CIS 200 - Lab0501

Vehicle Individual

Algorithm

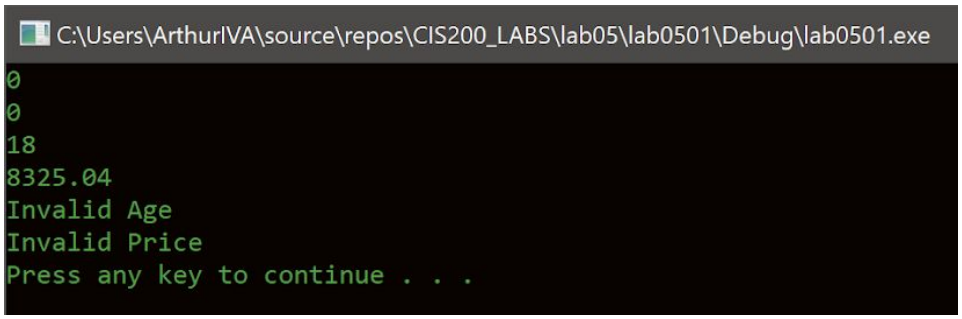
1. Create Class Vehicle
    a. Create Private Variables
        i. Integer age
        ii. Float price
    b. Create Function
        i. Name: setAge
        ii. Parameters: integer input
        iii. Return: void
    c. Create Function
        i. Name: setPrice
        ii. Parameters: float input
        iii. Return: void
    d. Create Function
        i. Name: getAge
        ii. Parameters: n/a
        iii. Return: integer age
    e. Create Function
        i. Name: getPrice
        ii. Parameters: n/a
        iii. Return: float price
2. Create Main Function
    a. Show default constructor
    b. Attempt setting age to 18
    c. Attempt setting price to $8,325.04
    d. Attempt setting age to -1
    e. Attempt setting price to -1

Test Plan

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Default Constructor	Default	Age = 0 Price = 0.0	Age = 0 Price = 0.0	Pass
Valid Data	2	Set Age	18	Age = 18	Age = 18	Pass
Valid Data	3	Get Age	^^	Age = 18	Age = 18	Pass

Valid Data	4	Set Price	8325.04	Price = 8325.04	Price = 8325.04	Pass
Valid Data	5	Get Price	^^	Price = 8325.04	Price = 8325,04	Pass
Invalid Data	1	Set Age	-1	Invalid Age	Invalid Age	Pass
Invalid Data	2	Set Price	-1	Invalid Price	Invalid Price	Pass

Snippets



Code

[vehicleIndividual.cpp]

```
//Vehice Individual Definition and Test
#include <iostream>
#include "vehicle.h"

int main()
{
	vehicle example;

	//Testing Defaults
	std::cout << example.getAge() << std::endl;
	std::cout << example.getPrice() << std::endl;

	//Testing Valid
	example.setAge(18);
	std::cout << example.getAge() << std::endl;
	example.setPrice(8325.04);
	std::cout << example.getPrice() << std::endl;

	//Testing Invalid
	example.setAge(-1);
	example.setPrice(-1);
```

```
	system("pause");
	return 0;
}
```

[vehicle.h]

```
#pragma once
#include <iostream>

#ifndef VEHICLE_H
#define VEHICLE_H

class vehicle
{
public:

	vehicle();		//Default Constructor Primitive

	void setAge(int);
	void setPrice(float);

	int getAge();
	float getPrice();

	~vehicle();		//Default Constrcutor Primitive

private:

	int age;		//Age of the vehicle
	float price; //Price of the vehicle
};
#endif VEHICLE_H
```

