Module 1: Java Basics

Premanand S

Assistant Professor
School of Electronics Engineering (SENSE)
Vellore Institute of Technology
Chennai Campus

premanand.s@vit.ac.in

January 13, 2025

Topics covered in Module 1,

- OOP Paradigm
- Features of JAVA Language
- JVM
- Bytecode
- Java Program Structure
- Basic Programming Construct
- Data Types
- Variables
- Java naming conventions
- Operators

Relational (Comparison) Operators

Description: Relational operators are used to compare two values. These operators return a boolean value: `true` or `false`. The following are the relational operators in Java:

Operator	Description	Example
==	Equal to	`a == b`
!=	Not equal to	`a != b`
>	Greater than	`a > b`
<	Less than	`a < b`
>=	Greater than or equal to	`a >= b`
<=	Less than or equal to	`a <= b`

== : Equal to

The == operator compares if two values are equal. It returns `true` if the values on both sides are the same, and `false` if they are different.

`a == b` returns true if a is equal to b

Example:

$5 == 5$ returns `true`

!= : Not equal to

The `!=` operator compares if two values are not equal. It returns `true` if the values are different, and `false` if they are the same.

`a != b` returns true if a is not equal to b

Example:

$5! = 3$ returns `true`

> : Greater than

The > operator checks if the value on the left is greater than the value on the right. It returns `true` if the left operand is greater, and `false` otherwise.

`a > b` returns true if a is greater than b

Example:

$10 > 5$ returns `true`

< : Less than

The < operator checks if the value on the left is less than the value on the right. It returns `true` if the left operand is less, and `false` otherwise.

`a < b` returns true if a is less than b

Example:

$5 < 10$ returns `true`

>= : Greater than or equal to

The >= operator checks if the left operand is greater than or equal to the right operand. It returns `true` if the left value is greater than or equal, and `false` if it is less.

`a >= b` returns true if *a* is greater than or equal to *b*

Example:

$10 >= 10$ returns `true`

<= : Less than or equal to

The <= operator checks if the left operand is less than or equal to the right operand. It returns `true` if the left value is less than or equal, and `false` if it is greater.

`a <= b` returns true if a is less than or equal to b

Example:

$3 <= 7$ returns `true`

Examples

Example (Java Code)

```
public class Main {
    public static void main(String[] args) {
        int a = 10, b = 5;

        // Using relational operators
        System.out.println("a == b: " + (a == b));  // false
        System.out.println("a != b: " + (a != b));  // true
        System.out.println("a > b: " + (a > b));    // true
        System.out.println("a < b: " + (a < b));    // false
        System.out.println("a >= b: " + (a >= b));  // true
        System.out.println("a <= b: " + (a <= b));  // false
    }
}
```

Relational Operators Practice Questions

Practice Questions:

1. **Basic Relational Comparisons:** Write a Java program to compare two integers and check:
   - If they are equal.
   - If one is greater than the other.
   - If one is less than the other.
2. **Largest of Two Numbers:** Write a program that takes two numbers as input and prints which number is larger or if they are equal.
3. **Pass or Fail:** Write a program to check if a student has passed an exam. The program should take the marks as input and print "Pass" if the marks are greater than or equal to 40, otherwise print "Fail."
4. **Age Check:** Write a Java program that takes a person's age as input and checks if the person is eligible to vote (age ≥ 18).
5. **Number Range Check:** Write a program to check if a given number lies between 1 and 100 (inclusive).

Relational Operators Practice Questions

Practice Questions:

1. **Temperature Comparison:** Write a program that takes two temperatures (in degrees Celsius) as input and determines:
   - If the first temperature is hotter than the second.
   - If the temperatures are equal.
   - If the first temperature is cooler than the second.
2. **Triangle Validity Check:** Write a program to check if three sides can form a valid triangle. A triangle is valid if:

   sum of any two sides is greater than the third side.

3. **Discount Eligibility:** Write a program to check if a customer is eligible for a discount. The customer is eligible if their total purchase amount is greater than or equal to $100.
4. **Logical Use of Relational Operators:** Write a program to check if a number is divisible by 5 and lies between 50 and 100.

