**LOCKER FACILITY SYSTEM:**

**Purpose of the project:**The aim of the project is to create a locker management system. In the initialisation phase, it sets up empty lockers depending on the configuration(no of lockers, row\*col arrangement) given by the user. Once the system is setup, user can book lockers by choosing a chamber and setting a pin to it. Each locker has different fares depending on its size; after the user releases the locker by giving the correct pin the charges for usage are conveyed.   
  
The edge cases, for instance  
- unavailability of required locker chambers  
- trying to book an already occupied locker  
- providing incorrect locker pins  
- too many incorrect pins  
- choosing a locker number out of bounds  
are handled by the code.  
  
This project is command line driven, where the user can issue the commands and appropriate actions will be taken by the program.  
  
**Design:**  
The code is designed and organised using object oriented programming paradigm. The design of the program ensures reusability, extendibility, easy maintenance and debugging.   
  
  
Diagram

Description automatically generated

The code is organised into classes, packing the associated attributes and methods. This provides encapsulation allowing data and the methods that operate on the data to be wrapped together. Constructors are used to do object initialisation. Inheritance principle is used to achieve generalisation, specificity, better organisation, and code reuse. Polymorphism allows us to use the same methods (eg: get\_type()) across multiple classes. The code benefits from method overriding (eg: get\_fare()) to levy special fee for medium sized lockers.

Further the command operations,  
- Booking a locker  
- Releasing a locker  
- Viewing the fares of lockers  
- Displaying the locker configuration and status  
- Displaying the entire locker system.   
 space -> empty locker  
 X -> occupied locker  
are all organised as separate methods. This keeps the code modular, improves reusability and makes it easily modifiable.

**Sample (3\*3 locker system):**  
  
 0 1 2  
0 | X | | |  
1 | | X | |  
2 | | X | |

**Project File Organisation:**  
The project files are as follows,  
- .docs/configuration.txt -> contains user configuration for the locker system(no. of chambers, row \* col configuration)  
- .docs/readme.txt -> introduces and explains the project  
- classes.py -> contains the classes of the project  
- locker\_system.py -> Interacts with the user via the command line interface; receives the commands and executes the appropriate methods to operate on the command  
- unit\_test.py -> contains sample testcases to unit test the code for various user command operations.