[vehicle.cpp]

```
#include "vehicle.h"

//Default Constructor
vehicle::vehicle()
{
	age = 0;
	price = 0.0;
}

//Setter Commands
void vehicle::setAge(int input)
{
```

```
	if (input >= 0)
	{
		age = input;
	}
	else
	{
		std::cout << "Invalid Age" << std::endl;
	}
}

void vehicle::setPrice(float input)
{
	if (input >= 0)
	{
		price = input;
	}
	else
	{
		std::cout << "Invalid Price" << std::endl;
	}
}

//Getter Commands
int vehicle::getAge()
{
	return age;
}

float vehicle::getPrice()
{
	return price;
}

//Default Destructor
vehicle::~vehicle()
{
}
```

Car Individual

Algorithm

3. Create Class Car
    a. Create Private Variables
        i. Bool Race Car Status

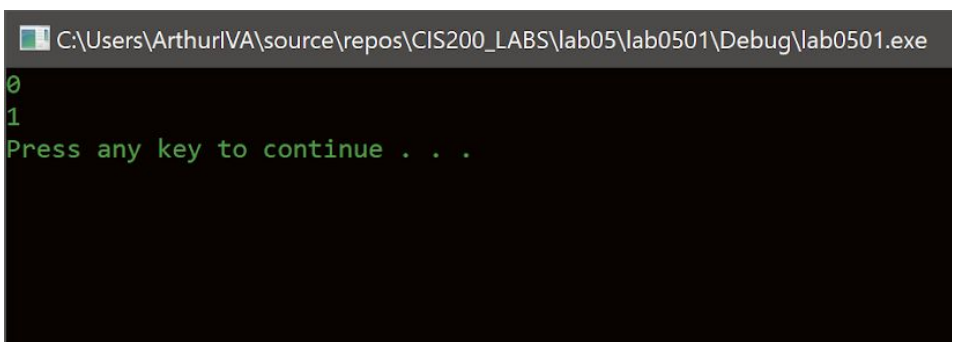
b. Create Function
    i. Name: setRaceCarStatus
    ii. Parameters: bool input
    iii. Return: void
c. Create Function
    i. Name: getRaceCarStatus
    ii. Parameters: n/a
    iii. Return: bool

4. Create Main Function
    a. Show Default Constructor
    b. Attempt setting RaceCarStatus to true

Test Plan

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Default Constructor	Default	0	0	Pass
Valid Data	2	Set Race Car	true	1	1	Pass

Snippets



Code

[carIndividual.cpp]

```
//Car Individual Definition and Test
#include <iostream>
#include "car.h"

int main()
{
	car mustang;
```

```
	//Testing Valid
	std::cout << mustang.getRaceCarStatus() << std::endl;
	mustang.setRaceCarStatus(true);
	std::cout << mustang.getRaceCarStatus() << std::endl;

	system("pause");
	return 0;
}
```

[car.h]

```
#pragma once
#include <iostream>
#include "vehicle.h"

#ifndef CAR_H
#define CAR_H

class car: public vehicle
{
public:

	car();		//Default Constructor

	void setRaceCarStatus(bool);
	bool getRaceCarStatus();

	~car();		//Default Destructor
private:

	bool RaceCarStatus;
};

#endif CAR_H
```

[car.cpp]

```
#include "car.h"

//Default Constructor
car::car()
{
	RaceCarStatus = 0;
}

//Setter
void car::setRaceCarStatus(bool input)
```

```
{
	RaceCarStatus = input;
}

//Getter
bool car::getRaceCarStatus()
{
	return RaceCarStatus;
}

//Default Destructor
car::~car()
{
}
```

