

TEST REPORT

for

Railway Booking Software

Prepared by -
Suhas Jain (19CS30048)

Assignment - 5
Software Engineering (CS20006)

Indian Institute of Technology, Kharagpur

April 9, 2021

Contents

1	Testing the Stations class	7
1.1	Testing the constructor	7
1.2	Testing the static <code>Station& CreateStation(const string&)</code> function	7
1.3	Testing the <code>string GetName()</code> const function	7
1.4	Testing the <code>int GetDistance(const Station&)</code> const function	7
1.5	Testing the friend <code>ostream& operator<<(ostream&, const Station&)</code> function	7
1.6	Testing the static <code>Station& CreateStation(const string&)</code> function	7
2	Testing the Railways class	7
2.1	Testing the constructor	7
2.2	Testing the <code>int GetDistance(const Station&, const Station&)</code> function	7
2.3	Testing the <code>Station GetStation(const string& name)</code> const function	7
2.4	Testing the friend <code>ostream& operator<<(ostream&, const Railways&)</code> function	8
2.5	Testing the <code>Station GetStation(const string& name)</code> const function	8
3	Testing the Date class	8
3.1	Testing the constructor	8
3.2	Testing the copy constructor <code>Date(const Date&)</code>	8
3.3	Testing the <code>Day day()</code> const function	8
3.4	Testing the friend <code>int operator-(const Date&, const Date&)</code> function	8
3.5	Testing the <code>bool operator>(const Date&)</code> function	8
3.6	Testing the <code>bool operator==(const Date&)</code> function	8
3.7	Testing the friend <code>ostream& operator<<(ostream&, const Date&)</code> function	8
4	Testing the Name class	9
4.1	When First, Middle and Last name are present	9
4.2	When First and Last name are present	9
4.3	When Last name is present	9
4.4	When First name is present	9
4.5	When First and Middle are present	9
4.6	When Middle and Last name are present	9
4.7	When only middle name is present	9
4.8	When none of the names are present	9
5	Testing the Gender class and hierarchy	9
5.1	Testing <code>Male</code> derived class made using template <code>GenderTypes;T_i</code>	9
5.1.1	Testing the constructor	9
5.1.2	Testing the <code>const string GetName()</code> const function	10
5.1.3	Testing the <code>const string GetTitle()</code> const function	10
5.1.4	Testing the friend <code>ostream& operator<<(ostream&, const Gender&)</code> function	10
5.2	Testing <code>Female</code> derived class made using template <code>GenderTypes;T_i</code>	10
5.2.1	Testing the constructor	10
5.2.2	Testing the <code>const string GetName()</code> const function	10
5.2.3	Testing the <code>const string GetTitle()</code> const function	10
5.2.4	Testing the friend <code>ostream& operator<<(ostream&, const Gender&)</code> function	10

