Practical Usage of Methods of CompletableFuture

# **thenCompose()**

## **ThenCompose1.java**

**package** com.ddlab.rnd.type1;  
  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenCompose1 {  
 **public static** String task1(String inParam1) {  
 **try** {  
 System.***out***.println(**"Task 1 thread started running"**);  
 TimeUnit.***SECONDS***.sleep(5);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Task Thread-1 running ..."**);  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **return "outParam1"**;  
 }  
  
 **public static** String task2(String inParam2) {  
 **try** {  
 System.***out***.println(**"Get value -2 thread started running"**);  
 TimeUnit.***SECONDS***.sleep(2);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Task Thread-2 running ..."**);  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **if** (inParam2 == **null**) **throw new** NullPointerException(**"Input parameter is null or blank ..."**);  
 **return "outParam2"**;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<String> cf1 = CompletableFuture.*supplyAsync*(() -> *task1*("Input1"));** **CompletableFuture<String> cf2 = CompletableFuture.*supplyAsync*(() -> *task2*("Input2"));**  
 **CompletableFuture<String> cf3 = cf1.thenCompose(s -> cf2);** System.***out***.println(**cf3.get()**);//**outParam2**  
 }  
}

**OUPUT**

Task 1 thread started running

Get value -2 thread started running

Task Thread-2 running ...

Task Thread-2 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-2 running ...

Task Thread-1 running ...

Task Thread-1 running ...

Task Thread-1 running ...

outParam2

## **ThenCompose2.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenCompose2 {  
  
 **public static** String getUserName(String userId) {  
 **try** {  
 System.***out***.println(**"User Name thread started running & user Id-"** + userId);  
 TimeUnit.***SECONDS***.sleep(5);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"User Name running ..."**);  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **return "John Abraham"**;  
 }  
  
 **public static** String getUserRating(String userName) {  
 **try** {  
 System.***out***.println(**"User Rating thread started running & user Name-"** + userName);  
 TimeUnit.***SECONDS***.sleep(1);  
 **for** (**int** i = 0; i < 5; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"User Rating thread running ..."**);  
 }  
  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **if** (userName == **null**) **throw new** NullPointerException(**"Input parameter is null or blank ..."**);  
 **return "5 plus"**;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<String> cf1 = CompletableFuture.*supplyAsync*(() -> *getUserName*("ID-101"));** **CompletableFuture<String> cf3 =  
 cf1.thenCompose(resultCF1 -> CompletableFuture.*supplyAsync*(() -> *getUserRating*(resultCF1)));** System.***out***.println(cf3.get());*//5 plus* }  
}

**OUTPUT**

User Name thread started running & user Id-ID-101

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Name running ...

User Rating thread started running & user Name-John Abraham

User Rating thread running ...

User Rating thread running ...

User Rating thread running ...

User Rating thread running ...

User Rating thread running ...

5 plus

## **ThenCompose3.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenCompose3 {  
  
 **private static void** sleepNRun(String threadName, **int** time) {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(time);  
 **for** (**int** i = 0; i < 5; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Thread "** + threadName + **" is running"**);  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 **public static** CompletableFuture<String> computeSomething() {  
 **return** CompletableFuture.*supplyAsync*(  
 () -> {  
 *sleepNRun*(**"computeSomething"**, 5);  
 **return "test"**;  
 });  
 }  
  
 **public static** CompletableFuture<Integer> computeInteger() {  
 **return** CompletableFuture.*supplyAsync*(  
 () -> {  
 *sleepNRun*(**"computeInteger"**, 3);  
 **return** 42;  
 });  
 }  
  
 **public static** CompletableFuture<Boolean> computeBoolean() {  
 **return** CompletableFuture.*supplyAsync*(  
 () -> {  
 *sleepNRun*(**"computeBoolean"**, 2);  
 **return false**;  
 });  
 }  
  
 **public static** CompletableFuture<Boolean> computeBoolean(Integer i) {  
 **return** CompletableFuture.*supplyAsync*(  
 () -> {  
 *sleepNRun*(**"computeBoolean-Integer"**, 5);  
 **return** i > 100000;  
 });  
 }  
  
 **public static** CompletableFuture<Integer> computeInteger(**int** i) {  
 **return** CompletableFuture.*supplyAsync*(  
 () -> {  
 *sleepNRun*(**"computeInteger-Inti"**, 4);  
 **return** 42 + i;  
 });  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<Integer> runResult =  
 *computeSomething*()  
 .thenCompose(  
 s -> {  
 return *computeInteger*();  
 });** System.***out***.println(**"Now runResult :::"** + runResult.get());  
 **CompletableFuture<Boolean> booleanResult =  
 *computeSomething*()  
 .thenCompose(s -> *computeInteger*())  
 .thenCompose(i -> *computeInteger*(i))  
 .thenCompose(b -> *computeBoolean*(b));** System.***out***.println(**"booleanResult :::"** + booleanResult.get());  
 }  
}

