Selected design patterns:

1. Iterator – The iterator design pattern is used in order to promote reusability, extensibility, maintainability and modularity of iterating trough a specific collection. It does so by defining a scanning algorithm rather than supplying the actual data structure.

In the current project a simple txt file (written in csv format) was used as the cities' data base. As such it is highly recommended to supply an iterator in order to scan the data-base (especially since the only use of it is for searching a specific city).This the scanning logic is aggregated in one component and the city class is decoupled from it. Also, performance is dramatically improved since searching in a large is no longer requiring the creation of all cities. This structure institutes an infrastructure for the second pattern – strategy.

1. Strategy – The strategy design pattern is used in order to modify a specific selected part in an overall none-modifiable component. As such it allow to reuse a specific logic in different context. The modifiable part is stored in a different class or in a different function pointer (if supported by the language) and can be swapped upon need or even in runtime.

In the app it was implemented in order to regulate parsing format. As mentioned in the iterator design pattern, the app used a csv format. Thus the parsing logic was implemented to suit those needs. Nonetheless changing the format to a JSON or XML is certainly a feasible situation as the system grow.

Therefore the parsing mechanism is aggregated in a separate class – CsvParser.

Nevertheless the entity who uses the CsvParser knows it as an interface IParser rather than a class. This makes the format switching t in the future highly convenient.

1. Decorator -