DESIGN

Convention:

Class Attributes
NameOfAttribute{(AccessSpecifier(private,protected,public),Encapsulation(static/non-static),Data Type(string,long long int,double,int)}

Class Member Functions: Name_of_Function():
(AccessSpecifier(private,protected,public,friend),Encapsulation(static/non-static/const static/virtual),Return Type(string,long long int,double,int,void))---Use of the function

Design Plan for All the Classes:

1)Station(Non-Polymorphic)

Class Attributes:name_(private,non-static,string)

Class Member Functions:
Station(string name):
-(public,non-static)--parameterized Constructor
GetName():
-(public,non-static,string)--returns the name of station
GetDistance(const Station &s):
-(public,non-static,float)--returns the distance between two stations
operator<<(ostream &os, const Station &st):
-(**friend**,non-static,ostream &)--output streaming operator to help output process as well as debugging
Station& operator= (const Station& f) :
-(public,non-static,Station&)--copy assignment operator
static void UTStation() :
-(public,static,void)--static unit testing function
friend bool operator==(const Station& a, const Station& b)
-(**friend**,non-static,bool)--Equal To Operator(Utilization Operator)
friend bool operator<(const Station& a, const Station& b):
-(**friend**,non-static,bool)--Less-Than To Operator(Utilization Operator)
~Station():
-(public,non-static)--Destructor

2)Railways (Singleton Class)(Non-Polymorphic)

Class Attribute :
vector<Station>sStation(public,static,Station);map<Station,int>sHashMap(public,static,<Station,int>);map<pair<int,int>,double>sDistance(public,static,pair<int,int>,double); static Railways *sIndianRailways(public,static,Railways*);

Class Member Functions:
Railways();
    -(private,non-static)-non-parameterized
static Railways &GetRailways():
    -(public,static,Railways&)--Get instance of singleton class
friend ostream &operator<<(ostream &os, const Railways &r):
    -(**friend**,non-static,ostream&)--output ostream
double GetDistance(const Station& a, const Station& b):
    -(public,non-static,double)--Distance between two stations
Railways &operator=(const Railways &r):
    -(public,non-static,Railways&)--Copy Assignment operator
void printStations(vector<Station>sStation):
    -(public,non-static,void)--Utilization functions
void printDistance(map<pair<int,int>,double>sDistStations):
    -(public,non-static,void)-Utilization functions
static void UTRailways():
    -(public,static,void)--Unit Testing functions
~Railways():
    -(private,non-static)-Destructor

3)Date(Non-Polymorphic)

Class
Attribute:monthNames(private,const,char[][]),dayNames(private,const,char[][]),date_(private,non-static,unsigned int),month_,(private,non-static,enum:month)year_(private,non-static,unsigned int)

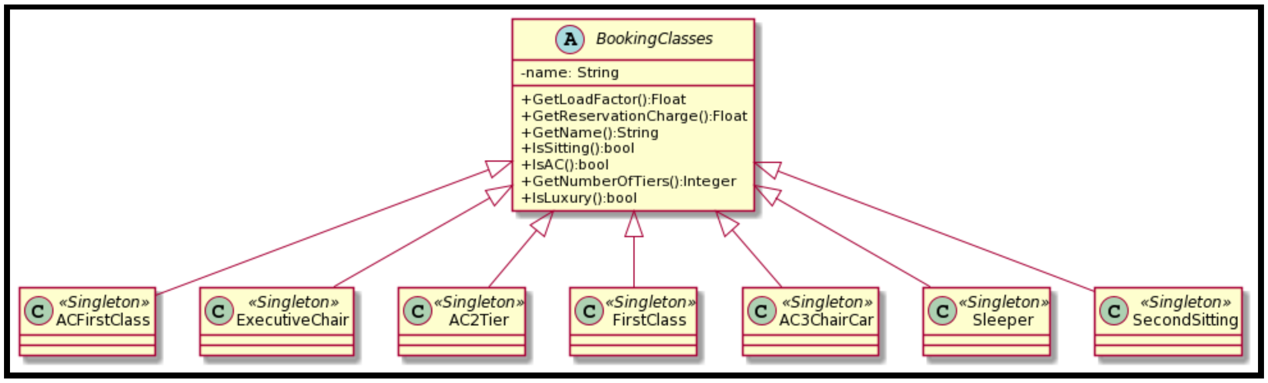
Class Member Functions:
Date(UINT d, UINT m, UINT y) :
    -(public,non-static)--parameterized Constructor
friend ostream &operator<<(ostream &os, const Date &date):
    -(**friend**,non-static,ostream&)--output ostream
bool validDate():
    -(public,non-static,bool)--Checks if a given date is valid or not
int getDays():
    -(public,non-static,integer)-Returns the number of days from 0/00/0000
static void UTDate():

