Design Patterns and Software Development Process

Final Project

This final assignment is composed of 3 exercises. The notions seen and presented during the course are expected to be employed, whenever helpful. No external libraries or tools can be used for solution development. Only .NET Microsoft's native packages and objects can be used.

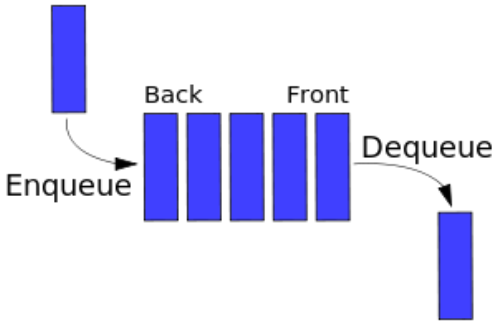
A document explaining the design of solutions is to be submitted along with project implementation. A template for it is provided on DVO – Final project section.

Table of contents

Exercise 1 – CustomQueue – Generics (4 points)

In line with the exercises proposed in TD 1, design and develop a **CustomQueue** data structure, formed by linked nodes that carry content of generic data type.

New nodes are added on the back of the queue (*enqueueing*), and retrieved (and deleted, one retrieved) from the top of the queue (*dequeueing*), so to implement a FIFO policy, as shown here below:

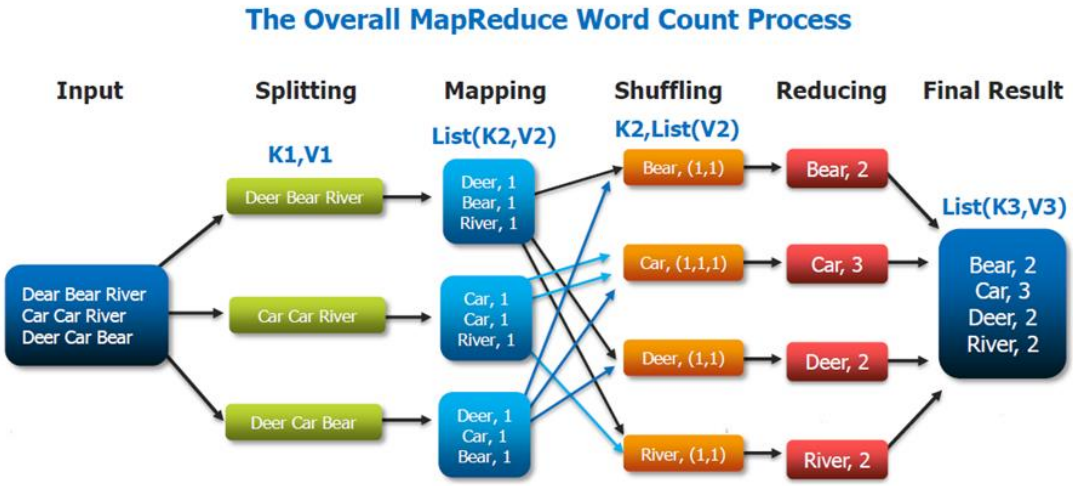


CustomQueue class must provide all the needed characteristics so to be used by a C# foreach loop.

Exercise 2 – MapReduce – Design patterns, Threads & IPC (8 points)

MapReduce is a framework for processing parallelizable tasks using a number of computers (nodes, collectively referred to as a cluster or grid), typically employed to elaborate or conduct operations on large sets of data (e.g., column-wise sum of matrix elements).

A MapReduce program is composed of a ***map*** procedure (or method) – which could also perform preliminary filtering and sorting over data (such as sorting students by first name into queues, one queue for each name) – and a ***reduce*** method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies).
The following picture shows how the principle can be applied on the word counting process on a given text:



Through multithreading and IPC (Inter-Process Communication) techniques implements a program:

- simulate a network of elaborating nodes,
- each node can execute an arbitrary task (i.e., a function) on a given data frame received in input
- data are supposed to be in the right format expected by the task

Suggestion: A (basic) data exchange protocol could be needed to be defined, so to allow nodes to receive the data frame to be then used by the task to execute.

Optional *(To be considered only if previous points have been fully addressed; to be avoided otherwise)*
Sockets and networking capabilities can be employed as IPC technique, so to allow the solution to indeed work on separate network nodes.

Exercise 3 – A Monopoly™ game (8 points) – Design patterns

Objective of the exercise is to simulate a simplified version of the **Monopoly™** game.

In the following, a summary of the rules – to necessary respect in the design and implementation – is provided. Other constraints and details are free to define by the students, and set as design hypotheses.

- The game is played on a circular game board, composed of 40 positions on the board, indexed from 0 to 39
- A set of players is given, each with name and initial position. Dice are rolled and players' positions on the game board will change
- If a player reaches position 39 and still needs to move forward, he'll continue from position 0. In other words, positions 38, 39, 0, 1, 2 are contiguous
- Each player rolls two dice and moves forward by a number of positions equal to the sum of the numbers told by the two dice
- A player's turn ends after having moved to its new position
- The same position can be occupied by more than one player
- If a player gets both dice with the same value, then he rolls the dice and moves again. If this happens three times in a row, the player goes to jail and ends his turn
- Jail can be a situation the player is in, or a place he pays visit to. The board therefore has a *Visit Only / In Jail* cell at position 10. A *Go To Jail* cell is also present, at position 30
- If at the end of a basic move, the player lands on *Go To Jail*, then he immediately moves to the position *Visit Only / In Jail* and is in jail. His turn ends
- If after moving, the player lands on *Visit Only / In Jail*, he is visiting only and is not in jail
- While the player is in jail, he still rolls the dice on his turn as usual, but does not move until either:

  (a) he gets a both dice with the same value, or

  (b) he fails to roll both dice with the same value for three times in a row (i.e., his previous two turns after moving to jail and his current turn)

  If either (a) or (b) happens in the player's turn, then he moves forward by the sum of the dice rolled positions and his turn ends. He does not roll the dice again even if he has rolled a both dice with the same value.