**Java HR Questions and Answers**

**Basic Java Concepts**

1. **What is Java? Why is it platform-independent?**  
   **Answer:** Java is a high-level, object-oriented programming language. It is platform-independent because Java bytecode can run on any system that has the JVM installed.
2. **What is the difference between JDK, JRE, and JVM?**  
   **Answer:**
   * **JDK (Java Development Kit):** Includes tools for developing Java programs, such as a compiler, debugger, and libraries.
   * **JRE (Java Runtime Environment):** Provides runtime libraries to execute Java programs.
   * **JVM (Java Virtual Machine):** Executes the Java bytecode.
3. **What are the features of Java?**  
   **Answer:** Object-oriented, platform-independent, secure, robust, multithreaded, high performance, distributed, and dynamic.
4. **What is bytecode in Java?**  
   **Answer:** Bytecode is the intermediate code generated after compiling Java source code. It is executed by the JVM.
5. **What are the principles of Object-Oriented Programming (OOP)?**  
   **Answer:** Abstraction, Encapsulation, Inheritance, and Polymorphism.

**Object-Oriented Programming**

1. **What is the difference between an abstract class and an interface?**  
   **Answer:**
   * Abstract class: Can have both abstract and concrete methods. Supports single inheritance.
   * Interface: Only has abstract methods (up to Java 7) or default and static methods (Java 8+). Supports multiple inheritance.
2. **What is encapsulation?**  
   **Answer:** Encapsulation is the wrapping of data (variables) and methods in a single unit (class). It is achieved using access modifiers.
3. **What is polymorphism?**  
   **Answer:** Polymorphism allows methods to perform different tasks based on the object. It is achieved through method overloading and method overriding.
4. **What is inheritance in Java?**  
   **Answer:** Inheritance is a mechanism where one class acquires the properties and behaviours of another class using the extends keyword.
5. **What is the difference between method overloading and method overriding?**  
   **Answer:**
   * **Method Overloading:** Same method name, different parameter lists (compile-time polymorphism).
   * **Method Overriding:** Subclass provides a specific implementation of a method already defined in the parent class (runtime polymorphism).

**Java Syntax and Concepts**

1. **What is the purpose of the static keyword?**  
   **Answer:** The static keyword makes methods or variables belong to the class rather than a specific instance of the class.
2. **What is a constructor in Java?**  
   **Answer:** A constructor is a special method used to initialize objects. It has the same name as the class and no return type.
3. **What is the difference between == and .equals()?**  
   **Answer:**
   * == checks reference equality (memory address).
   * .equals() checks logical equality (content).
4. **What is the purpose of the final keyword?**  
   **Answer:**
   * final variable: Cannot be reassigned.
   * final method: Cannot be overridden.
   * final class: Cannot be subclassed.
5. **What is a String in Java?**  
   **Answer:** A String is a sequence of characters. It is immutable in Java.

**Memory Management**

1. **What is garbage collection in Java?**  
   **Answer:** Automatic memory management in Java that removes unused objects from memory to free up space.
2. **What are the differences between stack and heap memory?**  
   **Answer:**
   * **Stack:** Stores local variables and function calls.
   * **Heap:** Stores objects and class-level variables.
3. **What is the finalize() method?**  
   **Answer:** It is called by the garbage collector before removing an object from memory. It can be overridden for cleanup operations.
4. **What are StringBuilder and StringBuffer? How are they different from String?**  
   **Answer:**
   * StringBuilder and StringBuffer are mutable, unlike String.
   * StringBuffer is thread-safe, whereas StringBuilder is faster but not thread-safe.
5. **What is a memory leak in Java?**  
   **Answer:** A memory leak occurs when objects are no longer used but are still referenced, preventing the garbage collector from freeing memory.

