Kamran Gul | 023-25-0161 | Section C

LAB–2: METHODS IN JAVA

Objectives

1. Understand the purpose and importance of methods in Java
2. Define and call different types of methods
3. Use methods with parameters and return values
4. Apply control statements including if-else, switch, and ternary inside methods
5. Write methods that work with Strings and Arrays
6. Use built-in Math class functions through methods
7. Understand and use command-line arguments
8. Organize logic into reusable and readable code

1. Methods

A method is a block of code that performs a specific task. Methods help reduce code repetition, improve readability, and make programs modular and manageable. In this lab, all methods are declared static so they can be called directly from the main method without creating objects.

Method Syntax

```
static returnType methodName(parameters) {
   // method body
}
```

2. Types of Methods

2.1 Method with No Parameters and No Return Value

This type of method is used when a task only performs an action such as displaying a message and does not need input or output.

```
static void greet() {
   System.out.println("Welcome to Java Methods Lab");
}
```

2.2 Method with Parameters and No Return Value

These methods accept input values as parameters and perform operations without returning a result.

```
static void printSquare(int num) {
   System.out.println("Square is: " + (num * num));
}
```

2.3 Method with Return Value and No Parameters

This type of method performs a calculation and returns a value without taking any input parameters.

```
static int getFixedNumber() {
   return 10;
}
```

2.4 Method with Parameters and Return Value

This is the most commonly used method type. It takes input, processes it, and returns a result.

```
static int add(int a, int b) {
   return a + b;
}
```

3. Methods with Control Statements

3.1 If–Else Inside a Method

If–else statements allow decision-making inside methods based on conditions.

```
static String checkResult(int marks) {
   if (marks >= 40)
      return "Pass";
   else
      return "Fail";
}
```

3.2 Ternary Operator in Methods

The ternary operator is a short form of if–else used for simple conditional expressions.

```
static int findMax(int a, int b) {
   return (a > b) ? a : b;
}
```

3.3 Switch Statement in Methods

The switch statement is used when multiple conditions are based on a single variable value.

```
static String dayName(int day) {
  switch (day) {
    case 1: return "Monday";
    case 2: return "Tuesday";
    case 3: return "Wednesday";
    case 4: return "Thursday";
    case 5: return "Friday";
    default: return "Invalid Day";
  }
}
```

4. Methods with String Basics

4.1 String Length

The length() method returns the number of characters present in a string.

```
static int getLength(String text) {
  return text.length();
}
```

4.2 String Comparison

Strings should be compared using equals() instead of == to compare actual content.

```
static boolean checkName(String name) {
  return name.equals("Ali");
}
```

4.3 String Case Conversion

```
static String convertToUpper(String word) {
  return word.toUpperCase();
}
```

5. Methods with Arrays

5.1 Sum of Array Elements

Arrays can be passed to methods to perform calculations on multiple values.

```
static int sumArray(int[] arr) {
  int sum = 0;
  for (int i = 0; i < arr.length; i++) {
    sum += arr[i];
  }
  return sum;
}
```

5.2 Find Maximum Element in Array

```
static int findMax(int[] arr) {
  int max = arr[0];
  for (int i = 1; i < arr.length; i++) {
    if (arr[i] > max)
      max = arr[i];
  }
  return max;
}
```

6. Methods Using Math Class

The Math class provides built-in static methods for common mathematical operations. These methods can be used directly without creating objects.

6.1 Power Function

```
static int calculatePower(int a, int b) {
  return (int) Math.pow(a, b);
}
```

6.2 Absolute Value

```
static int getAbsolute(int x) {
  return Math.abs(x);
}
```

7. Command-Line Arguments

Command-line arguments allow data to be passed to the program at the time of execution. They are received as an array of strings in the main method.