Relational Operators Practice Questions

Practice Questions:

1. **Relational Operators with Strings:** Java doesn't support == for comparing the content of strings directly. Write a program to demonstrate this and use `.equals()` to compare two strings.
2. **Challenge Problem:** Write a program that takes three numbers as input and finds the largest among them using relational operators.

Compare Two Integers

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int a = 10, b = 20;

        System.out.println("a == b: " + (a == b));
        System.out.println("a > b: " + (a > b));
        System.out.println("a < b: " + (a < b));
    }
}
```

Largest of two numbers

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int x = 30, y = 20;

        if (x > y) {
            System.out.println("x is larger.");
        } else if (y > x) {
            System.out.println("y is larger.");
        } else {
            System.out.println("Both are equal.");
        }
    }
}
```

Pass or Fail

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int marks = 45;

        System.out.println(marks >= 40 ? "Pass" : "Fail");
    }
}
```

Age Check for Voting

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int age = 19;

        System.out.println(age >= 18 ? "Eligible to vote" :
        "Not eligible to vote");
    }
}
```

Number Range Check

Example (Java)

```
public class Main{
    public static void main(String[] args) {
        int number = 50;

        System.out.println((number >= 1 && number <= 100)
        ? "Number is within range" :
        "Number is out of range");
    }
}
```

Temperature Comparison

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        double temp1 = 30.5, temp2 = 28.2;

        System.out.println(temp1 > temp2 ?
        "First temperature is hotter" :
                           temp1 < temp2 ?
                           "First temperature is cooler" :
                           "Both temperatures are equal");
    }
}
```

Triangle Validity Check

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int a = 3, b = 4, c = 5;

        boolean isValid = (a + b > c) && (b + c > a)
        && (c + a > b);
        System.out.println(isValid ? "Valid triangle"
        : "Invalid triangle");
    }
}
```

Discount Eligibility

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        double totalAmount = 120.0;

        System.out.println(totalAmount >= 100 ?
        "Eligible for discount" : "Not eligible for discount");
    }
}
```

Logical Use of Relational Operators

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int number = 75;

        boolean result = (number % 5 == 0) &&
        (number >= 50 && number <= 100);
        System.out.println(result ?
        "Number meets the conditions" : "Number does not meet
        the conditions");
    }
}
```

String Comparison

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "Hello";

        System.out.println("Using == : " +
        (str1 == str2));
        System.out.println("Using equals():
        " + str1.equals(str2));
    }
}
```

Challenge Problem: Largest of Three Numbers

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int a = 15, b = 30, c = 25;

        int largest = (a > b) ? (a > c ? a : c)
        : (b > c ? b : c);
        System.out.println("The largest number is:
        " + largest);
    }
}
```

Logical Operators in Java

Logical operators are used to perform logical operations on boolean expressions. They return a boolean value (`true` or `false`) based on the logic applied.

Types of Logical Operators

Operator	Description	Example
&&	Logical AND	`a > b && c < d`
||	Logical OR	`a > b || c < d`
!	Logical NOT	`!(a > b)`

Logical AND (&&)

- Returns `true` if **both** conditions are `true`.

Example:

Example (Java)

```
int a = 10, b = 20, c = 30;
if (a < b && b < c) {
    System.out.println("Both conditions are true.");
}
```

Output: `Both conditions are true.`

Logical OR (||)

- Returns `true` if **at least one** condition is `true`.

Example:

Example (Java)

```
int x = 5, y = 15;
if (x > 10 || y > 10) {
    System.out.println("At least one condition is true.");
}
```

Output: `At least one condition is true.`

Logical NOT (!)

- Reverses the logical state of a condition.

