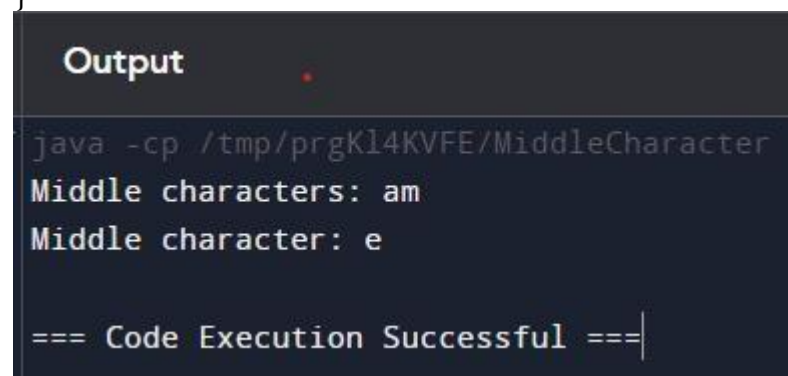


JAVA PROGRAMS (22-08-2024)

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.

```
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/prgKl4KVFE/MiddleCharacter  
Middle characters: am  
Middle character: e  
  
=== Code Execution Successful ===
```

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.

import java.util.Scanner;

```
public class PasswordValidator {

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

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

        return digitCount >= 2;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a password: ");
        String password = scanner.nextLine();

        if (isValidPassword(password)) {
            System.out.println("Valid password.");
        } else {
            System.out.println("Invalid password.");
        }

        scanner.close();
    }
}
```

Output

```
java -cp /tmp/ppAwrogo5S/PasswordValidato
Enter a password: 123USER123
Valid password.

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

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

```
public class ArrayUtil {

    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[] sortedArray = {1, 2, 3, 4, 5};
        int[] unsortedArray = {1, 3, 2, 4, 5};

        System.out.println("Is the sortedArray sorted? " + isSorted(sortedArray, 0));
        System.out.println("Is the unsortedArray sorted? " + isSorted(unsortedArray, 0));
    }
}
```

OUTPUT

```
Is the sortedArray sorted? true
Is the unsortedArray sorted? false

=== 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".

```
4]public class Initializer {  
    static int initialValue;  
  
    static {  
        initialValue = 1000;  
        System.out.println("Static block executed: initialValue = " + initialValue);  
    }  
  
    public Initializer() {  
        System.out.println("Constructor executed");  
    }  
  
    public static void main(String[] args) {  
        System.out.println("Value of initialValue before creating an instance: " +  
Initializer.initialValue);  
  
        Initializer obj = new Initializer();  
  
        System.out.println("Value of initialValue after creating an instance: " +  
Initializer.initialValue);  
    }  
}
```

OUTPUT

```
Static block executed: initialValue = 1000  
Value of initialValue before creating an instance: 1000  
Constructor executed  
Value of initialValue after creating an instance: 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.

```
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()); // Output:
Generated ID: 1
    System.out.println("Generated ID: " + IDGenerator.generateID()); // Output:
Generated ID: 2
    System.out.println("Generated ID: " + IDGenerator.generateID()); // Output:
Generated ID: 3
    System.out.println("Generated ID: " + IDGenerator.generateID()); // Output:
Generated ID: 4
}
}

```

OUTPUT

```

Generated ID: 1
Generated ID: 2
Generated ID: 3
Generated ID: 4

=== 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.

```

public class Dog {
    private String name;
    private String color;
    public Dog(String name, String color) {
        this.name = name;
        this.color = color;
    }
}