OUTPUT

Thread computeSomething is running

Thread computeSomething is running

Thread computeSomething is running

Thread computeSomething is running

Thread computeSomething is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger is running

Now runResult :::42

Thread computeSomething is running

Thread computeSomething is running

Thread computeSomething is running

Thread computeSomething is running

Thread computeSomething is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger is running

Thread computeInteger-Inti is running

Thread computeInteger-Inti is running

Thread computeInteger-Inti is running

Thread computeInteger-Inti is running

Thread computeInteger-Inti is running

Thread computeBoolean-Integer is running

Thread computeBoolean-Integer is running

Thread computeBoolean-Integer is running

Thread computeBoolean-Integer is running

Thread computeBoolean-Integer is running

booleanResult :::false

# **thenCombine()**

## **ThenCombine1.java**

**package** com.ddlab.rnd.type1;  
  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenCombine1 {  
 **public static** String task1(String inParam1) {  
 **try** {  
 System.***out***.println(**"Task 1 thread started running"**);  
 TimeUnit.***SECONDS***.sleep(5);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Task Thread-1 running ..."**);  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **return "outParam1"**;  
 }  
  
 **public static** String task2(String inParam2) {  
 **try** {  
 System.***out***.println(**"Get value -2 thread started running"**);  
 TimeUnit.***SECONDS***.sleep(2);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Thread-2 running ..."**);  
 }  
  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **if** (inParam2 == **null**) **throw new** NullPointerException(**"Input parameter is null or blank ..."**);  
 **return "outParam2"**;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<String> cf1 = CompletableFuture.*supplyAsync*(() -> *task1*("Input1"));**

**CompletableFuture<String> cf2 = CompletableFuture.*supplyAsync*(() -> *task2*("Input2"));**

CompletableFuture<String> cf3 =  
 cf1.thenCombine(  
 cf2,  
 (resultCF1, resultCF2) -> {  
 **return "First CF1 result ->"** + resultCF1  
 + **" || "** + **"Second CF2 result ->"** + resultCF2;  
 });  
 *// Here output will be  
 // First CF1 result ->outParam1 || Second CF2 result ->outParam2* System.***out***.println(**"Now combined result : "** + cf3.get());  
 }  
}

OUTPUT

Task 1 thread started running

Get value -2 thread started running

Thread-2 running ...

Thread-2 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Thread-2 running ...

Task Thread-1 running ...

Task Thread-1 running ...

Task Thread-1 running ...

Now combined result : First CF1 result ->outParam1 || Second CF2 result ->outParam2

# **thenApply()**

## **ThenApply1.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.ExecutionException;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenApply1 {  
 **public static** String getInfo1() {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(5);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return "info-1"**;  
 }  
  
 **public static** CompletableFuture<String> task1() {  
 **CompletableFuture<String> cf =  
 CompletableFuture.*supplyAsync*(  
 () -> {  
 return *getInfo1*();  
 });  
 return cf;** }  
  
 **public static void** main(String[] args) {  
 **CompletableFuture<String> cf1 = *task1*();** **CompletableFuture<String> cf2 =  
 cf1.thenApply(  
 value -> {  
 System.*out*.println("Actual Response : " + value);  
 return "Chain-1" + "-" + value;  
 })  
 .thenApply(  
 val1 -> {  
 System.*out*.println("Now New Value : " + val1);  
 return "Chain-2" + "-" + val1;  
 })  
 .thenApply(  
 val2 -> {  
 System.*out*.println("New Value in chain 3: " + val2);  
 return "Chain-3-" + val2;  
 });**  
 String finalValue = **null**;  
 **try** {  
 finalValue = cf2.get();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(**"Now final value :::"** + finalValue);  
 }  
}

OUTPUT

Actual Response : info-1

Now New Value : Chain-1-info-1

New Value in chain 3: Chain-2-Chain-1-info-1

Now final value :::Chain-3-Chain-2-Chain-1-info-1

## **ThenApplyWithException1.java**

**package** com.ddlab.rnd.type1;  
  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.ExecutionException;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenApplyWithException1 {  
 **public static** String getInfo1() {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(5);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return "info-1"**;  
 }  
  
