**Class 10 chapter 4**

**Operators in Java**

**A. Tick (✓) the correct answer.**

1. The '+' operator is an example of **c. Arithmetic** (✓).
2. Which of the following is an example of a postfix increment operator?
   * **b. a++** (✓).
3. Which of the following is an example of a prefix decrement operator?
   * **b. a--** (✓).
4. Which of the following is the correct use of a ternary operator?
   * **d. (a > b) ? "a is greater than b" : "b is greater than a";** (✓).
5. Which of the following operators can we use to initialize all non-primitive data types?
   * **c. new operator** (✓).

**B. Fill in the blanks.**

1. Arrange these operators (<, ++, %) in order of higher precedence to lower precedence:
   * **++, %, <**.
2. The equivalent Java expression for a2+b2+2aba^2 + b^2 + 2aba2+b2+2ab is:
   * **(a \* a) + (b \* b) + (2 \* a \* b)**.
3. The output of the expression a += a++ \* ++a % a if a = 5:
   * **a += 5 \* 7 % 5 → a += 35 % 5 → a += 0 → a = 5**.
4. The expression (true) || (true) will return:
   * **true**.
5. The expression 10 % 3 will return:
   * **1**.

**C. Short Answer Type Questions.**

1. **What is the output of the following expressions if executed sequentially? Where a = 10, b = 2, and c = 8**

c += a / b \* c;

* + c = c + (10 / 2 \* 8)
  + c = 8 + (5 \* 8)
  + c = 8 + 40
  + **c = 48**

a = c + b++ \* (a + b + c);

* + a = 48 + 2 \* (10 + 2 + 48)
  + a = 48 + 2 \* 60
  + a = 48 + 120
  + **a = 168**

1. **What will be the output of the following ternary expression if int a = 5, b = 3, and double d?**

d = (a >= b) ? (a + b) : (a \* b);

* + Since a (5) >= b (3), it executes (a + b) = (5 + 3).
  + **Output: d = 8**.

1. **What is the difference between the ternary operator and the unary operator? Give one example of each.**

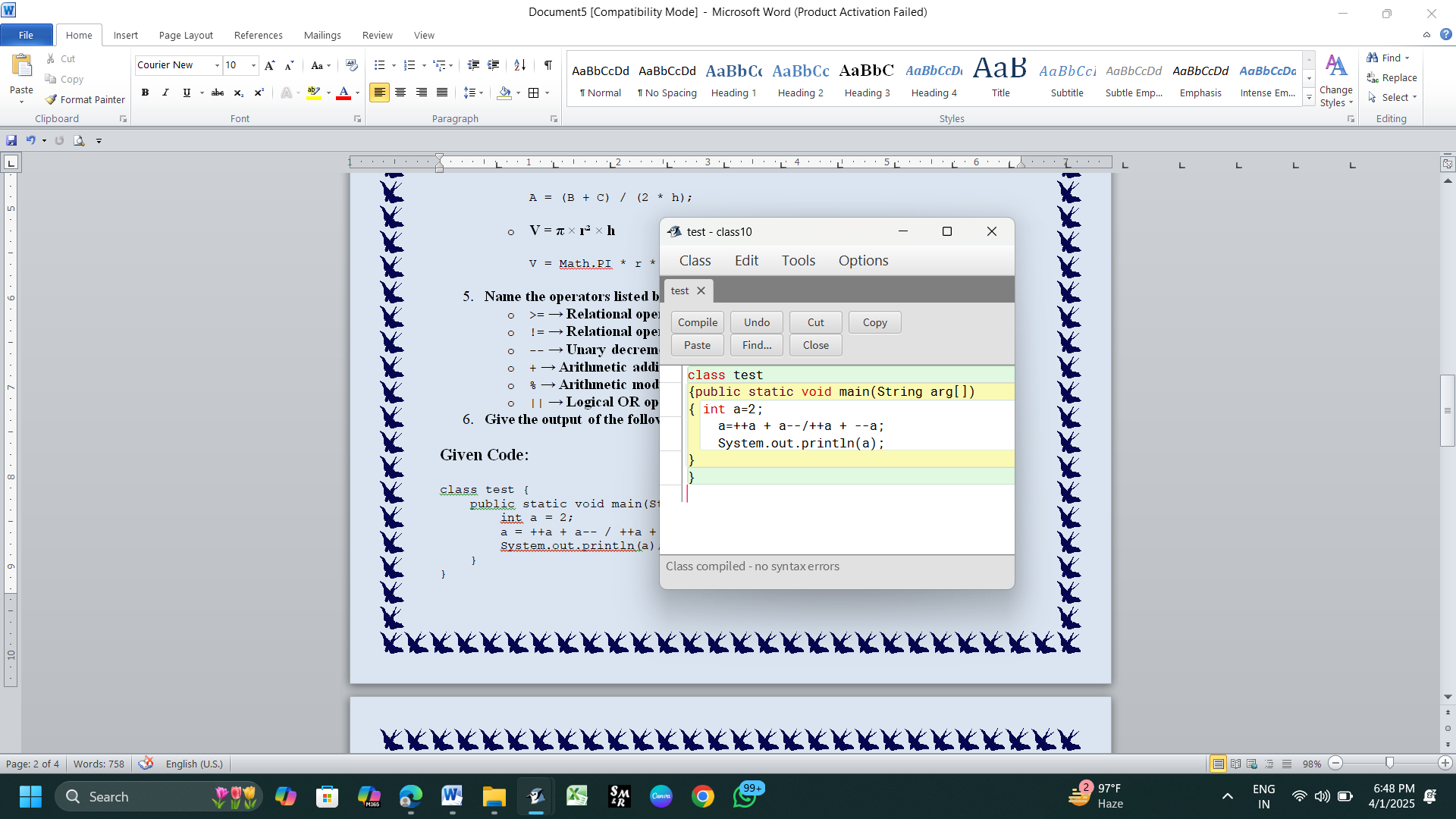
| **Feature** | **Ternary Operator** | **Unary Operator** |
| --- | --- | --- |
| Definition | Used for conditional expressions (? :) | Operates on a single operand |
| Example | int min = (a < b) ? a : b; | int x = -5; x++; |

