Programação_Concorrente-Studies/ Aula13: CompletableFuture, Classes e API's extras para MultiThread

- Essa Aula é apenas um extra para entendermos bem por alto o uso de algumas classes e curiosidades sobre elas no uso da MultiThreading.
- A primeira observação é ser feita é sobre as classes TimerTask e Timer, são classes que não serão citadas aqui porém caso você as encontre em alguns códigos ou exemplos por ai, saiba que elas funcionam de forma similar ao serviço de executores agendados porém atualmente quase não se usa essas classes, pois elas foram substituidas pelos executores.

- A Classe que iremos citar é a classe CompletableFuture:

```java
*/
public class CompletableFuture_1 {

    Run | Debug
    public static void main(String[] args) {
        CompletableFuture<String> processe = processe
        ();

        CompletableFuture<String> thenApply =
                processe.thenApply(s -> s + " Lembre-se
                do porquê você começou!");

        CompletableFuture<Void> thenAccept =
                thenApply.thenAccept(s -> System.out.
                println(s));

        System.out.println(x:"Nunca pare de
        estudar!");

        sleep();
        sleep();
        sleep();
    }
```

```java
private static CompletableFuture<String>
processe() {
  return CompletableFuture.supplyAsync(() -> {
    sleep();
    return "Fim das Aulas de Java Multithread!";
  });
}

private static final void sleep() {
  try {
    Thread.sleep(millis:2000);
  } catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    e.printStackTrace();
  }
}
```

- Obs: Essa classe é bem complexa e vale um estudo detalado por fora
- A primeira coisa que fazemos é declarar um CompletableFuture de String que é recebido de um métodod chamado processe()
- Esse método retorna um CompletableFuture que não é inteiro ele ainda pode receber mudanças, nesse caso usando o método supplyAsync() e após isso ele espera um tempo de 2 segundos definido no método sleep e retorna uma String.
- Logo após instanciar o objeto processe e receber uma String do método processe() eu chamo o método processe.thenApply(“passando mais uma String como parâmetro”), esse método concatena a string que eu retornei pelo método processe() com a String passada por parâmetro.
- E por último por meio do método thenApply.theAccept(“oq será realizado como parâmetro”) eu imprimo o meu resultado final no console.
- Agora o resultado disso é que quando chamamos o método processe() ele não retorna imediatamente a String para ser impressa na tela, ele retorna um completableFuture que é assincrono a execução do meu main, portanto caso eu não coloque os 3 sleeps posteriores ao System.out da main ele para o programa e não continua a execução.
- Logo ele é uma forma de chamar um processo que retorna um resultado de forma assíncrona.

#Concorrent