By Hritaban Ghosh 19CS30053

****
Name: Hritaban Ghosh
Roll Number: 19CS30053
****

Low-Level Design

Design Principles:

Encapsulation:

– Maximize encapsulation for every class.
– Use private access specifier for all data members that are not needed by derived classes, if any. Use protected otherwise.
– Use public access specifier for interface methods and static constants and friend functions only.

STL Containers:

– Use STL containers (like vector, map, hashmap, list, etc.) and their iterators. Do not use arrays.
– Use iterators for STL containers. Do not use bare for loops.

Pointers & References:

– Minimize the use of pointers. Use pointers only if you need null-able entities.
– If you use pointer for dynamically allocated objects (should be minimized), remember to delete at an appropriate position.
– Use const reference wherever possible.

P.T.O.

By Hritaban Ghosh 19CS30053

Design of Classes, Data Members and Methods:

Class Passenger

This class is used to encapsulate Passenger Information. It stores the date, month and year of the the date.

Data members

data fields:
(All data fields are private and are only accessible to the class. They are not needed to be accessible to other classes and encapsulation is intensified.)
1) std::string name
    - Stores the name of the Passenger
2) int aadhaarNumber
    - Stores the unique Aadhaar Number of the Passenger
3) Date dateOfBirth
    - Stores the DOB of the Passenger using Date Class.
4) std::string gender
    - Stores the Gender of the Passenger
5) int mobileNumber
    - Stores the Mobile Number of the Passenger or -1 if not provided.

Methods

public member methods:
1) Constructor :
    - The Constructor of this class takes five arguments name, aadhaarNumber, dateOfBirth, gender and mobileNumber with mobileNumber being defaulted to -1. These are initialized in the initializer list.
    - The body of the Constructor is empty.
2) Copy Constructor
    - The copy constructor takes a Passenger object as argument by const reference and copies each data member of the argument into the now being created object via the initializer list.
    - The body of the Copy Constructor is empty.
3) virtual Destructor
    - The Destructor of Date Class is a default destructor.
    - The body of the Destructor is empty.

P.T.O.

By Hritaban Ghosh 19CS30053

Class Date

This class is used to encapsulate date (dd/MMM/yy). It stores the date, month and year of the the date.

Friend Functions:
1) ostream& operator<<(ostream&, const Date&)
- The insertion operator is overloaded as a friend function and it prints out the date to the console in a dd/MMM/yyy format.
- It takes a output streaming operator by reference and returns the same by reference in order to support chaining. It also takes a Date object by const reference in order to access it's data members and print to the console in dd/MMM/yyy format.

Static Data Members:
(All static constants are private and are only accessible to the class. They are not needed to be accessible to other classes and encapsulation is intensified.)
1) const vector of strings of monthNames (vector containing names of the months in a year)
2) const vector of strings of dayNames (vector containing names of the days in a week)
3) const int MAX_VALID_YR = 9999 (const value of year used to check validity of the date)
4) const int MIN_VALID_YR = 1800 (const value of year used to check validity of the date)

Data members

data fields:
(All data fields are private and are only accessible to the class. They are not needed to be accessible to other classes and encapsulation is intensified.)
1) enum type of Month (to enumerate all the 12 months and indexed starting from 1)
2) enum type of Day (to enumerate all the 7 days in a week and indexed starting from 0)
3) unsigned int date_ (to store the dd part of the Date)
4) Month month_ (to store MMM part of the Date)
5) unsigned int year_ (to store yy part of the Date)

Static Methods:
private static methods:
1) bool isLeap(int) :
- This method takes an integer value as argument (passed by value) and returns a boolean value (return by value).
- It returns true(1) if the integer passed to it is a leap year and returns false(0) otherwise.
- This method is a utility function i.e. it is to be used by the class alone for its internal computation and should not be exposed to the user and thus is declared as private static.

Methods

public member methods:
1) Constructor :

- The Constructor of this class takes three arguments d, m and y as unsigned integers and initializes the date_, month_ and year_ respectively.
- The body of the Constructor is empty.

2) Destructor
- The Destructor of Date Class is a default destructor.

3) void print() const
- This member function prints out the date in dd/MMM/yy format.
- This function is qualified as constant and is not allowed to change any data members of the object.

4) bool validDate() const
- This member function returns true if the date stored in the object is valid or not. It considers the Gregorian Calendar between the years 1800 and 9999 both inclusive.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) std:: string day() const
- This member function returns the day of the week corresponding to the date. It uses system calendar for it's operation.
- This function is qualified as constant and is not allowed to change any data members of the object.

