1. Print even numbers from 1 to 20 using a `for` loop:

```
public class EvenNumbers {
    public static void main(String[] args) {
        for (int i = 1; i <= 20; i++) {
            if (i % 2 == 0) {
                System.out.println(i);
            }
        }
    }
}
```

2. Prompt user for a valid flight choice using a `while` loop:

```
import java.util.Scanner;

public class FlightChoice {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.print("Enter a valid flight choice (1-10): ");
            choice = scanner.nextInt();
        } while (choice < 1 || choice > 10);
        System.out.println("You selected flight " + choice);
    }
}
```

3. Discuss the pros and cons of different loop types for iterating arrays:

- **For loop**: Ideal for scenarios with a known range. Easier to control iteration.
- **While loop**: Useful when the number of iterations isn't known beforehand.
- **Enhanced for loop**: Cleaner syntax for traversing collections or arrays but lacks index control.

4. Print the first 10 numbers of the Fibonacci sequence:

```
public class FibonacciSequence {
    public static void main(String[] args) {
        int n1 = 0, n2 = 1, n3;
        System.out.print(n1 + " " + n2);
        for (int i = 2; i < 10; i++) {
            n3 = n1 + n2;
            System.out.print(" " + n3);
            n1 = n2;
            n2 = n3;
        }
    }
}
```

5. Sum integers from 1 to 100 using a `while` loop:

```
public class SumUsingWhile {
    public static void main(String[] args) {
        int sum = 0, i = 1;
        while (i <= 100) {
            sum += i;
            i++;
        }
        System.out.println("Sum is: " + sum);
    }
}
```

6. Prompt user until they enter a negative number using a `do-while` loop:

```
import java.util.Scanner;

public class NegativeNumberPrompt {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int num;
        do {
            System.out.print("Enter a number: ");
            num = scanner.nextInt();
        } while (num >= 0);
        System.out.println("You entered a negative number.");
    }
}
```

7. Demonstrate `continue` statement in a loop:

```
public class ContinueDemo {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i % 2 == 0) {
                continue;
            }
            System.out.println(i);
        }
    }
}
```

8. Initialize and print a 2D array:

```
public class TwoDArray {
    public static void main(String[] args) {
        int[][] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print(array[i][j] + " ");
            }
            System.out.println();
```

```
        }
    }
}
```

9. Compare `for` and `while` loops:

- **For loop**: Best for definite iterations (fixed ranges).
- **While loop**: More flexible for indefinite conditions.
- Use **for loop** for arrays and **while loop** for reading user input until valid data is entered.

10. Impact of `break` in nested loops:

A `break` in nested loops will exit only the innermost loop unless labeled breaks are used.

Example:

```
public class NestedBreak {
    public static void main(String[] args) {
        outer: for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                if (j == 2) break outer;
                System.out.println("i=" + i + ", j=" + j);
            }
        }
    }
}
```

11. How arrays improve data management:

- Arrays offer structured, indexed storage for similar data types.
- Example:

```
public class ArrayExample {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40};
        for (int num : numbers) {
            System.out.println(num);
        }
    }
}
```

---

12. Differences in memory allocation for single-dimensional and multi-dimensional arrays:

- **Single-dimensional arrays**: Contiguous memory allocation.
- **Multi-dimensional arrays**: Memory allocated row-wise; non-contiguous.

Example:

```
public class ArrayMemory {
    public static void main(String[] args) {
        int[] singleArray = {1, 2, 3};
        int[][] multiArray = {
            {1, 2},
            {3, 4},
            {5, 6}
        };
        System.out.println("Single-dimensional array: " + singleArray[1]);
        System.out.println("Multi-dimensional array: " + multiArray[1][1]);
    }
}
```

---

13. Potential pitfalls of using uninitialized arrays:

Accessing uninitialized arrays throws `NullPointerException`. Ensure proper initialization.

```
public class UninitializedArray {
    public static void main(String[] args) {
        int[] arr = new int[5]; // Initialized with default values (0).
        for (int i : arr) {
            System.out.println(i);
        }
    }
}
```

---

14. Method to return the maximum value in an array using a `for` loop:

```
public class MaxValue {
    public static int findMax(int[] array) {
        int max = array[0];
        for (int i = 1; i < array.length; i++) {
            if (array[i] > max) {
                max = array[i];
            }
        }
        return max;
    }
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        System.out.println("Maximum value: " + findMax(numbers));
    }
}
```

