Design For Railway Ticket Booking System

Abhishek Gandhi
19CS10031

RAILWAYS (Singleton)

Data Members
static:
private-

1. sStations : vector<const Station*>
2. sDistStations: map<pair<string,string>, int>

Methods
static:

- IndianRailways :
    - Params: none
    - Returns: Railways (public) singleton representing Railways

non-static:

- GetDistance : const
    - Returns: int
    - Params: Station, Station]
- GetStation : const
    - Returns: station
    - Params: string (public)

DATE

Data Members
Static (private):

- sMonthNames : const vector<string>

Non-static (private):

- date_ : const uint32_t
- month_ : const uint32_t
- year_ : const uint32_t

Methods
static:

- CreateDate (public) :
    - Params: uint32_t, uint32_t, uint32_t
    - Returns: Date

  The static function does appropriate error handling before creating a date.

Methods
non-static (private):

- CheckValidity :
    - Params : none
    - Returns : none

BOOKING CLASSES

They are designed as a single-level flat hierarchy with parametric polymorphism.

The sub-types are:

- ACFirstClass
- ExecutiveChairCar
- AC2Tier
- FirstClass
- AC3Tier
- ACChairCar
- Sleeper
- SecondSitting

BASE CLASS (Abstract)

Data Members
non-static (private):

- name_ : const string

Methods
non-static consts (public):

- GetName :
    - [Params: none]
    - [Returns: string]
- IsAC :
    - [Params: none]
    - [Returns: bool] (polymorphic)
- IsAC :

    - [Params: none]
    - [Returns: bool] (polymorphic)
- IsAC :
    - [Params: none]
    - [Returns: bool] (polymorphic)
- GetNumberOfTiers :
    - [Params: none]
    - [Returns: int] (polymorphic)
- GetLoadFactor :
    - [Params: none]
    - [Returns: double] (polymorphic)
- GetReservationCharge :
    - [Params: none]
    - [Returns: double] (polymorphic)
- GetTatkalLoadFactor :
    - [Params: none]
    - [Returns: double] polymorphic)
- GetMinimumTatkalCharge :
    - [Params: none]
    - [Returns: double] (polymorphic)
- GetMaximumTatkalCharge :
    - [Params: none]
    - [Returns: double](polymorphic)
- GetMinimumTatkalDistance :
    - [Params: none]
    - [Returns: int] (polymorphic)

SUB-TYPES (Singletons)

Data Members
static:
private:

- sName : const string
- sReservationCharge : const double
- sTatkalLoadFactor : const double
- sMinimumTatkalCharge : const double
- sMaximumTatkalCharge : const double
- sMinimumTatkalDistance : const int
- sAC : const bool
- sLuxury : const bool
- sSitting : const bool
- sNumberOfTiers : const int
- sLoadFactor : const double

Methods
The pure virtual functions from the base class are implemented by returning the appropriate static attribute. A static function to access each of the singleton sub-types is added.
static:

- Type :
    - [Params: none]
    - [Returns: BookingClassesTypes&] (public) - static function to access singleton booking class

STATION

Data Members
NON-STATIC:

- name_: const string (private)

Methods
NON-STATIC:

- GetName : const
    - [Params: none]
    - [Returns: string] (public)
- GetDistance: const
    - [Params: Station]
    - [Returns: int] (public)

DIVYAANG

They are designed as a single-level flat hierarchy with parametric polymorphism.

The sub-types are:

- Blind
- CancerPatient
- TBPatient
- OrthopaedicallyHandicaped

ABSTRACT BASE CLASS

Data Members (non statics)

- name_ : const string (private)

Methods (non statics)

- GetName : const
    - [Params: none]
    - [Returns: string] (public)
- The following public function is overloaded for 8 different BookingClassesTypes
    - GetConcessionFactor : const
        - [Params: BookingClassesTypes]
        - [Returns: double] (polymorphic)

sub-types (Singletons)

Data Members (static constants)
private:

- sName : string
- sACFirstClassConcession : double
- sExecutiveChairCarConcession : double
- sAC2TierConcession : double
- sFirstClassConcession : double
- sAC3TierConcession : double
- sACChairCarConcession : double
- sSleeperConcession : double
- sSecondSittingConcession : double

Methods
The pure virtual functions from the base class are implemented by returning the appropriate static attribute. A public static function to access each of the singleton sub-types is added.

- Type :
    - [Params: none]
    - [Returns: DivyaangTypes] - static function representing singleton divyaang

CONCESSION

They are designed as a single-level flat hierarchy rooted at Concession with ad-hoc polymorphism.

The subtypes are all singletons as follows.