6) bool operator==(const Date&) const
- The equality operator is overloaded as a member function-Returns true if the dates are the same else false.
- This method takes a Date object by const reference as an argument and the current object is implicitly passed as this pointer.
- This function is qualified as constant and is not allowed to change any data members of the object.

7)bool operator!=(const Date&) const
- The inequality operator is overloaded as a member function-Returns true if the dates are different else false.
- This method takes a Date object by const reference as an argument and the current object is implicitly passed as this pointer.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class Railways

This class is a singleton class and is used to encapsulate the list of stations and the distance matrix of the Railways system.

Friend Functions:

1) ostream& operator<<(ostream&, const Railways&)
- The insertion operator is overloaded as a friend function and it prints out the all the information of the Indian Railways System to the console. (The List of Stations and the Distance Matrix)

P.T.O.

- It takes a output streaming operator by reference and returns the same by reference in order to support chaining. It also takes a Railways object by const reference in order to access it's data and print to the console.

Static Data Members:
1) static const std::vector<Station> sStations
- Collection of Stations
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is public. It is made public because testing applications and other units of the program must be able to view the Stations accessible through this Railways system.

2) static const std::vector<std::vector <double>> sDistances
- Distance Matrix to get distance between any two stations
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private and is only accessible to the class. They are not needed to be accessible to other classes and encapsulation is intensified.
- Information from this matrix can be obtained through the GetDistance public member method.

Methods

Static member methods:
1) static const Railways& IndianRailways()
- Singleton instance of the class Railways called IndianRailways
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is declared private in order to make it inaccessible. An instance of this class can only be created using the static member method IndianRailways.
- The body of the Constructor is empty.

2) Destructor
- The Destructor of Railways Class is a default destructor.

public member methods:
1) double GetDistance(const Station&, const Station&) const
- It returns the distance between two stations passed to it by const reference. It returns a double value.
- It uses the list of stations and the matrix to access the required distance and return the same.
- This function is qualified as constant and is not allowed to inadvertently change any data members of this singleton object.

Note: The **public static unit test function** is described in the test plan.

By Hritaban Ghosh 19CS30053

Class Station

This class is used to encapsulate a station. It stores it's name.

Friend Functions:
1) ostream& operator<<(ostream&, const Station&)
- The insertion operator is overloaded as a friend function and it prints out the name of the Station.
- It takes a output streaming operator by reference and returns the same by reference in order to support chaining. It also takes a Station object by const reference in order to access it's data members and print to the console.

Data members

data fields:
(All data fields are private and are only accessible to the class. They are not needed to be accessible to other classes and encapsulation is intensified.)
1) string name_ (stores the name of the station), for example "Mumbai".

Methods

public member methods:
1) Constructor :
- The Constructor of this class takes one argument which is the name of the Station and initializes name_ in the initializer list of the Constructor.
- The body of the Constructor is empty.

2) Destructor
- The Destructor of Station Class is a default destructor.

3) string GetName() const:
- This member function returns the name of the station.
- This function is qualified as constant and is not allowed to change any data members of the object.

4) double GetDistance(const Station&) const:
- This method returns the distance between two Stations as per the Distance Matrix initialized in Railways.
- This method takes one of the Station as an argument passed by const reference and the other is implicitly passed to the method as this pointer.
- This method is qualified as constant and is not allowed to change any data members of the object.
- It uses the double Railways::GetDistance(const Station&, const Station&) const to compute the distance between the two stations and return the same.

5) bool operator==(const Station&) const:
- The equality operator is overloaded as a member function-Returns true if the names of the Stations are the same else false

P.T.O.