Example:

Example (Java)

```
boolean flag = false;
if (!flag) {
    System.out.println("The condition is false.");
}
```

Output: `The condition is false.`

Truth Tables

Logical AND (&&):

Expression 1	Expression 2	Result (&&)
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

Logical OR (||):

Expression 1	Expression 2	Result (||)
`true`	`true`	`true`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

Practical Example

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int age = 25;
        boolean citizen = true;

        // Logical AND
        if (age >= 18 && citizen) {
            System.out.println("Eligible to vote.");
        }

        // Logical OR
        int marks = 45;
        if (marks > 90 || marks > 40) {
            System.out.println("Pass in the exam.");
        }
```

Practical Example (Contd...)

Example (Java)

```
        // Logical NOT
        boolean isRaining = false;
        if (!isRaining) {
            System.out.println("You can go outside.");
        }
    }
}
```

Assignment Operators

Assignment Operators:

- = (Assign) `a = 5;`
- += (Add and assign) `a += 5;`
- -= (Subtract and assign) `a -= 5;`
- *= (Multiply and assign) `a *= 5;`
- /= (Divide and assign) `a /= 5;`

Assignment Operator (=)

- The assignment operator = assigns the value on the right to the variable on the left.

Example:

Example (Java)

```
int a;
a = 10;
System.out.println("The value of a is: " + a);
```

Output: `The value of a is:  10`

Add and Assign (+=)

- Adds the right operand to the left operand and assigns the result to the left operand.

Example:

Example (Java)

```
int a = 5;
a += 3;
System.out.println("The value of a is: " + a);
```

Output: `The value of a is:  8`

Subtract and Assign (-=)

- Subtracts the right operand from the left operand and assigns the result to the left operand.

Example:

Example (Java)

```
int a = 10;
a -= 4;
System.out.println("The value of a is: " + a);
```

Output: `The value of a is:  6`

Multiply and Assign (*=)

- Multiplies the left operand by the right operand and assigns the result to the left operand.

Example:

Example (Java)

```
int a = 4;
a *= 2;
System.out.println("The value of a is: " + a);
```

Output: `The value of a is:  8`

Divide and Assign (/=)

- Divides the left operand by the right operand and assigns the result to the left operand.

Example:

Example (Java)

```
int a = 20;
a /= 4;
System.out.println("The value of a is: " + a);
```

Output: `The value of a is:  5`

Modulo and Assign (%=)

- Takes the modulus (remainder) of the left operand divided by the right operand and assigns the result to the left operand.

Example:

Example (Java)

```
int a = 10;
a %= 3;
System.out.println("The value of a is: " + a);
```

Output: `The value of a is:  1`

Practical Example for Assignment Operators

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int x = 5;
        x += 3; // Adds 3 to x
        System.out.println("After +=: " + x);
        x -= 2; // Subtracts 2 from x
        System.out.println("After -=: " + x);
        x *= 4; // Multiplies x by 4
        System.out.println("After *=: " + x);
        x /= 2; // Divides x by 2
        System.out.println("After /=: " + x);
        x %= 3; // Takes modulus of x by 3
        System.out.println("After %=: " + x);
    }
}
```

Bitwise Operators

Bitwise Operators:

- & (AND)
- | (OR)
- ^ (XOR)
- >> (Right shift)
- << (Left shift)

Bitwise AND (&)

- Performs a bitwise AND operation between two integers.
- Each bit of the result is 1 if the corresponding bits of both operands are 1.

Example:

Example (Java)

```
int a = 5; // 0101 in binary
int b = 3; // 0011 in binary
int result = a & b; // Result is 0001 in binary
System.out.println("Result of a & b: " + result);
```

Output: `Result of a & b:  1`

Bitwise OR (|)

- Performs a bitwise OR operation between two integers.
- Each bit of the result is 1 if at least one of the corresponding bits of the operands is 1.

Example:

Example (Java)

```
int a = 5; // 0101 in binary
int b = 3; // 0011 in binary
int result = a | b; // Result is 0111 in binary
System.out.println("Result of a | b: " + result);
```

Output: `Result of a | b:  7`

Bitwise XOR (^)

- Performs a bitwise XOR operation between two integers.
- Each bit of the result is 1 if the corresponding bits of the operands are different.