```

```

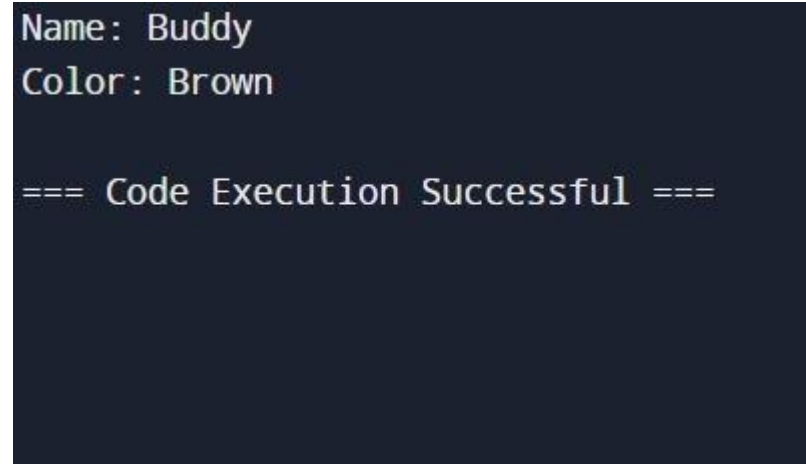
public void printDogDetails() {
    System.out.println("Name: " + name);
    System.out.println("Color: " + color);
}

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

    myDog.printDogDetails();
}
}

```

OUTPUT



```

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.

```

public class Book {
    private String title;
    private String author;
    private double price;

    public Book() {
        this.title = "Unknown";
        this.author = "Unknown";
        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 printBookDetails() {
    System.out.println("Title: " + title);
    System.out.println("Author: " + author);
    System.out.println("Price: $" + price);
    System.out.println();
}

public static void main(String[] args) {
    Book book1 = new Book(); // Default constructor
    Book book2 = new Book("1984", "George Orwell");
    Book book3 = new Book("To Kill a Mockingbird", "Harper Lee", 15.99);

    System.out.println("Book 1:");
    book1.printBookDetails();

    System.out.println("Book 2:");
    book2.printBookDetails();

    System.out.println("Book 3:");
    book3.printBookDetails();
}
}

```

OUTPUT

```
Book 1:
Title: Unknown
Author: Unknown
Price: $0.0

Book 2:
Title: 1984
Author: George Orwell
Price: $0.0

Book 3:
Title: To Kill a Mockingbird
Author: Harper Lee
Price: $15.99

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

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.

```
public class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double initialBalance) {
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }
}
```



```

    }
    public double getBalance() {
        return balance;
    }
    public void setBalance(double balance) {
        if (balance >= 0) {
            this.balance = balance;
        } else {
            System.out.println("Balance cannot be negative.");
        }
    }
}
public static void main(String[] args) {
    BankAccount account = new BankAccount("123456789", 1000.0);

    System.out.println("Account Number: " + account.getAccountNumber());
    System.out.println("Balance: " + account.getBalance());

    account.setAccountNumber("987654321");
    account.setBalance(1500.0);

    System.out.println("Updated Account Number: " +
account.getAccountNumber());
    System.out.println("Updated Balance: " + account.getBalance());

    account.setBalance(-500.0);
}
}

```

OUTPUT

```
Account Number: 123456789
Balance: 1000.0
Updated Account Number: 987654321
Updated Balance: 1500.0
Balance cannot be negative.

=== 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.

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

class Football implements Playable {
    @Override
    public void play() {
        System.out.println("Playing Football");
    }
}

class Volleyball implements Playable {
    @Override
    public void play() {
        System.out.println("Playing Volleyball");
    }
}

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

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

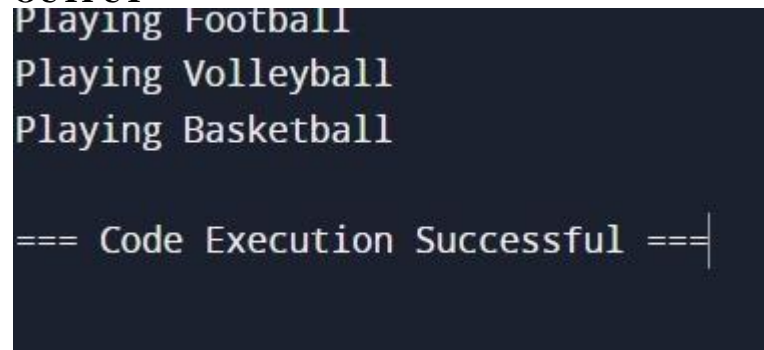
```

    Playable football = new Football();
    Playable volleyball = new Volleyball();
    Playable basketball = new Basketball();

    football.play();
    volleyball.play();
    basketball.play();
}
}

```

OUTPUT



```

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.

```

public class OddNumberCheck {

    public static void main(String[] args) {
        try {
            checkEven(4);
            checkEven(7);
            checkEven(10);
        } catch (OddNumberException e) {
            System.out.println(e.getMessage());
        }
    }

    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.");
        }
    }
}

class OddNumberException extends Exception {

```

```

    public OddNumberException(String message) {
        super(message);
    }
}

```

OUTPUT

```

The number 4 is even.
The number 7 is odd.

=== 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.

