Future vs CompletableFuture

Future

1. A Future is used as a reference to the result of an asynchronous computation. It provides an isDone() method to check whether the computation is done or not, and a get() method to retrieve the result of the computation when it is done.

CompletableFuture

1. CompletableFuture is used for asynchronous programming in Java. Asynchronous programming is a means of writing non-blocking code by running a task on a separate thread than the main application thread and notifying the main thread about its progress, completion or failure.
2. This way, your main thread does not block/wait for the completion of the task and it can execute other tasks in parallel.
3. Having this kind of parallelism greatly improves the performance of your programs.

Limitation of Future

1. It cannot be manually completed :

Let's say that you've written a function to fetch the latest price of an e-commerce product from a remote API. Since this API call is time-consuming, you're running it in a separate thread and returning a Future from your function. Now, let's say that If the remote API service is down, then you want to complete the Future manually by the last cached price of the product. Can you do this with Future? No!

2. You cannot perform further action on a Future's result without blocking

   Future does not notify you of its completion. It provides a get() method which blocks until the result is available.

   You don't have the ability to attach a callback function to the Future and have it get called automatically when the Future's result is available.

3. Multiple Futures cannot be chained together :

   Sometimes you need to execute a long-running computation and when the computation is done, you need to send its result to another long-running computation, and so on. You can not create such asynchronous workflow with Futures.

4. You can not combine multiple Futures together

   Let's say that you have 10 different Futures that you want to run in parallel and then run some function after all of them completes. You can't do this as well with Future.

5. No Exception Handling :

Future API does not have any exception handling construct.

Summary all the topics learnt in the 5 days

Day 1.

1. Maven
    a. Local
    b. Central
    c. Remote
2. Git

Day2.

1. Data type
    a. Primitive type
    b. Wrapper class
    c. Autoboxing &Unboxing
2. String/ StringBuilder/ StringBuffer
3. String constant pool
4. Integer constant pool
5. Equals / hashcode
6. Collections
7. Comparator vs Comparable
8. JVM
    a. Class loader
    b. Runtime data area
    c. Execution Engine
    d. Class loader
    e. Garbage collector
9.

Day3.

1. Final
2. Immutable class
    a. Final class
    b. Private final fields
    c. No Setter
    d. Return deep copy of the collections for getter
3. Static
4. OOP
5. Different exceptions
6. Handle exception
7. Generics

<? extends E>
<? super T>
<T extends E>

8. IO stream
9. Serialization & deserialization

Day 4.

1. Java 8
    a. Lambda
    b. Functional Interface
    c. Optional (OrElse to print error messages)
    d. Stream API

Day 5.

1. Thread
    a. States of Thread
    b. How to create thread
    c. Customize the thread pool
    d. In-built thread pool (Not popular in industry)
2. Lock
    a. How to implement a lock.
    b. Different usages of “Synchronized”.
    c. Lock interfaces.