Programs:

1.Write a Java method to display the middle character of a string. Note: a) If the length of the string is odd there will be two middle characters. b) If the length of the string is even there will be one middle character.

Code:

```
public class MiddleCharacter {

    public static void displayMiddleCharacter(String str) {
        int length = str.length();
        if (length % 2 == 0) {
            System.out.println("Middle character: " + str.charAt(length / 2 - 1));
        } else {
            System.out.println("Middle characters: " + str.charAt(length / 2 - 1) + str.charAt(length / 2));
        }
    }

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

```
        displayMiddleCharacter("example");
        displayMiddleCharacter("test");
    }
}
```

Output:

```
java -cp /tmp/vQZDfk4gSy/MiddleCharacter
Middle characters: am
Middle character: e

=== Code Execution Successful ===
```

2) 2. Write a Java method to check whether a string is a valid password. Password rules: A password must have at least ten characters. A password consists of only letters and digits. A password must contain at least two digits.

Code and output:

```
public class PasswordValidator {

    public static boolean isValidPassword(String password) {
        if (password.length() < 10) return false;

        int digitCount = 0;
        for (char ch : password.toCharArray()) {
            if (!Character.isLetterOrDigit(ch)) return false;
            if (Character.isDigit(ch)) digitCount++;
        }

        return digitCount >= 2;
    }

    public static void main(String[] args) {
        String password = "Password123";
        System.out.println(isValidPassword(password) ? "Valid password" :
            "Invalid password");
    }
}
```

```
java -cp /tmp/aV19lUncCL/PasswordValidator
Valid password

=== Code Execution Successful ===
```

3.Write a Java recursive method to check if a given array is sorted in ascending order

Code and output:

```java
public class ArraySortChecker {

    public static boolean isSorted(int[] array, int index) {
        if (index == array.length - 1) {
            return true;
        }
        if (array[index] > array[index + 1]) {
            return false;
        }
        return isSorted(array, index + 1);
    }

    public static void main(String[] args) {
        int[] array = {7, 6, 3, 4, 1};
        System.out.println(isSorted(array, 0) ? "Array is sorted" : "Array is not
            sorted");
    }
}
```

```
java -cp /tmp/eXhe8ZspE9/ArraySortChecker
Array is not sorted

=== Code Execution Successful ===
```

4. write a Java program to create a class called "Initializer" with a static block that initializes a static variable 'initialValue' to 1000. Print the value of 'initialValue' before and after creating an instance of "Initializer".

Code and output:

```java
public class Initializer {
    static int initialValue;
    static {
        initialValue = 1000;
        System.out.println("Static block executed. initialValue = " +
            initialValue);
    }
    public Initializer() {
        System.out.println("Initializer instance created.");
    }

    public static void main(String[] args) {
        System.out.println("Before creating an instance, initialValue = " +
            Initializer.initialValue);

        Initializer initializer = new Initializer();

        System.out.println("After creating an instance, initialValue = " +
            Initializer.initialValue);
    }
}
```

```
java -cp /tmp/JdOS3LzPjY/Initializer
Static block executed. initialValue = 1000
Before creating an instance, initialValue = 1000
Initializer instance created.
After creating an instance, initialValue = 1000

=== Code Execution Successful ===
```

5. write a Java program to create a class called "IDGenerator" with a static variable 'nextID' and a static method "generateID()" that returns the next ID and increments 'nextID'. Demonstrate the usage of generateID in the main method

Code and output:

```java
public class IDGenerator {
    private static int nextID = 1;
    public static int generateID() {
        return nextID++;
    }

    public static void main(String[] args) {
        System.out.println("Generated ID: " + IDGenerator.generateID());
        System.out.println("Generated ID: " + IDGenerator.generateID());
        System.out.println("Generated ID: " + IDGenerator.generateID());
    }
}
```

```
java -cp /tmp/NRMa3OG7ad/IDGenerator
Generated ID: 1
Generated ID: 2
Generated ID: 3

