Day-5: String, Arrays and Command Line Argument

- String
- Arrays
- Command Line Arguments
- Variable Arguments

String

- It is the most extensively class in Java.
- It can store maximum of 2 billion Unicode characters (256 KB)

Constructors

String()
String(String)
String(char c[],int startIndex, int num)
String(byte b[],int startIndex, int num)

```
String s1 = new String();
String s2 = new String("Welcome");
char c[] = {'W','e','l','c','o','m','e'};
String s3 = new String(c,3,4);                    //come
byte b[]={65,66,67,68,69,70};
String s4 = new String(b,0,5);                    //ABCDE

String s5 = new String("Welcome");
String s6 = "Welcome";
```

Methods

int indexOf(String str)
int indexOf(String str,int startIndex)
int lastIndexOf(String str)
int lastIndexOf(String str,int startIndex)
char charAt(int index)

W	e	l	c	o	m	e
0	1	2	3	4	5	6

```
String str = new String("Welcome");
int a = str.indexOf("e");                         //1
int b = str.indexOf("e",a+1);                     //6
int c = str.lastIndexOf("e");                     //6
int d = str.lastIndexOf("e",c-1);                 //1
int e = str.indexOf("me");                        //5
int f = str.indexOf("zee");                       //-1
char g = str.charAt(3);                           //c
char h = “Hello”.charAt(4);                       //o
```

String toUpperCase()
String toLowerCase()
boolean equals(String)
boolean equalsIgnoreCase(String)

```
boolean startsWith(String)
boolean endsWith(String)

String s1 = new String("Hello World");
String s2 = s1.toUpperCase();                    //s1 = Hello World , s2 = HELLO WORLD
boolean b = s1.equals(s2);                       //false
boolean b1 = s1.equalsIgnoreCase(s2);            //true
boolean b2 = s1.startsWith("Hell");              //true
boolean b3 = s2.endsWith("ing");                 //false
String trim()
int length()
String substring(int stindex)
String substring(int stindex,int endindex)

String s1 = new String("  Hello   World  ");
int a = s1.length();                             //16
String  s2 = s1.trim();
int b = s2.length();                             //12
String s3 = s2.substring(0,5);                   //Hello
String s4 = s2.substring(7);                     //World

String[] split(String separator)
String concat(String str)
String replace(char original, char replacement)

String str = new String("This is a new world");
String s[] = str.split(" ");

String s1 = "one";
String s2 = s1.concat("two");                    //onetwo
String s = "Hello".replace('l', 'w');            //Hewwo
```

Write a program that accepts an RGB value from user and print parts of red, green and blue.

```
import java.util.Scanner;
class Rgb {
        public static void main(String args[]) {
                String str = new String();
                Scanner a = new Scanner(System.in);
                System.out.println("Enter Rgb Value:");
                str = a.nextLine();
                String s[] = str.split(",");
                System.out.println("Red is : " + s[0]);
                System.out.println("Green is : " + s[1]);
                System.out.println("Blue is : " + s[2]);
        }
}
```

Write a program that accepts a string. The string should have at least 5 characters and should not contain any special characters or digits.

```
import java.util.Scanner;
class TestString {
	public static void main(String args[]) {
		String str = new String();
		Scanner in = new Scanner(System.in);
		main:
		while(true) {
			System.out.print("Enter String: ");
			str = in.nextLine();
			if(str.length() < 5) {
				System.out.println("Error !! Atleast 5 character is required.");
				continue;
			}
			for(int i = 0 ; i < str.length(); i++) {
				char c = str.charAt(i);
				if ( !(c >= 'A' && c <= 'Z'  || c >= 'a' && c <= 'z')) {
					System.out.println("Error !! Special characters or digits found. ");
					continue main;
				}
			}
			break;
		}
	}
}
```

StringBuffer and StringBuilder

- It is mutable (Operation is performed on the same object)
- StringBuilder provides similar functionality like StringBuffer, but the methods are faster as these methods are non-synchronized.

