Singletons

An airplane's software system can only ever have one (and only one) AdminController instance.
To ensure that this is enforced, the **singleton** pattern can be used.

Task 1

Copy the `AdminController` class code into a new project in your IDE. Follow the video and see if you can get the singleton pattern working.

Task 2

Answer the following questions.

1. In the original version of **`AdminController`**, how many objects could be created?

2. Why is the constructor made **`private`** in the Singleton implementation?

3. What is the purpose of the static variable that stores the single instance?

4. Explain what the `getInstance()` method does when:

   a) No object exists yet:

   b) An object already exists:

5. Why must the instance variable be declared static rather than non-static?

6. What problem could occur in the airplane system if multiple `AdminController` objects were allowed to exist?

7. Give one other real-world system where the Singleton pattern would be appropriate. Explain why.

8. How does the Singleton pattern help improve control and safety in software systems?