=== Code Execution Successful ===
```

6. Write a Java program to create a class called Dog with instance variables name and color. Implement a parameterized constructor that takes name and color as parameters and initializes the instance variables. Print the values of the variables.

Code and output:

```java
public class Dog {
    private String name;
    private String color;
    public Dog(String name, String color) {
        this.name = name;
        this.color = color;
    }
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Color: " + color);
    }

    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy", "Brown");
        myDog.printDetails();
    }
}
```

```
java -cp /tmp/lcvQLnqvBs/Dog
Name: Buddy
Color: Brown

=== Code Execution Successful ===
```

7. Write a Java program to create a class called "Book" with instance variables title, author, and price. Implement a default constructor and two parameterized constructors: One constructor takes title and author as parameters. The other constructor takes title, author, and price as parameters. Print the values of the variables for each constructor.

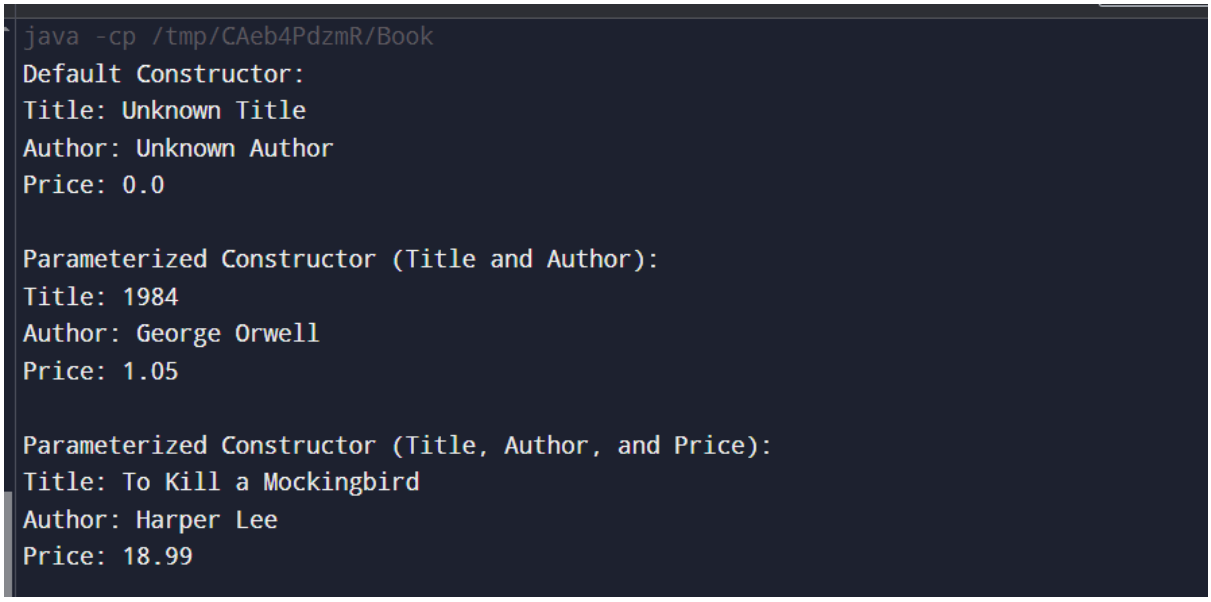
Code:

```
public class Book {
    // Instance variables
    private String title;
    private String author;
    private double price;

    // Default constructor
    public Book() {
        this.title = "Unknown Title";
        this.author = "Unknown Author";
        this.price = 0.0;
    }
    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.price = 0.0;
    }
    public Book(String title, String author, double price) {
        this.title = title;
        this.author = author;
        this.price = price;
    }
```

```
    public void printDetails() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: " + price);
    }

    public static void main(String[] args) {
        // Create an instance using the default constructor
        Book defaultBook = new Book();
        System.out.println("Default Constructor:");
        defaultBook.printDetails();

        // Create an instance using the parameterized constructor with title and author
        Book bookWithTitleAndAuthor = new Book("1984", "George Orwell");
        System.out.println("\nParameterized Constructor (Title and Author):");
        bookWithTitleAndAuthor.printDetails();