-(public,static,void)--Unit Testing functions
~Date():
-(private,non-static)-Destructor

4)BookingClasses(Abstract Class) Polymorphic Hierarchy

Class Attributes:None

Class Member Functions:
BookingClasses():
-(public, non-static,non-virtual)-- **inline defined**(Constructor)
~BookingClasses()
- (public,non-static,virtual)-- **inline defined** (Destructor)
GetLoadFactor():
-(public,non-static,virtual,float)--Gets Load Factor
GetName():
- (public,non-static,virtual,string) --Gets Name
IsSitting():
- (public,non-static,virtual,boolean)--Return status of sitting
IsAC():
- (public, non-static,virtual,boolean)-- Return status of AC
GetNumberOfTiers():
- (public, non-static, virtual,int)-- Get number of Tiers
IsLuxury():
-(public,non-static,virtual,boolean)-- Status of Luxury
Here the **single-level polymorphic hierarchy** is rooted at BookingClasses which is an abstract base class.Here there is one level of single inheritance which has 7 concrete classes.
Here static sub-typing polymorphism with inclusion and parametric polymorphism is used .
Then we have 7 concrete classes . Every concrete booking class has all fixed properties and there is no need to construct more than one object for any of them. So there is a singleton constant object for
each which, kind of, will stand for its polymorphic type.
The 7 concrete classes are-
1)AC2Tier
2)AC3Tier
3)ACChairCar
4)ACFirstClass
5)FirstClass
6)SecondSiiting
7)Sleeper



<u>UML Class Diagram</u>

5)Divyaang (Abstract) Polymorphic Hierarchy

Class Attribute:sName(private,static,const string),sConcessionFactor(private,static,map<const BookingClasses&, float>)

Class Member Functions:
DivyaangType(const string& name):
 -(private,non-static)-parameterized Constructor
~DivyaangType():
 -(private,non-static)-parameterized Destructor
GetConcessionFactor(const BookingClasses& bookingClass):
 -(public,non-static,float)-returns Concession Factor

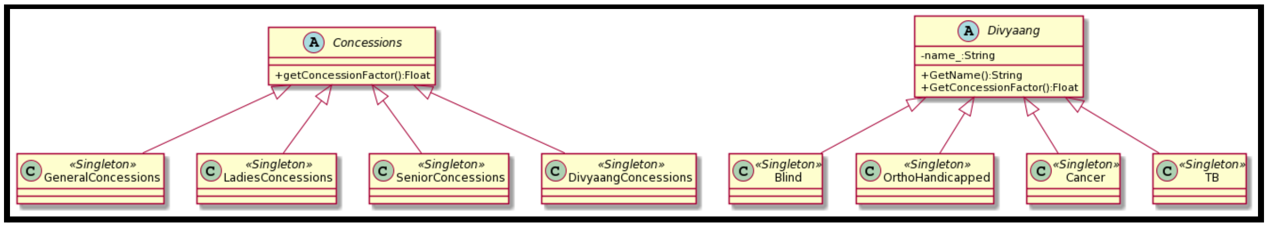
Divyaang has a **<u>single-level polymorphic hierarchy</u>**.Here there is one level of single inheritance which has 4 concrete classes.
Here static sub-typing polymorphism with inclusion and parametric polymorphism is used .
Then we have 4 concrete classes . Every concrete class has all fixed properties and there is no need to construct more than one object for any of them. So there is a singleton constant object for
each which, kind of, will stand for its polymorphic type.
The 4 concrete classes are-
1)Blind
2)OrthoHandicapped
3)CancerPatient
4)TBPatient

6)Concession (Abstract) Polymorphic Hierarchy

Class Attribute:None

Class Member Functions:
1)Concessions():
	(public,non-static):Constructor
2)~Concessions():
	(public,non-static): virtual Destructor
3)getConcessionFactor():
	(public,non-static,float):virtual function
4)isEligible():
	(public,non-static,bool):virtual function