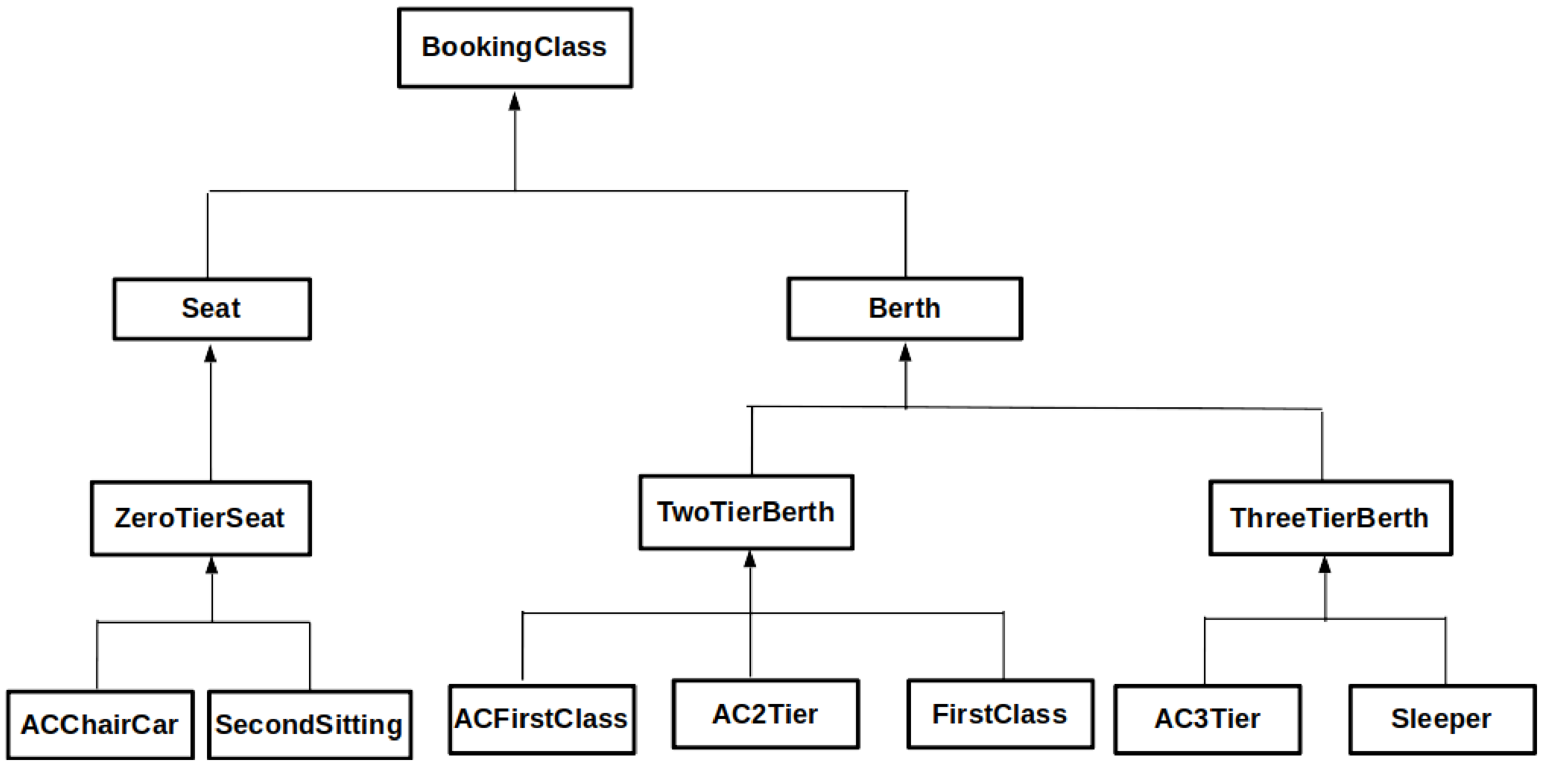
- This function is qualified as constant and is not allowed to change any data members of the object.
- This method takes a Station object by const reference as an argument and the current object is implicitly passed as this pointer.

6) bool operator!=(const Station&) const:

- The inequality operator is overloaded as a member function-Returns true if the names of the Stations are different else false
- This method takes a Station object by const reference as an argument and the current object is implicitly passed as this pointer.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Booking Class Hierarchy Design



Note: The above diagram is not upto UML standards and is for schematic representation only.

By Hritaban Ghosh 19CS30053

Class BookingClass

This class is an abstract base class and is used as an interface for the BookingClass Hierarchy.

Methods

public member methods:
1) Constructor :
   - The Constructor of this class is the default constructor.
   - The body of the Constructor is empty.
2) virtual Destructor
   - The Destructor of this Class is a default destructor.
   - It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
   - The body of the Destructor is empty.
3) virtual double GetLoadFactor() const = 0
   - It is a **pure virtual** function and has not been implemented for this class. It is to be implemented in subsequent specializations of this class. It’s semantic meaning is given in the next line.
   - It’s role is to return the Load Factor of the Booking Class
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
4) virtual std::string GetName() const = 0
   - It is a **pure virtual** function and has not been implemented for this class. It is to be implemented in subsequent specializations of this class. It’s semantic meaning is given in the next line.
   - It’s role is to return the Name of the Booking Class
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
5) virtual bool IsSitting() const = 0
   - It is a **pure virtual** function and has not been implemented for this class. It is to be implemented in subsequent specializations of this class. It’s semantic meaning is given in the next line.
   - It’s role is to return true if the Booking Class has Seat else false
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
6) virtual bool IsAC() const = 0
   - It is a **pure virtual** function and has not been implemented for this class. It is to be implemented in subsequent specializations of this class. It’s semantic meaning is given in the next line.
   - It’s role is to return true if the Booking Class has AC else false
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
7) virtual int GetNumberOfTiers() const = 0

P.T.O.

