University of Aveiro
Department of Electronics, Telecommunications and Informatics (DETI)
Object-Oriented Programming (OOP)
Practical Test (04/06/2025); Duration: 1h30

Name: **Nº Mec.**

Important Notes:

1. Create a project on your computer with your mechanographic number.
2. Download the base files for this exam from the submission link in the elearning "Testes (ATP1, ATP2, AP) > Teste Prático 2025 - !! Só Enunciado !!"
3. At the end of the test, access the *elearning again* and **use the submission link** to send the project code by compressing it into a single file with the same name as its mechanographic number.

Management of Concerts of a Band

Write a program that models a band's concert management system. Each concert is represented by a Concert object. The program must use Java Collections to manage the billing of these concerts.

Each Concert must contain the following attributes:

- Unique identifier (int) – must be automatic, incremental starting at one (i.e., 1)
- Duration (double) – unit is minute
- Concert venue (String) – "City, Country" format
- Start date and time (LocalDateTime)
  Additional information:
  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
  LocalDateTime date = LocalDateTime.parse("2025-05-29 16:00", formatter);
  System.out.println(date.format(formatter));

The program should include the following classes and interfaces: **[15% modeling]**

- Concert: is the main class that represents a concert. This class must have the attributes mentioned above, a constructor, toString() function, and the appropriate setters and getters.
- IConcertProfitCalculator: An interface that defines the behavior of objects (i.e., calculators) that allow you to project the profit of each concert. It should contain the following method:
  - calculateConcertProfit(**Concert c**): Returns the estimated profit for concert "c".
  - You can use the IConcertProfitCalculator.java file and make any changes you think are necessary.
- StandardConcertProfitCalculator: **[10%]** A class that implements the IConcertProfitCalculator interface. The cost of this transaction is calculated as follows:
  - Base profit of €1500/h for all concerts.
  - It doubles the profit whenever the concert is outside Portugal.

    - Increases the cost by €800 (fixed) whenever the concert is outside the Iberian Peninsula.
- ConcertManager: A class that manages concert billing. The class must use Java Collections to store and manage the transactions. The class should offer the following methods:
    - addConcert(Concert c): Adds a new concert to the system. **[5%]**
    - removeConcert(int id): Removes a concert using the unique identifier. **[5%]**
    - getConcert(int id): Gets the concert on the system based on the identifier. Returns the concert or null if the concert does not exist. **[5%]**
    - calculateConcertProfit(**int id**): Calculates the profit from making this concert using the StandardConcertProfitCalculator. In case this concert does not exist, it returns -1. **[10%]**
    - printAllConcerts(): Prints the information of all transactions in the system. **[10%]**
    - sortConcertsByProfit(): Prints on the screen all concerts **separated by month of the year**. Within each month, concerts must be printed in ***descending order*** of their profit. **[15%]**
    - readFile(String fich): Imports concert information from a file. **[10%]**
        - Note #1: If you prefer, you can replace this method with a constructor.
        - Note #2: If the file contains a concert with an existing identifier, that concert is replaced with the one in the file.
        - Note #3: Ensure that the automatic, incremental unique identifier remains consistent with the higher-valued imported identifier.
    - writeFile(String fich): Writes the list of concerts in the ConcertManager to a file. The file must have the data separated by ";" and include in each line the following information for each concert: **[15%]**
        - Unique Identifier
        - Duration
        - Concert venue
        - Start date and time
        - Concert profit

Use the ConcertTester.java file to test and demonstrate the use of the ConcertManager class *(no menu needed).* You can edit this file freely, but you should ensure that it continues to test the following features:

- Creates an instance of ConcertManager.
- Adds concerts to the system.
- Removes system fixes.
- Reads from a file the data for several concerts.
    - i.e., the program must be able to read the file classicpimba.txt
- Get a specific concert through its handle.
- Prints the information of all the concerts registered in the system.
- Calculates the profit of a concert.
- Prints on the screen the list of concerts separated by month, sorted by profit (decreasing).
- Writes the list of concerts in ConcertManager to a file.