Phase 2 lines of code – 998

Phase 1 lines of code: 1663

Emilia 489

Nikki 609

Sam 565

NAME: Geek Shop

DESCRIPTION: Inventory manager for technology shop – commercial

AUTHORS: Nikki Perez, Emiliia Dyrenkova, Samuel Nguyen

PHASE 2 DEVELOPMENT PROBLEM: Successfully implemented the program using one List class (employing polymorphism) after running into some challenges. We used GitHub for communication among team members as well as STEM Center support and office hours.

**<TechProductMenu.cpp>**

#include <iostream>

#include <iomanip>

#include <string>

#include "TechProduct.h"

#include "TechProductList.h"

#include "Drone.h"

#include "VoiceAssistant.h"

#include "Headphones.h"

#include "Laptops.h"

#include "Battery.h"

#include "Camera.h"

void addToTheList(TechProductList& list, string productType){

string brand, model;

int quantity,userChoice;

double price;

if(productType=="Drone"){

Drone \*product = new Drone();

double time, diameter;

string camera;

cout << "Enter Brand: ";

getline(cin, brand);

cout << "Enter Model: ";

getline(cin, model);

cout << "Enter Time of Life per charge: ";

cin >> time;

cout << "Enter Diameter of Signal: ";

cin >> diameter;

cout << "Enter camera (n/a if no camera is present): ";

cin >> camera;

cout << "Enter Quantity: ";

cin >> quantity;

cout << "Enter Price $";

cin >> price;

product->setBrand(brand);

product->setModel(model);

product->setPrice(price);

product->setQuantity(quantity);

product->setTimePerCharge(time);

product->setSignalDiameter(diameter);

product->setCamera(camera);

if(!list.addTechProduct(product)){

cout<<"The list is full!\n";

}else cout<<"Added!\n";

}

else if(productType=="Voice Assistant"){

VoiceAssistant \*product = new VoiceAssistant();

double weight;

string color, searchEngine;

cout << "Enter Brand: ";

getline(cin, brand);

cout << "Enter Model: ";

getline(cin, model);

cout << "Enter Search Engine: ";

cin >> searchEngine;

cout << "Enter Color: ";

cin >> color;

cout << "Enter weight: ";

cin >> weight;

cout << "Enter Quantity: ";

cin >> quantity;

cout << "Enter Price $";

cin >> price;

product->setBrand(brand);

product->setModel(model);

product->setPrice(price);

product->setQuantity(quantity);

product->setSearchEngine(searchEngine);

product->setColor(color);

product->setWeight(weight);

if(!list.addTechProduct(product)){

cout<<"The list is full!\n";

}

else cout<<"Added!\n";

}

else if(productType=="Headphone"){

Headphones \*product = new Headphones();

int impedance;

string driverSize;

cout << "Enter Brand: ";

getline(cin, brand);

cout << "Enter Model: ";

getline(cin, model);

cout << "Enter driver size (small, medium, large): ";

cin >> driverSize;

cout << "Enter Impedance: ";

cin >> impedance;

cout << "Enter Quantity: ";

cin >> quantity;

cout << "Enter Price $";

cin >> price;

product->setBrand(brand);

product->setModel(model);

product->setPrice(price);

product->setQuantity(quantity);

product->setImpedance(impedance);

product->setDriverSize(driverSize);

if(!list.addTechProduct(product)){

cout<<"The list is full!\n";

}

else cout<<"Added!\n";

}

else if(productType=="Laptop"){

Laptops \*product = new Laptops();

int storage;

string cpu;

cout << "Enter Brand: ";

getline(cin, brand);

cout << "Enter Model: ";

getline(cin, model);

cout << "Enter CPU name: ";

cin >> cpu;

cout << "Enter Storage Size (in GB): ";

cin >> storage;

cout << "Enter Quantity: ";

cin >> quantity;

cout << "Enter Price $";

cin >> price;

product->setBrand(brand);

product->setModel(model);

product->setPrice(price);

product->setQuantity(quantity);

product->setStorage(storage);

product->setCpu(cpu);

if(!list.addTechProduct(product)){

cout<<"The list is full!\n";

}

else cout<<"Added!\n";