---

15. Find the average of numbers in an integer array:

```
public class AverageArray {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        int sum = 0;
```

```
        for (int num : numbers) {
            sum += num;
        }
        System.out.println("Average: " + (double) sum / numbers.length);
    }
}
```

---

16. Sum the elements of a 2D array:

```
public class Sum2DArray {
    public static void main(String[] args) {
        int[][] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        int sum = 0;
        for (int[] row : array) {
            for (int val : row) {
                sum += val;
            }
        }
        System.out.println("Sum: " + sum);
    }
}
```

---

17. Find the minimum and maximum values in an array:

```
public class MinMaxArray {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 5, 40, 50};
        int min = numbers[0], max = numbers[0];
        for (int num : numbers) {
            if (num < min) min = num;
            if (num > max) max = num;
        }
        System.out.println("Minimum: " + min);
        System.out.println("Maximum: " + max);
    }
}
```

---

18. Benefits and drawbacks of static vs. dynamic arrays:

- **Static arrays**: Fixed size; efficient memory use but limited flexibility.
- **Dynamic arrays**: Resizable; more memory overhead but greater flexibility.

---

19. Merge two sorted arrays into a single sorted array:

```
import java.util.Arrays;

public class MergeSortedArrays {
    public static int[] merge(int[] arr1, int[] arr2) {
        int[] merged = new int[arr1.length + arr2.length];
        int i = 0, j = 0, k = 0;

        while (i < arr1.length && j < arr2.length) {
            if (arr1[i] < arr2[j]) {
                merged[k++] = arr1[i++];
            } else {
                merged[k++] = arr2[j++];
            }
        }
        while (i < arr1.length) merged[k++] = arr1[i++];
        while (j < arr2.length) merged[k++] = arr2[j++];

        return merged;
    }
    public static void main(String[] args) {
        int[] arr1 = {1, 3, 5};
        int[] arr2 = {2, 4, 6};
        System.out.println(Arrays.toString(merge(arr1, arr2)));
    }
}
```

---

20. Reverse an array:

```
import java.util.Arrays;

public class ReverseArray {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        int start = 0, end = array.length - 1;
        while (start < end) {
            int temp = array[start];
            array[start] = array[end];
            array[end] = temp;
            start++;
            end--;
        }
        System.out.println(Arrays.toString(array));
    }
}
```

---

21. Find the second largest element in an array:

```
public class SecondLargest {
    public static void main(String[] args) {
        int[] array = {10, 20, 30, 40, 50};
        int largest = Integer.MIN_VALUE, secondLargest = Integer.MIN_VALUE;

        for (int num : array) {
```

```
            if (num > largest) {
                secondLargest = largest;
                largest = num;
            } else if (num > secondLargest && num != largest) {
                secondLargest = num;
            }
        }
        System.out.println("Second largest: " + secondLargest);
    }
}
```

---

22. Find the first even number in a list and break the loop:

```
public class FirstEven {
    public static void main(String[] args) {
        int[] numbers = {1, 3, 5, 6, 7};
        for (int num : numbers) {
            if (num % 2 == 0) {
                System.out.println("First even number: " + num);
                break;
            }
        }
    }
}
```

---

23. Print odd numbers using `continue`:

```
public class PrintOdd {
    public static void main(String[] args) {
        for (int i = 1; i <= 20; i++) {
            if (i % 2 == 0) continue;
            System.out.println(i);
        }
    }
}
```

---

24. Prompt user until they enter a negative number:

```
import java.util.Scanner;

public class PromptNegative {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int num;
        do {
            System.out.print("Enter a number: ");
            num = scanner.nextInt();
        } while (num >= 0);
        System.out.println("Negative number entered.");
    }
}
```

---

25. Print multiplication table skipping 5:

```
public class SkipFiveTable {
    public static void main(String[] args) {
        int number = 2; // Example for 2's table
        for (int i = 1; i <= 10; i++) {
            if (i == 5) continue;
            System.out.println(number + " x " + i + " = " + (number * i));
        }
    }
}
```

---

26. Count from 1 to 10 but break when it reaches 6:

```
public class BreakAtSix {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 6) break;
            System.out.println(i);
        }
    }
}
```

---

27. Print numbers from 1 to 10 but skip 5:

```
public class SkipFive {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 5) continue;
            System.out.println(i);
        }
    }
}
```

---

28. Check whether a number is prime using a `for` loop:

```
import java.util.Scanner;

public class PrimeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        boolean isPrime = true;

        if (number <= 1) isPrime = false;
        for (int i = 2; i <= Math.sqrt(number); i++) {
```

```
            if (number % i == 0) {
                isPrime = false;
                break;
            }
        }
        System.out.println(number + (isPrime ? " is a prime number." : " is
not a prime number."));
    }
}
```

