DESIGN FOR RAILWAY BOOKING SYSTEM

Name: Nisarg Upadhyaya
Roll No: 19CS30031

- STATION

  Data Members
  NON-STATIC:
  1. name_ : const string (private)

  Methods
  NON-STATIC:
  1. GetName : const [Params: none] [Returns: string] (public)
  2. GetDistance: const [Params: Station] [Returns: int] (public)

- RAILWAYS (Singleton)

  Data Members
  STATIC:
  1. sStations : vector<const Station*> (private)
  2. sDistStations: map<pair<string,string>, int> (private)

  Methods
  STATIC:
  1. IndianRailways : [Params: none] [Returns: Railways] (public) - static function to access singleton Railways

  NON-STATIC:
  1. GetDistance : const [Params: Station, Station] [Returns: int] (public)
  2. GetStation : const [Params: string] [Returns: station] (public)

- DATE

  Data Members
  STATIC:
  1. sMonthNames : const vector<string> (private)

  NON-STATIC:
  1. date_ : const int (private)
  2. month_ : const int (private)
  3. year_ : const int (private)

Methods

STATIC:

1. CreateDate : [Params: int, int, int] [Returns: Date] (public)
   Static function which does appropriate error handling before creating a date.

FRIEND:

1. operator- : [Params: date1, date2] Returns an integer which gives the number of days between two dates, is +ve if date1 comes after date2. This operator alone allows various other operations like comparing dates, finding span.

- BOOKING CLASSES

Designed as a single level flat hierarchy rooted at BookingClasses with parametric polymorphism.

The following structs are included in base class which act as tag-types (the respective sub-types are shown in brackets)

- ACFirstClassType (ACFirstClass)
- ExecutiveChairCarType (ExecutiveChairCar)
- AC2TierType (AC2Tier)
- FirstClassType (FirstClass)
- AC3TierType (AC3Tier)
- ACChairCarType (ACChairCar)
- SleeperType (Sleeper)
- SecondSittingType (SecondSitting)

BASE CLASS (Abstract)

Data Members

NON-STATIC:

1. name_ : const string (private)

Methods

NON-STATIC:

1. GetName : const [Params: none] [Returns: string] (public)
2. IsAC : const [Params: none] [Returns: bool] (public, virtual, pure)
3. IsAC : const [Params: none] [Returns: bool] (public, virtual, pure)
4. IsAC : const [Params: none] [Returns: bool] (public, virtual, pure)
5. GetNumberOfTiers : const [Params: none] [Returns: integer] (public, virtual, pure)
6. GetLoadFactor : const [Params: none] [Returns: double] (public, virtual, pure)
7. GetReservationCharge : const [Params: none] [Returns: double] (public, virtual, pure)
8. GetTatkalLoadFactor : const [Params: none] [Returns: double] (public, virtual, pure)

9. GetMinimumTatkalCharge : const [Params: none] [Returns: double] (public, virtual, pure)
10. GetMaximumTatkalCharge : const [Params: none] [Returns: double] (public, virtual, pure)
11. GetMinimumTatkalDistance : const [Params: none] [Returns: integer] (public, virtual, pure)

SUB-TYPES (Singletons)

Data Members
STATIC:

1. sName : const string (private)
2. sAC : const boolean (private)
3. sLuxury : const boolean (private)
4. sSitting : const boolean (private)
5. sNumberOfTiers : const integer (private)
6. sLoadFactor : const double (private)
7. sReservationCharge : const double (private)
8. sTatkalLoadFactor : const double (private)
9. sMinimumTatkalCharge : const double (private)
10. sMaximumTatkalCharge : const double (private)
11. sMinimumTatkalDistance : const integer (private)

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:

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

- DIVYAANG

Designed as a single level flat hierarchy rooted at Divyaang with parametric polymorphism.

The following structs are included in base class which act as tag-types (the respective sub-types are shown in brackets)

- BlindType (Blind)
- CancerPatientType (CancerPatient)
- TBPatientType (TBPatient)
- OrthopaedicallyHandicapedType (OrthopaedicallyHandicaped)

BASE CLASS (Abstract)

Data Members
NON-STATIC:

1. name_ : const string (private)

Methods
NON-STATIC:

1. GetName : const [Params: none] [Returns: string] (public)
2. The following function is overloaded for different BookingClassesTypes
   GetConcessionFactor : const [Params: BookingClassesTypes] [Returns: double] (public, virtual, pure)
   8 overloads for each of the 8 booking types

SUB-TYPES (Singletons)

Data Members
STATIC:

1. sName : const string (private)
2. sACFirstClassConcession : const double (private)
3. sExecutiveChairCarConcession : const double (private)
4. sAC2TierConcession : const double (private)
5. sFirstClassConcession : const double (private)
6. sAC3TierConcession : const double (private)
7. sACChairCarConcession : const double (private)
8. sSleeperConcession : const double (private)
9. sSecondSittingConcession : const double (private)

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:

1. Type : [Params: none] [Returns: DivyaangTypes] (public) - static function to access singleton divyaang type

- CONCESSION

Designed as a single level flat hierarchy rooted at Concession with ad-hoc polymorphism.