Example: Using Command-Line Arguments

```
class CommandLineExample {
  public static void main(String[] args) {
    int a = Integer.parseInt(args[0]);
    int b = Integer.parseInt(args[1]);
    System.out.println("Sum: " + (a + b));
  }
}
```

8. Complete Example Program

```
class MethodLab2 {

  static int square(int x) {
    return x * x;
  }

  static boolean isPositive(int x) {
    return x > 0;
  }

  static String checkDay(int d) {
    return dayName(d);
  }

  static String dayName(int day) {
    switch (day) {
      case 1: return "Monday";
      case 2: return "Tuesday";
      default: return "Invalid";
    }
  }

  public static void main(String[] args) {
    int n = 5;
    System.out.println("Square: " + square(n));
    System.out.println("Is Positive: " + isPositive(n));
```

```
        System.out.println("Day: " + checkDay(1));
    }
}
```

Lab Guidelines

• All methods must be static
• Write logic inside methods, not directly in main
• Use meaningful method names
• Follow proper indentation and formatting

Outcome

After completing this lab, students will be able to design modular Java programs using methods and apply control logic, strings, arrays, math operations, and command-line arguments effectively.

Kamran Gul | 023-25-0161 | Section C

Exercises

Instructions:
- Solve all tasks using methods only.
- main() should only call methods and print results.
- All methods must be static.
- Focus on logic and clean implementation.

Task 0: Syntax Errors

Broken Code	Find the Error	Fix It
static int add(int a int b){ return a+b; }	Comma is missing in parameters	static int add(int a, int b){ return a+b; }
static void show(){ System.out.println("Hi") }	Terminator is missing	static void show(){ System.out.println("Hi"); }
static int square(int x){ System.out.println(x*x); }	No return value	static int square(int x){ return (x*x); }
switch(x){ case 1: System.out.println("One") break; }	Missing Terminator	switch(x){ case 1: System.out.println("One"); break; }
int[] arr = new int[5]	Missing terminator	int[] arr = new int[5];
public static void main(String args){ }	Missing square brackets	public static void main(String[] args){ }
static int max(int[] arr){ return arr.length }	Missing terminator	static int max(int[] arr){ return arr.length; }

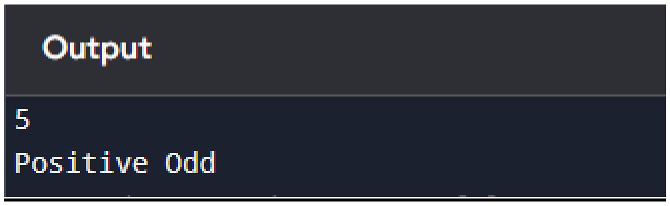
Task 1: Smart Number Analyzer

Method Type:
Return: String
Parameters: int n

Return:
- "Positive Even" if number is positive and even
- "Positive Odd" if positive and odd
- "Negative" if number is negative
- "Zero" if number is 0

Solution:

```
import java.util.Scanner;
class first{
  public static String smartNumberAnalyzer(int num){
    if(num==0){
      return ("Zero");
    }
    else if(num>0){
      if(num%2==0){
        return ("Positive Even"); }
      else{
        return ("Positive Odd");
      } }
    else{
      return ("Negative");}}
  public static void main(String[] args){
    Scanner input = new Scanner(System.in);
    int num = input.nextInt();
    String output = smartNumberAnalyzer(num);
    System.out.print(output); } }
```



```
5
Positive Odd
```

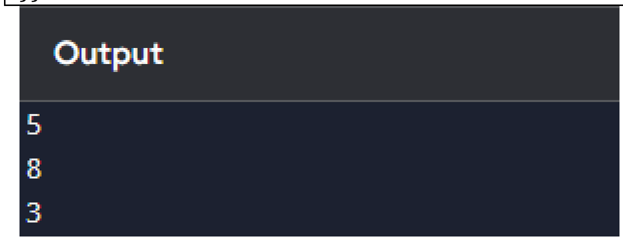
Task 2: Conditional Calculator

Method Type:
Return: int
Parameters: int a, int b

Rules:
- If a > b, return a – b
- If a < b, return b – a
- If equal, return 0

Solution:

```
import java.util.Scanner;

class second{

static int conditionalCalculator(int a, int b){
		if(a>b){
			return a-b;
		}
		else if(a==b){
			return 0;
		}
		else{
			return b-a;
		}
	}

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

	int a = in.nextInt();
	int b = in.nextInt();

	int result = conditionalCalculator(a,b);

	System.out.print(result);
}}
```



```
5
8
3
```