6	Testing the Passenger class	10
6.1	Testing the static Passenger& CreatePassenger(const Name, const Date, const Gender&, const string&, const string&, const Divyaang&, const string&) function	10
6.2	Testing the const Date GetDateOfBirth() const function	10
6.3	Testing the const Gender& GetGender() const function	10
6.4	Testing the const Divyaang& GetDisability() const function	11
7	Testing the BookingClass class and hierarchy	11
7.1	Testing ACFirstClass the derived class modelled using the template BookingClassType<T>	11
7.1.1	Testing the constructor	11
7.1.2	Testing the bool IsAC() const function	11
7.1.3	Testing the bool IsLuxury() const function	11
7.1.4	Testing the bool IsSitting() const function	11
7.1.5	Testing the double GetLoadFactor() const function	11
7.1.6	Testing the int GetNumberOfTiers() const function	11
7.1.7	Testing the double GetReservationCharge() const function	11
7.1.8	Testing the double GetTatkalFactor() const function	11
7.1.9	Testing the double GetTatkalMinCharge() const function	11
7.1.10	Testing the double GetTatkalMaxCharge() const function	12
7.1.11	Testing the int GetMinTatkalDistance() const function	12
7.2	Testing the ExecutiveChairCar derived class modelled using the template BookingClassType<T>	12
7.2.1	Testing the constructor	12
7.2.2	Testing the bool IsAC() const function	12
7.2.3	Testing the bool IsLuxury() const function	12
7.2.4	Testing the bool IsSitting() const function	12
7.2.5	Testing the double GetLoadFactor() const function	12
7.2.6	Testing the int GetNumberOfTiers() const function	12
7.2.7	Testing the double GetReservationCharge() const function	12
7.2.8	Testing the double GetTatkalFactor() const function	12
7.2.9	Testing the double GetTatkalMinCharge() const function	12
7.2.10	Testing the double GetTatkalMaxCharge() const function	12
7.2.11	Testing the int GetMinTatkalDistance() const function	13
7.3	Testing the AC2Tier derived class modelled using the template BookingClassType<T> . .	13
7.3.1	Testing the constructor	13
7.3.2	Testing the bool IsAC() const function	13
7.3.3	Testing the bool IsLuxury() const function	13
7.3.4	Testing the bool IsSitting() const function	13
7.3.5	Testing the double GetLoadFactor() const function	13
7.3.6	Testing the int GetNumberOfTiers() const function	13
7.3.7	Testing the double GetReservationCharge() const function	13
7.3.8	Testing the double GetTatkalFactor() const function	13
7.3.9	Testing the double GetTatkalMinCharge() const function	13
7.3.10	Testing the double GetTatkalMaxCharge() const function	13
7.3.11	Testing the int GetMinTatkalDistance() const function	13
7.4	Testing the FirstClass derived class modelled using the template BookingClassType<T>	13
7.4.1	Testing the constructor	14
7.4.2	Testing the bool IsAC() const function	14
7.4.3	Testing the bool IsLuxury() const function	14
7.4.4	Testing the bool IsSitting() const function	14

7.4.5	Testing the double <code>GetLoadFactor()</code> const function	14
7.4.6	Testing the int <code>GetNumberOfTiers()</code> const function	14
7.4.7	Testing the double <code>GetReservationCharge()</code> const function	14
7.4.8	Testing the double <code>GetTatkalFactor()</code> const function	14
7.4.9	Testing the double <code>GetTatkalMinCharge()</code> const function	14
7.4.10	Testing the double <code>GetTatkalMaxCharge()</code> const function	14
7.4.11	Testing the int <code>GetMinTatkalDistance()</code> const function	14
7.5	Testing the <code>AC3Tier</code> derived class modelled using the template <code>BookingClassType<T></code> . .	14
7.5.1	Testing the constructor	14
7.5.2	Testing the bool <code>IsAC()</code> const function	14
7.5.3	Testing the bool <code>IsLuxury()</code> const function	15
7.5.4	Testing the bool <code>IsSitting()</code> const function	15
7.5.5	Testing the double <code>GetLoadFactor()</code> const function	15
7.5.6	Testing the int <code>GetNumberOfTiers()</code> const function	15
7.5.7	Testing the double <code>GetReservationCharge()</code> const function	15
7.5.8	Testing the double <code>GetTatkalFactor()</code> const function	15
7.5.9	Testing the double <code>GetTatkalMinCharge()</code> const function	15
7.5.10	Testing the double <code>GetTatkalMaxCharge()</code> const function	15
7.5.11	Testing the int <code>GetMinTatkalDistance()</code> const function	15
7.6	Testing the <code>ACChairCar</code> derived class modelled using the template <code>BookingClassType<T></code>	15
7.6.1	Testing the constructor	15
7.6.2	Testing the bool <code>IsAC()</code> const function	15
7.6.3	Testing the bool <code>IsLuxury()</code> const function	15
7.6.4	Testing the bool <code>IsSitting()</code> const function	15
7.6.5	Testing the double <code>GetLoadFactor()</code> const function	16
7.6.6	Testing the int <code>GetNumberOfTiers()</code> const function	16
7.6.7	Testing the double <code>GetReservationCharge()</code> const function	16
7.6.8	Testing the double <code>GetTatkalFactor()</code> const function	16
7.6.9	Testing the double <code>GetTatkalMinCharge()</code> const function	16
7.6.10	Testing the double <code>GetTatkalMaxCharge()</code> const function	16
7.6.11	Testing the int <code>GetMinTatkalDistance()</code> const function	16
7.7	Testing the <code>Sleeper</code> derived class modelled using the template <code>BookingClassType<T></code> . .	16
7.7.1	Testing the constructor	16
7.7.2	Testing the bool <code>IsAC()</code> const function	16
7.7.3	Testing the bool <code>IsLuxury()</code> const function	16
7.7.4	Testing the bool <code>IsSitting()</code> const function	16
7.7.5	Testing the double <code>GetLoadFactor()</code> const function	16
7.7.6	Testing the int <code>GetNumberOfTiers()</code> const function	16
7.7.7	Testing the double <code>GetReservationCharge()</code> const function	17
7.7.8	Testing the double <code>GetTatkalFactor()</code> const function	17
7.7.9	Testing the double <code>GetTatkalMinCharge()</code> const function	17
7.7.10	Testing the double <code>GetTatkalMaxCharge()</code> const function	17
7.7.11	Testing the int <code>GetMinTatkalDistance()</code> const function	17
7.8	Testing the <code>SecondSitting</code> derived class modelled using the template <code>BookingClassType<T></code>	17
7.8.1	Testing the constructor	17
7.8.2	Testing the bool <code>IsAC()</code> const function	17
7.8.3	Testing the bool <code>IsLuxury()</code> const function	17
7.8.4	Testing the bool <code>IsSitting()</code> const function	17
7.8.5	Testing the double <code>GetLoadFactor()</code> const function	17