Example:

Example (Java)

```
int a = 5; // 0101 in binary
int b = 3; // 0011 in binary
int result = a ^ b; // Result is 0110 in binary
System.out.println("Result of a ^ b: " + result);
```

Output: `Result of a ^ b:  6`

Bitwise NOT (~)

- Performs a bitwise NOT operation, which inverts each bit of the operand.

Example:

Example (Java)

```
int a = 5; // 0101 in binary
int result = ~a; // Result is 1010 in binary (inverted)
System.out.println("Result of ~a: " + result);
```

Output: `Result of ã:  -6`

Left Shift (<<)

- Shifts the bits of the left operand to the left by the number of positions specified by the right operand.
- Zeros are shifted into the rightmost bits.

Example:

Example (Java)

```
int a = 5; // 0101 in binary
int result = a << 2;
// Shifting left by 2, result is 10100 in binary
System.out.println("Result of a << 2: " + result);
```

Output: `Result of a << 2:  20`

Right Shift (>>)

- Shifts the bits of the left operand to the right by the number of positions specified by the right operand.
- The sign bit (leftmost bit) is shifted into the leftmost bits in case of signed integers.

Example:

Example (Java)

```
int a = 20; // 10100 in binary
int result = a >> 2;
// Shifting right by 2, result is 101 in binary
System.out.println("Result of a >> 2: " + result);
```

Output: `Result of a >> 2:  5`

Unsigned Right Shift (>>>)

- Shifts the bits of the left operand to the right by the number of positions specified by the right operand.
- Unlike the regular right shift, this operator does not preserve the sign bit; zeros are shifted into the leftmost bits.

Example:

Example (Java)

```
int a = -20; // 11111111111111111111111111101100 in binary
int result = a >>> 2;
// Shifting right by 2, result is
// 00111111111111111111111111111011
System.out.println("Result of a >>> 2: " + result);
```

Output: `Result of a >>> 2:  1073741821`

Practical Example for Bitwise Operators

Example (Java)

```
public class Main {
    public static void main(String[] args) {
        int a = 5, b = 3;

        // Bitwise AND
        int andResult = a & b;
        System.out.println("a & b = " + andResult);

        // Bitwise OR
        int orResult = a | b;
        System.out.println("a | b = " + orResult);

        // Bitwise XOR
        int xorResult = a ^ b;
        System.out.println("a ^ b = " + xorResult);
```

Practical Example for Bitwise Operators (Contd...)

Example (Java)

```
        // Bitwise NOT
        int notResult = ~a;
        System.out.println("~a = " + notResult);
        // Left Shift
        int leftShiftResult = a << 2;
        System.out.println("a << 2 = " + leftShiftResult);
        // Right Shift
        int rightShiftResult = a >> 2;
        System.out.println("a >> 2 = " + rightShiftResult);

        // Unsigned Right Shift
        int unsignedRightShiftResult = a >>> 2;
        System.out.println("a >>> 2 = " + unsignedRightShiftRe
    }}
```

Unary and Ternary Operators

Unary Operators:

- + (Unary plus)
- - (Unary minus)
- ++ (Increment)
- -- (Decrement)
- ! (Logical NOT)

Ternary Operator:

- `condition ? expr1 : expr2;`
- Example: `result = (a > b) ? a : b;`

Unary Plus (+) and Unary Minus (-)

- `+` : Indicates a positive value (default, usually redundant).
- `-` : Negates the value of the operand.

Example:

Example (Java)

```
int num = 5;
System.out.println(+num); // Output: 5
System.out.println(-num); // Output: -5
```

Increment (++) and Decrement (--)

- ++ : Increases the value by 1.
- -- : Decreases the value by 1.
- Can be used in two forms:
  - **Pre-Increment/Decrement:** First modifies the value, then returns it.
  - **Post-Increment/Decrement:** Returns the value, then modifies it.

Example:

Example (Java)

```
int x = 10;
System.out.println(++x); // Pre-Increment: Output: 11
System.out.println(x--); // Post-Decrement: Output: 11
System.out.println(x);   // Output: 10
```

Logical Complement (!)

- Inverts the logical state of a boolean value.
- `true` becomes `false`, and vice versa.