---

29. Reverse the digits of a given integer:

```
import java.util.Scanner;

public class ReverseDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        int reversed = 0;

        while (number != 0) {
            reversed = reversed * 10 + number % 10;
            number /= 10;
        }
        System.out.println("Reversed number: " + reversed);
    }
}
```

---

30. Print multiplication table for a given number:

```
import java.util.Scanner;

public class MultiplicationTable {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        System.out.print("Enter the range: ");
        int range = scanner.nextInt();

        for (int i = 1; i <= range; i++) {
            System.out.println(number + " x " + i + " = " + (number * i));
        }
    }
}
```

---

31. Count vowels and consonants in a string:

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

```
public class CountVowelsConsonants {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = scanner.nextLine().toLowerCase();
        int vowels = 0, consonants = 0;

        for (char ch : str.toCharArray()) {
            if ("aeiou".indexOf(ch) != -1) {
                vowels++;
            } else if (ch >= 'a' && ch <= 'z') {
                consonants++;
            }
        }
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
    }
}
```

---

32. Print a pattern:

```
public class PrintPattern {
    public static void main(String[] args) {
        for (int i = 5; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("1 ");
            }
            System.out.println();
        }
    }
}
```

---

33. Feedback collection system with average rating:

```
import java.util.Scanner;

public class FeedbackSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int totalRating = 0, count = 0, rating;

        while (true) {
            System.out.print("Enter a rating (1-5) or 0 to exit: ");
            rating = scanner.nextInt();
            if (rating == 0) break;
            if (rating < 1 || rating > 5) {
                System.out.println("Invalid rating. Try again.");
                continue;
            }
            totalRating += rating;
            count++;
        }
        if (count > 0) {
```

```
            System.out.println("Average rating: " + (double) totalRating /
count);
            System.out.println("Total ratings: " + count);
        } else {
            System.out.println("No ratings provided.");
        }
    }
}
```

---

34. Track monthly expenses:

```
import java.util.Scanner;

public class MonthlyExpenses {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double total = 0;

        while (true) {
            System.out.print("Enter an expense or type 'done' to finish: ");
            String input = scanner.next();
            if (input.equalsIgnoreCase("done")) break;
            try {
                total += Double.parseDouble(input);
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a number or
'done'.");
            }
        }
        System.out.println("Total expenses: $" + total);
    }
}
```

---

35. Password validation system:

```
import java.util.Scanner;

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

        while (true) {
            System.out.print("Create a password: ");
            password = scanner.nextLine();
            if (password.length() >= 8 && password.matches(".*[A-Z].*") &&
                password.matches(".*[a-z].*") && password.matches(".*\\d.*")
&&
                password.matches(".*[!@#$%^&*].*")) {
                System.out.println("Password accepted.");
                break;
            } else {
```

```
                System.out.println("Password must be at least 8 characters
long and include uppercase, lowercase, digits, and special characters.");
            }
        }
    }
}
```

---

36. Log daily steps for a week:

```
import java.util.Scanner;

public class FitnessApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] steps = new int[7];
        int totalSteps = 0;

        for (int i = 0; i < 7; i++) {
            System.out.print("Enter steps for day " + (i + 1) + ": ");
            steps[i] = scanner.nextInt();
            totalSteps += steps[i];
        }
        System.out.println("Total steps: " + totalSteps);
        System.out.println("Average steps per day: " + (totalSteps / 7));
    }
}
```

---

37. Temperature conversion tool:

```
import java.util.Scanner;

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

        while (true) {
            System.out.print("Enter temperature (e.g., 30C or 86F) or type
'exit' to quit: ");
            String input = scanner.next();
            if (input.equalsIgnoreCase("exit")) break;

            try {
                char scale = input.charAt(input.length() - 1);
                double temp = Double.parseDouble(input.substring(0,
input.length() - 1));
                if (scale == 'C' || scale == 'c') {
                    System.out.println(temp + "C = " + (temp * 9 / 5 + 32) +
"F");
                } else if (scale == 'F' || scale == 'f') {
                    System.out.println(temp + "F = " + ((temp - 32) * 5 / 9)
+ "C");
                } else {
                    System.out.println("Invalid input. Use 'C' or 'F'.");
```

```
                }
            } catch (Exception e) {
                System.out.println("Invalid input format.");
            }
        }
    }
}
```