7.8.6	Testing the <code>int GetNumberOfTiers()</code> const function	17
7.8.7	Testing the <code>double GetReservationCharge()</code> const function	17
7.8.8	Testing the <code>double GetTatkalFactor()</code> const function	17
7.8.9	Testing the <code>double GetTatkalMinCharge()</code> const function	18
7.8.10	Testing the <code>double GetTatkalMaxCharge()</code> const function	18
7.8.11	Testing the <code>int GetMinTatkalDistance()</code> const function	18
8	Testing the BookingCategory class and hierarchy	18
8.1	Testing the General derived class modelled by the template <code>BookingCategoryType<T></code> . .	18
8.1.1	Testing the <code>GetName()</code> function	18
8.1.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	18
8.2	Testing the Ladies derived class modelled by the template <code>BookingCategoryType<T></code> . .	18
8.2.1	Testing the <code>GetName()</code> function	18
8.2.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	18
8.2.3	Testing the <code>bool IsEligible(Passenger&)</code> const function	18
8.3	Testing the SeniorCitizen derived class modelled by the template <code>BookingCategoryType<T></code>	18
8.3.1	Testing the <code>GetName()</code> function	18
8.3.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	19
8.3.3	Testing the <code>bool IsEligible(Passenger&)</code> const function	19
8.4	Testing the Tatkal derived class modelled by the template <code>BookingCategoryType<T></code> . .	19
8.4.1	Testing the <code>GetName()</code> function	19
8.4.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	19
8.5	Testing the PremiumTatkal derived class modelled by the template <code>BookingCategoryType<T></code>	19
8.5.1	Testing the <code>GetName()</code> function	19
8.5.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	19
9	Testing the Divyaang class and hierarchy	19
9.1	Testing the Blind derived class modelled by the template <code>DivyaangType<T></code>	19
9.1.1	Testing the <code>GetName()</code> function	19
9.1.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	19
9.1.3	Testing the <code>bool IsEligible(Passenger&)</code> const function	19
9.2	Testing the OrthoHandicapped derived class modelled by the template <code>DivyaangType<T></code>	20
9.2.1	Testing the <code>GetName()</code> function	20
9.2.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	20
9.2.3	Testing the <code>bool IsEligible(Passenger&)</code> const function	20
9.3	Testing the Cancer derived class modelled by the template <code>DivyaangType<T></code>	20
9.3.1	Testing the <code>GetName()</code> function	20
9.3.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	20
9.3.3	Testing the <code>bool IsEligible(Passenger&)</code> const function	20
9.4	Testing the TB derived class modelled by the template <code>DivyaangType<T></code>	20
9.4.1	Testing the <code>GetName()</code> function	20
9.4.2	Testing the <code>bool IsEligible(Passenger&)</code> const function	20
9.4.3	Testing the <code>bool IsEligible(Passenger&)</code> const function	20
10	Testing the Concessions class and hierarchy	21
10.1	Testing the GeneralConcession derived class	21
10.1.1	Testing the constructor	21
10.1.2	Testing the <code>double GetFactor()</code> function	21
10.2	Testing the LadiesConcession derived class	21
10.2.1	Testing the <code>double GetFactor()</code> function	21

