Object Class in Java

What is `Object` Class?

- The `Object` class is the root (superclass) of all Java classes.
- Every class in Java implicitly or explicitly inherits from `Object`.
- Defined in the `java.lang` package.
- If no `extends` keyword is used in a class, it automatically extends the `Object` class.

Why `Object` Class is Important?

- It provides basic methods that every Java object has.
- These methods can be used or overridden by derived classes.
- Examples include: `toString()`, `equals()`, `hashCode()`, `clone()`, etc.

`Object` Class Methods

S.No	Method Name	Description
1	`public String toString()`	**Converts object to String**
2	`public boolean equals(Object obj)`	**Compares two objects**
3	`public int hashCode()`	**Returns object's hash code**
4	`protected Object clone()`	**Creates and returns a copy**
5	`public final Class<?> getClass()`	**Returns runtime class info**

6	`protected void finalize()`	Cleanup before object destruction
7	`public final void wait()`	Pauses current thread (no timeout)
8	`public final void wait(long timeout)`	Waits with timeout
9	`public final void wait(long timeout, int nanos)`	High-precision wait
10	`public final void notify()`	Wakes up one waiting thread
11	`public final void notifyAll()`	Wakes up all waiting threads

🔷 1. `toString()`

Purpose:

Returns a string representing the object.

Default:

```
ClassName@HexHashCode
```

Syntax:

```
public String toString()
```

Example:

```
class Student {
    int id = 1;
    String name = "Ram";
}
Student s = new Student();
System.out.println(s.toString()); // Student@15db9742
```

Override to get readable output:

```
public String toString() {
    return id + " " + name;
```

🔷 2. `equals(Object obj)`

Purpose:

Compares objects. Default: checks references (==). Override to compare content.

Syntax:

```
public boolean equals(Object obj)
```

Example:

```
String s1 = new String("A");
String s2 = new String("A");

System.out.println(s1.equals(s2)); // true
System.out.println(s1 == s2);      // false
```

Override Example:

```
public boolean equals(Object obj) {
    Student s = (Student)obj;
    return id == s.id && name.equals(s.name);
}
```

🔷 3. `hashCode()`

Purpose:

Returns integer hash code, used in `HashMap`, `HashSet`.

Syntax:

```
public int hashCode()
```

Rule:

If `equals()` is overridden, then `hashCode()` must also be overridden.

Example:

```
System.out.println("A".hashCode()); // same hash for same content
```

Override:

```
public int hashCode() {
    return id;
}
```

🔷 4. `clone()`

Purpose:

Creates a copy (shallow) of the object.

Conditions:

- Class must implement `Cloneable`.
- Override `clone()` with `public`.

Syntax:

```
protected Object clone() throws CloneNotSupportedException
```

Example:

```
class Student implements Cloneable {
    int id = 10;
    public Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}
```

🔷 5. `getClass()`

Purpose:

Gives runtime class info. Useful in Reflection API.

Syntax:

```
public final Class<?> getClass()
```

Example:

```
Student s = new Student();
System.out.println(s.getClass().getName()); // Stud
```

🔷 6. `finalize()`

Purpose:

Called by Garbage Collector before destroying object.

Syntax:

```
protected void finalize() throws Throwable
```

Example:

```
protected void finalize() {
    System.out.println("Object is garbage collected");
}
```

Note: `finalize()` is deprecated in Java 9+

🔷 7. `wait()`

Purpose:

Pauses the current thread until another thread calls `notify()`.

Syntax:

```
public final void wait() throws InterruptedException
```

Use:

Must be called inside `synchronized` block.

Example:

```
synchronized(obj) {
    obj.wait();
}
```

🔷 8. `wait(long timeout)`

Purpose:

Waits for a specified timeout in milliseconds.

Syntax:

```
public final void wait(long timeout) throws InterruptedException
```

🔷 9. `wait(long timeout, int nanos)`

Purpose:

Waits with millisecond and nanosecond precision.

Syntax:

```
public final void wait(long timeout, int nanos) throws InterruptedException
```

🔷 10. `notify()`

Purpose:

Wakes up one thread waiting on the object.

Syntax:

```
public final void notify()
```

Example:

```
synchronized(obj) {
    obj.notify();
```

🔷 11. `notifyAll()`

Purpose:

Wakes up all threads waiting on the object.

Syntax:

```
public final void notifyAll()
```