The subtypes are all singletons as follows

1. DivyaangConcession
   Methods
   STATIC:
   1. Type : [Params: none] [Returns: DivyaangConcession] (public) - static function to access singleton

   NON-STATIC:
   1. GetConcessionFactor : const [Params: Passenger, BookingClasses] [Returns: double] (public)

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

   NON-STATIC:
   1. GetConcessionFactor : const [Params: Passenger] [Returns: double] (public)

3. LadiesConcession
   Members
   STATIC:
   1. sConcessionFactor : const double (private)

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

   NON-STATIC:
   1. GetConcessionFactor : const [Params: none] [Returns: double] (public)

4. GeneralConcession
   Members
   STATIC:
   1. sConcessionFactor : const double (private)

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

   NON-STATIC:
   1. GetConcessionFactor : const [Params: none] [Returns: double] (public)

- PASSENGER

Members

NON-STATIC (all private and const)

1. name_ : string
2. dateIfBirth_ : Date
3. gender_ : Gender
4. aadhar_ : string
5. mobile_ : string
6. disabilityType_ : Divyaang
7. disabilityID_ : string

Methods

NON-STATIC const Get methods for Name, DateOfBirth, Gender, DisabilityType and DisabilityID

STATIC CreatePassenger for exception handling before invoking constructor

Note: Booking Category and Booking use structs from a common namespace **categories** as tag-types. The namespace has the following structs

1. GeneralType
2. LadiesType
3. SeniorCitizenType
4. DivyaangType
5. TatkalType
6. PremiumTatkalType

- BOOKING CATEGORY

Designed as a single level flat hierarchy rooted at BookingCategory with parametric polymorphism.

The sub-types are as follows:

1. General
2. Ladies
3. SeniorCitizen
4. Divyaang
5. Tatkal
6. PremiumTatkal

BASE CLASS (Abstract)

Data Members

NON-STATIC:

1. name_ : const string (private)

Methods
NON-STATIC:

1. GetName : const [Params: none] [Returns: string] (public)
2. CheckEligibility : const [Params: Passenger] [Returns: boolean] (public, virtual, pure)
3. MakeReservation : const [Params: …] [Returns: none] (public, virtual, pure)

SUB-TYPES (Singletons)

Data Members
STATIC:

1. sName : const string (private)

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:

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

- BOOKING

Designed as a single level flat hierarchy rooted at Booking with parametric polymorphism.

The sub-types are as follows:

1. GeneralBooking
2. LadiesBooking
3. SeniorCitizenBooking
4. DivyaangBooking
5. TatkalBooking
6. PremiumTatkalBooking

BASE CLASS (Abstract)

Data Members
STATIC:

1. sBookings: vector<Booking *> (protected)
2. sBaseFarePerKM: const double (protected)
3. sBookingPNRSerial: integer (protected)

NON-STATIC:

1. pnr_ : const integer (protected)
2. dateOfBooking_ : const Date (protected)
3. dateOfReservation_ : const Date (protected)
4. fromStation_ : const Station (protected)
5. toStation_ : const Station (protected)
6. bookingClass_ : const BookingClasses (protected)
7. bookingCategory_ : const BookingCategory (protected)
8. passenger_ : const Passenger (protected)
9. fare_ : integer (protected)
10. bookingStatus_ : boolean (protected)
11. bookingMessage_ : string (protected)

Methods
STATIC:

1. MakeReservation : [Params: …] [Returns: none] (public)
2. PrintBookings: [Params: none] [Returns: none] (public)

NON-STATIC:

1. 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.

- EXCEPTIONS

1. Bad_Date
    a. YearOutOfLimit
    b. InvalidDate

2. Bad_Station
    a. EmptyName
    b. AlreadyExists

3. Bad_Booking
    a. InvalidStation
        i. SameStations
        ii. StationNotFound
    b. InvalidDates
        i. InvalidTatkal

    ii. MoreThan1Year
    iii. PastDate
c. Ineligible
    i. DivyaangIneligible
    ii. SeniorCitizenIneligibleMale
    iii. SeniorCitizenIneligibleFemale
    iv. LadiesIneligble