# Exercises: Async and Await

Problems for exercises and homework for the "Concurrent Programming" course from the official "Applied Programmer" curriculum.

## Dice

Create a **program** that **simulates** a **dice** **roll**. Create **two** **asynchronous** **methods** DiceRollAsync() and RollAsync().

The **first** **method** **prints** "Dice rolling..." and then **rolls** **asynchronously** **twice** and **returns** the **sum** of the **two** **rolls**.

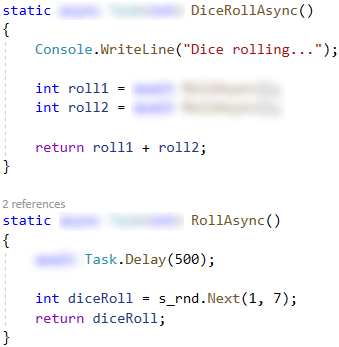
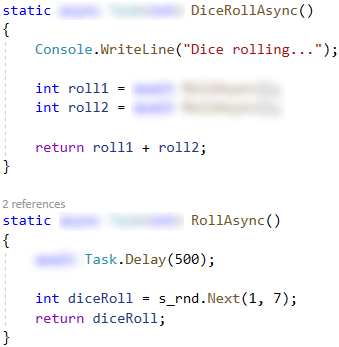
The **second** **method** **sleeps** the **task** for **500** **milliseconds** and actually **rolls** a **die** **once** and **returns** the **result**. The **result** of the dice roll is a **random** **number** from 1 to 7 of the Random **class**.

**Finally**, **print** to the console: "Your dice roll result is {the sum of the two rolls}"

### Examples

|  |  |
| --- | --- |
| **Output** | **Output** |
| Dice rolling...  Your dice roll result is 8 | Dice rolling...  Your dice roll result is 3 |

### Hints

## Day Off

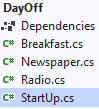
Use the provided **skeleton. Do not change** its methods, classes and namespaces!

You have a **day** **off** and you have a **morning** **routine** on those days.

First **while** **listening** to **music** you **make** **breakfast** and **when** you are **ready** you **eat** **and** **read** the news.



You are **provided** with three **classes**: Newspaper, Radio and Breakfast, **each** of which **has** **fields** and **methods**. **Use** **objects** from the classes and **methods** in StartUp.cs to **implement** this **asynchronous** **morning** **routine**.



### The Breakfast class

There are **two** **properties** bool IsReady and bool IsEaten and by default the bool types are false.

Has **two** **methods**:

* public string Cooked() – changes the IsReady property to **true** and **returns** a string "Breakfast is ready."
* public string Eaten() – changes the IsEaten property to **true** and **returns** a string "Breakfast is eaten."

### The Newspaper class

Has one **property** IsReaded and one method public string Reading() that **changes** the **property** to **true** and **returns** the string "Reading newspaper.".

### The Radio class

Has one **property** IsOn and **two** **methods**:

* public string RadioOn() – **changes** the **property** to **true** and **returns** the string "Music onn."
* public string RadioOff() – **changes** the **property** to **false** and **returns** the string "Music off."

### The StartUp.cs

**Make** the below **methods** **asynchronous** and **follow** the **instructions** on how each **method** should **work**.

The method MakingBreakfastAsync() should:

* **Accept** a Breakfast **object** as a **parameter**
* **Print** "Making breakfast."
* Then it should **sleep** the **task** for **1000** milliseconds
* **Change** the **state** of the **breakfast** **object** to **cooked**
* **Print** the **message** **from** the object’s **method**
* **Finally** **return** the Breakfast **object**

The method ListeningMusicAsync() should:

* **Accept** a Radio **object** as a **parameter**
* **Turn on the radio** and **print** the **message**  from the object’s method
* Then it should **sleep** the **task** for **1000** milliseconds
* **Turn off the radio** and **print** the **message**  from the object’s method
* **Finally** **return** the Radio **object**

The method EatingBreakfastAsync() should:

* **Accept** a Breakfast object as a **parameter**
* **Print** "Eating breakfast."
* Then it should **sleep** the **task** for **1700** milliseconds
* **Change** the **state** of the **breakfast object** to eaten
* **Print** the **message**  **from** the **object’s** **method**
* **Finally** **return** the **breakfast** **object**

The method ReadingNewsAsync() should:

* **Accept** a Newspaper **object** as a **parameter**
* **Read the newspaper** and **print** the **message** from the object’s method
* Then it should **sleep** the **task** for **1700** milliseconds
* **Finally** **return** the Newspaper **object**

In the Main() **method** after the initial message, **instantiate** the **objects** from the classes and **run** the **methods**.

You have to **cook** **breakfast** **and** **listen** to **music** at the **same** **time**.