10.3	Testing the SeniorCitizenConcession derived class	21
10.3.1	Testing the double GetFactor() function	21
10.4	Testing the DivyaangConcessions derived class	21
10.5	Testing the constructor and singleton behavior	21
10.5.1	Testing the double GetFactor() function	21
11	Testing the Booking class and hierarchy	23
12	When booking is done in General	24
13	When booking is done in Ladies	24
14	When booking is done in Senior Citizen (Male)	24
15	When booking is done in Senior Citizen (Female)	24
16	When booking is done in Tatkal (for General person)	24
17	When booking is done in Premium Tatkal (for General person)	24
18	When booking is done in Tatkal (for person who can avail concession)	24
19	When booking is done in Premium Tatkal (for person who can avail concession)	24
20	When booking is done in Divyaang of type Blind	24
21	When booking is done in Divyaang of type Orthopedically Handicapped	24
22	When booking is done in Divyaang of type Cancer	24
23	When booking is done in Divyaang of type TB	24
24	When the source station is misspelled	25
25	When the destination station is misspelled	25
26	When the source station and destination are the same	25
27	When the date of booking is out of range of guidelines	25
28	When date of booking and reservation are the same	25
29	Date of booking cannot be beyond 1 year from date of reservation	25
30	When the person is not eligible for the booking category applied (Divyaang)	25
31	When the person is not eligible for the booking category applied (Senior Citizen)	25

Unit Test Cases

1 Testing the Stations class

Positive test cases

1.1 Testing the constructor `Station(const string&)`

1. PASS

1.2 Testing the static `Station& CreateStation(const string&)` function

1. PASS

1.3 Testing the string `GetName()` const function

1. PASS

1.4 Testing the int `GetDistance(const Station&)` const function

1. PASS

1.5 Testing the friend `ostream& operator<<(ostream&, const Station&)` function

1. PASS

Negative test cases

1.6 Testing the static `Station& CreateStation(const string&)` function

1. **Input:** Passing an empty string to the function : `Station::CreateStation("")`

Golden Output: An exception should be printed saying "Station name cannot be empty"

2 Testing the Railways class

Positive test cases

2.1 Testing the constructor `Railways()` and static const `Railways& IndianRailways()` function

1. **Input:** Checking the singleton creation via constructor: `firstPointer = &Railways::IndianRailways()` and `secondPointer = &Railways::IndianRailways()`