Vehicle and Car Combined

Algorithm

5. Create Class Vehicle
    a. Create Private Variables
        i. Integer age
        ii. Float price
    b. Create Function
        i. Name: setAge
        ii. Parameters: integer input
        iii. Return: void
    c. Create Function
        i. Name: setPrice
        ii. Parameters: float input
        iii. Return: void
    d. Create Function
        i. Name: getAge
        ii. Parameters: n/a
        iii. Return: integer age
    e. Create Function
        i. Name: getPrice
        ii. Parameters: n/a
        iii. Return: float price
6. Create Class Car
    a. Create Private Variables
        i. Bool Race Car Status
    b. Create Function

        i. Name: setRaceCarStatus
        ii. Parameters: bool input
        iii. Return: void
    c. Create Function
        i. Name: getRaceCarStatus
        ii. Parameters: n/a
        iii. Return: bool
7. Create Main Function
    a. Show default constructor
    b. Attempt setting age to 31
    c. Attempt setting price to $12,345.67
    d. Attempt setting RaceCarStatus to true
    e. Attempt setting age to -1
    f. Attempt setting price to -1

Test Plan

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Default Constructor	Default	Age = 0 Price = 0.0 Race Car = 0	Age = 0 Price = 0.0 Race Car = 0	Pass
Valid Data	2	Set Age	18	Age = 18	Age = 18	Pass
Valid Data	3	Get Age	^^	Age = 18	Age = 18	Pass
Valid Data	4	Set Price	12,345.67	Price = 12,345.67	Price = 12,345.67	Pass
Valid Data	5	Get Price	^^	Price = 12,345.67	Price = 12,345.67	Pass
Valid Data	6	Set Race Car	true	1	1	Pass
Invalid Data	1	Set Age	-1	Invalid Age	Invalid Age	Pass
Invalid Data	2	Set Price	-1	Invalid Price	Invalid Price	Pass

Snippets

Code

[vehicleCar.cpp]

```
//Vehice and Car Definition and Test
#include <iostream>
#include "vehicle.h"
#include "car.h"

int main()
{
	car mustang;

	//Testing Defaults
	std::cout << mustang.getAge() << std::endl;
	std::cout << mustang.getPrice() << std::endl;
	std::cout << mustang.getRaceCarStatus() << std::endl;

	//Testing Valid
	mustang.setAge(18);
	std::cout << mustang.getAge() << std::endl;
	mustang.setPrice(12345.7);
	std::cout << mustang.getPrice() << std::endl;
	mustang.setRaceCarStatus(true);
	std::cout << mustang.getRaceCarStatus() << std::endl;

	//Testing Invalid
	mustang.setAge(-1);
	mustang.setPrice(-1);
```

```
	system("pause");
	return 0;
}
```

[vehicle.h]

```
#pragma once
#include <iostream>

#ifndef VEHICLE_H
#define VEHICLE_H

class vehicle
{
public:

	vehicle();		//Default Constructor Primitive

	void setAge(int);
	void setPrice(float);

	int getAge();
	float getPrice();

	~vehicle();		//Default Constrcutor Primitive

private:

	int age;		//Age of the vehicle
	float price; //Price of the vehicle
};
#endif VEHICLE_H
```

[vehicle.cpp]

```
#include "vehicle.h"

//Default Constructor
vehicle::vehicle()
{
	age = 0;
	price = 0.0;
}

//Setter Commands
void vehicle::setAge(int input)
{
```

```
	if (input >= 0)
	{
		age = input;
	}
	else
	{
		std::cout << "Invalid Age" << std::endl;
	}
}

void vehicle::setPrice(float input)
{
	if (input >= 0)
	{
		price = input;
	}
	else
	{
		std::cout << "Invalid Price" << std::endl;
	}
}

//Getter Commands
int vehicle::getAge()
{
	return age;
}

float vehicle::getPrice()
{
	return price;
}

//Default Destructor
vehicle::~vehicle()
{
}
```

[car.h]

```
#pragma once
#include <iostream>
#include "vehicle.h"

#ifndef CAR_H
#define CAR_H
```

```
class car: public vehicle
{
public:

	car();		//Default Constructor

	void setRaceCarStatus(bool);
	bool getRaceCarStatus();

	~car();		//Default Destructor
private:

	bool RaceCarStatus;
};

#endif CAR_H
```