        // Create an instance using the parameterized constructor with title, author, and price
```

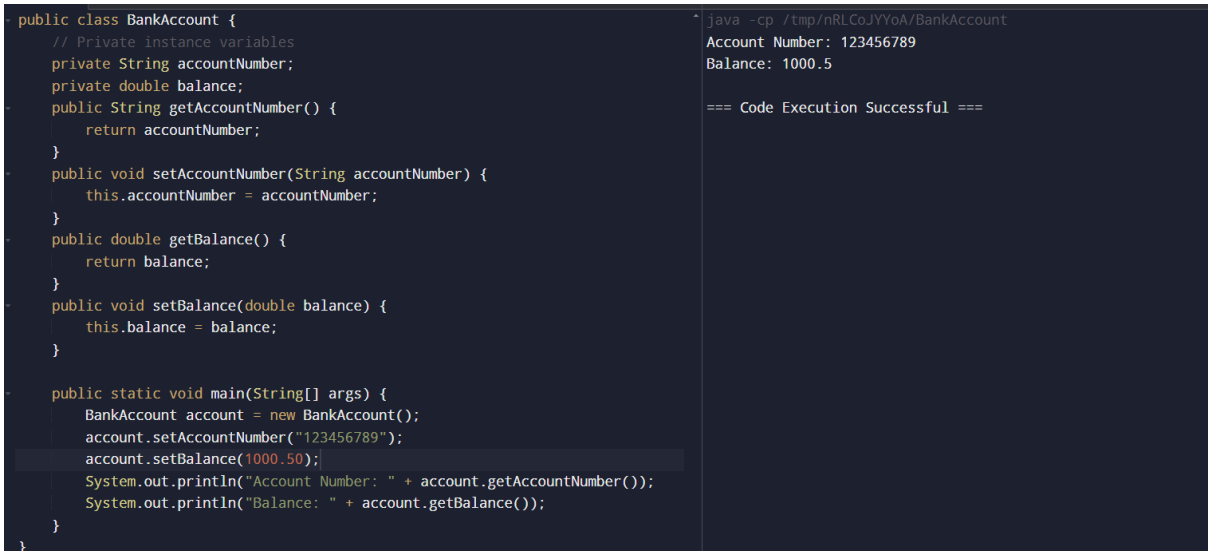
```
        Book bookWithAllDetails = new Book("To Kill a
Mockingbird", "Harper Lee", 18.99);

        System.out.println("\nParameterized Constructor
(Title, Author, and Price):");

        bookWithAllDetails.printDetails();

    }

}
```

Output:



8.Write a Java program to create a class called BankAccount with private instance variables accountNumber and balance. Provide public getter and setter methods to access and modify these variables.

Code and output:



```java
public class BankAccount {
    // Private instance variables
    private String accountNumber;
    private double balance;
    public String getAccountNumber() {
        return accountNumber;
    }
    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }
    public double getBalance() {
        return balance;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }

    public static void main(String[] args) {
        BankAccount account = new BankAccount();
        account.setAccountNumber("123456789");
        account.setBalance(1000.50);
        System.out.println("Account Number: " + account.getAccountNumber());
        System.out.println("Balance: " + account.getBalance());
    }
}
```

```
java -cp /tmp/nRLCoJYYoA/BankAccount
Account Number: 123456789
Balance: 1000.5

=== Code Execution Successful ===
```

9. Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

Code and output:

```
interface Playable {
    void play();
}


class Football implements Playable {
    public void play() {
        System.out.println("Playing football!");
    }
}
```

```
class Volleyball implements Playable {
    public void play() {
        System.out.println("Playing volleyball!");
    }
}

class Basketball implements Playable {
    public void play() {
        System.out.println("Playing basketball!");
    }
}

public class SportGame {
    public static void main(String[] args) {
        Playable footballGame = new Football();
        footballGame.play();

        Playable volleyballGame = new Volleyball();
        volleyballGame.play();

        Playable basketballGame = new Basketball();
        basketballGame.play();
```

```
    }
}
```

```
java -cp /tmp/vbz4qjyNYl/SportGame
Playing football!
Playing volleyball!
Playing basketball!