Golden Output: On asserting both the pointers should be equal

2.2 Testing the int `GetDistance(const Station&, const Station&)` function

1. PASS

2.3 Testing the Station `GetStation(const string& name)` const function

1. PASS

2.4 Testing the friend ostream& operator<<(ostream&, const Railways&) function

1. **PASS**

Negative test cases

2.5 Testing the Station GetStation(const string& name) const function

1. **FAIL (Exception Thrown)**

3 Testing the Date class

Positive test cases

3.1 Testing the constructor Date(int, int, int)

1. **PASS**

3.2 Testing the copy constructor Date(const Date&)

1. **PASS**

3.3 Testing the Day day() const function

1. **PASS**

3.4 Testing the friend int operator-(const Date&, const Date&) function

1. **PASS**

3.5 Testing the bool operator>(const Date&) function

1. **PASS**

2. **PASS**

3.6 Testing the bool operator==(const Date&) function

1. **PASS**

2. **PASS**

3.7 Testing the friend ostream& operator<<(ostream&, const Date&) function

1. **PASS**

Negative test cases

1. **FAIL (Exception Thrown)**

2. **FAIL (Exception Thrown)**

3. **FAIL (Exception Thrown)**

4. **FAIL (Exception Thrown)**

5. FAIL (Exception Thrown)
6. FAIL (Exception Thrown)
7. FAIL (Exception Thrown)

4 Testing the Name class

Positive test cases

4.1 When First, Middle and Last name are present

1. PASS

4.2 When First and Last name are present

1. PASS

4.3 When Last name is present

1. PASS

4.4 When First name is present

1. PASS

4.5 When First and Middle are present

1. PASS

4.6 When Middle and Last name are present

1. PASS

Negative test cases

4.7 When only middle name is present

1. FAIL (Exception Thrown)

4.8 When none of the names are present

1. FAIL (Exception Thrown)

5 Testing the Gender class and hierarchy

5.1 Testing Male derived class made using template GenderTypes<T>

Positive test cases

5.1.1 Testing the constructor GenderTypes(const string& name = GenderTypes<T>::sName)

1. PASS

5.1.2 Testing the const string GetName() const function

1. PASS

5.1.3 Testing the const string GetTitle() const function

1. PASS

5.1.4 Testing the friend ostream& operator<<(ostream&, const Gender&) function

1. PASS

5.2 Testing Female derived class made using template GenderTypes<T>

Positive test cases

5.2.1 Testing the constructor GenderTypes(const string& name = GenderTypes<T>::sName)

1. PASS

5.2.2 Testing the const string GetName() const function

1. PASS

5.2.3 Testing the const string GetTitle() const function

1. PASS

5.2.4 Testing the friend ostream& operator<<(ostream&, const Gender&) function

1. PASS

6 Testing the Passenger class

Positive test cases

6.1 Testing the static Passenger& CreatePassenger(const Name, const Date, const Gender&, const string&, const string&, const Divyaang&, const string&) function

1. PASS

6.2 Testing the const Date GetDateOfBirth() const function

1. PASS

6.3 Testing the const Gender& GetGender() const function

1. PASS

6.4 Testing the const Divyaang& GetDisability() const function

1. **PASS**

Negative test cases

1. **FAIL (Exception Thrown)**
2. **FAIL (Exception Thrown)**
3. **FAIL (Exception Thrown)**
4. **FAIL (Exception Thrown)**