Methods of StringBuffer/StringBuilder Class

append(String s)
reverse()
charAt(int index)
setCharAt(int index, char c)
insert(int index, String str)
deleteCharAt(int index)
delete(int index, int endindex)
String substring(int startIndex)
String substring(int startIndex, int endIndex*) * endindex is exclusive
int length( )
int capacity( )

Array

Collection of similar data types that shares common name and memory.

Terms:

- Each data item in an array is called "element"
- The counting of elements is called "index" or "subscript"

- The total number of elements is called "length" or "size"
- Index always starts from 0.

Declaration and Initialization

<Data Type> <arrayVariable>[] = new <Data Type>[size];

Example:

```
int num[] = new int[10];
float f[] = new float[5];
```

Declaration

```
int[] num, a;        int num[];        int []num;
float[] f;           float f[];        float []f;
```

Initialization

```
num = new int[10];
f = new float[5];
```

The length property

This property returns length of an array

```
num.length          // 10;
f.length            // 5;
```

```
int num[][] = new int[4][5];
String wdays[]={ "Sunday", "Monday", "Saturday" };
int n[] = {1, 2, 3, 4, 5};
int num[][]={
        {1,2,3,4,5},
        {1,2,3,4,5},
        {1,2,3,4,5}
        };
```

<u>Arrays Class</u>

Java's Arrays class, part of the java.util package, provides various static utility methods for array operations. Below is a basic listing of commonly used functions from the Arrays class, along with examples:

1. Sorting an Array

- **Method:** Arrays.sort(array)
- **Example:**

```
import java.util.Arrays;

public class ArraysSortExample {
   public static void main(String[] args) {
      int[] numbers = {5, 3, 8, 1, 9};
      Arrays.sort(numbers);
      System.out.println(Arrays.toString(numbers)); // [1, 3, 5, 8, 9]
   }
}
```

2. Binary Search

- **Method:** Arrays.binarySearch(array, key)
- **Example:** *(Array must be sorted)*

```
import java.util.Arrays;

public class ArraysBinarySearchExample {
   public static void main(String[] args) {
      int[] numbers = {1, 3, 5, 7, 9};
      int index = Arrays.binarySearch(numbers, 5);
      System.out.println("Index of 5: " + index); // Output: 2
   }
}
```

3. Filling an Array with a Specific Value

- **Method:** Arrays.fill(array, value)
- **Example:**

```
import java.util.Arrays;

public class ArraysFillExample {
   public static void main(String[] args) {
      int[] arr = new int[5];
      Arrays.fill(arr, 7);
      System.out.println(Arrays.toString(arr)); // [7, 7, 7, 7, 7]
   }
}
```

4. Copying an Array

- **Method:** Arrays.copyOf(originalArray, newLength)
- **Example:**

```
import java.util.Arrays;

public class ArraysCopyExample {
   public static void main(String[] args) {
      int[] original = {1, 2, 3};
      int[] copy = Arrays.copyOf(original, 5);
      System.out.println(Arrays.toString(copy)); // [1, 2, 3, 0, 0]
   }
}
```

5. Copying a Range of an Array

- **Method:** Arrays.copyOfRange(originalArray, fromIndex, toIndex)
- **Example:**

```
import java.util.Arrays;

public class ArraysCopyRangeExample {
   public static void main(String[] args) {
      int[] numbers = {10, 20, 30, 40, 50};
      int[] subArray = Arrays.copyOfRange(numbers, 1, 4);
      System.out.println(Arrays.toString(subArray)); // [20, 30, 40]
   }
```

```
}
```

6. Comparing Two Arrays (Equality)

- **Method:** Arrays.equals(array1, array2)
- **Example:**

```
import java.util.Arrays;

public class ArraysEqualsExample {
   public static void main(String[] args) {
      int[] arr1 = {1, 2, 3};
      int[] arr2 = {1, 2, 3};
      boolean isEqual = Arrays.equals(arr1, arr2);
      System.out.println("Are arrays equal? " + isEqual); // true
   }
}
```

