1. **Code**

**Car.cpp**

#include "Car.h"

Car::Car()

{

raceCarStatus = false;

}

void Car::setRaceCarStatus(bool thisStatus)

{

raceCarStatus = thisStatus;

}

bool Car::getRaceCarStatus()

{

return raceCarStatus;

}

**Car.h**

#pragma once

#pragma once

#ifndef CAR\_H

#define CAR\_H

class Car

{

public:

Car();

void setRaceCarStatus(bool thisStatus);

bool getRaceCarStatus();

private:

bool raceCarStatus;

};

#endif // !CAR\_H

**Stub\_main\_Car.cpp**

#include "Car.h"

int main()

{

Car Civic;

int tempStatus;

do {

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStatus;

if (tempStatus < 0 || tempStatus > 1)

{

cout << endl << "Invalid input. Please try again with number 1 or number 0" << endl;

}

} while (tempStatus < 0 || tempStatus > 1);

Civic.setRaceCarStatus(tempStatus);

cout << "Race Car Status stored" << endl;

if (Civic.getRaceCarStatus())

{

cout << "Your car, CIvic, IS a race car" << endl;

}

else {

cout << "Your car, Civic, IS NOT a race car" << endl;

}

return 0;

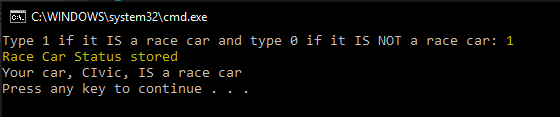
}

1. **Test Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | User enters corresponding number for choosing either true or false for race car status | User enter “1” for true for car status | “Race Car Status stored” | “Race Car Status stored” | Pass |
| Invalid | 1 | User enters number not corresponding to choosing either true or false for race car status | User enters “10” for true for race car status | “Invalid input. Please try again with number 1 or number 0” | “Invalid input. Please try again with number 1 or number 0” | Pass |

1. **Screenshots**

Valid Test Case 1



Invalid Test Case 2

