Група ТВ-22

П.І.Б. Іщук П.О.

Варіант за списком групи 5

**Практична робота №5**

**Завдання** 1

Створіть два асинхронних завдання. Перше завдання отримує дані з бази даних (імітуємо затримку), а друге завдання залежить від результату першого і обробляє отримані дані.

**Завдання** 2

Користувач хоче порівняти погоду в трьох різних містах. Потрібно отримати температуру, вологість та швидкість вітру для кожного міста паралельно, а потім порівняти ці дані. Дати висновки про те, де зараз можна сходити на пляж, а де варто вдягнутись тепліше.

**Код програми**

import java.util.concurrent.\*;  
  
public class Task1 {  
  
 public static void main(String[] args) throws Exception {  
 // Task 1: Database Fetch and Processing  
 CompletableFuture<String> fetchFromDatabase = CompletableFuture.*supplyAsync*(() -> {  
 *simulateDelay*(2000);  
 return "Data from database";  
 });  
  
 CompletableFuture<String> processData = fetchFromDatabase.thenCompose(data -> CompletableFuture.*supplyAsync*(() -> {  
 *simulateDelay*(1000);  
 return "Processed: " + data;  
 }));  
  
 System.*out*.println(processData.get());  
  
 // Task 2: Weather Comparison in Three Cities  
 CompletableFuture<CityWeather> city1Weather = *getWeatherAsync*("City1");  
 CompletableFuture<CityWeather> city2Weather = *getWeatherAsync*("City2");  
 CompletableFuture<CityWeather> city3Weather = *getWeatherAsync*("City3");  
  
 CompletableFuture<Void> allWeatherFetched = CompletableFuture.*allOf*(city1Weather, city2Weather, city3Weather);  
  
 allWeatherFetched.thenRun(() -> {  
 try {  
 CityWeather weather1 = city1Weather.get();  
 CityWeather weather2 = city2Weather.get();  
 CityWeather weather3 = city3Weather.get();  
  
 System.*out*.println("City1 Weather: " + weather1);  
 System.*out*.println("City2 Weather: " + weather2);  
 System.*out*.println("City3 Weather: " + weather3);  
  
 String bestCityForBeach = *compareWeatherForBeach*(weather1, weather2, weather3);  
 System.*out*.println("Best city for the beach: " + bestCityForBeach);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }).join();  
  
 CompletableFuture<Object> anyWeatherFetched = CompletableFuture.*anyOf*(city1Weather, city2Weather, city3Weather);  
 anyWeatherFetched.thenAccept(weather -> System.*out*.println("First fetched weather: " + weather));  
 }  
  
 private static CompletableFuture<CityWeather> getWeatherAsync(String city) {  
 return CompletableFuture.*supplyAsync*(() -> {  
 *simulateDelay*((int) (Math.*random*() \* 2000));  
 return new CityWeather(city, (int) (Math.*random*() \* 35), (int) (Math.*random*() \* 100), (int) (Math.*random*() \* 20));  
 });  
 }  
  
 private static void simulateDelay(int millis) {  
 try {  
 Thread.*sleep*(millis);  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 }  
 }  
  
 private static String compareWeatherForBeach(CityWeather... cities) {  
 CityWeather bestCity = null;  
 for (CityWeather city : cities) {  
 if (city.getTemperature() > 25 && city.getHumidity() < 70 && city.getWindSpeed() < 15) {  
 if (bestCity == null || city.getTemperature() > bestCity.getTemperature()) {  
 bestCity = city;  
 }  
 }  
 }  
 return bestCity != null ? bestCity.getCityName() : "No suitable city for the beach";  
 }  
  
 static class CityWeather {  
 private final String cityName;  
 private final int temperature;  
 private final int humidity;  
 private final int windSpeed;  
  
 public CityWeather(String cityName, int temperature, int humidity, int windSpeed) {  
 this.cityName = cityName;  
 this.temperature = temperature;  
 this.humidity = humidity;  
 this.windSpeed = windSpeed;  
 }  
  
 public String getCityName() {  
 return cityName;  
 }  
  
 public int getTemperature() {  
 return temperature;  
 }  
  
 public int getHumidity() {  
 return humidity;  
 }  
  
 public int getWindSpeed() {  
 return windSpeed;  
 }  
  
 @Override  
 public String toString() {  
 return "CityWeather{" +  
 "cityName='" + cityName + '\'' +  
 ", temperature=" + temperature +  
 ", humidity=" + humidity +  
 ", windSpeed=" + windSpeed +  
 '}';  
 }  
 }  
}

**Скриншот результатів роботи**

