1. Java Basics

- **Java** is **object-oriented, platform-independent**, and **compiled into bytecode** that runs on the JVM (Java Virtual Machine).
- **JDK vs JRE vs JVM**:
    - **JDK** (Java Development Kit) → tools to write, compile, and run Java programs.
    - **JRE** (Java Runtime Environment) → JVM + libraries needed to run Java programs.
    - **JVM** → interprets bytecode and runs Java programs on any platform.
- **Data types:**
    - Primitive: int, long, float, double, char, boolean, byte, short.
    - Non-primitive: String, Arrays, Classes, Interfaces.
- **Variables**:
- int x = 10; // integer variable
- final int y = 20; // constant variable
- **Operators**: +, -, *, /, %, ==, !=, <, >, <=, >=, &&, ||, !.

---

2. Control Structures

- **If-else**:
- if(x > 0) {
- System.out.println("Positive");
- } else {
- System.out.println("Non-positive");
- }
- **Switch**:
- switch(day) {
- case 1: System.out.println("Monday"); break;
- default: System.out.println("Other day");
- }
- **Loops**:
    - **For**:
    - for(int i=0; i<5; i++) { System.out.println(i); }

- **While**:
- int i=0;
- while(i<5) { System.out.println(i); i++; }
- **Do-While**:
- int i=0;
- do { System.out.println(i); i++; } while(i<5);

---

3. Object-Oriented Programming (OOP)

- **Class**: blueprint for objects.
- **Object**: instance of a class.
- **Constructor**: special method to initialize objects.
- class Car {
- String model;
- Car(String m) { model = m; }
- }
- **Inheritance**: class Child extends Parent {}
- **Polymorphism**: method overloading (same name, different params) and overriding (subclass changes method).
- **Encapsulation**: use private fields + public getters/setters.
- **Abstraction**: use abstract class or interface.

---

4. Exception Handling

```
try {
   int result = 10/0;
} catch(ArithmeticException e) {
   System.out.println("Cannot divide by zero");
} finally {
   System.out.println("This always runs");
}
```

---

5. Common Java Classes

- **String**:
- String s = "Hello";
- s.length(); // 5
- s.charAt(0); // 'H'
- s.substring(1,4);// "ell"
- s.equals("Hi"); // false
- **ArrayList**:
- ArrayList<Integer> list = new ArrayList<>();
- list.add(10); list.add(20);
- list.remove(0); list.get(0);
- **HashMap**:
- HashMap<String,Integer> map = new HashMap<>();
- map.put("A",1);
- map.get("A"); // 1

---

6. Threads

```
class MyThread extends Thread {
   public void run() { System.out.println("Running"); }
}
MyThread t = new MyThread();
t.start();
```

---

7. Java 8+ Features

- **Lambda expressions**:
- (a,b) -> a+b
- **Streams**:
- List<Integer> nums = List.of(1,2,3);
- nums.stream().filter(n -> n%2==0).forEach(System.out::println);
- **Optional**: avoids null pointer exceptions.