7 Testing the BookingClass class and hierarchy

7.1 Testing ACFirstClass the derived class modelled using the template BookingClassType<T>

Positive test cases

7.1.1 Testing the constructor BookingClassType(const string& name = BookingClassType<T>::sName)

1. **PASS**

7.1.2 Testing the bool IsAC() const function

1. **PASS**

7.1.3 Testing the bool IsLuxury() const function

1. **PASS**

7.1.4 Testing the bool IsSitting() const function

1. **PASS**

7.1.5 Testing the double GetLoadFactor() const function

1. **PASS**

7.1.6 Testing the int GetNumberOfTiers() const function

1. **PASS**

7.1.7 Testing the double GetReservationCharge() const function

1. **PASS**

7.1.8 Testing the double GetTatkalFactor() const function

1. **PASS**

7.1.9 Testing the double GetTatkalMinCharge() const function

1. **PASS**

7.1.10 Testing the double GetTatkalMaxCharge() const function

1. PASS

7.1.11 Testing the int GetMinTatkalDistance() const function

1. PASS

7.2 Testing the ExecutiveChairCar derived class modelled using the template BookingClassType<T>

Positive test cases

7.2.1 Testing the constructor BookingClassType(const string& name = BookingClassType<T>::sName)

1. PASS

7.2.2 Testing the bool IsAC() const function

1. PASS

7.2.3 Testing the bool IsLuxury() const function

1. PASS

7.2.4 Testing the bool IsSitting() const function

1. PASS

7.2.5 Testing the double GetLoadFactor() const function

1. PASS

7.2.6 Testing the int GetNumberOfTiers() const function

1. PASS

7.2.7 Testing the double GetReservationCharge() const function

1. PASS

7.2.8 Testing the double GetTatkalFactor() const function

1. PASS

7.2.9 Testing the double GetTatkalMinCharge() const function

1. PASS

7.2.10 Testing the double GetTatkalMaxCharge() const function

1. PASS

7.2.11 Testing the `int GetMinTatkalDistance()` const function

1. PASS

7.3 Testing the AC2Tier derived class modelled using the template `BookingClassType<T>`

Positive test cases

7.3.1 Testing the constructor `BookingClassType(const string& name = BookingClassType<T>::sName)`

1. PASS

7.3.2 Testing the `bool IsAC()` const function

1. PASS

7.3.3 Testing the `bool IsLuxury()` const function

1. PASS

7.3.4 Testing the `bool IsSitting()` const function

1. PASS

7.3.5 Testing the `double GetLoadFactor()` const function

1. PASS

7.3.6 Testing the `int GetNumberOfTiers()` const function

1. PASS

7.3.7 Testing the `double GetReservationCharge()` const function

1. PASS

7.3.8 Testing the `double GetTatkalFactor()` const function

1. PASS

7.3.9 Testing the `double GetTatkalMinCharge()` const function

1. PASS

7.3.10 Testing the `double GetTatkalMaxCharge()` const function

1. PASS

7.3.11 Testing the `int GetMinTatkalDistance()` const function

1. PASS

7.4 Testing the FirstClass derived class modelled using the template `BookingClassType<T>`

Positive test cases

7.4.1 Testing the constructor `BookingClassType(const string& name = BookingClassType<T>::sName)`

1. PASS

7.4.2 Testing the bool `IsAC()` const function

1. PASS

7.4.3 Testing the bool `IsLuxury()` const function

1. PASS

7.4.4 Testing the bool `IsSitting()` const function

1. PASS

7.4.5 Testing the double `GetLoadFactor()` const function

1. PASS

7.4.6 Testing the int `GetNumberOfTiers()` const function

1. PASS

7.4.7 Testing the double `GetReservationCharge()` const function

1. PASS

7.4.8 Testing the double `GetTatkalFactor()` const function

1. PASS

7.4.9 Testing the double `GetTatkalMinCharge()` const function

1. PASS

7.4.10 Testing the double `GetTatkalMaxCharge()` const function

1. PASS

7.4.11 Testing the int `GetMinTatkalDistance()` const function

1. PASS

7.5 Testing the AC3Tier derived class modelled using the template `BookingClassType<T>`

7.5.1 Testing the constructor `BookingClassType(const string& name = BookingClassType<T>::sName)`

1. PASS

7.5.2 Testing the bool `IsAC()` const function

1. PASS

7.5.3 Testing the bool IsLuxury() const function

1. PASS

7.5.4 Testing the bool IsSitting() const function

1. PASS

7.5.5 Testing the double GetLoadFactor() const function

1. PASS

7.5.6 Testing the int GetNumberOfTiers() const function

1. PASS

7.5.7 Testing the double GetReservationCharge() const function

1. PASS

7.5.8 Testing the double GetTatkalFactor() const function

1. PASS

7.5.9 Testing the double GetTatkalMinCharge() const function

