] charArray2 = str2.toCharArray();
        Arrays.sort(charArray1);
        Arrays.sort(charArray2);

        // Compare the sorted arrays
        return Arrays.equals(charArray1, charArray2);
    }

    public static void main(String[] args) {
        String input1 = "listen";
        String input2 = "silent";
        System.out.println("Are \"" + input1 + "\" and \"" + input2 + "\" anagrams? " + areAnagrams(input1, in
put2));
    }
}
```

2. Create a pangram checker that returns a Boolean TRUE if an input string is a pangram and FALSE if it isn’t.

```
public class PangramChecker {

    public static boolean isPangram(String input) {
        // Convert the input string to lowercase (optional)
        input = input.toLowerCase();

        // Create a boolean array to keep track of letter occurrence
        boolean[] isLetterPresent = new boolean[26];

        // Traverse through the input string and mark letter occurrence
        for (int i = 0; i < input.length(); i++) {
            char currentChar = input.charAt(i);
            if (Character.isLetter(currentChar)) {
                int index = currentChar - ’a’;
                isLetterPresent[index] = true;
            }
        }

        // Check if all letters are present (i.e., array is filled with true values)
        for (boolean isPresent : isLetterPresent) {
            if (!isPresent) {
                return false;
            }
```

```
        }

        return true;
    }

    public static void main(String[] args) {
        String pangram = "The quick brown fox jumps over the lazy dog.";
        String nonPangram = "Hello, World!";

        System.out.println("\"" + pangram + "\" is a pangram? " + isPangram(pangram));
        System.out.println("\"" + nonPangram + "\" is a pangram? " + isPangram(nonPangram));
    }
}
```

3. create a perfect pangram checker. A perfect pangram is a sentence that uses each letter of the alphab et only once.

```
public class PangramChecker {

    public static boolean isPangram(String input) {
        // Convert the input string to lowercase (optional)
        input = input.toLowerCase();

        // Create a boolean array to keep track of letter occurrence
        boolean[] isLetterPresent = new boolean[26];

        // Traverse through the input string and mark letter occurrence
        for (int i = 0; i < input.length(); i++) {
            char currentChar = input.charAt(i);
            if (Character.isLetter(currentChar)) {
                int index = currentChar - 'a';
                isLetterPresent[index] = true;
            }
        }

        // Check if all letters are present (i.e., array is filled with true values)
        for (boolean isPresent : isLetterPresent) {
            if (!isPresent) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {
        String pangram = "The quick brown fox jumps over the lazy dog.";
        String nonPangram = "Hello, World!";

        System.out.println("\"" + pangram + "\" is a pangram? " + isPangram(pangram));
        System.out.println("\"" + nonPangram + "\" is a pangram? " + isPangram(nonPangram));
    }
}
```

4. reverse a decimal number

```
public class DecimalNumberReverser {

    public static int reverseDecimalNumber(int number) {
        String numberStr = Integer.toString(number);
        String reversedStr = new StringBuilder(numberStr).reverse().toString();
        return Integer.parseInt(reversedStr);
    }

    public static void main(String[] args) {
        int originalNumber = 12345;
        int reversedNumber = reverseDecimalNumber(originalNumber);
        System.out.println("Original Number: " + originalNumber);
        System.out.println("Reversed Number: " + reversedNumber);
    }
}
```

5. create an Armstrong number calculator that returns all Armstrong numbers between 0 and the input nu
mber.

```
public class ArmstrongNumberCalculator {

    public static int calculatePower(int base, int exponent) {
        int result = 1;
        for (int i = 0; i < exponent; i++) {
            result *= base;
        }
        return result;
    }

    public static int countDigits(int number) {
        return Integer.toString(number).length();
    }

    public static boolean isArmstrongNumber(int number) {
        int numDigits = countDigits(number);
        int sum = 0;
        int temp = number;

        while (temp > 0) {
            int digit = temp % 10;
            sum += calculatePower(digit, numDigits);
            temp /= 10;
        }

        return sum == number;
    }

    public static void findArmstrongNumbers(int inputNumber) {
        System.out.println("Armstrong numbers between 0 and " + inputNumber + ":");
        for (int i = 0; i <= inputNumber; i++) {
            if (isArmstrongNumber(i)) {
                System.out.print(i + " ");
            }
        }
```

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

    public static void main(String[] args) {
        int inputNumber = 1000;
        findArmstrongNumbers(inputNumber);
    }
}
```