# Principles of Software Engineering and Data Bases

Davide Yi Xian Hu
Email: davideyi.hu@polimi.it
Date: 19 December 2024
Exercise Lecture: 11 - Software Engineering

POLITECNICO
MILANO 1863

# Exercise 1A - Transportation System

Design a simple object-oriented system to allow a person to reach the airport using different modes of transportation. A person could drive their own car, take a taxi, use an airport shuttle, take a bus, or use a limousine service. In some cases, they might also use the subway or a helicopter. All these modes of transportation could be used by the same person at different times, and all enable the person to reach the airport. The choice will be influenced by cost, time required or available, and the desired level of comfort.

- Define a UML class diagram, utilizing the Strategy Design Pattern
- Write the structure of the main classes in Python.

# Exercise 1B - Concurrent Trains

Write a concurrent Python program (using classes and threads) to manage a train station with a number of tracks (nTracks) and a maximum number of trains (nTrains). A train can enter the station only if at least one track is free, and it will leave the station after a variable amount of time. For simplicity, each train will occupy the first available track starting from the first.

Additionally, model the case where the station has 5 tracks and 20 trains, all trying to enter the station in parallel. Each train will wait for a random amount of time before attempting to occupy a track. Note that random.random() returns a value between 0 and 1.

# Exercise 1C - Testing

Consider the following Python code snippet:

```
def foo(x, y):
    z = 0
    if (x > 0) and (y > 0):
        z = x
    return z
```

Draw the control flow diagram.
Identify a minimal set of tests to cover:

- all statements
- all decisions (branches)
- all conditions

What would change if the expression `(x > 0) and (y > 0)` were replaced by `(z > 0) and (y > 0)`?

# Exercise 2A - Sandwiches

A sandwich shop wants to offer its customers maximum flexibility in creating sandwiches. Each sandwich starts with a base, which is the type of bread: white, pita, or multigrain. Customers can then add ingredients such as Prosciutto, Cooked Ham, Salami, or Tuna, followed by vegetables like Tomato, Mozzarella, Lettuce, Iceberg Salad, or Capers. The shop also wants a system that can be easily extended in the future to add new ingredients.

Design a UML class diagram to "solve" the problem described above. The use of appropriate design patterns will be evaluated positively. Write the structure of the main classes in Python.

# Exercise 2B - Wellness Center

Consider a wellness center where the pool water has therapeutic properties. As a result, the center is frequented both by regular visitors and individuals with health conditions (patients). Guests access the pool one at a time, stay for an arbitrary amount of time, and then exit. The pool has a maximum capacity MAXP, which specifies the maximum number of people it can accommodate. The pool rules prohibit the simultaneous presence of visitors and patients.

Develop a concurrent Python application that represents patients and visitors as concurrent threads. The application must implement a synchronization policy that satisfies the given constraints and also prioritizes patients over visitors when accessing the pool. Create a main program to simulate the scenario with at least 50 visitors, 20 patients, and a maximum capacity of 60 guests.

# Exercise 2C - Testing

Consider the following Java code snippet:

```
def foo(a, d1, d2):
   if d1 > 0 and d2 ⩽ d1:
       for i in range(d2 + 1):
           if a[i] > 0:
               print(a[i])
           if a[i] < 0:
               print(-a[i])
   print("End")
```

Where **a** is an array of integers, **d1** is the length of the array, and **d2** is a positive integer.

1. Draw the control flow diagram.
2. Identify a minimal set of test cases to cover all statements, all decisions (branches), and all conditions.
3. What (likely) errors are revealed by the test set defined in the previous points?

# Exercise 3A - Functional

Consider the following classes:

```
class Element:
    def __init__(self, name: str, size: int):
        self.name = name
        self.size = size
    def get_name(self):
        return self.name
    def get_size(self):
        return self.size

class Group:
    def __init__(self, members: List[Element]):
        self.members = members
    def get_members(self):
        return self.members
```

Implement the following functions using functional programming constructs:
The function **print_type**(group) must print the average size of the elements in the group g. If the group is empty, the method should not print anything.
The function **names_in_groups**(group) returns a list of the names of all elements within the groups in gs that have a size greater than 10.

# Exercise 3B - State Notification

Design a class diagram to manage the notification modes of your smartphone. The proposed solution, leveraging an appropriate design pattern, must handle the following scenarios: if the phone is in silent mode, the notification should appear on the screen without any additional action. If the phone is in airplane mode, the notification should not be displayed at all. If the phone is in outdoor mode, the notification should trigger an appropriate sound signal. If the phone is configured for noisy environments, the notification should also make the phone's light flash.

# Exercise 3C - Liskov Substitution

Explain briefly, preferably through an example (not previously discussed in class or in other exam topics), the **Liskov Substitution Principle**, highlighting its strengths.

# Exercise 3D - Concurrent Robots

An assembly line includes four robots: two robots produce parts of type A and B, while the other two use or pick up these parts. The two producing robots generate parts of type A or B randomly, and the two robots picking up the parts take the first available piece. The conveyor belt can hold no more than three parts (of any type). Implement the concurrent system in Python using basic synchronization mechanisms without relying on external concurrency libraries. Simulate the operation of the system with the four robots and the conveyor belt.

# Exercise 4A - Concurrent Parking

Design a simple object-oriented system to allow a person to reach the airport using different modes of transportation. A person could drive their own car, take a taxi, use an airport shuttle, take a bus, or use a limousine service. In some cases, they might also use the subway or a helicopter. All these modes of transportation could be used by the same person at different times, and all enable the person to reach the airport. The choice will be influenced by cost, time required or available, and the desired level of comfort.

- Define a UML class diagram, utilizing the Strategy Design Pattern
- Write the structure of the main classes in Python.

# Exercise 4B - Testing

Consider the following Java code snippet:

```
def method(x):
    if x < 3:
        return
    m = x - 2
    while x > 0:
        x = x % m
        if m == 1 or x >= 0:
            x = x - 1
    return
```

Define the following:

- The control flow diagram for the method.
- A set of test cases that cover all instructions.
- A set of test cases that exercise all decisions (branches).
- A test case that executes the while loop only once.