=== Code Execution Successful ===
```

10. Write a Java program to create a method that takes an integer as a parameter and throws an exception if the number is odd.

Code and output:

```
class OddNumberException extends Exception {
    public OddNumberException(String message) {
        super(message);
    }
}


public class OddNumberChecker {


    // Method to check if the number is odd
    public static void checkEven(int number) throws
OddNumberException {
```

```
        if (number % 2 != 0) {
            throw new OddNumberException("The number " + number + " is odd.");
        } else {
            System.out.println("The number " + number + " is even.");
        }
    }

    public static void main(String[] args) {
        // Test the method with different numbers
        int[] testNumbers = {2, 3, 4, 5, 6};

        for (int number : testNumbers) {
            try {
                checkEven(number);
            } catch (OddNumberException e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
```

```
java -cp /tmp/OlNotRk4N2/OddNumberChecker
The number 2 is even.
The number 3 is odd.
The number 4 is even.
The number 5 is odd.
The number 6 is even.

=== Code Execution Successful ===
```

11. Write a Java program to create a method that takes a string as input and throws an exception if the string does not contain vowels.

Code and output:

```
class NoVowelException extends Exception {
    public NoVowelException(String message) {
        super(message);
    }
}


public class VowelChecker {
    public static void checkForVowels(String input) throws NoVowelException {
        String lowerCaseInput = input.toLowerCase();

        // Check for vowels
        if (!lowerCaseInput.matches(".[aeiou]")) {
```

```
            throw new NoVowelException("The string does not
contain any vowels.");
        } else {
            System.out.println("The string contains vowels.");
        }
    }

    public static void main(String[] args) {
        // Test the method with different strings
        String testString1 = "Hello World";
        String testString2 = "Rhythm";

        try {
            checkForVowels(testString1);
            checkForVowels(testString2);
        } catch (NoVowelException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
java -cp /tmp/V0d6ELszYN/VowelChecker
The string does not contain any vowels.
```

12. Write a Java program to print the following grid. Expected

Code and output:

```
public class GridPrinter {
    public static void main(String[] args) {
        // Define the number of rows and columns
        int rows = 10;
        int columns = 10;

        // Print the grid
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                System.out.print("- ");
            }
            System.out.println(); // Move to the next line after each row
        }

        // Print the last row with two dashes
        System.out.println("- -");
    }
}
```

```
java -cp /tmp/37THV4nSG5/GridPrinter
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- -

=== Code Execution Successful ===
```

13. Write a Java program to create a generic method that takes two lists of the same type and merges them into a single list. This method alternates the elements of each list.

Code and output:

```
import java.util.ArrayList;

import java.util.List;


public class ListMerger {

    public static <T> List<T> mergeLists(List<T> list1, List<T> list2) {

        List<T> mergedList = new ArrayList<>();

        int maxLength = Math.max(list1.size(), list2.size());


        for (int i = 0; i < maxLength; i++) {

            if (i < list1.size()) {
```

```
            mergedList.add(list1.get(i));
        }
        if (i < list2.size()) {
            mergedList.add(list2.get(i));
        }
    }

    return mergedList;
}

public static void main(String[] args) {
    // Example usage
    List<String> list1 = List.of("A", "B", "C");
    List<String> list2 = List.of("1", "2", "3", "4", "5");

    List<String> mergedList = mergeLists(list1, list2);

    // Print the merged list
    System.out.println("Merged List: " + mergedList);
}
}
```

```
java -cp /tmp/LTiPOMZGKQ/ListMerger
Merged List: [A, 1, B, 2, C, 3, 4, 5]
```

14. Write a Java program to sort an array of given integers using the Selection Sort Algorithm.

Code and output:

```
public class SelectionSort {
    public static void selectionSort(int[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {

