**PCPF Lab**

**Lab Assignment number 02**

**Name:** Aamir Ansari **Batch:** A **Roll no.** 01

**Aim:** Implementation of Inheritance

**Problem Statement:**

Write the code to implement the concept of inheritance for Vehicles. You are required to

implement inheritance between classes. There would be 3 classes - one superclass and two sub

classes. Vehicle is the super class, whereas Bus and Truck are subclasses of Vehicle class.

Detailed description of Vehicle (Super class):

The class Vehicle must have following attributes (private):

1. Vehicle model

2. Registration number

3. Vehicle speed (km/hour)

4. Fuel capacity (liters)

5. Mileage (kilometers/liter)

The Vehicle class must have following functions:

1. Parameterized constructor that will initialize all the data members with the given values.

2. Getters and Setters for each data member that will get and set the values of data members of

class.

3. A function fuelNeeded() that will take distance (in kilometer) as an argument. It will

calculate the amount of fuel needed for the given distance and will return the value of fuel

needed for given distance. You can use the attributes ‘ Mileage’ defined within the above

Vehicle class to determine the fuel needed for the given distance. You are required to

implement this functionality by yourself.

4. A function distanceCovered() that will take time (in hours) as an argument. It will calculate

the distance for the given time and speed and returns the value of distance. The formula to

calculate speed is given as speed = distance/time . You can use this formula to calculate the

distance.

5. A display() function to display all details of the Vehicle.

Detailed description of Truck (Sub class):

The class Truck must have following attribute (private):

Cargo weight limit (Kilo grams)

The above class must have following functions:

1. Parameterized constructor that will initialize all data members with the given values.

2. Getters and setters for each data member that will get and set the values of data members of

class.

3. It must override the display() function of Vehicle and display the details of the Truck object.

Detailed description of Bus (Sub class):

The class Bus must have following attribute (private):

No of passengers

The above class must have following functions:

1. Parameterized constructor that will initialize all the data members with given values.

2. Getters and setters that will get and set the value of each data member of class.

3. It must override the display() function of Vehicle and display the details of the Truck object.

In main function, perform the following:

● Create an instance of class Truck and initialize all the data members with proper values.

● Create an instance of class Bus and initialize all the data members with proper values.

● Now, call fuelNeeded () , distanceCovered () and display() functions using objects of these

classes.

**Theory:**

Inheritance and its Advantages:

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.

**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.

**Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

**Advantages of inheritance:**

1. It allows the code to be reused as many times as needed
2. The base class once defined and once it is compiled, it need not be reworked
3. Saves time and effort as the main code need not be written again
4. Saves time in program development

Discuss with examples, the implications of deriving a class from an existing class by the private, public and protected access specifiers

**Public mode:** If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.

**Protected mode:** If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.

**Private mode:** If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

When Constructors and Destructors are executed, in base and consecutive derive classes

**Constructor** of base class is executed before that of derived class. Inn case of multiple inheritance, the order in which they are inherited is the order is the order of constructor calling

**Destructor** works the other way around

Difference: Overriding and Overloading

|  |  |
| --- | --- |
| **Overloading** | **Overriding** |
| Method overloading is used to increase the readability of the program. | Method overriding is used to provide the specific implementation of the method that is already provided by its super class. |
| Method overloading is performed within class. | Method overriding occurs in two classes that have IS-A (inheritance) relationship. |
| In case of method overloading, parameter must be different. | In case of method overriding, parameter must be same. |
| Method overloading is the example of compile time polymorphism. | Method overriding is the example of run time polymorphism. |

**Program:**

// code

#include <iostream>

using namespace std;

//Super class

class Vehicle {

private:

string vehicleModel;

int regNum;

float speed, fuel, mileage;

public:

//Initialization

Vehicle(string vehicleModel, int regNum, float speed, float fuel, float mileage) {

this->vehicleModel = vehicleModel;

this->regNum = regNum;

this->speed = speed;

this->fuel = fuel;

this->mileage = mileage;

}

//Model getter setter

string getVehicleModel () {

return this->vehicleModel;