- It is a **pure virtual** function and has not been implemented for this class. It is to be implemented in subsequent specializations of this class. It's semantic meaning is given in the next line.
- It's role is to return the Number Of Tiers of the Booking Class
- This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.

8) virtual bool IsLuxury() const = 0
- It is a **pure virtual** function and has not been implemented for this class. It is to be implemented in subsequent specializations of this class. It's semantic meaning is given in the next line.
- It's role is to return true if the Booking Class is Luxury else false
- This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.

Class Seat : Specialization of BookingClass

This class is a Specialization of the Abstract Base Class BookingClass. This class represents the category of Booking Classes that have Seats.
This is still an abstract class as not all pure virtual functions of the Base Class have been implemented.

Methods

public member methods:

1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor
- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

3) virtual double GetLoadFactor() const = 0
- Inherited from Class BookingClass.

4) virtual std::string GetName() const = 0
- Inherited from Class BookingClass.

5) virtual bool IsSitting() const
- It is a **virtual** function and has been implemented for this class.
- It returns true as this class represents the category of Booking Classes that have Seats.
- This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.

6) virtual bool IsAC() const = 0
- Inherited from Class BookingClass.

7) virtual int GetNumberOfTiers() const = 0
- Inherited from Class BookingClass.

8) virtual bool IsLuxury() const = 0
- Inherited from Class BookingClass.

By Hritaban Ghosh 19CS30053

Class Berth : Specialization of BookingClass

This class is a Specialization of the Abstract Base Class BookingClass. This class represents the category of Booking Classes that have Berths.
This is still an abstract class as not all pure virtual functions of the Base Class have been implemented.

Methods

public member methods:

1) Constructor :
   - The Constructor of this class is the default constructor.
   - The body of the Constructor is empty.
2) virtual Destructor
   - The Destructor of this Class is a default destructor.
   - It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
   - The body of the Destructor is empty.
3) virtual double GetLoadFactor() const = 0
   - Inherited from Class BookingClass.
4) virtual std::string GetName() const = 0
   - Inherited from Class BookingClass.
5) virtual bool IsSitting() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns false as this class represents the category of Booking Classes that have Berths.
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
6) virtual bool IsAC() const = 0
   - Inherited from Class BookingClass.
7) virtual int GetNumberOfTiers() const = 0
   - Inherited from Class BookingClass.
8) virtual bool IsLuxury() const = 0
   - Inherited from Class BookingClass.

P.T.O.

By Hritaban Ghosh 19CS30053

Class ZeroTierSeat : Specialization of Seat

This class is a Specialization of the Abstract Base Class Seat. This class represents the category of Booking Classes that have Seats and have Zero Tiers.
This is still an abstract class as not all pure virtual functions of the Base Class have been implemented.

Methods

public member methods:

1) Constructor :
   - The Constructor of this class is the default constructor.
   - The body of the Constructor is empty.
2) virtual Destructor
   - The Destructor of this Class is a default destructor.
   - It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
   - The body of the Destructor is empty.
3) virtual double GetLoadFactor() const = 0
   - Inherited from Class Seat.
4) virtual std::string GetName() const = 0
   - Inherited from Class Seat.
5) virtual bool IsSitting() const
   - Inherited from Class Seat.
6) virtual bool IsAC() const = 0
   - Inherited from Class Seat.
7) virtual int GetNumberOfTiers() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns 0 as this class represents the category of Booking Classes with Seats that have Zero Tiers.
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
8) virtual bool IsLuxury() const = 0
   - Inherited from Class Seat.

By Hritaban Ghosh 19CS30053

Class TwoTierSeat : Specialization of Berth

This class is a Specialization of the Abstract Base Class Berth. This class represents the category of Booking Classes that have Berths and have Two Tiers.
This is still an abstract class as not all pure virtual functions of the Base Class have been implemented.

