Perfect — here's your **short, clean, note-style summary** of all that content you just went through 👇

🧩 JAVA BASIC STRUCTURE — SHORT NOTES

🔹 Smallest Java Program

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

🔹 Explanation

Part	Meaning
`class Main`	Defines a class (container for code)
`{ ... }`	Code block
`public`	Visible to all
`static`	Belongs to class, not object
`void`	Returns nothing
`main()`	Program entry point
`String[] args`	Command-line arguments
`System.out.println()`	Prints output
`;`	Ends a statement

🔹 Rules

- Every program must have **one `main()`** method.
- **Code must be inside a class.**

- **Filename = public class name.**
- JVM starts execution from `main()`.
- Use `{}` for blocks, `;` for line end.

🔹 Example with Function

```
class Main {
    static void sayHello() {
        System.out.println("Hello Asmi!");
    }

    public static void main(String[] args) {
        sayHello();
    }
}
```

✅ Output → `Hello Asmi!`

🔹 Can We Have Other Classes?

Yes ✅
You can create **multiple classes** in one file.

```
class Helper {
    void greet() {
        System.out.println("Hello from Helper!");
    }
}

class Main {
    public static void main(String[] args) {
        Helper h = new Helper();  // object of Helper class
        h.greet();                // calling method from Helper
    }
}
```

✅ Output → `Hello from Helper!`

🧠 **Main points:**

- Only one class (usually `Main`) has `main()`.
- Other classes can exist to organize logic.
- You can put them in the same file (non-public) or in separate files.

Would you like me to continue this short-note style for **“Class & Object”** (how they work, constructors, `this`, etc.) next?

Absolutely 👍 here’s your **crisp + clean short notes** version for all the JS, Python, and Java array input discussion above 👇

🧠 ARRAY INPUT — SHORT NOTES

🔹 JavaScript

```
let n = parseInt(prompt("Enter number of students: "));
let arr = [];

for (let i = 0; i < n; i++) {
  arr[i] = parseInt(prompt("Enter score of student " + (i + 1)));
}

arr.forEach(val => {
  if (val < 40) console.log("FAIL");
  else console.log("PASS");
});
```

✅ Key Points

- Use `prompt()` to take user input.
- `parseInt()` / `parseFloat()` → convert string → number.
- `arr.push(value)` or `arr[i]=value` → store in array.
- `forEach()` → loop through array.
- No need to convert numbers to string when printing with `+`.

🔹 Python

```
n = int(input("Enter number of students: "))
arr = []

for i in range(n):
    score = int(input("Enter score of student " + str(i+1) + ": "))
    arr.append(score)

print("All scores:", arr)

for score in arr:
    if score < 40:
        print("FAIL")
    else:
        print("PASS")
```

✅ Key Points

- `input()` always returns **string** → use `int()` to convert.
- Use `arr.append()` to add values.
- Use `str()` to combine numbers with text.
- Use `for score in arr:` to loop.

🔹 Java

```
import java.util.Scanner;

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

        int[] arr = new int[n];
        System.out.println("Enter the scores:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("Scores are:");
        for (int score : arr) {
            System.out.println(score + (score < 40 ? " -> FAIL" : " -> PASS"));
        }
        sc.close();
    }
}
```

✅ **Key Points**

- Use `Scanner` for input.
- Declare array: `int[] arr = new int[n];`
- Use `arr[i] = sc.nextInt();` to store.
- To print array → use loop (`for (int score : arr)`).
- `System.out.println(arr);` ❌ prints memory address, not values.

🔹 **Summary Table**

Language	Input Function	Convert to Int	Add to Array	Loop	Print Check
JS	`prompt()`	`parseInt()`	`arr.push()`	`for` / `forEach()`	`console.log()`
Python	`input()`	`int()`	`append()`	`for in`	`print()`
Java	`Scanner.nextInt()`	—	`arr[i] =`	`for` / `for-each`	`System.out.println()`

Would you like me to export this as a **1-page PDF of “Array Input Notes (JS + Python + Java)”** with clean formatting?

Here’s a **short version of all your notes** so far 👇

🧠 IDE

- Full form: **Integrated Development Environment**
- Combines: **Editor + Compiler + Debugger + Build tools + Version control**
- Examples: VS Code, PyCharm, Eclipse, IntelliJ
- Purpose: One place to **write, run & debug code**.

💬 console.log()

- Used in **JavaScript** to print to console.

console.log("message"); // prints: message

- Quotes not shown in output.
- For debugging; shows result in console, not webpage.

🧮 Variables

A **container for storing data** (name = value).

Language	**Example**	**Notes**
JS	`let x=5; const y=10;`	Dynamic typing
Python	`x=5`	Dynamic typing

Java	`int x=5; final int y=10;`	Static typing

➕ Arithmetic Operators

`+ - * / % **`

- JS/Python: `2 ** 3`
- Java: `Math.pow(2,3)`

⚖️ Relational Operators

`== != > < >= <=` → return True/False.

✅ Booleans

JS	Python	Java
true/false	True/False	true/false

🔁 Variable Initialization & Re-assignment

Language	Change Allowed?	Constant Keyword

JS	`let` ✅ `const` ❌	`const`
Python	✅ always	No true constant (use CAPS)
Java	Normal ✅ `final` ❌	`final`

🧮 Basic Calculator

JavaScript

```
const prompt=require("prompt-sync")();

let a=parseFloat(prompt("Enter first: "));

let b=parseFloat(prompt("Enter second: "));

console.log("Add:", a+b);
```

Python

```
a=float(input("Enter first: "))

b=float(input("Enter second: "))

print("Add:", a+b)
```

Java

```
Scanner sc=new Scanner(System.in);

double a=sc.nextDouble(), b=sc.nextDouble();

System.out.println("Add: "+(a+b));
```

⚙️ require("prompt-sync")();

- `require("prompt-sync")` → imports a **function**.
- `()` → calls it, returns the actual **prompt** function.
  ✅ `const prompt=require("prompt-sync")();`

🔁 Function returning Function

```
function outer() {
  return function inner() {
    console.log("Hello");
  };
}
outer()(); // ✅ prints Hello
```

🧠 Not a callback — just a function returning another.

🧾 process.stdout.write()

- Low-level output (no newline auto).
- Accepts **string or buffer only** → so use:

```
process.stdout.write("Result: " + res + "\n");
```

Difference:

console.log( )	**process.stdout.write ()**

Any data type	Only string/buffer
Adds newline	No newline
High-level	Low-level

🐍 Python input() issue

```
len = input("Enter length ")
bre = input("Enter breadth ")
res = len * bre   # ❌ TypeError
```

👉 input() returns **string**, not number.

✅ Fix:

```
len = float(input("Enter length: "))
bre = float(input("Enter breadth: "))
res = len * bre
print(res)
```

Would you like me to make this a **well-formatted PDF** (with tables & syntax highlighting) for quick revision?