But **when** you **finish** **preparing** the **breakfast**, you should **start** **eating** the **prepared** **breakfast**, and **when** you **finish**, the **state** of the **breakfast** should be **cooked** and **eaten**.

When you **finish** **listening** to the **music**, you should **turn** **off** the **radio** and **start** **reading** the **newspaper**. When this is done, the **newspaper** **status** should be "**read**".

Since **listening** to **music** and **preparing breakfast** takes the **same** amount of time, the **following** **tasks** will **also** **run** in **parallel**.

At the **end** of the Main() **method**, **print** **each** on a **new** **line**:

* "Breakfast was cooked: {the property IsReady of the breakfast object}"
* "Breakfast was eaten: {the property IsEaten of the breakfast object}"
* "The radio is on: {the property IsOn of the radio object}"
* "The newspaper was read: {the property IsReaded of the newspaper object }"

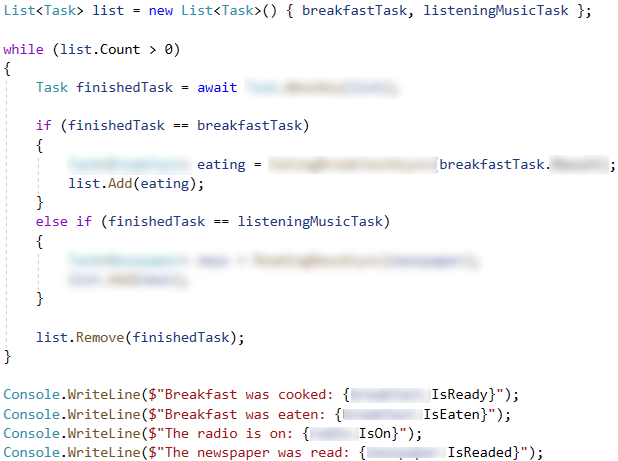
### Examples

It is possible that there may be small discrepancies in the results between different launches of the program. Your result should be similar to the example below.

|  |
| --- |
| **Output** |
| Good morning! Your plans for today are: making breakfast and listening to music, eating breakfast and reading the news.  Making breakfast.  Music onn.  Music off.  Breakfast is ready.  Reading a newspaper.  Eating breakfast.  Breakfast is eaten.  Breakfast was cooked: **True**  Breakfast was eaten: **True**  The radio is on: **False**  The newspaper was read: **True** |

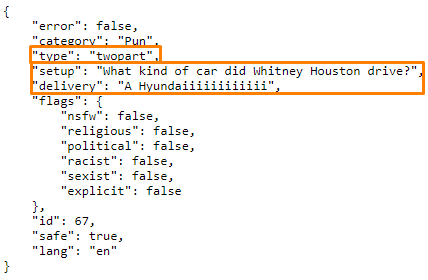
### Hints

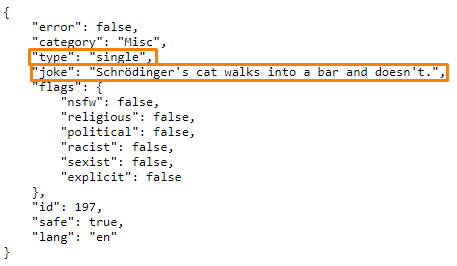
Eating



## Joke

**Read** the **API** **content** of the **address**: https://v2.jokeapi.dev/joke/Any?safe-mode as a **string** asynchronously, **map** it to a AJoke object, and **print** the **content**. The **API** **provides** **random** jokes**,** but they come in **two** **types**. The **first** is when the **joke** **consists** of **two** **parts** in the setup and delivery **properties**, and the **second** is when the **joke** text is on a **single** **line** in the joke **property**.





Create **class** AJoke with properties:

* int Id
* string Category
* string Type
* string Joke
* string Setup
* string Delivery

**Print** the **joke** in the **following** **format**:

"Joke id: {joke.Id}"

"Joke category: {joke.Category}"

**If** the joke is in **two** **parts**, **print**:

"{joke.Setup}"

"{joke.Delivery}"

**If** the joke is in **one** **part**, **print**:

"{joke.Joke}"

### Examples

|  |  |
| --- | --- |
| **Output** | **Output** |
| Joke id: 286  Joke category: Pun  A Roman walks into a bar and raises 2 fingers and says to the bartender...  "Five beers, please." | Joke id: 23  Joke category: Programming  The glass is neither half-full nor half-empty, the glass is twice as big as it needs to be. |

## Download Wikipedia HTML

**Download** **asynchronously** the **HTML** of this **URL**: <https://developer.wikimedia.org>.

### Examples

|  |
| --- |
| **Output** |
| <!doctype html>  <html lang="en" class="no-js">  <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width,initial-scale=1">  <meta name="description" content="Find technical documentation, and connect with Wikimedia developer communities.">  … |