Concession has a **single-level polymorphic hierarchy.**Here there is one level of single inheritance which has 4 concrete classes.
Here static sub-typing polymorphism with inclusion and parametric polymorphism is used .
Then we have 4 concrete classes . Every concrete class has all fixed properties and there is no need to construct more than one object for any of them. So there is a singleton constant object for
each which, kind of, will stand for its polymorphic type.
The 4 concrete classes are-
1)GeneralConcession
2)LadiesConcession
3)SeniorCitizenConcession
4)DivyaangConcession


Concessions
+getConcessionFactor():Float
«Singleton» GeneralConcessions
«Singleton» LadiesConcessions
«Singleton» SeniorConcessions
«Singleton» DivyaangConcessions
Divyaang
-name_:String
+GetName():String
+GetConcessionFactor():Float
«Singleton» Blind
«Singleton» OrthoHandicapped
«Singleton» Cancer
«Singleton» TB

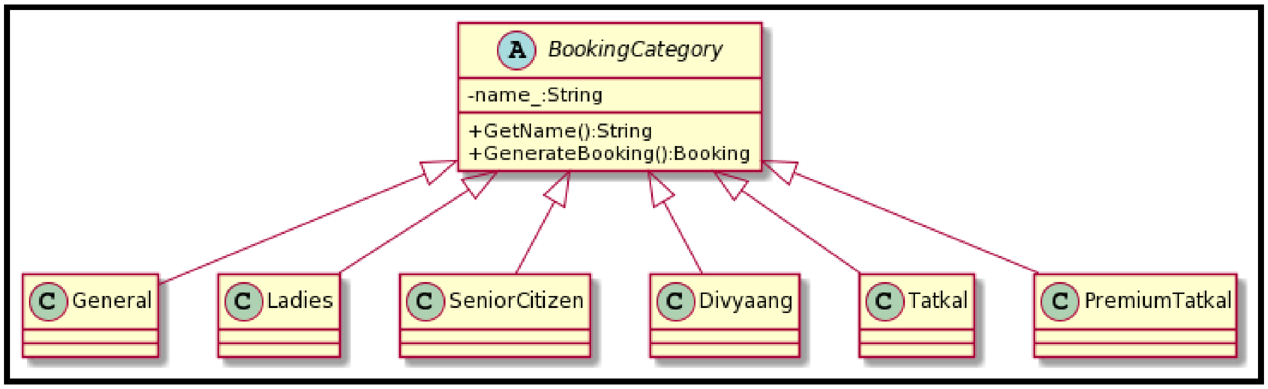
UML Class Diagram

7)Class Name:BookingCategory

Class Attribute:name_(private,non-static,string const)

Class Member Functions:
1)GetName():
	(public,non-static,string):virtual function
2) Booking& GenerateBooking():
	(public,non-static,Booking&):virtual function
BookingCategory has a **single-level polymorphic hierarchy**.Here there is one level of single inheritance which has 6 concrete classes.
Here static sub-typing polymorphism with inclusion and parametric polymorphism is used .

Then we have 6 concrete classes . Every concrete class has all fixed properties and there is no need to construct more than one object for any of them. So there is a singleton constant object for
each which, kind of, will stand for its polymorphic type.
The 6 concrete classes are-
1)General
2)Ladies
3)SeniorCitizen
4)Divyaang
5)Tatkal
6)PremiumTatkal



UML Class Diagram

8)Passenger (Non-Abstract) Non-Polymorphic

Class Attribute:
firstName_(private,non-static,string),middleName_(private,non-static,string),LastName_(private, non-static,string), aadharNumber_(private,non-static,string),Date dob_(private,non-static,Date), gender_(private,non-static,Gender),string mobileNumber_(private,non-static,string), disabilityID_(private,non-static,string),disabilityType_(private,non-static,Divyaang&);

Class Member Functions:
Passenger(string name_,string aadharNumber_,Date dob_,string gender_,string mobileNumber_,string category_):
    -(public,non-static)-parameterized Constructor
Passenger(const Passenger &p):
    -(public,non-static,Passenger)-Copy Assignment operator
friend ostream &operator<<(ostream &os, const Passenger &p):
    -(friend,non-static,ostream&)--output ostream
~Passenger():
    -(public,non-static)-Destructor
GetGender():