**Exception Handling**

1. **What is an exception in Java?**  
   **Answer:** An exception is an event that disrupts the normal flow of the program.
2. **What is the difference between checked and unchecked exceptions?**  
   **Answer:**
   * **Checked exceptions:** Must be handled or declared in the method (e.g., IOException).
   * **Unchecked exceptions:** Occur at runtime (e.g., ArithmeticException).
3. **What is the try-catch-finally block?**  
   **Answer:** Used for handling exceptions.
   * try: Code that might throw exceptions.
   * catch: Code to handle exceptions.
   * finally: Executes regardless of exceptions.
4. **What is the difference between throw and throws?**  
   **Answer:**
   * throw: Used to explicitly throw an exception.
   * throws: Declares exceptions a method might throw.
5. **What is a custom exception in Java?**  
   **Answer:** A user-defined exception by extending the Exception class.

**Multithreading**

1. **What is multithreading in Java?**  
   **Answer:** Multithreading allows multiple threads to run concurrently within a program.
2. **How do you create a thread in Java?**  
   **Answer:**
   * By extending the Thread class.
   * By implementing the Runnable interface.
3. **What is the purpose of the synchronized keyword?**  
   **Answer:** To prevent thread interference by allowing only one thread to access a critical section at a time.
4. **What is the difference between a process and a thread?**  
   **Answer:**
   * **Process:** An independent executing program.
   * **Thread:** A lightweight subprocess within a process.
5. **What is a daemon thread?**  
   **Answer:** A background thread that provides services to user threads. JVM exits when all user threads are completed.

**Collections Framework**

1. **What is the Java Collections Framework?**  
   **Answer:** A set of classes and interfaces for storing and manipulating groups of data.
2. **What is the difference between ArrayList and LinkedList?**  
   **Answer:**
   * ArrayList: Fast for random access, slow for insertion/deletion.
   * LinkedList: Efficient for insertion/deletion, slower for random access.
3. **What is a HashMap?**  
   **Answer:** A collection that stores key-value pairs and allows fast retrieval based on the key.
4. **What is the difference between HashSet and TreeSet?**  
   **Answer:**
   * HashSet: Unordered, uses hashing.
   * TreeSet: Ordered, uses a Red-Black tree for sorting.
5. **What is a Comparator and a Comparable?**  
   **Answer:**
   * Comparable: Used for natural ordering of objects.
   * Comparator: Used for custom ordering.

**Advanced Core Java Questions**

1. **What is the difference between Iterator and ListIterator?**  
   **Answer:**
   * Iterator: Can traverse collections in one direction only (forward).
   * ListIterator: Can traverse lists in both directions (forward and backward).
2. **What is the difference between HashMap and Hashtable?**  
   **Answer:**
   * HashMap: Not synchronized, allows one null key.
   * Hashtable: Synchronized, doesn’t allow null keys or values.
3. **What is the difference between Set and List?**  
   **Answer:**
   * Set: Unordered, unique elements.
   * List: Ordered, allows duplicate elements.
4. **What is the difference between Array and ArrayList?**  
   **Answer:**
   * **Array**: Fixed size, cannot grow or shrink.
   * **ArrayList**: Dynamic size, part of the Collections framework.
5. **What is the difference between Queue and Deque?**  
   **Answer:**
   * **Queue**: Follows the FIFO (First In, First Out) principle.
   * **Deque**: Allows insertion and removal from both ends.

**String Handling**

1. **Why is String immutable in Java?**  
   **Answer:** To ensure security, caching, and synchronization.
2. **What is the difference between String, StringBuilder, and StringBuffer?**  
   **Answer:**
   * **String**: Immutable.
   * **StringBuilder**: Mutable, not thread-safe.
   * **StringBuffer**: Mutable, thread-safe.
3. **How is memory managed in a String?**  
   **Answer:** Strings are stored in the **String Pool**, a special memory area in the heap.
4. **What is the String.intern() method?**  
   **Answer:** It adds the string to the String Pool and returns the reference from the pool if it already exists.
5. **What are the different ways to create a String in Java?**  
   **Answer:**
   * Using double quotes (e.g., String str = "Java";)
   * Using the new keyword (e.g., String str = new String("Java");)

