# Programming Patterns/Function Design

The purpose of patterns in the Bluebrook Airport system is to make the programming easier to understand, fix bugs, ensure future errors.  
  
Separating Methods   
--------------------------------------------------------------------------------------------------------------------------  
Name: Make Booking  
Description: The user will view empty seats, choose a seat then save their booking in XML  
Functions used: AddBooking / GetCustomerInfo  
--------------------------------------------------------------------------------------------------------------------------

To separate the various areas of the system, commented sections are used. Commented sections are useful as they help identify the purpose of each area to ensure the user can easily find and understand each section in a simple to understand manner before viewing the functions and variables. An example is displayed above for making a booking, giving the section a name and brief description of what occurs in this section to help other users easily understand, where appropriate, the commented section includes which functions are used in this section, this can help identify bugs easily and prevent future errors.

## Variable Names

|  |  |  |
| --- | --- | --- |
| **Customer** | **Booking** | **Plane** |
| cFirstName cLastName cPhoneNo cEmail | bBookingNo bSeats bFlightNo bFlightName | pCapacity pStatus pMileage pTerminal |

For the variable names, the declarations are split into sections using a lowercase identifier such as in the example above such as a lower case ‘c’ for customer variables, a lower case ‘b’ for booking variables. The purpose of this is to be able to identify where variables come from when they are called in later function to reduce ambiguity such as if the customer variables are used in the booking section to link a customer to a booking. These variable names ensure any user can understand the program easily and reduce errors in future as the user can identify the origin of a bug easier.

Function Names/Declaration  
For the naming of functions, nouns are used to clearly distinguish a function from variables or class names. Comments are not needed, as the functions should be descriptive of the purpose of function linked with the commented section, in which the function is defined and used.  
  
Public   
The purpose of using functions within public classes is to ensure any class has access to the function such as viewing flight info, which can be done by all types of users e.g. customer, staff or admin.  
  
Private -   
The purpose of using functions within private classes is to add security to the program, in which only the current private class will have access to a specific function such as admin functions cannot be used by staff or regular customers.  
  
Protected  
The purpose of using functions within protected classes is to ensure only the current class and subclasses have access to function, such as different levels of staff.  
  
void Functions – Verbs (an action)  
void functions are used throughout the Bluebrook Airport, they are unique functions which do not return a value, they are defined as camelCase using verbs as they are performing an action within the system such as the ‘createXML’ function, which is used create an xml file for the customer data to be stored in.

Pre-Processor directives – ALL CAPS  
Pre-processor directives are either defined in the C++ properites for the project or using a #define at the top of the program using #define followed by the directives in capitals. An example of this is the /////// directive in the /////// section of the system.

Booleans  
Boolean functions within the system are given correct Boolean names to return either a true or false output, an example for a function with the purpose of searching an xml file for a flight within the system is named ‘doesFlightExist’ as the function will return a value of True if the flight is found or a value of False if the flight cannot be found in the system. The purpose of this name is to include