1. DivyaangConcession
   Methods (statics)

- Type :
    - [Params: none]
    - [Returns: DivyaangConcession&] (public) - static function to access singleton

Methods (non-statics)

- GetConcessionFactor : const
    - [Params: Passenger, BookingClasses]
    - [Returns: double]

2. SeniorCitizenConcession
   Methods (statics)
   - Type :
       - [Params: none]
       - [Returns: SeniorCitizenConcession&] (public) - static function to access singleton

   Methods (non-statics)
   - GetConcessionFactor : const
       - [Params: Passenger]
       - [Returns: double] (public)

3. LadiesConcession
   Members
   static:
   - sConcessionFactor : const double (private)

   Methods
   static:
   - Type : [Params: none] [Returns: LadiesConcession&] (public) - static function to access singleton

   non-static:
   - GetConcessionFactor : const [Params: none] [Returns: double] (public)

4. GeneralConcession
   Members
   static:
   - sConcessionFactor : const double (private)

   Methods
   static:
   - Type :
       - [Params: none]
       - [Returns: GeneralConcession&] (public) - static function to access singleton

   non-static:
   - GetConcessionFactor : const

- [Params: none]
- [Returns: double] (public)

PASSENGER

Members (static constants and private)

- name_ : string
- dateOfBirth_: Date
- gender_: Gender
- aadhar_: string
- mobile_: string
- disabilityType_: Divyaang
- disabilityID_: string

Methods (non-static const functions)
GetName(), GetDateOfBirth(), GetGender(), GetDisabilityType(), and GetDisabilityID()

A static method CreateNewPassenger to handle exceptions in passenger data.

BOOKING

They are designed as a single-level flat hierarchy with parametric polymorphism.

The sub-types are:

- GeneralBooking
- LadiesBooking
- SeniorCitizenBooking
- DivyaangBooking
- TatkalBooking
- PremiumTatkalBooking

BASE CLASS (Abstract)

Data Members
static:
protected:

- sBookings: vector<Booking *>

- sBaseFarePerKM: const double
- sBookingPNRSerial: int

non-static:
protected:

- pnr_ : const int
- dateOfBooking_ : const Date
- dateOfReservation_ : const Date
- fromStation_ : const Station
- toStation_ : const Station
- bookingClass_ : const BookingClasses
- bookingCategory_ : const BookingCategory
- passenger_ : const Passenger
- fare_ : int
- bookingStatus_ : bool
- bookingMessage_ : string

Methods
static:

- MakeReservation :
  - [Params: …]
  - [Returns: none] (public)
- PrintBookings:
  - [Params: none]
  - [Returns: none] (public)

non-static:

- ComputeFare: const
  - [Params: none]
  - [Returns: int] (protected, virtual, pure)

SUB-TYPES **(Singletons)**

Methods
The pure virtual function ComputeFare from the base class is implemented with the appropriate logic based on the sub-type.

BOOKING CATEGORY

They are designed as a single-level flat hierarchy rooted at BookingCategory with parametric polymorphism.

The sub-types are as follows:

- General
- Ladies
- SeniorCitizen
- Divyaang
- Tatkal
- PremiumTatkal

BASE CLASS (Abstract)

Data Members
non-static:
private:

- name_ : const string

Methods
non-static:
public:

- GetName : const
  - [Params: none]
  - [Returns: string]
- CheckEligibility : const
  - [Params: Passenger]
  - [Returns: bool] (polymorphic)
- MakeReservation : const
  - [Params: …]
  - [Returns: none] (polymorphic)

SUB-TYPES

Data Members
static:
private:

- sName : const string

Methods
The pure virtual functions from the base class are implemented with the appropriate logic based on the sub-type. A static function to access each of the singleton sub-types is added.
STATIC:

- Type : [Params: none] [Returns: BookingCategoryTypes] (public) - static function to access singleton booking category type

K. 'Exceptions' classes

Bad_Name class

- virtual string what()

  Returns error message

Name_Too_Short class

- string what()

  Returns "Name too short" if first name or last name or both are missing

Bad_Date class

- virtual string what()

  Returns error message

Date_Out_Of_Bound class

- string what()

  Returns "Date out of bound" if the date is invalid

DOB_Too_Late

- string what()

  Returns "Date of Birth invalid" if it is a future date

Bad_Station class

- virtual string what()

  Returns error message

Name_Empty class

- string what()

Returns "Empty Station Name" if name of station is empty

Bad_Railway

- virtual what()

Returns error message

Duplicate_Station class

- string what()

Returns "Duplicate Stations" if there are any duplicate stations in master data

Valid_Stations class

- uses Bad_Station class to check if all Stations in master data are valid

Bad_Passenger

- virtual what()

Returns error message

Valid_Name class

- uses Bad_Name class to check if the name is valid

Valid_DOB class

- uses Bad_Date class to check if the date of birth is valid

Invalid_Aadhar class

- string what()

Returns "Invalid Aadhar" if there is any non-digit character or the length of aadhar is not 12

Invalid_Mobile class

- string what()

Returns "Invalid Mobile" if there is any non-digit character or the length of mobile is not 10

Invalid_Disability class

- string what()

Returns "Invalid Disability Type" if the disability type is not valid

Bad_Booking class

- virtual what()

Returns error message

Invalid_From_Station class

- string what()

Returns "Invalid From Station" if the from station is not present in master data of Railways class

Invalid_To_Station class

- string what()

Returns "Invalid To Station" if the to station is not present in master data of Railways class

Invalid_Reservation_Date

- string what()

Returns "Invalid date of reservation" if the date is from future

Invalid_Booking_Date

- string what()

Returns "Invalid date of booking" if the date is before or one year after reservation date

Valid_Passenger class

- uses Bad_Passenger class to check if the passenger is valid