```

public class VowelCheck {

    public static void main(String[] args) {
        try {
            checkVowels("Hello");
            checkVowels("Sky");
            checkVowels("Rhythm");
        } catch (NoVowelException e) {
            System.out.println(e.getMessage());
        }
    }

    public static void checkVowels(String input) throws NoVowelException {
        if (!input.matches("[aeiouAEIOU].")) {
            throw new NoVowelException("The string \"" + input + "\" does not contain any vowels.");
        } else {
            System.out.println("The string \"" + input + "\" contains vowels.");
        }
    }
}

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

```

OUTPUT

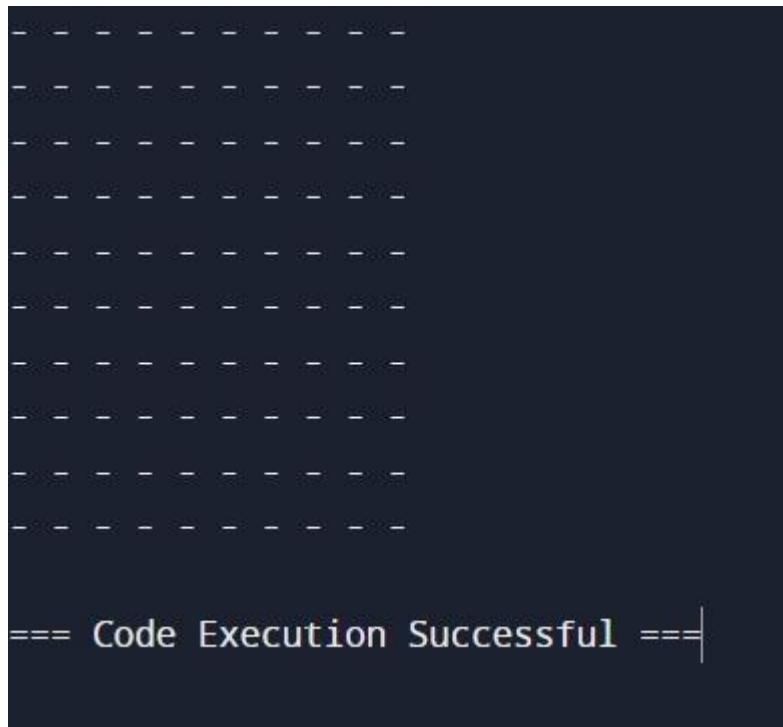
```
The string "Hello" contains vowels.  
The string "Sky" does not contain any vowels.  
  
=== Code Execution Successful ===
```

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

```
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -
```

```
public class PrintGrid {  
    public static void main(String[] args) {  
        int rows = 10;  
        int columns = 10;  
  
        for (int i = 0; i < rows; i++) {  
            for (int j = 0; j < columns; j++) {  
                System.out.print("- ");  
            }  
            System.out.println();  
        }  
    }  
}
```

OUTPUT



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.

```
import java.util.ArrayList;
import java.util.List;
```

```
public class MergeLists {
```

```
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>();
        List<Integer> list2 = new ArrayList<>();
```

```
        list1.add(1);
        list1.add(3);
        list1.add(5);
```

```
        list2.add(2);
        list2.add(4);
        list2.add(6);
```

```
        List<Integer> mergedList = mergeListsAlternating(list1, list2);
```

```
        System.out.println("Merged List: " + mergedList);
    }
```

```

public static <T> List<T> mergeListsAlternating(List<T> list1, List<T> list2) {
    List<T> mergedList = new ArrayList<>();
    int size1 = list1.size();
    int size2 = list2.size();
    int maxSize = Math.max(size1, size2);

    for (int i = 0; i < maxSize; i++) {
        if (i < size1) {
            mergedList.add(list1.get(i));
        }
        if (i < size2) {
            mergedList.add(list2.get(i));
        }
    }

    return mergedList;
}

```

OUTPUT

Merged List: [1, 2, 3, 4, 5, 6]

=== Code Execution Successful ===

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

