Certainly! Here’s a list of the **most common Core Java interview questions** that you can expect across various levels (from easier to tougher ones):

**1. Basics of Java**

**1.1 What are JDK, JRE, and JVM?**

* **JDK (Java Development Kit):** A full-featured development environment that includes JRE and tools like compilers.
* **JRE (Java Runtime Environment):** It provides libraries, Java Virtual Machine (JVM), and other components to run Java programs.
* **JVM (Java Virtual Machine):** Executes Java bytecode and provides platform independence.

**1.2 What is the difference between "==" and ".equals()" in Java?**

* **"=="** checks reference equality (whether two references point to the same object in memory).
* **".equals()"** checks for logical equality (whether the content of two objects is the same).

**1.3 What are the main features of Java?**

* **Platform independence** (WORA - Write Once, Run Anywhere)
* **Object-oriented programming (OOP)**
* **Automatic memory management (Garbage collection)**
* **Multithreading support**
* **Robust and secure**

**2. Object-Oriented Programming (OOP) Concepts**

**2.1 What is inheritance in Java?**

* Inheritance is the mechanism where one class acquires the properties and behavior (methods) of another class using the extends keyword.
  + Example: class Dog extends Animal {}

**2.2 What is polymorphism in Java?**

* **Method Overloading:** Same method name, but different method signatures (parameters).
* **Method Overriding:** A subclass provides a specific implementation of a method already defined in its superclass.

**2.3 What is encapsulation in Java?**

* Encapsulation is the concept of wrapping data (variables) and methods that operate on the data into a single unit (class). It helps in hiding the internal state and only exposing necessary operations (getter and setter methods).
  + Example:
  + class Person {
  + private String name;
  + private int age;
  + public String getName() { return name; }
  + public void setName(String name) { this.name = name; }
  + }

**2.4 What is abstraction in Java?**

* Abstraction is the process of hiding the implementation details and showing only the functionality to the user. This can be achieved using abstract classes or interfaces.
  + Example: Abstract class or interface with abstract methods.

**3. Java Memory Management**

**3.1 What is garbage collection in Java?**

* Garbage collection is the automatic process of reclaiming memory by deleting objects that are no longer referenced by the program. Java uses an automatic garbage collector (GC) to manage memory.

**3.2 What is the difference between the heap and the stack?**

* **Heap:** Used for storing objects and instance variables. Managed by the garbage collector.
* **Stack:** Stores method calls, local variables, and control flow. Memory is automatically managed when methods are invoked and returned.

**3.3 What is the role of finalize() method in Java?**

* finalize() is called by the garbage collector before an object is destroyed, allowing for cleanup (closing files, releasing resources).

**4. Exception Handling**

**4.1 What are the types of exceptions in Java?**

* **Checked exceptions:** Must be caught or declared in the method signature (e.g., IOException).
* **Unchecked exceptions:** Subclasses of RuntimeException, do not need to be explicitly caught (e.g., NullPointerException).

**4.2 What is the difference between throw and throws in Java?**

* **throw** is used to explicitly throw an exception from a method or block.
* **throws** is used in the method signature to declare that a method can throw an exception.

**4.3 What is the purpose of the finally block?**

* The finally block is used for cleanup operations and is executed whether or not an exception is thrown. It is typically used to close resources like files, database connections, etc.

**5. Java Collections Framework**

**5.1 What is the difference between ArrayList and LinkedList?**

* **ArrayList:** Implements the List interface with a dynamically resizing array. It's fast for random access but slower for insertions/deletions.
* **LinkedList:** Implements the List interface with a doubly linked list. It is slower for random access but faster for insertions and deletions.

**5.2 What is the difference between HashMap and TreeMap?**

* **HashMap:** Stores key-value pairs, unordered, allows null keys and values, faster for searching.
* **TreeMap:** Stores key-value pairs, ordered (sorted by keys), does not allow null keys, slower for searching compared to HashMap.

**5.3 What is the difference between a Set and a List?**

* **Set:** A collection that does not allow duplicate elements. Example: HashSet, TreeSet.
* **List:** An ordered collection that allows duplicate elements. Example: ArrayList, LinkedList.

**5.4 What is the difference between HashSet and TreeSet?**

* **HashSet:** Does not maintain any order of elements, faster for operations like add(), remove().
* **TreeSet:** Maintains elements in sorted order, slower for operations compared to HashSet.

**6. Threads and Concurrency**

**6.1 What is the difference between a Thread and a Runnable?**

* **Thread:** A class that represents a thread of execution.
* **Runnable:** An interface designed to represent a task that can be executed by a thread. It allows a thread to be passed around.

**6.2 How do you create a thread in Java?**

* **Extending the Thread class**:
* class MyThread extends Thread {
* public void run() {
* // Code to execute
* }
* }
* **Implementing the Runnable interface**:
* class MyRunnable implements Runnable {
* public void run() {
* // Code to execute
* }
* }

**6.3 What is synchronization in Java?**

* Synchronization ensures that only one thread can access a resource (method or block of code) at a time, preventing data inconsistency.

**6.4 What is a deadlock in Java?**

* A deadlock occurs when two or more threads are blocked forever, each waiting for the other to release resources, causing a standstill.

**7. Java 8 Features**

**7.1 What are Lambda expressions in Java?**

* Lambda expressions allow writing concise, functional-style code. They provide a way to pass behavior as parameters.
  + Example:
  + (a, b) -> a + b;

**7.2 What is the Stream API in Java?**

* The **Stream API** allows functional-style operations on streams of elements, such as filtering, mapping, and reducing. It supports operations like map(), filter(), collect(), and forEach().

**7.3 What are the default methods in Java interfaces?**

* Default methods in interfaces allow you to define methods in interfaces with a default implementation. This feature was introduced in Java 8.
* interface MyInterface {
* default void show() {
* System.out.println("Default method in interface");
* }
* }

**8. Miscellaneous**

**8.1 What is the difference between String, StringBuilder, and StringBuffer?**

* **String:** Immutable, every modification creates a new object.
* **StringBuilder:** Mutable, used in single-threaded environments for efficient string manipulation.
* **StringBuffer:** Similar to StringBuilder but thread-safe (synchronized), suitable for multi-threaded environments.

**8.2 What is method overloading and method overriding?**

* **Method Overloading:** Same method name, but different parameter types or number of parameters in the same class.
* **Method Overriding:** Same method signature in both parent and subclass, used for dynamic method dispatch.

**8.3 What are the access modifiers in Java?**

* **private:** Accessible only within the same class.
* **default:** Accessible only within the same package.
* **protected:** Accessible within the same package and subclasses.
* **public:** Accessible from anywhere.

These questions cover a broad range of Core Java concepts, and preparing them thoroughly will help you perform well in interviews. You can practice writing code for some of these questions and understand how different Java features work in real-world applications.

Let me know if you need more examples or deeper insights into any specific topic!