Anatomy of a Java Program :→ Pascal Naming Convention
Rate Of Interest
(constants) (interfaces)

```
public class Main {
    public static void main (String[] args) {
        System.out.println(" SNPSU ");
    }
}
```

main → camel naming convention

underscore case — python — rate_of_int( )

rateOfInterest — vars, methods (functions)

Operators :→

1. Arithmetic : +, -, /, *, % — Python: ** (exp), // (floor div)
2. Assignment : =, +=, -=, /=, *=, %=
   - Augmented / Shorthand
   - a = 5
   - a = a + 10
   - a += 10
3. Logical : && || ! — and or not
4. Relational | Comparison (Boolean) : >, <, >=, <=, ==, !=
5. Unary Operators : ++ / -- (Prefix | Postfix)
6. Ternary Operator (Short Hand If Else Operator)

(Condition) ? tv : fv ; — C, C++

Return Type var = (Condition) ? tv : fv ;
(Return Type and tv : fv — same)

* **Bitwise Operators :→** (They work on individual bits)

I	AND	→ &	→ Ampersand
II	OR	→ |	→ Pipe Symbol
III	XOR	→ ^	→ Caret
IV	Right Shift	→ >>	→ Angular Brackets
V	Left Shift	→ <<	→ Angular Brackets
VI	NOT	→ ~	→ Negation Tilde

3 → 0011
8 → 1000
3^8 = 1011 = 11

bin:
7 → 0111
9 → 1001
int: 0001

7 & 9 = 1 (int, int → int)

6 → 0110
5 → 0101
0111 = 7

Bitwise Shift :
value = 10
unit / step = 2

8 bit — 1 byte
0 0 0 0 1 0 1 0
(16 8 4 2 1; 2^4 2^3 2^2 2^1 2^0)

10 << 2 = 40 ↑
0 0 | 0 0 1 0 1 0 0 0 (added)
discarded — 32 8

(Bit Manipulations)
10 >> 2 = 2 ↓
0 0 | 0 0 0 0 1 0 1 0 — discarded (OOB)
Anything outside reference line is discarded.
To balance we add 0

NOT Table

T	F
F	T

0	1
1	0

5 → 0101
~5 → 1010 = 10 (TV)
−6 (PV)

abs (−6) = 6 → BIN = 0110
Negation of 2 / 2's Com { 1's Com = 1001; + 1 = 0001 } → 1010
or Double Negation

MCQ : Q: ~5 a) 5 b) 10 c) −6 ✓ d) 4

Formula : ~n = −n − 1 = −5 − 1 = −6
~(−499) = −(−499) − 1 = (498)

* Power of 2 — n = {7, 8, 2, 4, 9}

7 × 0111 / 1001
10
8 2^3 1000 / 0111
2 2^1 0010 / 0001
4 2^2 0100 / 0011
9 × 1001 / 1000

```
if ((n & (n-1)) == 0) {
    → po2
    → napo2
}
```

{ Decision Making — Looping — Branching — Jumps }

Conditional Statements :

* simple if → Only one condition
* if else → Exactly two conditions
* if else if else ladder → More than 2 conditions
* nested if → Condition inside condition
* switch case → more than 5 → optimal
* ternary operator → short hand if else

(22) e { d by 4 / e { nd by 4
(21) (29) o { d by 3 / o { nd by 3

if { if { else { } else { if { else {

<0 0 (==) >0

Nesting of Branching

Looping Statements :→

* while (condition) {
    Statements;
    inc / dec operations;
  }

(Uncertain) Entry Controlled Loop

At least once
do {
    statements;
    inc / dec operators;
} while (condition);

Exit Controlled Loop

for loop : (Finite Case) No of steps "known"

```
for (initialization ; condition ; inc/dec) {
    Statements;
}
```

```
for (int i=0; i<5; i++) {
    sout (i);
}
```
(i = initialization, c = condition, i/d = inc/dec; sout → Stmt)

4 2 1 5 2 4 1 — r = 0 — (twice / 1 unique) — (DRY RUN)

0000
0100
0100
0010
0110
0001
0111
0101
0010
0010
0000
0100
0100
0001
0101 = 5