7. Converting an Array to String

- **Method:** Arrays.toString(array)
- **Example:**

```
import java.util.Arrays;

public class ArraysToStringExample {
   public static void main(String[] args) {
      int[] numbers = {1, 2, 3, 4, 5};
      System.out.println(Arrays.toString(numbers)); // [1, 2, 3, 4, 5]
   }
}
```

8. Sorting in Reverse Order (For Integer Wrapper Class)

- **Method:** Arrays.sort(array, Collections.reverseOrder())
- **Example:**

```
import java.util.Arrays;
import java.util.Collections;

public class ArraysReverseSortExample {
   public static void main(String[] args) {
      Integer[] numbers = {5, 2, 8, 1, 3};
      Arrays.sort(numbers, Collections.reverseOrder());
      System.out.println(Arrays.toString(numbers)); // [8, 5, 3, 2, 1]
   }
}
```

Write a program that accepts 10 numbers in an array and display their sum and average.

```
import java.util.Scanner;
class Array1 {
        public static void main(String args[]) {
                int num[] = new int[10];
                int i, sum = 0, avg = 0;
                Scanner in = new Scanner(System.in);
                for(i = 0 ; i < num.length; i ++) {
```

```
                System.out.print("Enter Number " + (i + 1 ) + " : ");
                num[i] = in.nextInt();
                sum += num[i];
            }
            avg = sum / num.length;
            System.out.println("Sum : " + sum);
            System.out.println("Average : " + avg);
        }
}
```

Write a program that initializes an array by multiples of 5 from 100 to 200 and display them.

```
class Array2 {
    public static void main(String[] args) {
        int num[], j, i;
        for(i = 100 , j = 1; i< 200; i+=5, j++);
        num  = new int[j];
        for(i = 100 , j = 0; i< 200 ;i+=5,j++)
            num[j]=i;
        for(j=0;j<num.length;j++)
            System.out.println(num[j]);
    }
}
```

Write a program that creates matrix of size 4 by 4. Accepts all integer elements and print sum of all rows, all columns and entire matrix.

```
import java.util.Scanner;
class Array3 {
    public static void main(String args[]) {
        int num[][] = new int[4][4];
        int rowSum[] = new int[num.length];
        int columnSum[] = new int[num[0].length];
        int i, j, sum = 0;
        Scanner in = new Scanner(System.in);
        for(i = 0 ; i < num.length; i ++) {
            for(j = 0 ; j < num[i].length; j ++) {
                System.out.print("Enter Number ["+ i + "," + j + "] : ");
                num[i][j] = in.nextInt();
                sum += num[i][j];
                rowSum[i] += num[i][j];
                columnSum[j] += num[i][j];
            }
        }
        for(i = 0 ; i < num.length; i ++) {
            for(j = 0 ; j < num[i].length; j ++)
                System.out.print("\t" + num[i][j]);
            System.out.println("\t" + rowSum[i]);
        }
        for(j = 0 ; j < num[0].length; j ++)
            System.out.print("\t" + columnSum[j]);
```

```
                System.out.print("\t" + sum);
        }
}
```

Command Line Arguments

- These are details that are supplied to program at the time of execution.
- These details gets stored inside String args[]

Example:

```
        java Program One Two Three
        args[0]=One
        args[1]=Two
        args[2]=Three
        args.length=3
```

Objective: It is used to take short input during execution

Write a program to displays all the command line arguments

```
class Command1 {
        public static void main(String args[]) {
        for(int i = 0 ; i < args.length ; i++)
                System.out.println(args[i]);
        }
}
```

Write a program to accept a number as command line argument and display its table

```
class Table {
        public static void main(String args[]) {
                int num=0;
                if(args.length!=1) {
                        System.out.println("Invalid Arguments");
                        System.exit(0);
                }
                num = Integer.parseInt(args[0]);
                for(int i = 1 ; i <= 10; i++)
                        System.out.println(num + " * " + i + " = " + num*i);
        }
}
```

```
java Table 5            // args.length = 1
java Table              // args.length = 0
java Table   5 6 7      // args.length = 3
```

Variable Arguments

- It is a feature of Java 5
- It says number of arguments in a function can vary.
  public static int sum(int a, int b)
  public static int sum(int a, int b, int c)
  public static int sum(int a, int b, int c, int d)

- Syntax:
  <DataType>... <variableName>;
  Example:
  int... num;
- Internally variable argument is treated as an array (superset)
- Rules:
  - There can be only 1 variable argument in a function.
  - This should always be the last argument in the argument list.