---

38. Simple banking system:

```
import java.util.Scanner;

public class BankingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double balance = 0;

        while (true) {
            System.out.print("Enter transaction (deposit/withdraw) or 'exit':
");
            String action = scanner.next();
            if (action.equalsIgnoreCase("exit")) break;

            System.out.print("Enter amount: ");
            double amount = scanner.nextDouble();
            if (action.equalsIgnoreCase("deposit")) {
                balance += amount;
            } else if (action.equalsIgnoreCase("withdraw")) {
                if (amount <= balance) {
                    balance -= amount;
                } else {
                    System.out.println("Insufficient funds.");
                }
            } else {
                System.out.println("Invalid transaction type.");
            }
            System.out.println("Current balance: $" + balance);
        }
    }
}
```

---

39. Input student grades and calculate stats:

```
import java.util.Scanner;

public class StudentGrades {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int total = 0, count = 0, highest = Integer.MIN_VALUE, passing = 0;

        while (true) {
            System.out.print("Enter a grade or -1 to stop: ");
            int grade = scanner.nextInt();
```

```
            if (grade == -1) break;

            total += grade;
            count++;
            if (grade > highest) highest = grade;
            if (grade >= 50) passing++;
        }

        if (count > 0) {
            System.out.println("Average grade: " + (double) total / count);
            System.out.println("Highest grade: " + highest);
            System.out.println("Passing students: " + passing);
        } else {
            System.out.println("No grades entered.");
        }
    }
}
```

---

40. Shopping cart application:

```
import java.util.Scanner;

public class ShoppingCart {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double total = 0;
        StringBuilder cart = new StringBuilder();

        while (true) {
            System.out.print("Enter item name or 'checkout': ");
            String item = scanner.next();
            if (item.equalsIgnoreCase("checkout")) break;

            System.out.print("Enter price: ");
            double price = scanner.nextDouble();
            total += price;
            cart.append(item).append(": $").append(price).append("\n");
        }

        System.out.println("Items purchased:\n" + cart);
        System.out.println("Total amount due: $" + total);
    }
}
```

---

41. Calculate total sales and commission:

```
import java.util.Scanner;

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

```
        double totalSales = 0;
        int count = 0;

        while (true) {
            System.out.print("Enter sales amount or negative to stop: ");
            double sales = scanner.nextDouble();
            if (sales < 0) break;

            totalSales += sales;
            count++;
        }

        if (count > 0) {
            System.out.println("Total sales: $" + totalSales);
            System.out.println("Average sales: $" + (totalSales / count));
        } else {
            System.out.println("No sales data entered.");
        }
    }
}
```

---

42. Reverse a String:

```
public class ReverseString {
    public static void main(String[] args) {
        String str = "Hello";
        String reversed = new StringBuilder(str).reverse().toString();
        System.out.println("Reversed String: " + reversed);
    }
}
```

---

43. Check if a String is a palindrome:

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "racecar";
        String reversed = new StringBuilder(str).reverse().toString();
        System.out.println(str.equals(reversed) ? "Palindrome" : "Not a
palindrome");
    }
}
```

---

44. Count occurrences of each character in a String:

```
import java.util.HashMap;

public class CharOccurrences {
    public static void main(String[] args) {
        String str = "swiss";
        HashMap<Character, Integer> occurrences = new HashMap<>();
```

```
        for (char c : str.toCharArray()) {
            occurrences.put(c, occurrences.getOrDefault(c, 0) + 1);
        }
        System.out.println(occurrences);
    }
}
```

---

45. Reverse a String without `StringBuilder`:

```
public class ReverseStringManual {
    public static void main(String[] args) {
        String str = "Hello";
        String reversed = "";
        for (int i = str.length() - 1; i >= 0; i--) {
            reversed += str.charAt(i);
        }
        System.out.println("Reversed String: " + reversed);
    }
}
```

---

46. Check if a String is a palindrome (method):

```
public class PalindromeMethod {
    public static boolean isPalindrome(String str) {
        int start = 0, end = str.length() - 1;
        while (start < end) {
            if (str.charAt(start++) != str.charAt(end--)) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        System.out.println(isPalindrome("racecar"));
    }
}
```

---

47. Count vowels and consonants:

Already provided above in Question 31.

