Question 6: Producer/Consumer + Adapter pattern

IT-SDJ2-A21

Software Engineering

VIA University College

Jordi Lazo

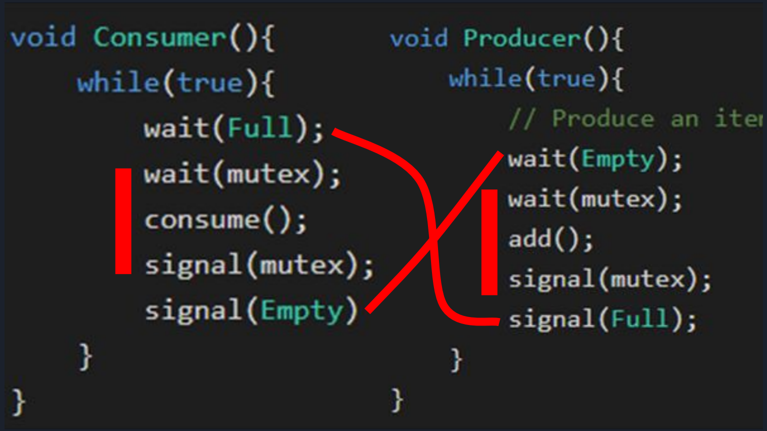
Producer/Consumer

Producer-Consumer problem

- ❖ There is a buffer of N slots and each slot is capable of storing one unit of data.
- ❖ There are two processes running, i.e. Producer and Consumer, which are currently operated in the buffer.
- ❖ There are certain restrictions/conditions for both the producer and consumer process, so that data synchronization can be done without interruption. These are as follows:
  - ➢ The producer tries to insert data into an empty slot of the buffer.
  - ➢ The consumer tries to remove data from a filled slot in the buffer.
  - ➢ The producer must not insert data when the buffer is full.
  - ➢ The consumer must not remove data when the buffer is empty.
  - ➢ The producer and consumer should not insert and remove data simultaneously.

Problem analysis

- semaphore Full: Tracks the space filled by the Producer process. It is initialized with a value of 0 as the buffer will have 0 filled spaces at the beginning.
- semaphore Empty: Tracks the empty space in the buffer. It is initially set to buffer_size as the whole buffer is empty at the beginning.
- semaphore mutex: Used for mutual exclusion so that only one process can access the shared buffer at a time.
-



```
void Consumer(){
    while(true){
        wait(Full);
        wait(mutex);
        consume();
        signal(mutex);
        signal(Empty)
    }
}
```

```
void Producer(){
    while(true){
        // Produce an ite
        wait(Empty);
        wait(mutex);
        add();
        signal(mutex);
        signal(Full);
    }
}
```

Java example

Adapter pattern

What is the purpose?

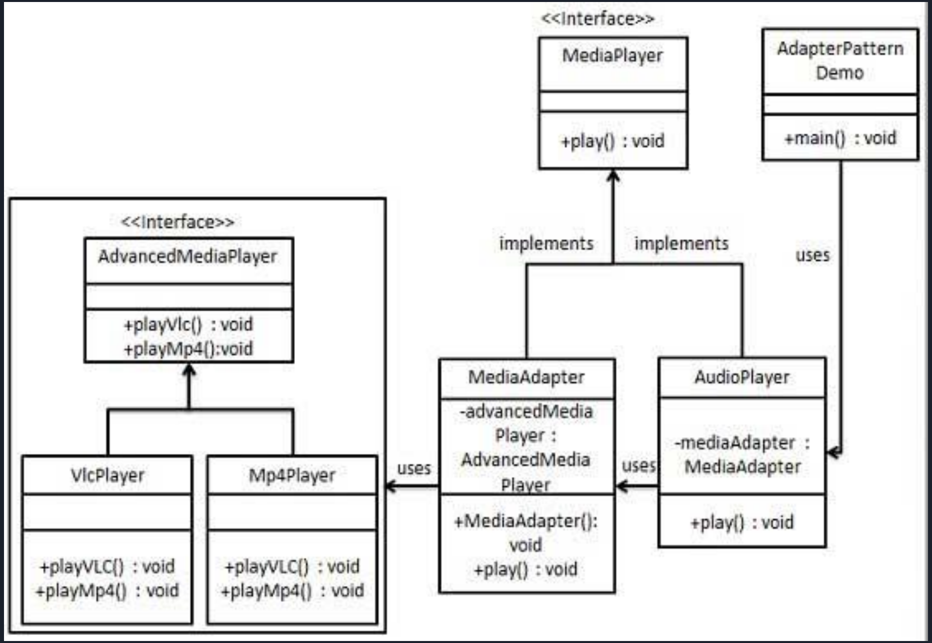
- Adapter pattern converts the interface of a class into another interface that a client want.
- It provides the interface according to client requirement while using the services of a class with a different interface.
- It is used:
  - When an object needs to utilize an existing class with an incompatible interface.
  - When you want to create a reusable class that cooperates with classes which don't have compatible interfaces.
  - When you want to create a reusable class that cooperates with classes which don't have compatible interfaces.
- It allows two or more previously incompatible objects to interact.
- It allows reusability of existing functionality.

What are the different parts involved?

- This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces.
- Target Interface: This is the desired interface class which will be used by the clients.
- Adapter class: This class is a wrapper class which implements the desired target interface and modifies the specific request available from the Adaptee class.
- Adaptee class: This is the class which is used by the Adapter class to reuse the existing functionality and modify them for desired use.
- Client: This class will interact with the Adapter class.

UML + Java example

❖



Java example