[car.cpp]

```
#include "car.h"

//Default Constructor
car::car()
{
	RaceCarStatus = 0;
}

//Setter
void car::setRaceCarStatus(bool input)
{
	RaceCarStatus = input;
}

//Getter
bool car::getRaceCarStatus()
{
	return RaceCarStatus;
}

//Default Destructor
car::~car()
{
}
```

Truck Individual

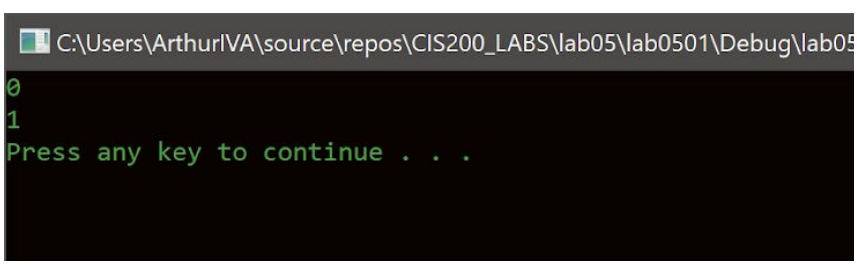
Algorithm

8. Create Class Truck
    a. Create Private Variables
        i. Bool Diesel Type Status
    b. Create Function
        i. Name: setDieselTypeStatus
        ii. Parameters: bool input
        iii. Return: void
    c. Create Function
        i. Name: getDieselTypeStatus
        ii. Parameters: n/a
        iii. Return: bool
9. Create Main Function
    a. Show Default Constructor
    b. Attempt setting DieselTypeStatus to true

Test Plan

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Default Constructor	Default	0	0	Pass
Valid Data	2	Set Diesel Type	true	1	1	Pass

Snippets



Code

[truckIndividual.cpp]

```
//Truck Individual Definition and Test
#include <iostream>
```

```
#include "truck.h"

int main()
{
	truck ranger;


	//Testing Valid
	std::cout << ranger.getDieselTypeStatus() << std::endl;
	ranger.setDieselTypeStatus(true);
	std::cout << ranger.getDieselTypeStatus() << std::endl;


	system("pause");
	return 0;
}
```

[truck.h]

```
#pragma once
#include "vehicle.h"

#ifndef TRUCK_H
#define TRUCK_H

class truck: public vehicle
{
public:
	truck();		//Default Constructor

	void setDieselTypeStatus(bool);
	bool getDieselTypeStatus();

	~truck();		//Default Destructor
protected:
	bool DieselTypeStatus;
};


#endif TRUCK_H
```

[truck.cpp]

```
#include "truck.h"

//Default Constructor
truck::truck()
```

```
{
	DieselTypeStatus = false;
}

//Setter
void truck::setDieselTypeStatus(bool input)
{
	DieselTypeStatus = input;
}

//Getter
bool truck::getDieselTypeStatus()
{
	return DieselTypeStatus;
}

//Default Destructor
truck::~truck()
{

}
```

Vehicle, Car, and Truck Combined

Algorithm

10. Create Class Vehicle
    a. Create Private Variables
        i. Integer age
        ii. Float price
    b. Create Function
        i. Name: setAge
        ii. Parameters: integer input
        iii. Return: void
    c. Create Function
        i. Name: setPrice
        ii. Parameters: float input
        iii. Return: void
    d. Create Function
        i. Name: getAge
        ii. Parameters: n/a
        iii. Return: integer age
    e. Create Function
        i. Name: getPrice