**Multithreading and Concurrency**

1. **What is the life cycle of a thread in Java?**  
   **Answer:** New → Runnable → Running → Blocked/Waiting → Terminated.
2. **What is the difference between Runnable and Callable?**  
   **Answer:**
   * Runnable: Doesn’t return a result.
   * Callable: Returns a result and can throw checked exceptions.
3. **What is a thread pool?**  
   **Answer:** A pool of reusable threads to execute tasks, managed by the ExecutorService.
4. **What is the difference between wait() and sleep()?**  
   **Answer:**
   * wait(): Releases the lock, used for inter-thread communication.
   * sleep(): Pauses execution without releasing the lock.
5. **What are the different thread states in Java?**  
   **Answer:** NEW, RUNNABLE, BLOCKED, WAITING, TIMED\_WAITING, TERMINATED.

**Input/Output (I/O)**

1. **What are the main classes in the Java I/O package?**  
   **Answer:**
   * Byte Stream Classes: InputStream, OutputStream.
   * Character Stream Classes: Reader, Writer.
2. **What is the difference between FileReader and BufferedReader?**  
   **Answer:**
   * FileReader: Reads data directly from the file.
   * BufferedReader: Reads data in chunks, more efficient.
3. **What is serialization in Java?**  
   **Answer:** Converting an object into a byte stream to save or transmit it.
4. **What is the purpose of the transient keyword?**  
   **Answer:** Prevents serialization of a variable.
5. **What is NIO in Java?**  
   **Answer:** Non-blocking I/O introduced in Java for high-performance file handling and networking.

**Java Features and Enhancements**

1. **What are the new features introduced in Java 8?**  
   **Answer:**
   * Lambda expressions
   * Stream API
   * Optional class
   * Default and static methods in interfaces
   * Date and Time API.
2. **What are functional interfaces?**  
   **Answer:** Interfaces with exactly one abstract method (e.g., Runnable, Callable).
3. **What is a lambda expression in Java?**  
   **Answer:** A concise way to represent a function using ->.
4. **What is the Stream API in Java?**  
   **Answer:** A functional programming model for processing collections of data.
5. **What is the purpose of the Optional class in Java?**  
   **Answer:** To handle null values and avoid NullPointerException.

**Exception Handling**

1. **What are the types of exceptions in Java?**  
   **Answer:**
   * Checked Exceptions: Compile-time (e.g., IOException).
   * Unchecked Exceptions: Runtime (e.g., NullPointerException).
   * Errors: Serious issues (e.g., OutOfMemoryError).
2. **What is a try-with-resources statement?**  
   **Answer:** Ensures that resources (e.g., files) are closed automatically after use.
3. **What is the finally block?**  
   **Answer:** It always executes after try and catch, used for cleanup.
4. **Can a finally block execute without a catch block?**  
   **Answer:** Yes, try and finally can be used without a catch.
5. **What happens if an exception is not handled in Java?**  
   **Answer:** The program terminates abruptly, and a stack trace is printed.

**Memory Management**

1. **What is the purpose of the volatile keyword?**  
   **Answer:** Ensures visibility of changes to variables across threads.
2. **What is the difference between SoftReference and WeakReference?**  
   **Answer:**
   * **SoftReference:** Cleared only when memory is low.
   * **WeakReference:** Cleared at the next garbage collection cycle.
3. **What is the role of the garbage collector?**  
   **Answer:** To reclaim memory from objects no longer referenced.
4. **What is the difference between shallow copy and deep copy?**  
   **Answer:**
   * **Shallow Copy:** Copies object references.
   * **Deep Copy:** Copies the entire object along with its references.
5. **What is a StackOverflowError?**  
   **Answer:** An error when the call stack exceeds its limit, often caused by infinite recursion.

**Collections and Data Structures**

1. **What is the difference between TreeSet and HashSet?**  
   **Answer:**
   * TreeSet: Maintains elements in a sorted order.
   * HashSet: Does not guarantee order.
2. **What is the ConcurrentHashMap?**  
   **Answer:** A thread-safe version of HashMap with better concurrency.
3. **How does HashMap handle collisions?**  
   **Answer:** By using linked lists or balanced trees.
4. **What is the LinkedHashMap?**  
   **Answer:** Maintains the insertion order of elements.
5. **What is the difference between Deque and Stack?**  
   **Answer:**
   * Deque: Can be used as both stack and queue.
   * Stack: LIFO (Last In, First Out) structure.

