Below are comprehensive revision notes from the class covering key concepts, as interpreted from the audio transcript and handwritten notes.

**Software Engineering Fundamentals: If-Else, Loops, and Operators**

**If-Else Statements**

**Concept**

* **If** and **Else** statements are fundamental in decision making within a program. They allow the program to execute a block of code based on a certain condition being true or false.

**Real-World Analogy**

* Imagine you are at a cafe:
  + **If** the cafe has coffee, **provide** it.
  + **Else** provide tea if coffee isn't available .

**Basic Structure**

if (condition) {

// code to execute if condition is true

} else {

// code to execute if condition is false

}

**Example: Voting Eligibility**

* **Problem:** Determine if a person is eligible to vote.
* **Condition:** Age should be 18 or older .

if (age >= 18) {

System.out.println("Eligible");

} else {

System.out.println("Not Eligible");

}

**Conditional Logs and Comparisons**

**Comparing Two Numbers**

* **Problem:** Given two integers, print the larger one or indicate if they are equal .
* **Solution:**

if (A > B) {

System.out.print(A + " is bigger");

} else if (B > A) {

System.out.print(B + " is bigger");

} else {

System.out.print("Both are equal");

}

**Working with Temperature**

**Temperature Classification**

* **Problem:** Given the temperature in Fahrenheit, determine if it's low, normal, or high .
* **Ranges:**
  + Low if below 98.2
  + Normal between 98.2 and 98.8
  + High if above 98.8

if (temperature < 98.2) {

System.out.print("low");

} else if (temperature > 98.8) {

System.out.print("high");

} else {

System.out.print("normal");

}

**Operators**

**Division**

* Division is carried out using the / operator.
* Resulting data type depends on the types of operands .

**Multiplication**

* Multiplication uses the \* operator, and follows similar rules regarding operand data types .

**Modulo**

* **Operator:** %
* **Function:** Returns the remainder of division .

**Typecasting and Precision**

* **Example:** Multiplying integers and ensuring data type fits result to avoid overflow .

int a = 100000;

int b = 100000;

long c = (long)a \* b; // Necessary typecasting to prevent overflow

* Proper casting ensures the results are stored without overflow by converting intermediate results to a larger data type if necessary.

**Understanding Modulo**

* **Calculations:** 7 % 3 = 1 because 7 divided by 3 is 2 with a remainder of 1.
* **Use Case:** Efficient way to check for even and odd numbers.
  + if (num % 2 == 0) the number is even .

**Control Flow with Conditions**

* Multiple if-else if-else statements can build comprehensive logic.
* Ensure conditions are logically ordered to prevent unintended fall-through or results .

**FizzBuzz Example**

* **Problem:** Print different strings ("Fizz", "Buzz", "FizzBuzz") based on divisibility conditions .
* **Approach:**

if (A % 3 == 0 && A % 5 == 0) {

System.out.print("Fizz-Buzz");

} else if (A % 3 == 0) {

System.out.print("Fizz");

} else if (A % 5 == 0) {

System.out.print("Buzz");

}

These notes cover the essentials discussed in class and provide a solid overview of using conditionals and operators in programming. Understanding these concepts is crucial for creating efficient and accurate software applications.

FizzBuzz

Print Fizz for multiples of 3 , Buzz for multiple of 5 , and FizzBuzz for both

Typecasting

Converting a variable from one data type to another, like int to long

Modulo Operator

Expression ‘a%b’ gives the remainder of a divided by b

If-Else

Conditional statement used to execute different block of code.

Eligibility to vote

A person is eligible to vote if their age is 18 or more

If(age > 10)

{

Eligible

}

Else

{

Not eligible

}

Logical AND

Return true only if both operands are true

Logical OR

Returns true if at least one operand is true

Electricity Bill calculation

Uses different rates for different consumption ranges

Integer Division

Dividing two integers gives an integer result by default

Float Division

Dividing an integer and float gives a float result

Last Digit Extraction

Use ‘n%10’ to get the last digit of an integer

Comparison of Two Integers

Identify the larger of two distinct integers

CLASS:

IF – ELSE:

Q. Given two integers A & B as input, print the larger. Note: A will not be equal to B

If(A>B)