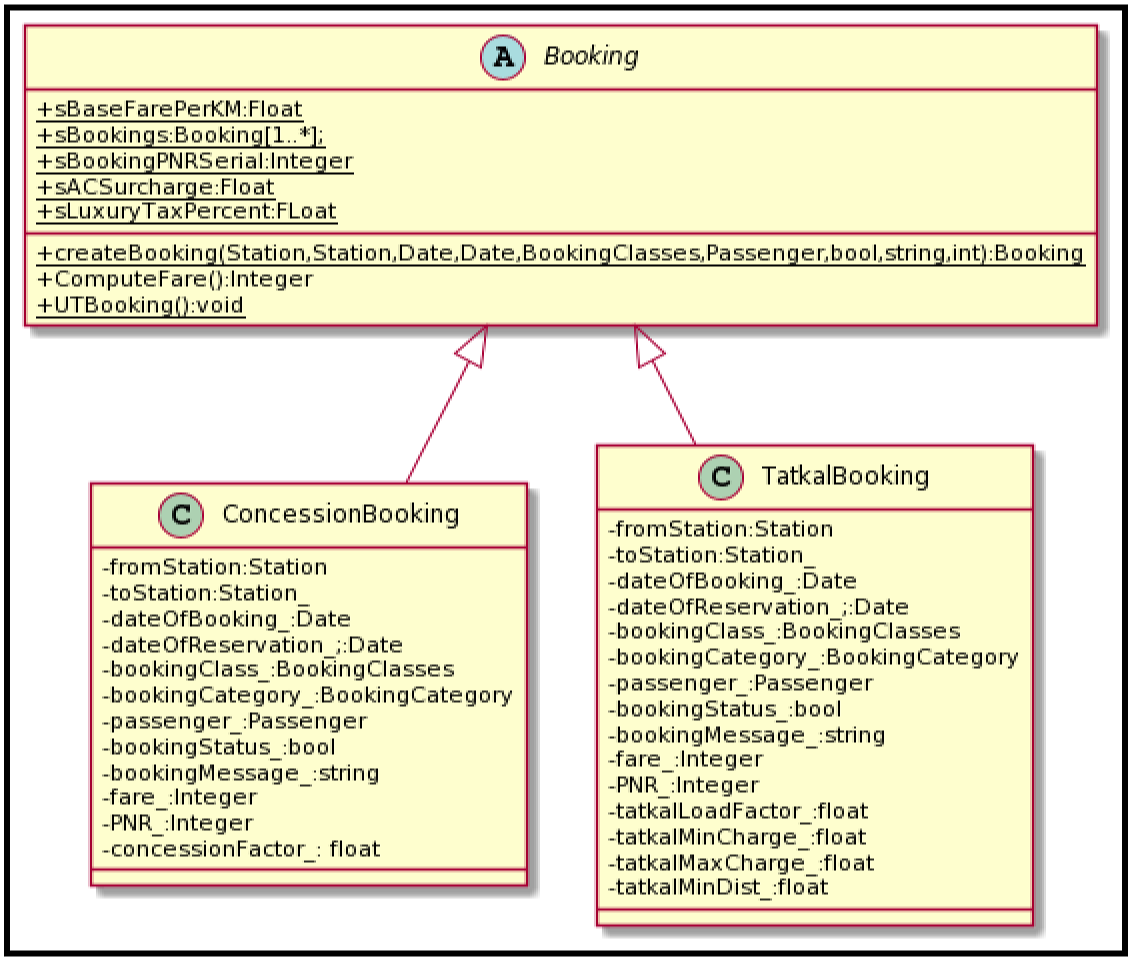
-(public,non-static,Gender):returns Gender
GetAge():
-(public,non-static,int):returns age
static void UTPassenger()
-(public,static,void)--Unit Testing functions

9)Booking (Abstract) Polymorphic hierarchy

Class Attributes:
sBaseFarePerKM(public,static const,double); vector<Booking> sBookings(public,static ,Booking); sBookingPNRSerial(public,static,long long int); sACSurcharge(public,static const,double);
sLuxuryTaxPercent(public,static const,double);

Class Member Functions:
Booking(Station fromStation,Station toStation,Date date,BookingClasses &bookingClass,Passenger *passenger,bool bookingStatus,string bookingMessage,double fare)
-(public,non-static)-parameterized Constructor
Booking(const Booking& f):
-(public,non-static,Passenger)-Copy Assignment operator
long long int ComputeFare():
-(public,non-static,long long int) Compute the fare
friend ostream &operator<<(ostream &os, const Booking &bk)
-(friend,non-static,ostream&)--output ostream
static void UTBooking():
-(public,static,void)--Unit Testing functions
~Booking():
-(private,non-static)-Destructor