        ii. Parameters: n/a
        iii. Return: float price

11. Create Class Car
    a. Create Private Variables
        i. Bool Race Car Status
    b. Create Function
        i. Name: setRaceCarStatus
        ii. Parameters: bool input
        iii. Return: void
    c. Create Function
        i. Name: getRaceCarStatus
        ii. Parameters: n/a
        iii. Return: bool
12. Create Class Truck
    a. Create Private Variables
        i. Bool Diesel Type Status
    b. Create Function
        i. Name: setDieselTypeStatus
        ii. Parameters: bool input
        iii. Return: void
    c. Create Function
        i. Name: getDieselTypeStatus
        ii. Parameters: n/a
        iii. Return: bool
13. Create Main Function
    a. Testing Car
        i. Show Default Constructor for car
        ii. Attempt setting age to 31
        iii. Attempt setting price to $12,345.67
        iv. Attempt setting RaceCarStatus to true
        v. Attempt setting age to -1
        vi. Attempt setting price to -1
    b. Testing Truck
        i. Show Default Constructor for truck
        ii. Attempt setting age to 18
        iii. Attempt setting price to $8,325.04
        iv. Attempt setting DieselTypeStatus to true
        v. Attempt setting age to -1
        vi. Attempt setting price to -1

Test Plan

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Default Constructor of car	Default	Age = 0 Price = 0.0 Race Car = 0	Age = 0 Price = 0.0 Race Car = 0	Pass
Valid Data	2	Set Age	31	Age = 31	Age = 31	Pass
Valid Data	3	Get Age	^^	Age = 31	Age = 31	Pass
Valid Data	4	Set Price	12,345.67	Price = 12,345.67	Price = 12,345.67	Pass
Valid Data	5	Get Price	^^	Price = 12,345.67	Price = 12,345.67	Pass
Valid Data	6	Set Race Car	true	1	1	Pass
Valid Data	7	Default Constructor of truck	Default	Age = 0 Price = 0.0 Diesel Type = 0	Age = 0 Price = 0.0 Diesel Type = 0	
Valid Data	8	Set Age	18	Age = 18	Age = 18	
Valid Data	9	Get Age	^^	Age = 18	Age = 18	
Valid Data	10	Set Price	8,325.04	Price = 8325.04	Price = 8325.04	
Valid Data	11	Get Price	^^	Price = 8325.04	Price = 8325.04	
Valid Data	12	Set Diesel	true	1	1	
Invalid Data	1	Set Car Age	-1	Invalid Age	Invalid Age	Pass
Invalid Data	2	Set Car Price	-1	Invalid Price	Invalid Price	Pass
Invalid Data	3	Set Truck Age	-2	Invalid Age	Invalid Age	Pass
Invalid Data	4	Set Truck Price	-2	Invalid Price	Invalid Price	Pass

Snippets

```
C:\Users\ArthurIVA\source\repos\CIS200_LABS\lab05\lab0501\Debug\lab0501.exe
Testing Default Constructor for Mustang
0
0
0
Testing Valid Data for Mustang
31
12345.7
1
Testing Invalid Data for Mustang
Invalid Age
Invalid Price

Testing Default Constructor for Ranger
0
0
0
Testing Valid Data for Ranger
18
8325.04
1
Testing Invalid Data for Ranger
Invalid Age
Invalid Price
Press any key to continue . . .
```

