Computer Science by AP Tutorial

Suplementary Notes for module 2 [addendum to notes-2]

Truth tables and Java concepts like typecasting, bitwise operators, and operator precedence form foundational knowledge in computer science. Here's a structured breakdown:

Truth Tables for Logical Operations

> You can read T (True) as 1 and F (False) as 0, both can also be read as high volt, and low volt respectively

AND (∧)

Outputs **True** only when **both inputs are True**:

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

Example: "You need a password AND biometrics to access the system."[2]

OR (∨)

Outputs **True** if **at least one input is True**:

A	B	A ∨ B
T	T	T
T	F	T
F	T	T
F	F	F

Example: "The alarm triggers if motion OR sound is detected."[2]

NOT (¬)

Outputs the **opposite** of the input:

A	¬A
T	F

A	¬A
F	T

Example: "NOT raining" is True when it’s sunny.[2]

XOR (⊕)

Outputs **True** when **inputs differ**:

A	B	A ⊕ B
T	T	F
T	F	T
F	T	T
F	F	F

Example: A light controlled by two switches where flipping either changes the state[3].

Typecasting in Java

Converting one data type to another:

Widening (Implicit)

- Automatic conversion from smaller to larger types (e.g., `int` → `double`).
- No data loss.

```
int num = 10;
double d = num; // 10.0
```

Narrowing (Explicit)

- Manual conversion from larger to smaller types (e.g., `double` → `int`).
- Risk of data loss.

```
double d = 10.5;
int num = (int) d; // 10
```

Syntax: `targetType variable = (targetType) sourceVariable;`[4][5]

Bitwise Operators in Java

Operate on individual bits:

Operator	Description	Example (A=5, B=3)
`&`	AND	5 & 3 = 1 (0101 & 0011 = 0001)
`|`	OR	5
`^`	XOR	5 ^ 3 = 6 (0101 ^ 0011 = 0110)
`~`	NOT (1's complement)	~5 = -6
`>`	Signed right shift	5 >> 1 = 2 (0101 → 0010)
`>>>`	Unsigned right shift	-5 >>> 1 = 2147483645

Operator Precedence in Java

Order of evaluation (highest to lowest):

1. **Postfix**: `expr++`, `expr--`
2. **Unary**: `++expr`, `--expr`, `~`, `!`
3. **Multiplicative**: `*`, `/`, `%`
4. **Additive**: `+`, `-`
5. **Shift**: `>`, `>>>`
6. **Relational**: ``, `=`, `instanceof`
7. **Equality**: `==`, `!=`
8. **Bitwise AND**: `&`
9. **Bitwise XOR**: `^`
10. **Bitwise OR**: `|`
11. **Logical AND**: `&&`
12. **Logical OR**: `||`
13. **Ternary**: `? :`
14. **Assignment**: `=`, `+=`, `-=`, etc.

Example: `int result = 5 + 3 * 2;` evaluates to 11 (3*2 first).

> Multiplication, addition, subtraction are all binary operators (since two data points are needed at minimum), and thus done after unary

---

These concepts are essential for programming logic and efficient code design. Truth tables help visualize boolean operations, while typecasting and operators enable precise data manipulation in Java.