

ChatGPT en el desarrollo de Person

Primer caso	1
Segundo caso	3


Primer caso

Después de implementar la operación de la clase persona, he decidido crear un test genérico de caso correcto con *ChatGPT* y así poder ahorrar un poco de tiempo en inventar personas:

```
J Knowing this is a Java class I have implemented 'public Person(String name, int age, String
gender) {
    if(name.isEmpty() || name.isBlank()){
        throw new BadArgumentsException("Not possible to create an unnamed person");
    }else if(age < 0){
        throw new BadArgumentsException("Not possible to create a negative aged
person");
    }else if(gender.isEmpty() || gender.isBlank()){
        throw new BadArgumentsException("Not possible to create a person without
gender");
    }else if(!gender.equals("Male") && !gender.equals("Female")){
        throw new BadArgumentsException("Not possible to create a person with a gender
other tan 'Male' or 'Female'");
    }

    this.name = name;
    this.age = age;
    this.gender = gender;
}' And knowing I have this operation '/*
 * Computes the average age of male and female persons in a list and returns the result in
 * an array of two elements (the first element is the male mean age and the second one is
the
 * female mean age)
 *
 * @param persons list of people to compute.
 * @return Array of two elements. res[0] = maleMeanAge and res[1] = femaleMeanAge
 */
public double[] averageAgePerGender(List<Person> persons)'. Can you generate me a
test for this method with a list of 2 males and 3 females
```

Esta ha sido la respuesta que he obtenido:

 Sure! Here's a test for the `averageAgePerGender` method using a list of 2 males and 3 females:

```
csharp
Copy code

@Test
public void testAverageAgePerGender() {
    List<Person> persons = new ArrayList<>();
    persons.add(new Person("John", 25, "Male"));
    persons.add(new Person("Michael", 30, "Male"));
    persons.add(new Person("Jane", 28, "Female"));
    persons.add(new Person("Emily", 26, "Female"));
    persons.add(new Person("Sarah", 32, "Female"));

    double[] result = person.averageAgePerGender(persons);

    assertEquals(27.5, result[0], 0.1); // Male average age should be 27.5
    assertEquals(29, result[1], 0.1); // Female average age should be 29
}
```

This test creates a list of 5 `Person` objects, 2 of them are male and 3 are female. Then it calls the `averageAgePerGender` method with the list and asserts that the result is as expected.

Tras implementarlo en la clase y ejecutar los tests me daba error, y después de preguntarle a *ChatGPT* no encontraba la solución, pero en unos pocos minutos he encontrado que el fallo era que la operación no estaba definida como estática, y al ser *person* un objeto nulo por el *@AfterEach*, obtenía un *NullPointerException*.

Por lo tanto, aunque *ChatGPT* no me dijera nada (y el esquema de código del enunciado no lo indicara) he cambiado la operación a estática, ya que de esta manera me parece más correcta.

Entonces al ejecutar el nuevo test, ya no obtenía un *NullPointerException*, pero, de cualquier manera, el test seguía saliendo incorrecto. Pero gracias a la interfaz de *IntelliJ IDEA* encontré el problema casi al instante. El problema se encontraba en que el test que había creado *ChatGPT* era incorrecto, es decir, la media de edad de los hombres estaba mal calculada, y en lugar de 27.5 tenía que ser 28.6. Tras corregir esto el test se ejecutaba de la manera esperada.

En mi experiencia con *ChatGPT*, se le da muy mal realizar operaciones matemáticas que no estén predefinidas, es decir, que las tenga que calcular sobre la marcha, por lo tanto, no me sorprende. Y tal vez, usarlo para generar algún tipo de test más complejo, podría ocasionar una mayor dificultad para encontrar el error.

Segundo caso

Para facilitar el trabajo de documentación, teniendo que escribir varias veces lo mismo, he usado *ChatGPT* para que me generara la documentación de los casos de prueba de la clase persona.

```
J Can you add documentation to this test '@Test
void negativeAgePerson(){
    assertThrows(BadArgumentsException.class, () -> person = new Person("Alice",-20,
"Female"));
}'

Sure! Here's the updated code with documentation:

csharp Copy code

/**
 * Test to verify the handling of negative age in the Person class.
 * Expects a BadArgumentsException to be thrown when a Person is created with negat
 */
@Test
void negativeAgePerson() {
    assertThrows(BadArgumentsException.class, () -> person = new Person("Alice", -20, "Female"));
}
```

```
J and to this test '@Test
void unnamedPerson(){
    assertThrows(BadArgumentsException.class, () -> person = new Person("",68,"Female"));
}'

Sure, here's the updated code with documentation:

csharp Copy code

/**
 * Test to verify the handling of empty name in the Person class.
 * Expects a BadArgumentsException to be thrown when a Person is created with an em
 */
@Test
void unnamedPerson() {
    assertThrows(BadArgumentsException.class, () -> person = new Person("", 68, "Female"));
}
```

Así con prácticamente todas las pruebas.

Además podemos observar cómo prácticamente para todas las pruebas ha seguido el mismo esquema de respuesta y siempre ha acertado en los objetivos de cada uno de los tests, por lo tanto, ha quedado una documentación muy clara.

Aunque, solamente en los casos de prueba para el método, para cada test me generaba un comentario distinto, por lo tanto, he escogido el esquema que mas me ha gustado y lo he repetido para los demás tests.