 **public static** CompletableFuture<String> task1() {  
 **CompletableFuture<String> cf =  
 CompletableFuture.*supplyAsync*(  
 () -> {  
 return *getInfo1*();  
 });  
 return cf;** }  
  
 **public static void** main(String[] args) {  
 **CompletableFuture<String> cf1 = *task1*();**

**CompletableFuture<String> chainCf =  
 cf1.thenApply(  
 returnValue1 -> {  
 System.*out*.println("First value : " + returnValue1);  
 return "Chain-1";  
 })  
 .thenApply(  
 returnValue2 -> {  
 String str1 = returnValue2;  
 System.*out*.println("Chain-1 return value : " + returnValue2);  
 str1 = null;  
 if (str1 == null)  
 throw new NullPointerException("An unknown exception thrown...");  
 return "Chain-2";  
 })  
 .thenApply(  
 returnValue3 -> {  
 System.*out*.println("Chain-3 response : " + returnValue3);  
 return "Chain-3";  
 })  
 .exceptionally(  
 someValue -> {  
 System.*out*.println("someValue = " + someValue);  
 return "Default Fallback value";  
 });** **try** {  
 String finalResponse = chainCf.get();  
 System.***out***.println(**"Final Response : "** + finalResponse);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

**OUTPUT**

First value : info-1

Chain-1 return value : Chain-1

someValue = java.util.concurrent.CompletionException: java.lang.NullPointerException: An unknown exception thrown...

Final Response : Default Fallback value

# **thenAccept()**

## **ThenAccept1.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** ThenAccept1 {  
 **public static** String getActualWork(String param1) {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(5);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return** param1 == **null** ? **"work-1"** : param1;  
 }  
  
 **public static** CompletableFuture<String> task1() {  
 **CompletableFuture<String> cf =  
 CompletableFuture.*supplyAsync*(  
 () -> {  
 return *getActualWork*(null);  
 });** **return** cf;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 StringBuilder result = **new** StringBuilder();  
 **CompletableFuture<Void> cf =  
 CompletableFuture.*completedFuture*("thenAcceptAsync message")  
 .thenAcceptAsync(s -> result.append(s));  
 cf.join();** System.***out***.println(**"Result :::"** + result);  
  
 **CompletableFuture<String> cf1 = *task1*();  
 cf1.thenAccept(value -> System.*out*.println("New Value : " + value));** System.***out***.println(**"Now Response : "** + cf1.get());  
 }  
}

**OUTPUT**

Result :::thenAcceptAsync message

New Value : work-1

Now Response : work-1

# **exceptionally()**

## **Exceptionally1.java**

*// Usage of exceptionally***package** com.ddlab.rnd.type1;  
  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** Exceptionally1 {  
  
 **public static** String getValue1(String inParam1) {  
 **try** {  
 System.***out***.println(**"Get value -1 thread started running"**);  
 TimeUnit.***SECONDS***.sleep(5);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Thread-1 running ..."**);  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **return "outParam1"**;  
 }  
  
 **public static** String getValue2(String inParam2) {  
 **try** {  
 System.***out***.println(**"Get value -2 thread started running"**);  
 TimeUnit.***SECONDS***.sleep(2);  
 **for** (**int** i = 0; i < 10; i++) {  
 TimeUnit.***SECONDS***.sleep(1);  
 System.***out***.println(**"Thread-2 running ..."**);  
 }  
  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 **if** (inParam2 == **null**) **throw new** NullPointerException(**"Input parameter is null or blank ..."**);  
 **return "outParam2"**;  
 }

**public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<String> cf1 = CompletableFuture.*supplyAsync*(() -> *getValue1*("Input1"));  
 CompletableFuture<String> cf2 = CompletableFuture.*supplyAsync*(() -> *getValue2*(null));  
 CompletableFuture<String> fallback1 = cf1.exceptionally(x -> "default outParam1");  
 CompletableFuture<String> fallback2 = cf2.exceptionally(x -> "default outParam2");  
 System.*out*.println("Fallback 1 :::" + fallback1.get());*//outParam1***System.***out***.println(**"Fallback 2 :::"** + fallback2.get());*//default outParam2* }  
}

OUTPUT

Get value -1 thread started running

Get value -2 thread started running

Thread-2 running ...

Thread-2 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-2 running ...

Thread-1 running ...

Thread-1 running ...

Thread-1 running ...