Methods

public member methods:

1) Constructor :
    - The Constructor of this class is the default constructor.
    - The body of the Constructor is empty.
2) virtual Destructor
    - The Destructor of this Class is a default destructor.
    - It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
    - The body of the Destructor is empty.
3) virtual double GetLoadFactor() const = 0
    - Inherited from Class Berth.
4) virtual std::string GetName() const = 0
    - Inherited from Class Berth.
5) virtual bool IsSitting() const
    - Inherited from Class Berth.
6) virtual bool IsAC() const = 0
    - Inherited from Class Berth.
7) virtual int GetNumberOfTiers() const
    - It is a **virtual** function and has been implemented for this class.
    - It returns 2 as this class represents the category of Booking Classes with Berths that have Two Tiers.
    - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
8) virtual bool IsLuxury() const = 0
    - Inherited from Class Berth.

P.T.O.

By Hritaban Ghosh 19CS30053

Class ThreeTierSeat : Specialization of Berth

This class is a Specialization of the Abstract Base Class Berth. This class represents the category of Booking Classes that have Berths and have Three Tiers.
This is still an abstract class as not all pure virtual functions of the Base Class have been implemented.

Methods

public member methods:
1) Constructor :
   - The Constructor of this class is the default constructor.
   - The body of the Constructor is empty.
2) virtual Destructor
   - The Destructor of this Class is a default destructor.
   - It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
   - The body of the Destructor is empty.
3) virtual double GetLoadFactor() const = 0
   - Inherited from Class Berth.
4) virtual std::string GetName() const = 0
   - Inherited from Class Berth.
5) virtual bool IsSitting() const
   - Inherited from Class Berth.
6) virtual bool IsAC() const = 0
   - Inherited from Class Berth.
7) virtual int GetNumberOfTiers() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns 3 as this class represents the category of Booking Classes with Berths that have Three Tiers.
   - This function is qualified as constant and is not allowed to change any data members of the object even in the specializations of the class.
8) virtual bool IsLuxury() const = 0
   - Inherited from Class Berth.

By Hritaban Ghosh 19CS30053

Class ACChairCar : Specialization of ZeroTierSeat

This class is a Specialization of the Abstract Base Class ZeroTierSeat. It stores information for the Booking Class Category “AC Chair Car”.
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:
1) static const double sloadFactor
- It stores the load factor that is to be levied in the computation of fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it’s value. It can be obtained through the GetLoadFactor method which is const qualified.

Methods

Static member methods:
1) static const ACChairCar& Instance()
- Singleton instance of the class Sleeper can be obtained by calling ACChairCar::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor
- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:
1) virtual double GetLoadFactor() const
- It is a **virtual** function and has been implemented for this class.
- It returns the static constant sloadFactor by value.
- This function is qualified as constant and is not allowed to change any data members of the object.

2) virtual std::string GetName() const
- It is a **virtual** function and has been implemented for this class.
- It returns the name of the Booking Class which is “AC Chair Car”.
- This function is qualified as constant and is not allowed to change any data members of the object.

3) virtual bool IsSitting() const

P.T.O.

- Inherited from Class ZeroTierSeat.

4) virtual bool IsAC() const
- It is a **virtual** function and has been implemented for this class.
- It returns true because this Booking Class has Air Conditioning.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) virtual int GetNumberOfTiers() const
- Inherited from Class ZeroTierSeat.

6) virtual bool IsLuxury() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because it is not considered Luxurious as per Government Standards.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class SecondSitting: Specialization of ZeroTierSeat

This class is a Specialization of the Abstract Base Class ZeroTierSeat. It stores information for the Booking Class Category “Second Sitting”.
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:
1) static const double sloadFactor
- It stores the load factor that is to be levied in the computation of fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it’s value. It can be obtained through the GetLoadFactor method which is const qualified.

Methods

Static member methods:
1) static const SecondSitting& Instance()
- Singleton instance of the class Sleeper can be obtained by calling SecondSitting::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor

- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:

1) virtual double GetLoadFactor() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns the static constant sloadFactor by value.
   - This function is qualified as constant and is not allowed to change any data members of the object.
2) virtual std::string GetName() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns the name of the Booking Class which is "Second Sitting".
   - This function is qualified as constant and is not allowed to change any data members of the object.
3) virtual bool IsSitting() const
   - Inherited from Class ZeroTierSeat.
4) virtual bool IsAC() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns false because this Booking Class has no Air Conditioning.
   - This function is qualified as constant and is not allowed to change any data members of the object.
5) virtual int GetNumberOfTiers() const
   - Inherited from Class ZeroTierSeat.
6) virtual bool IsLuxury() const
   - It is a **virtual** function and has been implemented for this class.
   - It returns false because it is not considered Luxurious as per Government Standards.
   - This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class ACFirstClass : Specialization of TwoTierBerth

This class is a Specialization of the Abstract Base Class TwoTierBerth. It stores information for the Booking Class Category "AC First Class".
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:

1) static const double sloadFactor
   - It stores the load factor that is to be levied in the computation of fare.
   - It is qualified as const as none of the data can be changed after initialization.
   - This static constant is private. It is made private so that no one can inadvertently change it's value. It can be obtained through the GetLoadFactor method which is const qualified.

