**Fraser Burch & Dino Bertoli**

**Thursday Lab**

**Lab 5**

**Automobile.h**

#ifndef AUTOMOBILE\_H

#define AUTOMOBILE\_H

#include <string>

class Automobile

{

public:

Automobile();

Automobile(std::string imake, std::string imodel, int iyear, double iprice, int istall);

bool park(bool lot[], int size);

std::string getMake(){return make\_;}

std::string getModel(){return model\_;}

int getYear(){return year\_;}

double getPrice(){return price\_;}

int getStall(){return stall\_;}

void setStall(int s){stall\_ = s;}

private:

std::string make\_;

std::string model\_;

int year\_;

double price\_;

int stall\_;

};

#endif // AUTOMOBILE\_H

**Automobile.cpp**

#include "automobile.h"

Automobile::Automobile(){

make\_ = "";

model\_="";

year\_=0;

price\_=0;

stall\_=-1;

}

Automobile::Automobile(std::string imake, std::string imodel, int iyear, double iprice, int istall){

make\_ = imake;

model\_=imodel;

year\_=iyear;

price\_=iprice;

stall\_=istall;

}

bool Automobile::park(bool lot[], int size){

int spot=size;

for(int i=0; i<size; i++){

if(!lot[i])

spot = i;

}

if(spot>=size)

return false;

else{

lot[spot] = true;

stall\_ = spot;

return true;

}

}

**Pickup.h**

#ifndef PICKUP\_H

#define PICKUP\_H

#include "automobile.h"

#include <string>

class Pickup: public Automobile{

private:

bool extCab;

int haulCap;

public:

Pickup();

Pickup(int haul, bool cab, std::string imake, std::string imodel, int iyear, double iprice, int istall);

int getHaul(){

return haulCap;

}

bool getCab(){

return extCab;

}

bool park(bool lot[], int size);

};

#endif

**Pickup.cpp**

#include "pickup.h"

Pickup::Pickup(): Automobile(){

haulCap = 0;

extCab = false;

}

Pickup::Pickup(int haul, bool cab, std::string imake, std::string imodel, int iyear, double iprice, int istall): Automobile(imake, imodel, iyear, iprice, istall){

haulCap = haul;

extCab = cab;

}

bool Pickup::park(bool lot[], int size){

int spot=size;

for(int i = 0; i < size-1; i++){

if(lot[i] == false && lot[i+1] == false){

spot = i;

}

}

if(spot != size){

setStall(spot);

lot[spot] = true;

lot[spot+1] = true;

return (true);

}

return false;

}

**Van.h**

#ifndef VAN\_H

#define VAN\_H

#include "automobile.h"

#include <string>

class Van: public Automobile{

private:

int seats;

public:

Van();

Van(int numSeats, std::string imake, std::string imodel, int iyear, double iprice, int istall);

int getSeats(){return seats;}

bool park(bool lot[], int size);

};

#endif

**Van.cpp**

#include "van.h"

Van::Van(){

seats = 0;

}

Van::Van (int numSeats, std::string imake, std::string imodel, int iyear, double iprice, int istall): Automobile(imake, imodel, iyear, iprice, istall){

seats = numSeats;

}

bool Van::park(bool lot[], int size){

if(seats <= 7){

Automobile::park(lot, size);

}

else{

int spot=size;

for(int i = 0; i < size-1; i++){

if(lot[i] == false && lot[i+1] == false){

spot = i;

}

}

if(spot != size){

setStall(spot);

lot[spot] = true;

lot[spot+1] = true;

return (true);

}

}

return false;

}

**Main.cpp**

#include "pickup.h"

#include "van.h"

#include <iostream>

using namespace std;

int main(){

bool parkSpots[8] = {true, false, true, false, false, true, true, false};

Automobile test = Automobile();

bool thisBool = test.park(parkSpots, 8);

cout<<(thisBool)<<endl;

//Testing Pickup Class

Pickup loadedTruck = Pickup(10000, true, "Ford", "F-150", 2019, 43760.98, 2);

Pickup defaultTruck = Pickup();

cout << loadedTruck.getCab() << endl;

cout << loadedTruck.getHaul() << endl;

loadedTruck.park(parkSpots, 8);

cout<<loadedTruck.getStall() <<endl;

//Testing Van Class

Van defaultVan = Van();

Van bigVan = Van(10, "Honda", "MiniVan", 2018, 12760.98, 3);

Van smallVan = Van(6, "Honda", "MiniVan", 2018, 12760.98, 5);

bool vanParkSpots[8] = {true, false, true, false, false, true, true, false};

bool bigVanParkSpots[8] = {true, false, true, false, false, true, true, false};

smallVan.park(vanParkSpots, 8);

bigVan.park(bigVanParkSpots, 8);

cout<<"Small van parked: " << smallVan.getStall() <<endl;

cout<<"Big van parked: " <<bigVan.getStall() <<endl;

return(0);

}

**FINAL OUTPUT:**

FHosts-MacBook-Pro:Lab 5 fhost$ g++ \*.cpp -o main

FHosts-MacBook-Pro:Lab 5 fhost$ ./main

1

1

10000

3

Small van parked: 7

Big van parked: 3

FHosts-MacBook-Pro:Lab 5 fhost$