Example:

Example (Java)

```
boolean flag = false;
System.out.println(!flag); // Output: true
```

Ternary Operator in Java (?:)

- A shorthand for `if-else` statements.
- Syntax:

$$result = (condition)?expression1 : expression2;$$

- If the `condition` is `true`, `expression1` is evaluated; otherwise, `expression2`.

Example:

Example (Java)

```
int a = 10, b = 20;
int max = (a > b) ? a : b;
System.out.println("Maximum: " + max); // Output: Maximum: 20
```

Practical Example: Ternary Operator

Example:

Example (Java)

```
int age = 18;
String eligibility = (age >= 18) ? "Adult" : "Minor";
System.out.println("You are an " + eligibility);
// Output: You are an Adult
```

Logical Operators - Questions to brush up!

1. Write a program to check if a number is both positive and divisible by 3. **Example:** For input 9, the output should be `"Number is positive and divisible by 3."`
2. Simulate a login system where the user enters a username and password. Check if the username is `"admin"` **OR** the password is `"1234"`. If either condition is true, print `"Login successful"`, otherwise print `"Login failed."`
3. Write a program that takes a person's age and citizenship status as input. Check if the person is eligible to vote (age $\geq$ `18 && citizen = true`). **Example:** Input: `Age = 20, Citizen = true` $\rightarrow$ Output: `"Eligible to vote."`
4. Write a program to check if a given number lies between 50 and 100 (inclusive) **AND** is even. **Example:** Input: 68 $\rightarrow$ Output: `"Number is in range and even."`

Assignment Operators - Questions to brush up!

1. Write a program to simulate a bank account balance update. Initialize the balance to 5000, then use assignment operators (+=, -=, *=) to:
   - Add 1000 for a deposit.
   - Deduct 1500 for a withdrawal.
   - Add 5% interest.

   Print the final balance.
2. A store offers a 10% discount for purchases over $100. Write a program to calculate the discounted price using *= and /= assignment operators.
3. In a game, a player earns points for different activities:
   - Collecting a coin adds 10 points.
   - Defeating an enemy adds 50 points.
   - Falling into a trap subtracts 20 points.

   Write a program that uses assignment operators to calculate the player's total score based on these activities.

Bitwise Operators - Questions to brush up!

1. Write a program to check if a number is odd or even using the bitwise AND operator. **Hint:** A number `n` is even if `n & 1 == 0`, otherwise it's odd.
2. Write a program to swap two numbers without using a temporary variable. Use the XOR operator (`^`) to achieve this.
3. - Write a program to take a number as input and print its value after shifting:
     - Left by 2 positions.
     - Right by 2 positions.

   Example: Input $= 8 \rightarrow$ Left shift: 32, Right shift: 2.
4. Write a program to check if a number is a power of 2 using bitwise operators. **Hint:** A number `n` is a power of 2 if `(n & (n - 1)) == 0` and `n > 0`.
5. Write a program to count the number of 1s in the binary representation of a number using bitwise operators.
6. Write a program that takes a number as input and toggles all its bits (flips 0s to 1s and vice versa). **Example:** Input: 5 (binary 00000101) $\rightarrow$ Output: 250 (binary 11111010 for 8-bit numbers).

Challenging Questions to brush up!

1. **Digital Lock System:** Write a program to simulate a digital lock with a 4-digit PIN. Allow the user 3 attempts to guess the PIN:
   - Use logical operators to check if the guess matches.
   - Use assignment operators to decrement the remaining attempts after each wrong guess.
2. **Prime Number Check Using Bitwise:** Write a program to check if a number is prime using bitwise operators to perform efficient modulus calculations.
3. **Bitwise Magic Number:** Write a program to check if a number is "magic." A number is "magic" if reversing its binary representation results in the same number. **Example:** Input 9 (binary 1001) $\rightarrow$ Output: `"Magic number."`

Logical Operators - Checking if a number is positive and divisible by 3

Example (Positive and Divisible by 3)

```
import java.util.Scanner;

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

        if (number > 0 && number % 3 == 0) {
            System.out.println("Number is positive
            and divisible by 3.");
        } else {
            System.out.println("Condition not met.");
        }    }}
```