---

48. Capitalize the first letter of each word:

```
public class CapitalizeWords {
    public static String capitalize(String str) {
        String[] words = str.split(" ");
        StringBuilder result = new StringBuilder();
        for (String word : words) {
```

```
            result.append(word.substring(0, 1).toUpperCase())
                  .append(word.substring(1)).append(" ");
        }
        return result.toString().trim();
    }

    public static void main(String[] args) {
        System.out.println(capitalize("hello world"));
    }
}
```

---

49. Check if two Strings are anagrams:

```
import java.util.Arrays;

public class AnagramCheck {
    public static boolean areAnagrams(String str1, String str2) {
        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        return Arrays.equals(arr1, arr2);
    }

    public static void main(String[] args) {
        System.out.println(areAnagrams("listen", "silent"));
    }
}
```

---

50. Remove duplicate characters:

```
public class RemoveDuplicates {
    public static String removeDuplicates(String str) {
        StringBuilder result = new StringBuilder();
        for (char c : str.toCharArray()) {
            if (result.indexOf(String.valueOf(c)) == -1) {
                result.append(c);
            }
        }
        return result.toString();
    }

    public static void main(String[] args) {
        System.out.println(removeDuplicates("programming"));
    }
}
```

---

51. Find the first non-repeating character:

```
import java.util.LinkedHashMap;
```

```
public class FirstNonRepeating {
    public static char firstNonRepeatingChar(String str) {
        LinkedHashMap<Character, Integer> map = new LinkedHashMap<>();
        for (char c : str.toCharArray()) {
            map.put(c, map.getOrDefault(c, 0) + 1);
        }
        for (char c : map.keySet()) {
            if (map.get(c) == 1) return c;
        }
        return '\0';
    }

    public static void main(String[] args) {
        System.out.println(firstNonRepeatingChar("swiss"));
    }
}
```

---

52. Compress a String with counts:

```
public class StringCompression {
    public static String compress(String str) {
        StringBuilder compressed = new StringBuilder();
        int count = 1;

        for (int i = 0; i < str.length() - 1; i++) {
            if (str.charAt(i) == str.charAt(i + 1)) {
                count++;
            } else {
                compressed.append(str.charAt(i)).append(count);
                count = 1;
            }
        }
        compressed.append(str.charAt(str.length() - 1)).append(count);
        return compressed.length() < str.length() ? compressed.toString() :
str;
    }

    public static void main(String[] args) {
        System.out.println(compress("aabcccccaaa"));
    }
}
```

---

53. Append " World" to StringBuffer:

```
public class AppendStringBuffer {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello");
        sb.append(" World");
        System.out.println(sb);
    }
}
```

---

54. Insert "Beautiful" at index 6 in StringBuffer:

```
public class InsertBeautiful {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello World");
        sb.insert(6, "Beautiful ");
        System.out.println(sb);
    }
}
```

---

55. Reverse StringBuffer contents:

```
public class ReverseStringBuffer {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Java Programming");
        sb.reverse();
        System.out.println(sb);
    }
}
```

---

56. Delete substring from StringBuffer:

```
public class DeleteSubstring {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello World");
        sb.delete(6, 11);
        System.out.println(sb);
    }
}
```

---

57. Reverse StringBuffer contents:

Duplicate of Question 55.

---

58. Delete "World" from StringBuffer:

Duplicate of Question 56.

---

59. Replace "Java" with "Python" in StringBuffer:

```
public class ReplaceStringBuffer {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("I love Java programming");
```

```
        int start = sb.indexOf("Java");
        int end = start + "Java".length();
        sb.replace(start, end, "Python");
        System.out.println(sb);
    }
}
```

---

60. StringBuffer capacity handling:

```
public class StringBufferCapacity {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer();
        System.out.println("Initial capacity: " + sb.capacity());
        sb.append("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        System.out.println("New capacity: " + sb.capacity());
    }
}
```

---

61. Convert StringBuffer to String:

```
public class BufferToString {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello World");
        String str = sb.toString();
        System.out.println(str);
    }
}
```

---

62. Count vowels in StringBuffer:

```
public class CountVowelsBuffer {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello World");
        int count = 0;

        for (int i = 0; i < sb.length(); i++) {
            char c = sb.charAt(i);
            if ("aeiouAEIOU".indexOf(c) != -1) count++;
        }
        System.out.println("Vowel count: " + count);
    }
}
```

---

63. Trim spaces from StringBuffer:

```
public class TrimStringBuffer {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("   Hello World   ");
```

```
        String trimmed = sb.toString().trim();
        System.out.println(trimmed);
    }
}
```

---

64. Merge two StringBuffers:

```
public class MergeStringBuffers {
    public static void main(String[] args) {
        StringBuffer sb1 = new StringBuffer("Hello");
        StringBuffer sb2 = new StringBuffer("World");
        sb1.append(" ").append(sb2);
        System.out.println(sb1);
    }
}
```