Java Notes:

1)Inheritance:

- Inheritance means on class inherits all the properties and methods of another class.
- It provide code reusability.

2)Polymorphism:

- Polymorphism means many forms.means ability to take many forms.
- There are two Types of polymorphism.

I)Compile Time Polymorphism(method overloading):

- Same method name but different parameter list
- It occurs Within the class.
- May or may not use inheritance

II)Runtime Time Polymorphism(method overriding):

- Same method name and same parameters but different classes (Parent-child relationship)
- Method declaration should be same in both Parent and Child class.
- Changing the implementation of parent class method into child class.

3)Abstract class:

- It consist both abstract methods(method without body) and non-abstract methods.
- You can't create object of Abstract class.
- It provides reusability.

4)Abstract Method:

- Abstract method means Method Without body only its name.

- Child class must override it(given its own code).

5)Interface:

- Its Like a Blueprint.
- Interface in java is a collection of abstract class.
- Support Multiple Inheritance

6)Constructor:

- When object is created it call automatically.
- Class name same as constructor name.
- No return type
- Automatically called

I)Default Constructor:

- No parameter
- Provided automatically by java if no constructor is written.

II)Parameterized Constructor:

- It is used to pass values.
- Accept argument to set initial values.

III)Copy Constructor:

- Manually Created by java to copy values from one object to another.

7)Call by value:

- Java madhe jevha tumhi function la variable deta tevha tyachi copy jate cannot change original value.

8)Call by reference:

- Java does not support call by ref.but object pass pass kelyavr reference sarakh work hot.
- Variable cha actual reference(address)method la dila jato Tyamule original value change hou shkte.

9)Generic :

- Asa code jo kontya pn data type sobat kam kru shkto.
- A way to write reusale code that works with different data types.

EX:

ArrayList<String>list=new Array List< >();//ethe string ha generic type ahe.

10)Wrapper class:

- Primitive data type la object madhe convert class.
  For ex:
  int→Integer
  double→Double

11)Collection :

- It is a framework.
- Java madhe data store ani manage karayla ready-made structure.
- Collections = object cha group
- Collection is Reusable, flexible, ani easy to use

Major Interfaces:

1. Collection → parent interface of List, Set, Queue

2. Map → Key-Value pair collection (Collection interface extend kart nahi).ha collection interface cha part nahi tari pn to collection framework madhe yeto.

I) List:

- Duplicate allowed
- Insertion order maintain(element jya order ne add kele tech order madhe store hotat)
- Index based access(0,1,2…..index vaprun access krta yet)
- Examples: ArrayList, LinkedList, Vector

- Example:

```
ArrayList<String> list = new ArrayList<>();

list.add("Apple");

list.add("Banana");

list.add("Apple"); // duplicate allowed
```

II) Set:

- Duplicate NOT allowed(Unique element collection).
- Order maintain nahi hot HashSet madhe / Sorted hava asel tr TreeSet use krto
- Fast searching sathi useful
- Examples: HashSet, LinkedHashSet, TreeSet
- Example:

```
HashSet<String> set = new HashSet<>();

set.add("Apple");

set.add("Banana");

set.add("Apple"); // ignored, duplicate
```

III) Queue:

- FIFO (First In First Out)
- Jo element pahil add hoto ,toch element pahila remove hoto.
- Insert element→magun(rear end).
- Remove element→pudhun(front end)
- Examples: PriorityQueue, LinkedList (as Queue), ArrayDeque
- Example:

```
Queue<String> q = new LinkedList<>();

q.add("Apple");

q.add("Banana");
```

```
System.out.println(q.remove()); // Apple
```

- **Map (Collection Framework pan Collection interface nahi)**
    - Key–Value pair madhye data store
    - Key unique(duplicate key not allowed),
    - Value duplicate allowed
    - Mostly fast access / search sathi use karto
    - Examples: HashMap, LinkedHashMap, TreeMap
    - Example:

```
HashMap<Integer, String> map = new HashMap<>();

map.put(1, "Siddhi");

map.put(2, "Yogita");

map.put(3, "Siddhi"); // value duplicate allowed
```