**Generics**

1. **What are generics in Java?**  
   **Answer:** Generics allow you to write code that can work with different types while maintaining type safety during compile time (e.g., List<String>).
2. **What are the benefits of using generics?**  
   **Answer:**
   * Type safety
   * Code reusability
   * Compile-time checking
3. **What is type erasure in generics?**  
   **Answer:** Type information is removed during runtime, meaning generic types are replaced by their raw types.
4. **What is a bounded type parameter in generics?**  
   **Answer:** Restricts the type to a specific range.  
   Example: <T extends Number> allows only Number and its subclasses.
5. **Can we use primitives with generics?**  
   **Answer:** No, generics work only with objects. Use wrapper classes like Integer for primitives.

**Annotations**

1. **What are annotations in Java?**  
   **Answer:** Metadata added to classes, methods, fields, or other elements for additional information. Example: @Override, @Deprecated.
2. **What is the @Override annotation?**  
   **Answer:** Indicates that a method overrides a method in its superclass.
3. **What is the purpose of @FunctionalInterface?**  
   **Answer:** Ensures that an interface has exactly one abstract method.
4. **What is a marker annotation?**  
   **Answer:** An annotation with no methods (e.g., @Deprecated).
5. **Can we create custom annotations in Java?**  
   **Answer:** Yes, using the @interface keyword.

**Java 8 Features**

1. **What are default methods in interfaces?**  
   **Answer:** Methods in interfaces with a default implementation. They allow adding new methods to interfaces without breaking existing code.
2. **What is the purpose of the Stream API?**  
   **Answer:** To process collections in a functional programming style with operations like map(), filter(), and reduce().
3. **What is a functional interface?**  
   **Answer:** An interface with exactly one abstract method, e.g., Runnable, Callable.
4. **What are method references in Java 8?**  
   **Answer:** A shorthand notation for calling a method using ClassName::methodName.
5. **What is the purpose of Optional?**  
   **Answer:** To avoid null checks and handle values that may or may not be present.

**Reflection**

1. **What is reflection in Java?**  
   **Answer:** The ability to inspect and manipulate classes, methods, and fields at runtime.
2. **What is the purpose of the Class class in reflection?**  
   **Answer:** Provides methods to obtain metadata about a class and its objects.
3. **How can you create an object using reflection?**  
   **Answer:** By using the Class.newInstance() or Constructor.newInstance() methods.
4. **What is the role of the Method class in reflection?**  
   **Answer:** Represents a method of a class, allowing invocation at runtime.
5. **What are the disadvantages of using reflection?**  
   **Answer:**
   * Performance overhead
   * Lack of compile-time checking
   * Security risks

**Java Virtual Machine (JVM)**

1. **What is the role of the JVM?**  
   **Answer:** The JVM executes Java bytecode and provides platform independence.
2. **What are the main components of the JVM?**  
   **Answer:**
   * Class Loader: Loads class files.
   * Bytecode Verifier: Validates bytecode.
   * Interpreter: Executes bytecode.
   * Just-In-Time (JIT) Compiler: Optimizes performance by compiling bytecode into native code.
3. **What is the difference between the heap and method area in JVM?**  
   **Answer:**
   * **Heap:** Stores objects and instance variables.
   * **Method Area:** Stores class-level data, such as static variables and method bytecode.
4. **What is the Just-In-Time (JIT) compiler?**  
   **Answer:** A component of the JVM that improves performance by compiling bytecode into native machine code during runtime.
5. **What are class loaders in Java?**  
   **Answer:** Class loaders are part of the JVM that load classes into memory. Types:  
   - Bootstrap ClassLoader  
   - Extension ClassLoader  
   - Application ClassLoader