1. PASS

7.5.10 Testing the double GetTatkalMaxCharge() const function

1. PASS

7.5.11 Testing the int GetMinTatkalDistance() const function

1. PASS

7.6 Testing the ACChairCar derived class modelled using the template BookingClassType<T>

Positive test cases

7.6.1 Testing the constructor BookingClassType(const string& name = BookingClassType<T>::sName)

1. PASS

7.6.2 Testing the bool IsAC() const function

1. PASS

7.6.3 Testing the bool IsLuxury() const function

1. PASS

7.6.4 Testing the bool IsSitting() const function

1. PASS

7.6.5 Testing the double GetLoadFactor() const function

1. PASS

7.6.6 Testing the int GetNumberOfTiers() const function

1. PASS

7.6.7 Testing the double GetReservationCharge() const function

1. PASS

7.6.8 Testing the double GetTatkalFactor() const function

1. PASS

7.6.9 Testing the double GetTatkalMinCharge() const function

1. PASS

7.6.10 Testing the double GetTatkalMaxCharge() const function

1. PASS

7.6.11 Testing the int GetMinTatkalDistance() const function

1. PASS

7.7 Testing the Sleeper derived class modelled using the template BookingClassType<T>

Positive test cases

7.7.1 Testing the constructor BookingClassType(const string& name = BookingClassType<T>::sName)

1. PASS

7.7.2 Testing the bool IsAC() const function

1. PASS

7.7.3 Testing the bool IsLuxury() const function

1. PASS

7.7.4 Testing the bool IsSitting() const function

1. PASS

7.7.5 Testing the double GetLoadFactor() const function

1. PASS

7.7.6 Testing the int GetNumberOfTiers() const function

1. PASS

7.7.7 Testing the double GetReservationCharge() const function

1. PASS

7.7.8 Testing the double GetTatkalFactor() const function

1. PASS

7.7.9 Testing the double GetTatkalMinCharge() const function

1. PASS

7.7.10 Testing the double GetTatkalMaxCharge() const function

1. PASS

7.7.11 Testing the int GetMinTatkalDistance() const function

1. PASS

7.8 Testing the SecondSitting derived class modelled using the template BookingClassType<T>

Positive test cases

7.8.1 Testing the constructor BookingClassType(const string& name = BookingClassType<T>::sName)

1. PASS

7.8.2 Testing the bool IsAC() const function

1. PASS

7.8.3 Testing the bool IsLuxury() const function

1. PASS

7.8.4 Testing the bool IsSitting() const function

1. PASS

7.8.5 Testing the double GetLoadFactor() const function

1. PASS

7.8.6 Testing the int GetNumberOfTiers() const function

1. PASS

7.8.7 Testing the double GetReservationCharge() const function

1. PASS

7.8.8 Testing the double GetTatkalFactor() const function

1. PASS

7.8.9 Testing the double GetTatkalMinCharge() const function

1. PASS

7.8.10 Testing the double GetTatkalMaxCharge() const function

1. PASS

7.8.11 Testing the int GetMinTatkalDistance() const function

1. PASS

8 Testing the BookingCategory class and hierarchy

8.1 Testing the General derived class modelled by the template BookingCategoryType<T>

Positive test cases

8.1.1 Testing the GetName() function

1. PASS

8.1.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

8.2 Testing the Ladies derived class modelled by the template BookingCategoryType<T>

Positive test cases

8.2.1 Testing the GetName() function

1. PASS

8.2.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

Negative test cases

8.2.3 Testing the bool IsEligible(Passenger&) const function

1. PASS

8.3 Testing the SeniorCitizen derived class modelled by the template BookingCategoryType<T>

Positive test cases

8.3.1 Testing the GetName() function

1. PASS

8.3.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

Negative test cases

8.3.3 Testing the bool IsEligible(Passenger&) const function

1. PASS

8.4 Testing the Tatkai derived class modelled by the template BookingCategoryType<T>

Positive test cases

8.4.1 Testing the GetName() function

1. PASS

8.4.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

8.5 Testing the PremiumTatkai derived class modelled by the template BookingCategoryType<T>

Positive test cases

8.5.1 Testing the GetName() function

1. PASS

8.5.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

9 Testing the Divyaang class and hierarchy

9.1 Testing the Blind derived class modelled by the template DivyaangType<T>;

Positive test cases

9.1.1 Testing the GetName() function

1. PASS

9.1.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