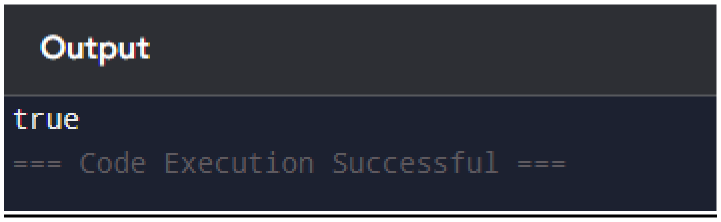
Task 3: Array Balance Checker

Method Type:
Return: boolean
Parameters: int[] arr

Return true if sum of even elements equals sum of odd elements.

Solution:

```
class third{

        static boolean arrayBalanceChecker(int[] arr){
                int evenSum =0;
                int oddSum=0;
                for(int i=0; i<5;i++){
                        if(arr[i] % 2 == 0){
                                evenSum = evenSum + arr[i];
                        }
                        else{
                                oddSum = oddSum + arr[i];
                        }
                }

                if(evenSum == oddSum){
                        return true;
                }
                else return false;
        }

        public static void main(String[] args){
                int[] arr = {5,2,4,4,5};
                boolean result = arrayBalanceChecker(arr);
                System.out.print(result);
        }}
```



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

Task 4: Switch-Driven Grading System

Method Type:
Return: String
Parameters: int marks

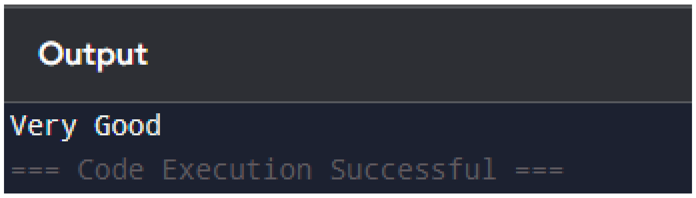
Use switch on (marks / 10):
• 10 → Distinction
• 9 → Excellent

• 8 → Very Good
• Others → Fail

Note: marks are in range (0-100).

Solution:

```
class fourth{

        static String gradingSystem(int marks){
        String result="";
                        switch(marks){
                        case 10:
                                result = "Distinction";
                                break;
                        case 9:
                                result = "Excellent";
                                break;

                        case 8:
                                result = "Very Good";
                                break;

                        default:
                                result = "Fail";
                                break;
                        }
                return result;
                }
        public static void main(String[] args){
                String a = gradingSystem(8);
                System.out.print(a);
        }}
```



```
Very Good
=== Code Execution Successful ===
```

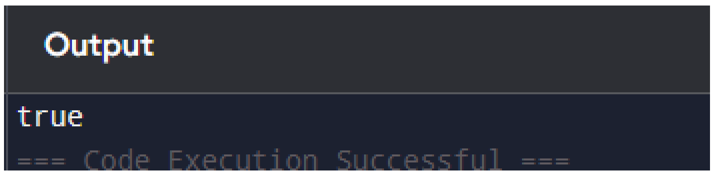
Task 5: String Pattern Validator

Method Type:
Return: boolean
Parameters: String s

Return true if:
• Length ≥ 6
• First character uppercase
• Last character is a digit

Solution:

```
class fifth{

        static boolean patternValidator(String str){
                        if(
                                (str.length() >=6 ) &&
                                (((int)str.charAt(0) >=65) && ((int)str.charAt(0) < 96 ))
                                &&
                                ((int)str.charAt(str.length()-1) <58)
                        )
                        {
                                return true;
                        }

                        else
                                return false;
        }


        public static void main(String[] args){
                String str = "HelloWorld9";

                boolean res = patternValidator(str);
                System.out.print(res);
        }
}
```



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

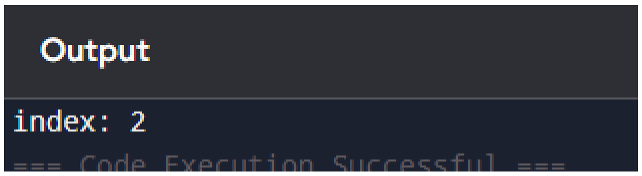
Task 6: Array Peak Finder

Method Type:
Return: int
Parameters: int[] arr

Return index of the largest element.
If multiple max values exist, return first index.