P.T.O.

By Hritaban Ghosh 19CS30053

Methods

Static member methods:
1) static const ACFirstClass& Instance()
- Singleton instance of the class Sleeper can be obtained by calling ACFirstClass::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor
- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:
1) virtual double GetLoadFactor() const
- It is a **virtual** function and has been implemented for this class.
- It returns the static constant sloadFactor by value.
- This function is qualified as constant and is not allowed to change any data members of the object.

2) virtual std::string GetName() const
- It is a **virtual** function and has been implemented for this class.
- It returns the name of the Booking Class which is “AC First Class”.
- This function is qualified as constant and is not allowed to change any data members of the object.

3) virtual bool IsSitting() const
- Inherited from Class TwoTierBerth.

4) virtual bool IsAC() const
- It is a **virtual** function and has been implemented for this class.
- It returns true because this Booking Class has Air Conditioning.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) virtual int GetNumberOfTiers() const
- Inherited from Class TwoTierBerth.

6) virtual bool IsLuxury() const
- It is a **virtual** function and has been implemented for this class.
- It returns true because it is considered Luxurious as per Government Standards.
- This function is qualified as constant and is not allowed to change any data members of the object.

By Hritaban Ghosh 19CS30053

Note: The **public static unit test function** is described in the test plan.

Class AC2Tier : Specialization of TwoTierBerth

This class is a Specialization of the Abstract Base Class TwoTierBerth. It stores information for the Booking Class Category “AC 2 Tier”.
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:
1) static const double sloadFactor
- It stores the load factor that is to be levied in the computation of fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it’s value. It can be obtained through the GetLoadFactor method which is const qualified.

Methods

Static member methods:
1) static const AC2Tier& Instance()
- Singleton instance of the class Sleeper can be obtained by calling AC2Tier::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor
- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:
1) virtual double GetLoadFactor() const
- It is a **virtual** function and has been implemented for this class.
- It returns the static constant sloadFactor by value.
- This function is qualified as constant and is not allowed to change any data members of the object.

2) virtual std::string GetName() const
- It is a **virtual** function and has been implemented for this class.
- It returns the name of the Booking Class which is “AC 2 Tier”.

P.T.O.

- This function is qualified as constant and is not allowed to change any data members of the object.

3) virtual bool IsSitting() const
- Inherited from Class TwoTierBerth.

4) virtual bool IsAC() const
- It is a **virtual** function and has been implemented for this class.
- It returns true because this Booking Class has Air Conditioning.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) virtual int GetNumberOfTiers() const
- Inherited from Class TwoTierBerth.

6) virtual bool IsLuxury() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because it is not considered Luxurious as per Government Standards.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class FirstClass : Specialization of TwoTierBerth

This class is a Specialization of the Abstract Base Class TwoTierBerth. It stores information for the Booking Class Category “First Class”.
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:
1) static const double sloadFactor
- It stores the load factor that is to be levied in the computation of fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it’s value. It can be obtained through the GetLoadFactor method which is const qualified.

Methods

Static member methods:
1) static const FirstClass& Instance()
- Singleton instance of the class Sleeper can be obtained by calling FirstClass::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor
- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:
1) virtual double GetLoadFactor() const
- It is a **virtual** function and has been implemented for this class.
- It returns the static constant sloadFactor by value.
- This function is qualified as constant and is not allowed to change any data members of the object.

2) virtual std::string GetName() const
- It is a **virtual** function and has been implemented for this class.
- It returns the name of the Booking Class which is “AC First Class”.
- This function is qualified as constant and is not allowed to change any data members of the object.

3) virtual bool IsSitting() const
- Inherited from Class TwoTierBerth.

4) virtual bool IsAC() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because this Booking Class has no Air Conditioning.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) virtual int GetNumberOfTiers() const
- Inherited from Class TwoTierBerth.