}

void setVehicleModel (string vehicleModel) {

this->vehicleModel = vehicleModel;

}

//regNum getter setter

int getRegNum () {

return this->regNum;

}

void setRegNum(int regNum) {

this->regNum = regNum;

}

//speed getter setter

float getSpeed() {

return this->speed;

}

void setSpeed (float speed) {

this->speed = speed;

}

//fuel getter setter

float getFuel() {

return this->fuel;

}

void setFuel(float fuel) {

this->fuel = fuel;

}

//mileage getter setter

float getMileage() {

return this->mileage;

}

void setMileage(float mileage) {

this->mileage = mileage;

}

//fuels needed in liter

float fuelNeeded (float km) {

return (this->mileage / km);

}

//Distance covered

float distanceCovered (float hour) {

return (this->speed \* hour);

}

//Display information of vehicle

virtual void display() {

cout << "Vehicle Model : " << this->vehicleModel << endl;

cout << "Registration number : " << this->regNum << endl;

cout << "Vehicle Speed : " << this->speed << " km/h" << endl;

cout << "Vehicle Fuel : " << this->fuel << " liters" << endl;

cout << "Vehicle Mileage : " << this->mileage << "km/l" << endl;

}

};

//Sub class

class Truck : public Vehicle {

private:

float cargoWeightLimit;

public:

//Initializing constructor

Truck (string vehicleModel, int regNum, float speed, float fuel, float mileage, float cargoWeightLimit)

:Vehicle(vehicleModel, regNum, speed, fuel, mileage) {

this->cargoWeightLimit = cargoWeightLimit;

}

//cargoWeightLimit getter setter

float getCargoWeightLimit() {

return this->cargoWeightLimit;

}

void setCargoWeightLimit() {

this->cargoWeightLimit = cargoWeightLimit;

}

//Display info of bus

void display() override {

cout << "Vehicle Model : " << getVehicleModel() << endl;

cout << "Registration number : " << getRegNum() << endl;

cout << "Vehicle Speed : " << getSpeed() << " km/h" << endl;

cout << "Vehicle Fuel : " << getFuel() << " liters" << endl;

cout << "Vehicle Mileage : " << getMileage() << " km/l" << endl;

cout << "Cargo Weight Limit : " << this->cargoWeightLimit << endl;

}

};

//Sub class

class Bus : public Vehicle {

private:

int numPassenger;

public:

//Initializing constructor

Bus (string vehicleModel, int regNum, float speed, float fuel, float mileage, int numPassenger)

:Vehicle(vehicleModel, regNum, speed, fuel, mileage) {

this->numPassenger = numPassenger;

}

//numPassenger getter setter

int getNumPassenger() {

return this->numPassenger;

}

void setNumPassenger() {

this->numPassenger = numPassenger;

}

//Display info of truck

void display() override {

cout << "Vehicle Model : " << getVehicleModel() << endl;

cout << "Registration number : " << getRegNum() << endl;

cout << "Vehicle Speed : " << getSpeed() << " km/h" << endl;

cout << "Vehicle Fuel : " << getFuel() << " liters" << endl;

cout << "Vehicle Mileage : " << getMileage() << " km/l" << endl;

cout << "Number of Passenger : " << this-> numPassenger << endl;

}

};

int main() {

//created object; initialized with constructor

Truck truckObj("Belaz", 2, 60, 50, 15, 80);

//created object; initialized with constructor

Bus busObj("Volvo", 1, 25, 50, 10, 100);

//Display truck

cout << "\*\*\*\*\*\*\*\*\*\*\* B U S \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

busObj.display();

cout << "Fuel needed : " << busObj.fuelNeeded(10) << endl;

cout << "Distance covered : " << busObj.distanceCovered(2) << endl;

cout << endl;

//Display bus

cout << "\*\*\*\*\*\*\*\*\*\*\* T R U C K \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

truckObj.display();

cout << "Fuel needed : " << truckObj.fuelNeeded(15) << endl;

cout << "Distance covered : " << truckObj.distanceCovered(2) << endl;

}

// output

