Great! These are **very common interview questions** when discussing Future and CompletableFuture. Let's go one by one with **clear answers + small examples**.

**1. What is the difference between Future and CompletableFuture?**

| **Feature** | **Future** | **CompletableFuture** |
| --- | --- | --- |
| **Asynchronous** | Yes (but limited) | Yes (powerful, flexible) |
| **Non-blocking** | ❌ Mostly blocking (e.g., get()) | ✅ Full non-blocking chaining |
| **Chaining** | ❌ Not possible | ✅ .thenApply(), .thenCompose(), etc. |
| **Manual Completion** | ❌ | ✅ Can complete manually (complete()) |
| **Combining futures** | ❌ Not possible | ✅ thenCombine(), allOf() |
| **Exception handling** | Limited | Rich exception handling via .exceptionally(), .handle() |

✅ **Example:**

CompletableFuture.supplyAsync(() -> "Hello")

.thenApply(msg -> msg + " World")

.thenAccept(System.out::println); // Prints "Hello World"

**2. How do you handle exceptions in CompletableFuture?**

**✅ .exceptionally() - handle exception & return fallback**

CompletableFuture.supplyAsync(() -> {

if (true) throw new RuntimeException("Failed");

return "Success";

}).exceptionally(ex -> {

System.out.println("Caught: " + ex.getMessage());

return "Fallback";

});

**✅ .handle() - handle success + exception**

cf.handle((result, ex) -> {

if (ex != null) return "Error";

else return result.toUpperCase();

});

**3. How do you chain multiple asynchronous operations?**

Use thenApply() or thenCompose().

**✅ thenApply() – Transform result (Sync)**

CompletableFuture.supplyAsync(() -> "Java")

.thenApply(str -> str + " Developer");

**✅ thenCompose() – FlatMap for async chaining**

CompletableFuture.supplyAsync(() -> "userId")

.thenCompose(id -> getUserDetailsAsync(id)); // returns another CompletableFuture

**4. What are thenApply(), thenAccept(), and thenRun()?**

| **Method** | **Purpose** | **Accepts Input?** | **Returns Output?** |
| --- | --- | --- | --- |
| thenApply() | Transform the result | ✅ Yes | ✅ Yes |
| thenAccept() | Consume result, no return | ✅ Yes | ❌ No |
| thenRun() | Run something after result, no input | ❌ No | ❌ No |

**✅ Example:**

CompletableFuture.supplyAsync(() -> "Java")

.thenApply(name -> name + " Dev") // returns String

.thenAccept(System.out::println) // consumes

.thenRun(() -> System.out.println("Done")); // final runnable

**5. How do you combine multiple futures in Java?**

**✅ Using thenCombine() – Combine results of two futures**

CompletableFuture<String> f1 = CompletableFuture.supplyAsync(() -> "Hello");

CompletableFuture<String> f2 = CompletableFuture.supplyAsync(() -> "World");

CompletableFuture<String> combined = f1.thenCombine(f2, (s1, s2) -> s1 + " " + s2);

**✅ Using allOf() – Wait for multiple futures (returns Void)**

CompletableFuture<Void> all = CompletableFuture.allOf(f1, f2);

all.join(); // Waits for both to complete

**✅ Using anyOf() – Return result of fastest**

CompletableFuture<Object> any = CompletableFuture.anyOf(f1, f2);

Would you like to practice 3–4 **interview-style Q&A** with real coding examples now?