Logical Operators - Simulating a Login System

Example (Login System)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter username: ");
        String username = scanner.nextLine();
        System.out.print("Enter password: ");
        String password = scanner.nextLine();

        if (username.equals("admin") ||
        password.equals("1234")) {
            System.out.println("Login successful.");
        } else {
            System.out.println("Login failed.");
        }    } }
```

Logical Operators - Checking Voting Eligibility

Example (Voting Eligibility)

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter age: ");
        int age = scanner.nextInt();
        System.out.print("Are you a citizen (true/false): ");
        boolean isCitizen = scanner.nextBoolean();

        if (age >= 18 && isCitizen) {
            System.out.println("Eligible to vote.");
        } else {
            System.out.println("Not eligible to vote.");
        }    }}
```

Logical Operators - Checking if a number is in range and even

Example (In Range and Even)

```
import java.util.Scanner;

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

        if (number >= 50 && number <= 100 &&
        number % 2 == 0) {
            System.out.println("Number is in
             range and even");
        } else {
            System.out.println("Condition not met.");
```

Assignment Operators - Bank Account Balance Update

Example (Bank Account Balance Update)

```
public class BankAccount {
    public static void main(String[] args) {
        double balance = 5000;

        // Deposit
        balance += 1000;

        // Withdrawal
        balance -= 1500;

        // Add 5% interest
        balance *= 1.05;

        System.out.println("Final Balance: " + balance);
    }}
```

Assignment Operators - Discount Calculation

Example (Discount Calculation)

```
import java.util.Scanner;

public class DiscountCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter purchase amount: ");
        double amount = scanner.nextDouble();

        if (amount > 100) {
            amount *= 0.90; // Apply 10% discount
        }

        System.out.println("Discounted Price: " + amount);
    }
}
```

Assignment Operators - Game Points Calculation

Example (Game Points Calculation)

```
public class GamePoints {
    public static void main(String[] args) {
        int points = 0;

        // Collecting a coin
        points += 10;

        // Defeating an enemy
        points += 50;

        // Falling into a trap
        points -= 20;

        System.out.println("Total Points: " + points);
    }}
```

Bitwise Operators - Odd or Even Check

Example (Odd or Even Check)

```
import java.util.Scanner;

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

        if ((number & 1) == 0) {
            System.out.println("Even number.");
        } else {
            System.out.println("Odd number.");
        }
    }
}
```

Bitwise Operators - Swapping Numbers

Example (Swapping Numbers)

```
import java.util.Scanner;
public class SwapNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number: ");
        int a = scanner.nextInt();
        System.out.print("Enter second number: ");
        int b = scanner.nextInt();

        a = a ^ b;
        b = a ^ b;
        a = a ^ b;
        System.out.println("After swap:
        a = " + a + ", b = " + b);
    }}
```

Bitwise Operators - Shifting Operations

Example (Shifting Operations)

```
import java.util.Scanner;

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

        System.out.println("Left shift by 2: " +
        (number << 2));
        System.out.println("Right shift by 2: " +
        (number >> 2));
    }}
```

Bitwise Operators - Check Power of 2

Example (Check Power of 2)

```
import java.util.Scanner;

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

        if ((number & (number - 1)) == 0 && number > 0) {
            System.out.println("Power of 2.");
        } else {
            System.out.println("Not a power of 2.");
        }
    }
}
```

Challenging Questions - Digital Lock System

Example (Digital Lock System)

```
import java.util.Scanner;
public class DigitalLock {
    public static void main(String[] args) {
        final int PIN = 1234; // Predefined PIN
        int attempts = 3; // Number of attempts allowed
        Scanner scanner = new Scanner(System.in);
        // First attempt
        System.out.print("Enter PIN: ");
        int input = scanner.nextInt();
        if (input == PIN) {
            System.out.println("Access granted.");
        } else {
            attempts--;
            System.out.println("Incorrect PIN. Attempts
            left: " + attempts);
```

Challenging Questions - Digital Lock System (Contd...)