1. **Write the Java expressions of the following mathematical expressions:**
   * **A = (B + C) / 2 × h**

A = (B + C) / (2 \* h);

* + **V = π × r² × h**

V = Math.PI \* r \* r \* h;

1. **Name the operators listed below:**
   * >= → **Relational operator (Greater than or equal to)**
   * != → **Relational operator (Not equal to)**
   * -- → **Unary decrement operator**
   * + → **Arithmetic addition operator**
   * % → **Arithmetic modulus operator**
   * || → **Logical OR operator**
2. **Give the output of the following expressions:**

### ****Given Code:**** 1st part

class test {

public static void main(String arg[])

{

int a = 2;

a = ++a + a-- / ++a + --a;

System.out.println(a);

}

}

### ****Step-by-Step Execution:****

1. **Initial Value:** a = 2
2. **Evaluate ++a (Pre-increment, increases a first and then uses it)**
   * ++a → a = 3
   * **Value used:** 3
3. **Evaluate a-- (Post-decrement, uses a first, then decreases)**
   * **Value used:** a = 3
   * a-- → a = 2 (Now a = 2)
4. **Evaluate ++a (Pre-increment again)**
   * ++a → a = 3
   * **Value used:** 3
5. **Evaluate --a (Pre-decrement, decreases a first)**
   * --a → a = 2
   * **Value used:** 2

### ****Substituting Values in Expression****

a = 3 + 3 / 3 + 2

1. **Evaluate Division / First (According to Operator Precedence)**

3 / 3 = 1

Now the expression reduces to:

