**1. What is the Object class?**

* Object is the **root class** of the Java class hierarchy.
* Every Java class **implicitly extends** java.lang.Object.
* It provides essential methods like:
  + toString()
  + equals(Object obj)
  + hashCode()
  + clone()
  + wait(), notify(), notifyAll()
  + getClass()

**2. Difference between wait() and sleep()**

| **Feature** | **wait()** | **sleep()** |
| --- | --- | --- |
| Belongs to | Object class | Thread class |
| Lock | Releases the lock | Does **not** release the lock |
| Wake-up | Needs notify()/notifyAll() | Wakes up after specified time |
| Used for | Inter-thread communication | Pausing execution |
| Exception | Throws InterruptedException | Throws InterruptedException |

**3. Class-level lock vs Object-level lock**

| **Lock Type** | **Description** |
| --- | --- |
| **Object-level lock** | Used when a thread locks a non-static synchronized method or block. Lock is **on the instance**. |
| **Class-level lock** | Used when a thread locks a static synchronized method or block. Lock is **on the class’s Class object**. |

🔒 Example:

java

CopyEdit

public synchronized void instanceMethod() { } // object-level lock

public static synchronized void staticMethod() { } // class-level lock

**🎯 Design Patterns**

**4. Factory Design Pattern**

✅ **Implementation:**

java

CopyEdit

interface Shape {

void draw();

}

class Circle implements Shape {

public void draw() { System.out.println("Circle drawn"); }

}

class Rectangle implements Shape {

public void draw() { System.out.println("Rectangle drawn"); }

}

class ShapeFactory {

public Shape getShape(String type) {

if (type.equalsIgnoreCase("CIRCLE")) return new Circle();

else if (type.equalsIgnoreCase("RECTANGLE")) return new Rectangle();

return null;

}

}

**Usage:**

java

CopyEdit

ShapeFactory factory = new ShapeFactory();

Shape shape = factory.getShape("CIRCLE");

shape.draw();

**5. Advantages of Factory Pattern**

* Loose coupling: client code doesn't need to know concrete classes.
* Adds abstraction to object creation.
* Easier to maintain and extend.
* Follows **Open/Closed Principle**.

**6. Thread-safe Singleton Implementation**

✅ Best option in modern Java:

java

CopyEdit

public class Singleton {

private static volatile Singleton instance;

private Singleton() {}

public static Singleton getInstance() {

if (instance == null) {

synchronized(Singleton.class) {

if (instance == null)

instance = new Singleton();

}

}

return instance;

}

}

* Uses **Double-Checked Locking**.
* Thread-safe and avoids unnecessary synchronization.

**7. Create Immutable Employee Class**

java

CopyEdit

final class Employee {

private final String name;

private final List<Project> projects;

public Employee(String name, List<Project> projects) {

this.name = name;

this.projects = new ArrayList<>(projects); // defensive copy

}

public String getName() {

return name;

}

public List<Project> getProjects() {

return new ArrayList<>(projects); // return copy

}

}

🛡️ **Rules for Immutability:**

* Make class final
* Make fields private final
* No setters
* Defensive copy of mutable fields

**📊 Data Structures**

**8. Time complexity of put() and get() in a HashMap**

| **Operation** | **Average Case** | **Worst Case** |
| --- | --- | --- |
| put() | **O(1)** | O(n) |
| get() | **O(1)** | O(n) |

**Why Worst Case O(n)?**

* Due to **hash collisions**: when multiple keys map to the same bucket.
* Java 8+ uses **balanced trees** (O(log n)) instead of linked lists if collisions grow.