```

import java.util.Scanner;

public class SelectionSortExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();

        int[] array = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            array[i] = scanner.nextInt();
        }
    }
}

```

```

        selectionSort(array);

        System.out.println("Sorted array:");
        for (int element : array) {
            System.out.print(element + " ");
        }

        scanner.close();
    }
    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;
                }
            }

            int temp = array[minIndex];
            array[minIndex] = array[i];
            array[i] = temp;
        }
    }
}

```

OUTPUT

```

Enter the number of elements in the array: 5
Enter the elements of the array:
6 7 8 9 6
Sorted array:
6 6 7 8 9
=== Code Execution Successful ===

```

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


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

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

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();

        int[] array = new int[n];

        System.out.println("Enter the elements of the array (sorted order):");
        for (int i = 0; i < n; i++) {
            array[i] = scanner.nextInt();
        }

        System.out.print("Enter the element to search for: ");
        int key = scanner.nextInt();

        int result = binarySearch(array, key);

        if (result == -1) {
            System.out.println("Element not found in the array.");
        } else {
            System.out.println("Element found at index: " + result);
        }

        scanner.close();
    }

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

        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (array[mid] == key) {
                return mid; // Element found, return index
            }
        }
    }
}
```

```

        if (array[mid] < key) {
            left = mid + 1;
        }
        else {
            right = mid - 1;
        }
    }

    return -1;
}
}

```

OUTPUT

```

Enter the number of elements in the array: 6
Enter the elements of the array (sorted order):
6
6
6
6

```

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

```

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

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

        String regex = "\\b[a-z]+_[a-z]+\\b";

        Pattern pattern = Pattern.compile(regex);

        System.out.println("Enter a sentence:");
        String input = scanner.nextLine();

        Matcher matcher = pattern.matcher(input);

        System.out.println("Sequences matching the pattern:");
    }
}

```

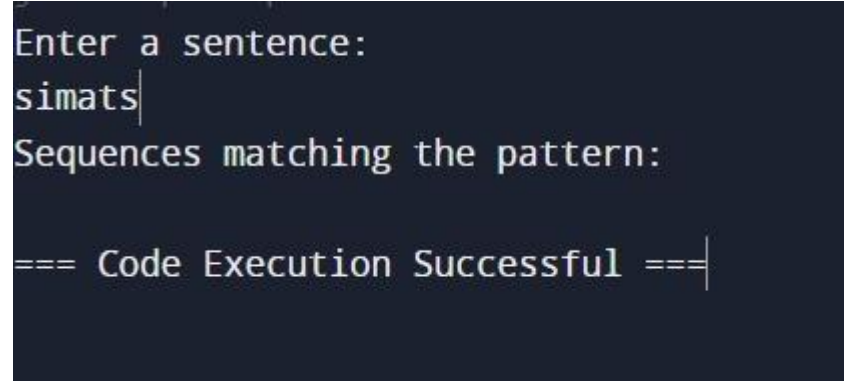
```

        while (matcher.find()) {
            System.out.println(matcher.group());
        }

        scanner.close();
    }
}

```

OUTPUT



```

Enter a sentence:
simats|
Sequences matching the pattern:

=== Code Execution Successful ===|

```

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

```

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

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

        String regex = "\\b[a-fh-zA-FH-Z]g[a-zA-Z]\\b";

        Pattern pattern = Pattern.compile(regex);

        System.out.println("Enter a sentence:");
        String input = scanner.nextLine();

        Matcher matcher = pattern.matcher(input);

        System.out.println("Words matching the pattern:");
        while (matcher.find()) {
            System.out.println(matcher.group());
        }

        scanner.close();
    }
}

```

OUTPUT

Enter a sentence:

saveetha

Words matching the pattern:

=== Code Execution Successful ===