Example (Digital Lock System)

```
// Second attempt
            System.out.print("Enter PIN: ");
            input = scanner.nextInt();
            if (input == PIN) {
                System.out.println("Access granted.");
            } else {
                attempts--;
                System.out.println("Incorrect PIN. Attempts le
                // Third attempt
                System.out.print("Enter PIN: ");
                input = scanner.nextInt();
                if (input == PIN) {
                    System.out.println("Access granted.");
```

Challenging Questions - Digital Lock System (Contd...)

Example (Digital Lock System)

```
                } else {
                    attempts--;
                    System.out.println("Incorrect PIN.
                    No attempts left. Access denied.");
                }
            }
        }

        scanner.close(); // Close the scanner resource
    }
}
```

What is a Switch Statement?

- A control flow statement used to select one of many code blocks to execute.
- Evaluates an expression and matches it against defined cases.
- Alternative to multiple `if-else-if` statements.
- Supports `byte`, `short`, `int`, `char`, `String`, and enums.

Syntax of Switch in Java

Example (Java Code)

```
switch (expression) {
    case value1:
        // Code to execute if expression == value1
        break;
    case value2:
        // Code to execute if expression == value2
        break;
    default:
        // Code to execute if no case matches
}
```

Example 1: Basic Switch Statement

Example (Java Code)

```
public class SwitchExample {
    public static void main(String[] args) {
        int day = 3;
        String dayName;

        switch (day) {
            case 1: dayName = "Monday"; break;
            case 2: dayName = "Tuesday"; break;
            case 3: dayName = "Wednesday"; break;
            default: dayName = "Invalid day";
        }

        System.out.println("Day is: " + dayName);
    }
}
```

Output of Example 1

Input: `day = 3`
Output: `Day is:  Wednesday`

Example 2: String in Switch

Example (Java Code)

```
public class SwitchStringExample {
    public static void main(String[] args) {
        String fruit = "Apple";

        switch (fruit) {
            case "Apple":
                System.out.println("Fruit is Apple");
                break;
            case "Mango":
                System.out.println("Fruit is Mango");
                break;
            default:
                System.out.println("Unknown Fruit");
        }
    }}
```

Key Points

- **Break Statement:** Prevents fall-through to the next case.
- **Default Case:** Executes if no match is found.
- **Fall-Through:** If `break` is omitted, subsequent cases execute.

Problem 1: Day of the Week

Write a program that takes an integer (1 to 7) as input and prints the corresponding day of the week. Use the following mapping:

- 1: Monday, 2: Tuesday, ..., 7: Sunday
- If the input is not between 1 and 7, print `Invalid Input`.

Solution 1: Day of the Week

Example (Java Code)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number (1-7): ");
        int day = scanner.nextInt();
        switch (day) {
            case 1: System.out.println("Monday"); break;
            case 2: System.out.println("Tuesday"); break;
            case 3: System.out.println("Wednesday"); break;
            case 4: System.out.println("Thursday"); break;
            case 5: System.out.println("Friday"); break;
            case 6: System.out.println("Saturday"); break;
            case 7: System.out.println("Sunday"); break;
            default: System.out.println("Invalid Input");
```

Problem 2: Calculator

Create a simple calculator program that takes two numbers and an operator (`+`, `-`, `*`, `/`) as input. Use a `switch` statement to perform the corresponding operation and display the result.

- Handle invalid operators with a `default` case.

Solution 2: Calculator

Example (Java Code)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter an operator (+, -, *, /): ");
        char operator = scanner.next().charAt(0);
        System.out.print("Enter second number: ");
        double num2 = scanner.nextDouble();
        switch (operator) {
            case '+': System.out.println("Result:
            " + (num1 + num2)); break;
            case '-': System.out.println("Result:
            " + (num1 - num2)); break;
```

Solution 2: Calculator (Contd...)