6) virtual bool IsLuxury() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because it is not considered Luxurious as per Government Standards.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class AC3Tier : Specialization of ThreeTierBerth

This class is a Specialization of the Abstract Base Class ThreeTierBerth. It stores information for the Booking Class Category “AC 3 Tier”.
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:

P.T.O.

1) static const double sloadFactor

- It stores the load factor that is to be levied in the computation of fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it's value. It can be obtained through the GetLoadFactor method which is const qualified.

Methods

Static member methods:

1) static const AC3Tier& Instance()

- Singleton instance of the class Sleeper can be obtained by calling AC3Tier::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:

1) Constructor :

- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor

- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:

1) virtual double GetLoadFactor() const

- It is a **virtual** function and has been implemented for this class.
- It returns the static constant sloadFactor by value.
- This function is qualified as constant and is not allowed to change any data members of the object.

2) virtual std::string GetName() const

- It is a **virtual** function and has been implemented for this class.
- It returns the name of the Booking Class which is "AC 3 Tier".
- This function is qualified as constant and is not allowed to change any data members of the object.

3) virtual bool IsSitting() const

- Inherited from Class ThreeTierBerth.

4) virtual bool IsAC() const

- It is a **virtual** function and has been implemented for this class.
- It returns true because this Booking Class has Air Conditioning.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) virtual int GetNumberOfTiers() const

- Inherited from Class ThreeTierBerth.

6) virtual bool IsLuxury() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because it is not considered Luxurious as per Government Standards.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class Sleeper : Specialization of ThreeTierBerth

This class is a Specialization of the Abstract Base Class ThreeTierBerth. It stores information for the Booking Class Category “Sleeper”.
This is a singleton concrete class as all virtual functions of the Base Class have been implemented.

Static Data Members:
1) static const double sloadFactor
- It stores the load factor that is to be levied in the computation of fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it’s value. It can be obtained through the GetLoadFactor method which is const qualified.

Methods

Static member methods:
1) static const Sleeper& Instance()
- Singleton instance of the class Sleeper can be obtained by calling Sleeper::Instance().
- This method is public and is used to create an instance of the class statically and return the same. This means that it creates a single instance of the class which is used throughout the program.
- The constructor is made private and thus the only way to access an instance of this class is through this method.

private member methods:
1) Constructor :
- The Constructor of this class is the default constructor.
- The body of the Constructor is empty.

2) virtual Destructor
- The Destructor of this Class is a default destructor.
- It is qualified as virtual to handle the deletion of objects in the polymorphic hierarchy.
- The body of the Destructor is empty.

public member methods:
1) virtual double GetLoadFactor() const
- It is a **virtual** function and has been implemented for this class.
- It returns the static constant sloadFactor by value.

P.T.O.

- This function is qualified as constant and is not allowed to change any data members of the object.

2) virtual std::string GetName() const
- It is a **virtual** function and has been implemented for this class.
- It returns the name of the Booking Class which is "Sleeper".
- This function is qualified as constant and is not allowed to change any data members of the object.

3) virtual bool IsSitting() const
- Inherited from Class ThreeTierBerth.

4) virtual bool IsAC() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because this Booking Class has no Air Conditioning.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) virtual int GetNumberOfTiers() const
- Inherited from Class ThreeTierBerth.

6) virtual bool IsLuxury() const
- It is a **virtual** function and has been implemented for this class.
- It returns false because it is not considered Luxurious as per Government Standards.
- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Class Booking

This class is used to encapsulate information of a Booking that has been made.

Friend Functions:

1) ostream& operator<<(ostream&, const Date&)
- The insertion operator is overloaded as a friend function and it prints out all the Booking Information to the console.
- It takes a output streaming operator by reference and returns the same by reference in order to support chaining. It also takes a Booking object by const reference in order to access it's data members and print to the console.

Example:

```
BOOKING SUCCEEDED:
PNR Number : 1
From Station : Mumbai
To Station : Delhi
Travel Date : 15/Feb/2021
Travel Class : AC First Class
--> Mode : Sleeping
--> Comfort : AC
```

--> Bunks : 2
--> Luxury : Yes
Fare : 2776

Static Data Members:
1) static const double sBareFarePerKM
- Stores the Bare Fare per Kilometer that is to levied in the computation of the fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it's value.

2) static const double sACSurcharge
- Stores the AC Surcharge that is to levied in the computation of the fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it's value.

3) static const double sLuxuryTaxPercent
- Stores the Luxury Tax Percent that is to levied in the computation of the fare.
- It is qualified as const as none of the data can be changed after initialization.
- This static constant is private. It is made private so that no one can inadvertently change it's value.