Fallback 1 :::outParam1

Fallback 2 :::default outParam2

# **handle()**

## **HandleError1.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** HandleError1 {  
 **private static void** sleep(**int** seconds) {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(seconds);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public static** String performValidation(String someValue) {  
 System.***out***.println(**"Started validation ..."**);  
 *sleep*(5);  
 **if** (someValue == **null**)  
 **throw new** IllegalArgumentException(**"Parameter cannot be null or empty ..."**);  
 **return "Your value is "** + someValue;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<String> cf1 =  
 CompletableFuture.*supplyAsync*(() -> *performValidation*(null))  
 .handle(  
 (response, error) -> {  
 System.*out*.println("Actual Response : " + response);  
 if (error != null) {  
 System.*out*.println("Actual Error : " + error.getMessage());  
 return "Since there is an error, I am sending default value.";  
 }  
 return response;  
 });** System.***out***.println(**"Now result : "** + cf1.get());  
 }  
}

OUTPUT

Started validation ...

Actual Response : null

Actual Error : java.lang.IllegalArgumentException: Parameter cannot be null or empty ...

Now result : Since there is an error, I am sending default value.

# **allOf()**

## **AllOf1.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** AllOf1 {  
  
 **private static void** sleep(**int** seconds) {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(seconds);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public static boolean** validateAadharNo(String aadharNo) {  
 System.***out***.println(**"Started Aadhar validation ..."**);  
 *sleep*(7);  
 **return** aadharNo == **null** ? **false** : **true**;  
 }  
  
 **public static boolean** validatePanNo(String panNo) {  
 System.***out***.println(**"Started Pan No validation ..."**);  
 *sleep*(5);  
 **return** panNo == **null** ? **false** : **true**;  
 }  
  
 **public static boolean** validatePassportNo(String passportNo) {  
 System.***out***.println(**"Started Passport validation ..."**);  
 *sleep*(5);  
 **return** passportNo == **null** ? **false** : **true**;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<Boolean> cf1 = CompletableFuture.*supplyAsync*(() -> *validateAadharNo*("12345"));**

**CompletableFuture<Boolean> cf2 = CompletableFuture.*supplyAsync*(() -> *validatePanNo*("12345"));**

**CompletableFuture<Boolean> cf3 =  
 CompletableFuture.*supplyAsync*(() -> *validatePassportNo*("12345"));  
  
 CompletableFuture<Void> cf4 = CompletableFuture.*allOf*(cf1, cf2, cf3);**

**cf4.get();**

System.***out***.println(**"Aadhar Validation : "** + cf1.get());  
 System.***out***.println(**"Pan No Validation : "** + cf2.get());  
 System.***out***.println(**"Passport Validation : "** + cf3.get());  
 }  
}

**OUTPUT**

Started Aadhar validation ...

Started Pan No validation ...

Started Passport validation ...

Aadhar Validation : true

Pan No Validation : true

Passport Validation : true

# **anyOf()**

## **AnyOf.java**

**package** com.ddlab.rnd.type1;  
**import** java.util.concurrent.CompletableFuture;  
**import** java.util.concurrent.TimeUnit;  
  
**public class** AnyOf {  
 **private static void** sleep(**int** seconds) {  
 **try** {  
 TimeUnit.***SECONDS***.sleep(seconds);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public static** String validateAadharNo(String aadharNo) {  
 System.***out***.println(**"Started Aadhar validation ..."**);  
 *sleep*(7);  
 **return "Your Aadhar No AA-12345 is valid"**;  
 }  
  
 **public static** String validatePanNo(String panNo) {  
 System.***out***.println(**"Started Pan No validation ..."**);  
 *sleep*(5);  
 **return "Your Pan No PN-1277 is valid"**;  
 }  
  
 **public static** String validatePassportNo(String passportNo) {  
 System.***out***.println(**"Started Passport validation ..."**);  
 *sleep*(5);  
 **return "Your Passport No PP-99 is valid"**;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 **CompletableFuture<String> cf1 = CompletableFuture.*supplyAsync*(() -> *validateAadharNo*("12345"));**

**CompletableFuture<String> cf2 = CompletableFuture.*supplyAsync*(() -> *validatePanNo*("12345"));**

**CompletableFuture<String> cf3 =  
 CompletableFuture.*supplyAsync*(() -> *validatePassportNo*("12345"));  
  
 CompletableFuture<Object> cf4 = CompletableFuture.*anyOf*(cf1, cf2, cf3);  
 cf4.thenAccept(s -> System.*out*.println("Intermediate value = " + s));** **Object obj = cf4.get();**  
 System.***out***.println(**"Now result : "** + obj.toString());  
 }  
}

OUTPUT

Started Aadhar validation ...

Started Pan No validation ...

Started Passport validation ...

Intermediate value = Your Passport No PP-99 is valid

Now result : Your Passport No PP-99 is valid