Example

```
class VarArgs {
	public static int sum(int... n) {
		int sum = 0;
		for(int i = 0 ; i < n.length ; i++)
			sum += n[i];
		return sum;
	}
	public static void main(String... args) {
		System.out.println(sum(10, 20, 30));
		System.out.println(sum());
		System.out.println(sum(10, 20, 30, 40, 50));
		int num[] = {9, 9, 9};
		System.out.println(sum(num));
	}
}
```

Assignment

Theory Questions

1. What is the range of String class? Write different ways to create an object of String class. Explaing why it is the most extensively used class in Java.
2. What is an array? How does arrays in Java differ from arrays in C++?
3. Why do we need command line arguments and how does it differ from input using Scanner?
4. What are variable arguments? Can variable argument be used as a substitute to function overload.

Practical Questions

1. Write a program that initializes a String with “Betty got a bit of bitter better butter” and perform the following operations:
   a. Display all the occurrences (index) of “b” and “t” in the String.
   b. Display each word along with its length in a new line.
   c. Display all the words that starts with “b” is the String.
2. Write a program that accepts a String from user and perform the following checks:
   a. The string should have atleast 3 characters.
   b. The string should not have any special characters or digits except alphabets.
   If any condition is not satisfied appropriate error message should be displayed.
3. Write a program that accepts a String from user and perform the following checks:
   a. The string should have atleast 5 characters.
   b. The string should not have any special characters or alphabets except digits.
   If any condition is not satisfied appropriate error message should be displayed.
4. Write a program that accepts a String from user and perform the following checks:
   a. The string should have atleast 6 characters.

b. The string should not have any special characters except @.
   If any condition is not satisfied appropriate error message should be displayed.

5. Write a program that accepts 10 names in an array and display all names that starts with “A”.
6. Write a program that accepts 10 names in an array and display all names that ends with “h”.
7. Write a program that accepts 10 names in an array and display all names that contains “he”.
8. Write a program that accepts 10 numbers in an array and display then in ascending order.
9. Write a program that accepts numbers in 2D array of 4 by 5 and display the sum of rows, columns and the entire matrix.
10. Write a program that accepts a number as command line argument and display its square. In case of invalid arguments proper error message should be displayed.
11. Write a program that accepts 5 numbers as command line argument and display their sum and average, also display these numbers in ascending order.
12. Create a matrix that contains sales of 4 products in 5 regions and perform the following operations:
    a. Total sales made by the company region-wise.
    b. Total sales made by the company product-wise.
    c. Highest and the lowest selling products.
    d. Total sales made by the company.
13. Accept numbers in 2 matrices of 4 by 4 and display the product of the matrices.
14. Write a Java program that: (**Sorting and Searching**)
    a. Takes an integer array as input from the user.
    b. Sorts the array in ascending order using Arrays.sort().
    c. Asks the user to enter a number to search for.
    d. Uses Arrays.binarySearch() to find the index of the number and displays the result.
15. Write a Java program that: (**Copying and Filling**)
    a. Creates an array of size 10 and fills it with the number 5 using Arrays.fill().
    b. Copies the first 5 elements of the filled array into a new array using Arrays.copyOf().
    c. Prints both arrays using Arrays.toString().
16. Write a Java program that: (**Comparing Arrays**)
    a. Takes two integer arrays as input from the user.
    b. Compares them using Arrays.equals().
    c. Prints whether they are equal or not.