Code

[vehicleCarTruckCombined.cpp]

```
//Vehice + Car + Truck Definition and Test
#include <iostream>
#include "vehicle.h"
#include "car.h"
#include "truck.h"

int main()
{
	car mustang;
	truck ranger;

	//Testing Car Defaults
	std::cout << "Testing Default Constructor for Mustang" << std::endl;
	std::cout << mustang.getAge() << std::endl;
	std::cout << mustang.getPrice() << std::endl;
	std::cout << mustang.getRaceCarStatus() << std::endl;
```

```
	//Testing Car Valid
	std::cout << "Testing Valid Data for Mustang" << std::endl;
	mustang.setAge(31);
	std::cout << mustang.getAge() << std::endl;
	mustang.setPrice(12345.7);
	std::cout << mustang.getPrice() << std::endl;
	mustang.setRaceCarStatus(true);
	std::cout << mustang.getRaceCarStatus() << std::endl;

	//Testing Car Invalid
	std::cout << "Testing Invalid Data for Mustang" << std::endl;
	mustang.setAge(-1);
	mustang.setPrice(-1);

	std::cout << std::endl;

	//Testing Truck Defaults
	std::cout << "Testing Default Constructor for Ranger" << std::endl;
	std::cout << ranger.getAge() << std::endl;
	std::cout << ranger.getPrice() << std::endl;
	std::cout << ranger.getDieselTypeStatus() << std::endl;

	//Testing Truck Valid
	std::cout << "Testing Valid Data for Ranger" << std::endl;
	ranger.setAge(18);
	std::cout << ranger.getAge() << std::endl;
	ranger.setPrice(8325.04);
	std::cout << ranger.getPrice() << std::endl;
	ranger.setDieselTypeStatus(true);
	std::cout << ranger.getDieselTypeStatus() << std::endl;

	//Testing Truck Invalid
	std::cout << "Testing Invalid Data for Ranger" << std::endl;
	ranger.setAge(-1);
	ranger.setPrice(-1);


	system("pause");
	return 0;
}
```

[vehicle.h]

```
#pragma once
#include <iostream>
```

```
#ifndef VEHICLE_H
#define VEHICLE_H

class vehicle
{
public:

	vehicle();		//Default Constructor Primitive

	void setAge(int);
	void setPrice(float);

	int getAge();
	float getPrice();

	~vehicle();		//Default Constrcutor Primitive

private:

	int age;		//Age of the vehicle
	float price; //Price of the vehicle
};
#endif VEHICLE_H
```

[vehicle.cpp]

```
#include "vehicle.h"

//Default Constructor
vehicle::vehicle()
{
	age = 0;
	price = 0.0;
}

//Setter Commands
void vehicle::setAge(int input)
{
	if (input >= 0)
	{
		age = input;
	}
	else
	{
		std::cout << "Invalid Age" << std::endl;
	}
```

```
}

void vehicle::setPrice(float input)
{
	if (input >= 0)
	{
		price = input;
	}
	else
	{
		std::cout << "Invalid Price" << std::endl;
	}
}

//Getter Commands
int vehicle::getAge()
{
	return age;
}

float vehicle::getPrice()
{
	return price;
}

//Default Destructor
vehicle::~vehicle()
{
}
```

[car.h]

```
#pragma once
#include <iostream>
#include "vehicle.h"

#ifndef CAR_H
#define CAR_H

class car: public vehicle
{
public:

	car();		//Default Constructor

	void setRaceCarStatus(bool);
	bool getRaceCarStatus();
```

```
	~car();		//Default Destructor
private:

	bool RaceCarStatus;
};

#endif CAR_H
```

[car.cpp]

```
#include "car.h"

//Default Constructor
car::car()
{
	RaceCarStatus = 0;
}

//Setter
void car::setRaceCarStatus(bool input)
{
	RaceCarStatus = input;
}

//Getter
bool car::getRaceCarStatus()
{
	return RaceCarStatus;
}

//Default Destructor
car::~car()
{
}
```

[truck.h]

```
#pragma once
#include "vehicle.h"

#ifndef TRUCK_H
#define TRUCK_H

class truck: public vehicle
{
public:
```

```
	truck();		//Default Constructor

	void setDieselTypeStatus(bool);
	bool getDieselTypeStatus();

	~truck();		//Default Destructor
protected:
	bool DieselTypeStatus;
};


#endif TRUCK_H
```

[truck.cpp]

```
#include "truck.h"

//Default Constructor
truck::truck()
{
	DieselTypeStatus = false;
}

//Setter
void truck::setDieselTypeStatus(bool input)
{
	DieselTypeStatus = input;
}

//Getter
bool truck::getDieselTypeStatus()
{
	return DieselTypeStatus;
}

//Default Destructor
truck::~truck()
{

}
```

UML / Generalization Diagram