Negative test cases

9.1.3 Testing the bool IsEligible(Passenger&) const function

1. PASS

9.2 Testing the OrthoHandicapped derived class modelled by the template DivyaangType<T>

Positive test cases

9.2.1 Testing the GetName() function

1. PASS

9.2.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

Negative test cases

9.2.3 Testing the bool IsEligible(Passenger&) const function

1. PASS

9.3 Testing the Cancer derived class modelled by the template DivyaangType<T>

Positive test cases

9.3.1 Testing the GetName() function

1. PASS

9.3.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

Negative test cases

9.3.3 Testing the bool IsEligible(Passenger&) const function

1. PASS

9.4 Testing the TB derived class modelled by the template DivyaangType<T>

Positive test cases

9.4.1 Testing the GetName() function

1. PASS

9.4.2 Testing the bool IsEligible(Passenger&) const function

1. PASS

Negative test cases

9.4.3 Testing the bool IsEligible(Passenger&) const function

1. PASS

10 Testing the Concessions class and hierarchy

10.1 Testing the GeneralConcession derived class

Positive test cases

10.1.1 Testing the constructor GeneralConcession(string&)

- 1. PASS**

10.1.2 Testing the double GetFactor() function

- 1. PASS**

10.2 Testing the LadiesConcession derived class

Positive test cases

- 1. PASS**

10.2.1 Testing the double GetFactor() function

- 1. PASS**

10.3 Testing the SeniorCitizenConcession derived class

- 1. PASS**

10.3.1 Testing the double GetFactor() function

- 1. PASS**
- 2. PASS**

10.4 Testing the DivyaangConcessions derived class

Positive test cases

10.5 Testing the constructor and singleton behavior

- 1. PASS**

10.5.1 Testing the double GetFactor() function

For Diyaang of type Blind

- 1. PASS**
- 2. PASS**
- 3. PASS**
- 4. PASS**
- 5. PASS**

6. **PASS**

7. **PASS**

8. **PASS**

For Diyaang of type Orthopedically Handicapped

1. **PASS**

2. **PASS**

3. **PASS**

4. **PASS**

5. **PASS**

6. **PASS**

7. **PASS**

8. **PASS**

For Diyaang of type Cancer Patient

1. **PASS**

2. **PASS**

3. **PASS**

4. **PASS**

5. **PASS**

6. **PASS**

7. **PASS**

8. **PASS**

For Diyaang of type TB Patient

1. **PASS**

2. **PASS**

3. **PASS**

4. **PASS**

5. **PASS**

6. **PASS**

7. **PASS**

8. **PASS**

11 Testing the Booking class and hierarchy

Positive test cases

Here instead of testing individual functions we book for various circumstances and check if the fare and all the other details of the booking are printed correctly

1. **PASS**
2. **PASS**
3. **PASS**
4. **PASS**
5. **PASS**
6. **PASS**
7. **PASS**
8. **PASS**
9. **PASS**
10. **PASS**
11. **PASS**
12. **PASS**
13. **PASS**
14. **PASS**
15. **PASS**
16. **PASS**
17. **PASS**
18. **PASS**

Positive Application Test Cases

12 When booking is done in General

1. PASS

13 When booking is done in Ladies

1. PASS

14 When booking is done in Senior Citizen (Male)

1. PASS

15 When booking is done in Senior Citizen (Female)

1. PASS

16 When booking is done in Tatkal (for General person)

1. PASS

17 When booking is done in Premium Tatkal (for General person)

1. PASS

18 When booking is done in Tatkal (for person who can avail concession)

1. PASS

19 When booking is done in Premium Tatkal (for person who can avail concession)

1. PASS

20 When booking is done in Divyaang of type Blind

1. PASS

21 When booking is done in Divyaang of type Orthopedically Handicapped

1. PASS

22 When booking is done in Divyaang of type Cancer

1. PASS

23 When booking is done in Divyaang of type TB

1. PASS

Negative Application Test Cases

24 When the source station is misspelled

1. **FAIL** (Exception Thrown)

25 When the destination station is misspelled

1. **FAIL** (Exception Thrown)

26 When the source station and destination are the same

1. **FAIL** (Exception Thrown)

27 When the date of booking is out of range of guidelines

1. **FAIL** (Exception Thrown)

28 When date of booking and reservation are the same

1. **FAIL** (Exception Thrown)

29 Date of booking cannot be beyond 1 year from date of reservation

1. **FAIL** (Exception Thrown)

30 When the person is not eligible for the booking category applied (Divyaang)

1. **FAIL** (Exception Thrown)

31 When the person is not eligible for the booking category applied (Senior Citizen)

1. **FAIL** (Exception Thrown)