Solution:

```
class sixth{
	static int arrayPeakFinder(int[] arr){
		int largest = arr[0];
		int ind = 0;

		for(int i=1;i<5;i++){
			if(arr[i]>largest){
				largest = arr[i];
				ind = i;
			}
		}
		return ind;
	}


	public static void main(String[] args){
		int[] array = {0,9,15,4,8};
		int result = arrayPeakFinder(array);
		System.out.print("index: " + result);
	}
}
```



```
index: 2
=== Code Execution Successful ===
```

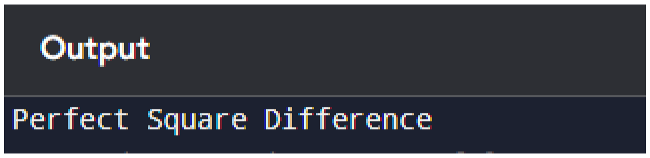
Task 7: Math-Based Decision Maker

Method Type:
Return: String
Parameters: int a, int b

Return "Perfect Square Difference" if |a − b| is a perfect square.
Otherwise return "Not Perfect Square".

Solution:

```
class seventh{
        static String decisionMaker(int a, int b){
                int res = Math.abs(a-b);

                double r = Math.sqrt(res);
                if(r*r == res){
                        return "Perfect Square Difference";
                }
                else
                return "Not Perfect Square";
        }

        public static void main(String[] args){

                String result = decisionMaker(25,9);
                System.out.print(result);

        }
}
```


Output

```
Perfect Square Difference
```

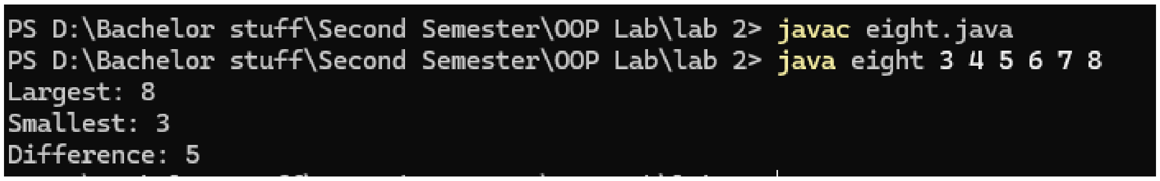
Task 8: Command-Line Validator

Method Type:
Return: void
Parameters: String[] args

Accept 3 numbers via command-line and print:
• Largest number
• Smallest number
• Difference between them

Solution:

```
class eight{
	public static void main(String[] args){
		int largest = Integer.parseInt(args[0]);
		int smallest = Integer.parseInt(args[0]);

		for(int i=1; i< args.length; i++){
		if(Integer.parseInt(args[i]) > largest){
			largest = Integer.parseInt(args[i]);
		}

		if(Integer.parseInt(args[i]) < smallest){
			smallest = Integer.parseInt(args[i]);
		}
	}

	System.out.println("Largest: " + largest);
	System.out.println("Smallest: " + smallest);
	System.out.print("Difference: " + (largest-smallest));
}
}
```



```
PS D:\Bachelor stuff\Second Semester\OOP Lab\lab 2> javac eight.java
PS D:\Bachelor stuff\Second Semester\OOP Lab\lab 2> java eight 3 4 5 6 7 8
Largest: 8
Smallest: 3
Difference: 5
```