4) static int sBookingPNRSerial
- Stores the current PNR Serial starting from 1. It is unique to a Booking Object. It is incremented after every construction of a new Booking object.
- This static variable is private. It is made private so that no one can inadvertently change it's value.

5) static std::vector<Booking*> sBookings
- Stores pointers to Bookings made in the Railway Booking System. This is to keep in track all the Bookings that have been made.
- This static variable is public. It is made public so that it can be accessed in the testing applications and in other parts of the program.

Data members

data fields:
(All data fields are private and are only accessible to the class. They are not needed to be accessible to other classes and encapsulation is intensified.)
1) Station fromStation
- Stores the Station of Departure.

2) Station toStation
- Stores the Station of Arrival.

3) Date date
- Stores the Date of the Journey.

4) const BookingClass& bookingClass
- Stores the reference to the Booking Class.

P.T.O.

- It is qualified as const as none of the data can be changed after initialization.

5) int fare
- Stores the fare required to pay for the ticket. This is computed using member function ComputeFare.

6) int PNR
- Stores the unique PNR Number of the ticket.
- Uses the sBookingPNRSerial to initialize it's value and after that the sBookingPNRSerial is incremented.

7) bool bookingStatus = true
- Stores the Booking Status.
- By default the booking status value is "true".

8) std::string bookingMessage = "BOOKING SUCCEEDED"
- Stores the Booking Message to be displayed when Booking is done.
- By default the booking message is " BOOKING SUCCEEDED".

9) Passenger* passenger
- Stores Passenger Information.
- It is initialized to NULL if no Passenger information is provided.

Methods

public member methods:

1) Constructor :
- The Constructor of this class takes five arguments frromStation, toStation, date, bookingClass and passenger and initializes the data members using the initializer list.
- The body of the Constructor is empty.

2) virtual Destructor
- The destructor deletes the pointer to the object in the vector sBookings.
- The destructor also deletes the passenger object that is associated with the booking.
- It is qualified as virtual to provide for further futuristic polymorphic hierarchy.

3) virtual int ComputeFare()
- It is a **virtual** function and has been implemented for this class.
- It computes the fare based on the data provided in the data members of the Booking object and the Business Logic and returns the fare after rounding it to the nearest integer.

4) bool operator==(const Booking&) const
- The equality operator is overloaded as a member function-Returns true if the PNR Numbers of the Bookings are the same else false.
- This method takes a Booking object by const reference as an argument and the current object is implicitly passed as this pointer.
- This function is qualified as constant and is not allowed to change any data members of the object.

5) bool operator!=(const Booking&) const
- The inequality operator is overloaded as a member function-Returns true if the PNR Numbers of the Bookings are different else false.
- This method takes a Booking object by const reference as an argument and the current object is implicitly passed as this pointer.

- This function is qualified as constant and is not allowed to change any data members of the object.

Note: The **public static unit test function** is described in the test plan.

Coding Principles and Guidelines

- Use CamelCase for naming variables, classes, types and functions
- Every name should be indicative of its semantics
- Start every variable with a lower case letter
- Start every function and class with an upper case letter
- Use a trailing underscore (_) for every non-static data member
- Use a leading 's' for every static data member
- Do not use any global variable or function (except main(), and friends)
- No constant value should be written within the code - should be put in the application as static
- Prefer to pass parameters by value for build-in type and by const reference for UDT
- Every polymorphic hierarchy must provide a virtual destructor in the base class
- Prefer C++ style casting (like static_cast<cast>(x) over C style casting (like (int))
- The project should compile without any compiler warning
- Indent code properly
- Comment the code liberally and meaningfully
- Keep the Code Simple
- Design code with scalability and reuse in mind.
- Correct errors as they occur.
- Indicate a brief description of what a variable is for (reference to commenting)
- Try to define different sections of the code by segmenting blocks of code into a paragraph
- The code should be such that one should be able to understand it even after returning to it after some time gap, without that person having to look at every line of it

P.T.O.

By Hritaban Ghosh 19CS30053

- The language functions that are complex or the structure that is difficult to be comprehended should be avoided
- Avoid Commenting on Obvious Things
- Deep nesting structure should be avoided
  Too many nesting structures make it difficult to understand the code.
- Folllow the Principle of DRY which stands for Don’t Repeat Yourself, also known as DIE (duplication is evil).
- It is more important to be correct than to be fast. It is more important to be maintainable than to be fast.