Booking has a **single-level polymorphic hierarchy**.Here there is one level of single inheritance which has 2 concrete classes.
Here static sub-typing polymorphism with inclusion and parametric polymorphism is used .
Then we have 2 concrete classes . Every concrete class has all fixed properties and there is no need to construct more than one object for any of them. So there is a singleton constant object for
each which, kind of, will stand for its polymorphic type.
The 2 concrete classes are-
1)Concession Booking
2)Tatkal Booking



UML Class Diagram

10)Exceptions (Abstract) Polymorphic Hierarchy

This is a class to throw errors in case of invalid input.It has four sub-classes classes.

1)class BadDate:
Class Member Function:
- what():
  (public,none,const char*)- (In Case of Invalid Date)

2)class BadStation:
Class Member Function:
- what():
  (public,none,const char*)- (In Case of Invalid Station)

3)class BadPassenger: public Exception
Class Member Function:
- what():

(public,none,const char*)- (In Case of Inavlid Passenger)

4)class BadRailways:Abstract class

Class Member Function:

- what():

    (public,none,const char*)- (In Case of exception in Railways)

BadRailways is also a abstract class with three subclasses

Hierarchy for BadRailways:

class DuplicateStation: public BadRailways

Class Member Function:

- what():

    (public,none, const char*)-(In case of Duplicate Station)

class DuplicateDistance: public BadRailways

Class Member Function:

- what():

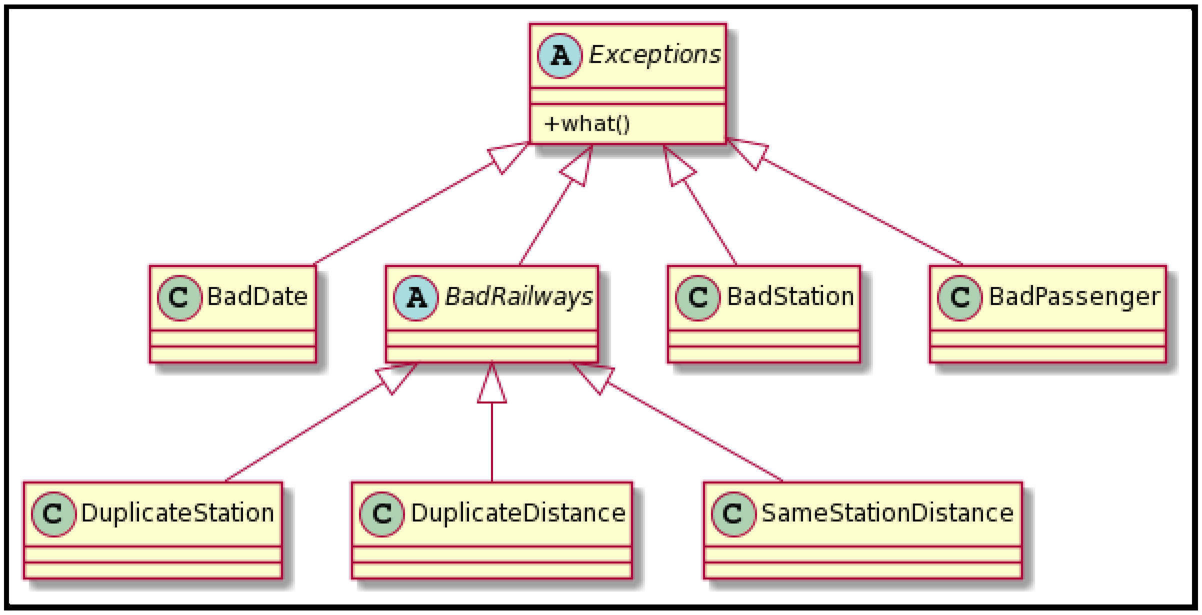
    (public,none, const char*)-(In case of Wrong Distance)

class SameStationDistance: public BadRailways

Class Member Function:

- what():

    (public,none, const char*)-(In case of Distance between same stations)



UML Class Diagram

Name:Yashica Patodia
Roll No:19CS10067