            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] < array[minIndex]) {
                    minIndex = j;
                }
            }
            if (minIndex != i) {
                int temp = array[minIndex];
                array[minIndex] = array[i];
                array[i] = temp;
            }
        }
```

```
}

// Method to print the array
public static void printArray(int[] array) {
    for (int value : array) {
        System.out.print(value + " ");
    }
    System.out.println();
}

// Main method to test the selection sort
public static void main(String[] args) {
    int[] array = {64, 25, 12, 22, 11};

    System.out.println("Original array:");
    printArray(array);

    selectionSort(array);

    System.out.println("Sorted array:");
    printArray(array);
}
```

}

```
Original array:
64 25 12 22 11
Sorted array:
11 12 22 25 64
```

15. Write a Java program to find a specified element in a given array of elements using Binary Search.

Code and output:

```
import java.util.Arrays;

import java.util.Scanner;


public class BinarySearchExample {


   public static int binarySearch(int[] arr, int target) {
      int left = 0;
      int right = arr.length - 1;


      while (left <= right) {
         int mid = left + (right - left) / 2; // Avoids potential overflow


         // Check if the target is present at mid
```

```
        if (arr[mid] == target) {
            return mid; // Target found
        }
        // If target is greater, ignore the left half
        else if (arr[mid] < target) {
            left = mid + 1;
        }
        // If target is smaller, ignore the right half
        else {
            right = mid - 1;
        }
    }
    // Target was not found
    return -1;
}

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

    // Sample array (must be sorted for binary search)
    int[] arr = {2, 3, 4, 10, 40, 50, 60, 70, 80, 90};
```

```
        // Display the array
        System.out.println("Array: " + Arrays.toString(arr));
        System.out.print("Enter the element to search: ");
        int target = scanner.nextInt();

        // Perform binary search
        int result = binarySearch(arr, target);
        if (result == -1) {
            System.out.println("Element not found in the
array.");
        } else {
            System.out.println("Element found at index: " +
result);
        }
        scanner.close();
    }
}
```

```
java -cp /tmp/Tcf7yGFmt9/BinarySearchExample
Array: [2, 3, 4, 10, 40, 50, 60, 70, 80, 90]
Enter the element to search: 50
Element found at index: 5
```

16. Write a Java program to find sequences of lowercase letters joined by an underscore.

Code aand output:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class LowercaseSequenceFinder {
    public static void main(String[] args) {
        // Sample input string
        String input = "this_is_a_test_string with
some_lowercase_sequences and some_Uppercase";

        findLowercaseSequences(input);
    }

    public static void findLowercaseSequences(String input) {
        String regex = "[a-z]+(_[a-z]+)+";

        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(input);

        System.out.println("Found sequences of lowercase
letters joined by underscores:");
        while (matcher.find()) {
```

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

```
java -cp /tmp/n6JymfbhNF/LowercaseSequenceFinder
Found sequences of lowercase letters joined by underscores:
this_is_a_test_string
some_lowercase_sequences
```

17. Write a Java program that matches a word containing 'g', not at the start or end of the word.

Code and output:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class WordMatcher {
    public static void main(String[] args) {
        // Sample words to test
        String[] words = {
            "giraffe",
            "dog",
            "cat",
            "glove",
            "raging",
```

```
            "swing",
            "growing",
            "egg"
        };
        String regex = "^[^g].g.[^g]$"; // g not at the start or
end
        Pattern pattern = Pattern.compile(regex);


        // Check each word
        for (String word : words) {
            Matcher matcher = pattern.matcher(word);
            if (matcher.find()) {
                System.out.println(word + " matches the
criteria.");
            } else {
                System.out.println(word + " does not match the
criteria.");
            }
        }
    }
}
```

```
java -cp /tmp/vxZYYYQVu/WordMatcher
giraffe does not match the criteria.
dog does not match the criteria.
cat does not match the criteria.
glove does not match the criteria.
raging does not match the criteria.
swing does not match the criteria.
growing does not match the criteria.
egg does not match the criteria.
```