Example (Java Code)

```
            case '*': System.out.println("Result:
            " + (num1 * num2)); break;
            case '/':
                if (num2 != 0)
                    System.out.println("Result:
                    " + (num1 / num2));
                else
                    System.out.println
                    ("Division by zero is not allowed.");
                break;
            default: System.out.println
            ("Invalid operator.");
        }   } }
```

Problem 3: Season Finder

Write a program that accepts a month number (1 to 12) and prints the season:

- Winter: December (12), January (1), February (2)
- Spring: March (3), April (4), May (5)
- Summer: June (6), July (7), August (8)
- Autumn: September (9), October (10), November (11)
- If the input is not between 1 and 12, print `Invalid Month`.

Solution 3: Season Finder

Example (Java Code)

```
import java.util.Scanner;
public class SeasonFinder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter month number (1-12): ");
        int month = scanner.nextInt();

        switch (month) {
            case 12: case 1: case 2:
                System.out.println("Winter");
                break;
            case 3: case 4: case 5:
                System.out.println("Spring");
                break;
            case 6: case 7: case 8:
```

Problem 4: Grade Evaluation

Write a program that takes a grade letter (`A`, `B`, `C`, `D`, `F`) as input and prints the corresponding description:

- A: Excellent
- B: Good
- C: Average
- D: Below Average
- F: Fail
- Use the `default` case to handle invalid grades.

Solution 4: Grade Evaluation

Example (Java Code)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter grade
        (A, B, C, D, F): ");
        char grade = scanner.next().
        toUpperCase().charAt(0);

        switch (grade) {
            case 'A': System.out.println("Excellent"); break;
            case 'B': System.out.println("Good"); break;
            case 'C': System.out.println("Average"); break;
            case 'D': System.out.println("Below Average"); bre
            case 'F': System.out.println("Fail"); break;
```

Problem 5: Days in a Month

Create a program that takes a month number (1 to 12) as input and prints the number of days in that month.

- Consider February as having 28 days.
- Handle invalid month numbers.

Solution 5: Days in a Month

Example (Java Code)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter month number (1-12): ");
        int month = scanner.nextInt();
        switch (month) {
            case 1: case 3: case 5: case 7: case 8:
            case 10: case 12:
                System.out.println("31 days");
                break;
            case 4: case 6: case 9: case 11:
                System.out.println("30 days");
                break;
            case 2:
                System.out.println("28 days");
```

Problem 6: Vowel or Consonant

Write a program that takes a single character as input and uses a `switch` statement to check if it is a vowel (`a`, `e`, `i`, `o`, `u`) or a consonant.

- Handle invalid inputs such as digits or symbols.

Problem 7: Electricity Bill Calculation

Write a program that takes the electricity unit slab as input and calculates the rate per unit:

- Slab 1: 0-100 units, 1.50/unit
- Slab 2: 101-300 units, 3.00/unit
- Slab 3: Above 300 units, 5.00/unit
- Use the `default` case for invalid slabs.

Problem 8: Traffic Light Simulator

Create a program that takes a traffic light color (`Red`, `Yellow`, `Green`) as input and prints the corresponding action:

- Red: Stop
- Yellow: Get Ready
- Green: Go
- Handle invalid inputs with the `default` case.

Problem 9: Number to Word Converter

Write a program that takes a single digit (0 to 9) as input and prints the corresponding word:

- 0: Zero, 1: One, ..., 9: Nine
- Use `default` to handle invalid inputs.

Problem 10: Simple Menu-Driven Program

Create a menu-driven program that displays the following options:

1. Add two numbers
2. Subtract two numbers
3. Multiply two numbers
4. Divide two numbers

Take the user's choice and two numbers as input, and perform the corresponding operation. Handle invalid menu options.

Problem 11: Animal Sound

Write a program that takes an animal name (`Dog`, `Cat`, `Cow`, `Duck`) as input and prints the corresponding sound:

- Dog: Woof
- Cat: Meow
- Cow: Moo
- Duck: Quack
- Use the `default` case to handle other animals.

Problem 12: Zodiac Sign

Write a program that takes a month number and a day as input and prints the zodiac sign based on the date range. Use a `switch` statement for months and handle the logic for day ranges inside each case.

Verdict

- Basic building blocks
- Java Syntax
- Compared with Python language