a = 3 + 1 + 2

1. **Final Addition**

a = 6

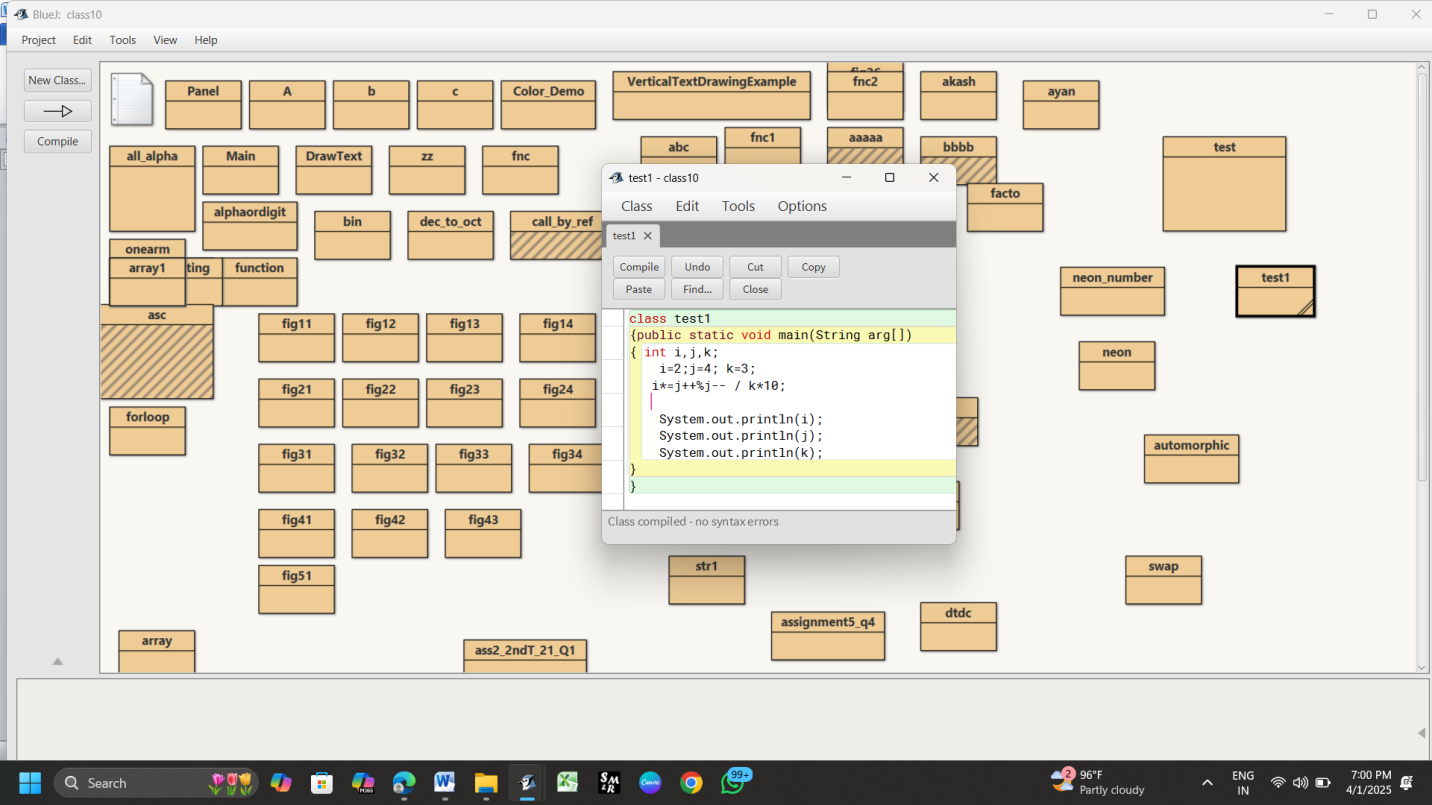
### ****Final Output:****

**6**

#### ****Key Learnings:****

* **Pre-increment (++a)** → Increases the value first, then uses it.
* **Post-decrement (a--)** → Uses the value first, then decreases it.
* **Operator Precedence:** / (division) is evaluated before + (addition).
* **Order of Execution Matters!**

**This expression contains a logic error (j++ % j-- / k \* 10). Please clarify it properly.**

1. **Give the output of the following expressions: 2nd part**

### ****Given Code:****

class test1 {

public static void main(String arg[]) {

int i, j, k;

i = 2;

j = 4;

k = 3;

i \*= j++ % j-- / k \* 10;

System.out.println(i);

System.out.println(j);

System.out.println(k);

}

}

### ****Step-by-Step Execution:****

#### ****Initial values:****

i = 2

j = 4

k = 3

#### ****Evaluating the Expression:****

i \*= j++ % j-- / k \* 10;

Since \*= is shorthand for:

i = i \* (j++ % j-- / k \* 10);

We need to evaluate the right-hand expression step by step.

### ****Step 1: Evaluate**** j++ ****(Post-increment)****

* j++ means **use the value of j first, then increment it.**
* j = 4, so the expression **uses 4** and then j becomes 5.
* So, we now have:

(4 % j-- / k \* 10)

and j is now 5.

### ****Step 2: Evaluate**** j-- ****(Post-decrement)****

* j-- means **use the value of j first, then decrement it.**
* j = 5, so the expression **uses 5** and then j becomes 4.
* So, we now have:

(4 % 5 / k \* 10)

and j is now 4.

### ****Step 3: Evaluate Modulus (****%****)****

4 % 5 = 4

(Since 4 is smaller than 5, the remainder is 4.)

So, the expression now reduces to:

(4 / k \* 10)

where k = 3.

### ****Step 4: Evaluate Division**** /

4 / 3 = 1 (Since integer division in Java truncates the decimal part)

Now, the expression reduces to:

(1 \* 10)

### ****Step 5: Evaluate Multiplication**** \*

1 \* 10 = 10

### ****Step 6: Evaluate**** i \*=

i = i \* 10

i = 2 \* 10

i = 20

### ****Final Values:****

**i = 20**

**j = 4**

**k = 3**

### ****Final Output:****

**20**

**4**

**3**

### ****Key Takeaways:****

* **Post-increment (j++)**: Uses j first, then increments.
* **Post-decrement (j--)**: Uses j first, then decrements.
* **Integer division (/)**: Truncates the decimal part.
* **Operator precedence**:
  + % (modulus) and / (division) have the same precedence and are evaluated from **left to right**.
  + \* (multiplication) follows /.
  + \*= is the last operation performed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*