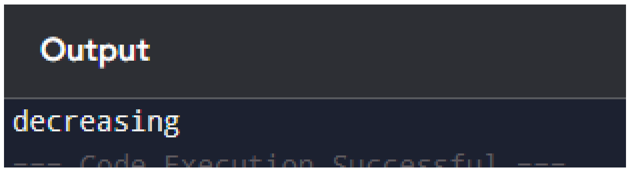
Task 9: Array Trend Detector

Method Type:
Return: String
Parameters: int[] arr

Return:
• "Increasing" if strictly increasing order in numbers
• "Decreasing" if strictly decreasing order in numbers
• "Mixed" otherwise

Solution:

```
class nine{
        static String arrayTrendDetector(int[] arr){
                boolean inc = true;
                boolean dec = true;
                for(int i=1; i<arr.length; i++){
                        if(arr[i] <= arr[i-1]){
                                inc = false;
                        }
                        if(arr[i] >= arr[i-1]){
                                dec = false;
                        }

                }

                if(inc){
                        return "increasing";
                }
                else if(dec){
                        return "decreasing";
                }
                else{
                        return "mixed";
                }
        }
        public static void main(String[] args){
                int[] array = {5,4,3,2,1};
                String result = arrayTrendDetector(array);
                System.out.print(result);
        }
}
```



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

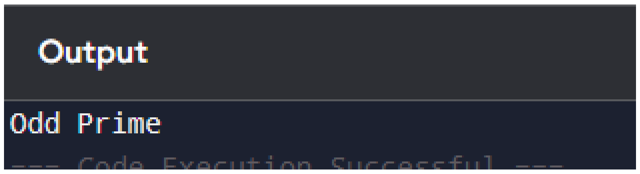
Task 10: Mini Decision Engine

Method Type:
Return: String
Parameters: int n

Return:
• "Prime Even" if n = 2
• "Prime Odd" if prime and odd
• "Composite" if not prime
• "Invalid" if n ≤ 1

Solution:

```
class ten{
        static String miniDecisionEngine(int num){
                int res=0;
                String r="";
                if(num <=1){
                        res = 1;
                }
                else if(num == 2){
                        res = 2;
                }
                else{
                        for(int i=2;i<num;i++){
                                if(num%i==0){
                                        res = 3;
                                        break;
                                }
                                else{
                                        res = 4;
                                }
                        }
                }
                switch(res){
                case 1:
                        r = "Invalid";
                        break;
                case 2:
                        r = "Even Prime";
                        break;
                case 3:
                        r = "Composite";
                        break;
                case 4:
                        r = "Odd Prime";
                        break;
                }
                return r;
        }
        public static void main(String[] args){
                String result = miniDecisionEngine(4);

                System.out.print(result);
}}
```



Output

```
Odd Prime
=== Code Execution Successful ===
```

Kamran Gul | 023-25-0161 | Section C

Task 11: Debugging Practice

Broken Code	Find the Error	Fix It
int[] a = new int[5]; a[5] = 10;	Array out of bound	int[] a = new int[6]; a[5] = 10;
int sum; for(int i=0;i<arr.length;i++) sum+=arr[i];	sum is not initialized	int sum = 0; for(int i=0;i<arr.length;i++) sum+=arr[i];
static int max(int[] arr){ int m=0; ... }		
switch(x){ case 1: System.out.println("One"); case 2: ... }	No break statement after case 1	switch(x){ case 1: System.out.println("One");break; case 2: ... }
Integer.parseInt(args[0]);	Args is string not array, it must use small brackets	Integer.parseInt(args(0));
static void calc(int a,int b){ return a+b; }	Void method cant return anything	static int calc(int a,int b){ return a+b; }
return arr[arr.length];		return arr.length;
static String day(int d){ switch(d){ case 1: return "Mon"; case 2: return "Tue"; default: "Invalid"; } }	No return in default	static String day(int d){ switch(d){ case 1: return "Mon"; case 2: return "Tue"; default: return "Invalid"; } }
static int add(int a,int b){ System.out.println(a+b); }	Method is of int type and cannot directly print anything directly	static void add(int a,int b){ System.out.println(a+b); }