{

Print A is greater

}

Else

{

Print B is greater

}

Q. A & B can be equal also find grater or equal

If(A==B)

{

Print equal

}

Else If(A>B)

{

Print A is greater

}

Else

{

Print B is greater

}

Q. Given a temperature of patient in Fahrenheit an input, print whether the temperature is low, normal or high.

Normal 98.2 to 98.8

If(temp < = 98.2)

{

Print low

}

Else If (temp > = 98.2 && temp <= 98.8)

{

Print normal

}

Else

{

Print high

}

OPERATORS:

Division (/):

Output datatype is based on dividend & divisor datatype

Int/int 🡪 int

Float/int 🡪 float

Int/float 🡪 float

Float/float 🡪 float

Long/int 🡪 long

Double/float 🡪 double

…

Whichever datatype is bigger in dividend or divisor that is taken as output datatype

Datatype Sequence:

Smaller Bigger

Byte 🡪 short 🡪 char 🡪 int 🡪 long 🡪 float 🡪double

(8bits) (16bits-signed) (16bits-unsigned) (32bits) (64bits) (32bits) (64bits)

byte

↓

short / char

↓

int

↓

long

↓

float

↓

double

**🎯 1. Size-wise:**

* short and char are **both 16 bits (2 bytes)** in memory.
  + short ➔ 16 bits signed (can be **negative or positive**).
  + char ➔ 16 bits **unsigned** (only **positive** values, 0 to 65,535).

So **by size**, they are equal — that's why they are shown at the same level.

**🎯 2. Type System Difference:**

* Even though they are the same size, **char and short are different types**:
  + short is a **numeric (integral)** type.
  + char is a **character** type (Unicode character).

**Important:**  
In expressions, when char and short are used in calculations, both are **automatically promoted to int**.

Example:

java

CopyEdit

char c = 'A'; // Unicode 65

short s = 10;

int result = c + s; // c and s both promoted to int

System.out.println(result); // 75

**🔥 Summary:**

| **Aspect** | **short** | **char** |
| --- | --- | --- |
| Size | 16 bits | 16 bits |
| Signedness | Signed (-32k to +32k) | Unsigned (0 to 65k) |
| Type | Integral numeric | Unicode character |
| Promotion | Promoted to int in expressions | Promoted to int in expressions |

👉 **So: "Same size, different roles" — but promotion behavior (both promoted to int) makes them stand at the same level in type hierarchy.**

Print 9/3 op: 3

Print 11/3 op: 3

Print 11f/3 op:3.6666667

Print 11.0/3 op:3.6666666666666667 // by default floating point is double

Multiplication (\*):

Same as division

Int \* int 🡪 int

Int \* long 🡪 long

Int \* float 🡪 float …

Int x=100000;

Int y=100000;

Int z= x\*y;

**Print z; // print some garbage value as int \* int 🡪 int**

Int x=100000;

Int y=100000;

**long** z= x\*y;

**Print z; // print again garbage value even if we are storing it into long data type as int \* int 🡪 int which overflow and generate garbage value then storing that garbage value to long datatype.**

Int x=100000;

Int y=100000;

**long** **z=(long) x\*y;**

**Print z; // 10000000000**

int a = 100000;

int b = 400000;

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

**System.out.println(c); //garbage value**

**1. int \* int 🡪 int : here overflow happens**

**2. typecast int 🡪 long**

**3. store to long**

Modulo (%):

Gives reminder when A is divided by B

7%3 =1 (3 \* 2 +1 =7)

8%5=3 (5\*1+3=8)

10%1=0 (10\*1+0=10)

5%12=5 (12\*0+5=5)

17%4=1 (4\*4+1=17)

Reminder always smaller than divisor

In A%B if B is completely divided A, then there is no reminder

Q. Given an integer as input print whether its even or odd

If(A%2==0)

{

Print even

}

Else

{

Print odd

}

A%2 only has 2 possible values =0,1

Q. Given an integer as input prints its last digit.

Int lastDigit= A%10;

Print lastDigit;

RELATIONAL OPERATOR:

A>B

A<B

A>=B

A<=B

A==B

A!=B

Input (Operand) of relational datatype could be anything (int/float/double…)

Output of relational operator always Boolean

LOGICAL OPERATOR:

Input (operand) datatype is Boolean.

Output is Boolean.