}

else if(productType=="Battery"){

Battery \*product = new Battery();

int capacity;

cout << "Enter Brand: ";

getline(cin, brand);

cout << "Enter Model: ";

getline(cin, model);

cout << "Enter Capacity: ";

cin >> capacity;

cout << "Enter Quantity: ";

cin >> quantity;

cout << "Enter Price $";

cin >> price;

product->setBrand(brand);

product->setModel(model);

product->setPrice(price);

product->setQuantity(quantity);

product->setCapacity(capacity);

if(!list.addTechProduct(product)){

cout<<"The list is full!\n";

}

else cout<<"Added!\n";

}

else if(productType=="Camera"){

Camera \*product = new Camera();

string aperture;

cout << "Enter Brand: ";

getline(cin, brand);

cout << "Enter Model: ";

getline(cin, model);

cout << "Enter Aperture: ";

cin >> aperture;

cout << "Enter Quantity: ";

cin >> quantity;

cout << "Enter Price $";

cin >> price;

product->setBrand(brand);

product->setModel(model);

product->setPrice(price);

product->setQuantity(quantity);

product->setAperture(aperture);

if(!list.addTechProduct(product)){

cout<<"The list is full!\n";

}

else cout<<"Added!\n";

}

}

void innerMenu(TechProductList& list, string productType){

string brand, model;

int quantity,userChoice, id;

double price;

do {

cout << "Please make a choice from the following options:" << endl;

cout <<"1) Add a new "<<productType<<" to Inventory" << endl;

cout <<"2) Remove a "<<productType<<" from Inventory" << endl;

cout <<"3) Update a "<<productType<<" from Inventory" << endl;

cout <<"4) Display all "<<productType<<"s in Inventory" << endl;

cout <<"5) Exit" << endl;

cout << ">> ";

cin >> userChoice;

cin.ignore(INT\_MAX, '\n');

switch (userChoice)

{

case 1:

{

addToTheList(list,productType);

break;

}

case 2:

cout << "\nWhich ID# would you like to remove? (or -1 to cancel) >> ";

cin >> id;

if(!list.removeTechProduct(id)){

cout<<"There is no such " << productType << " in the inventory\n";

}

else{

cout<<"Removed!\n";

}

break;

case 3:

cout << "\nWhich ID# would you like to update? (or -1 to cancel) >> ";

cin >> id;

if(!list.updateTechProduct(id))

{

cout<<"There is no such " << productType << " in the inventory\n";

}

break;

case 4:

cout << endl;

list.print();

cout << endl;

break;

case 5:

cout << endl << "Bzzz, bye!!!" << endl;

break;

}

cout << endl;

} while (userChoice != 5);

}

int main(int argv, char\* argc[])

{

TechProductList droneList;

TechProductList VAList;

TechProductList HeadphonesList;

TechProductList LaptopsList;

TechProductList BatteryList;

TechProductList CameraList;

int userChoice = 0;

do {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\* \*\*" << endl;

cout << "\*\* WELCOME TO THE \*\*" << endl;

cout << "\*\* GEEK SHOP \*\*" << endl;

cout << "\*\* \*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\* Please make a choice from the following options: \*\*" << endl;

cout << "\*\* 1) Manage Drones \*\*" << endl;

cout << "\*\* 2) Manage Voice Assistants \*\*" << endl;

cout << "\*\* 3) Manage Headphones \*\*" << endl;

cout << "\*\* 4) Manage Laptops \*\*" << endl;

cout << "\*\* 5) Manage Cameras \*\*" << endl;

cout << "\*\* 6) Manage Batteries \*\*" << endl;

cout << "\*\* 7) Exit \*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << ">> ";

cin >> userChoice;

cin.ignore(INT\_MAX, '\n');

switch (userChoice)

{

case 1:

{

innerMenu(droneList,"Drone");

break;

}

case 2:

innerMenu(VAList,"Voice Assistant");

break;

case 3:

innerMenu(HeadphonesList,"Headphone");

break;

case 4:

innerMenu(LaptopsList,"Laptop");

break;

case 5:

innerMenu(CameraList,"Camera");

break;

case 6:

innerMenu(BatteryList,"Battery");

break;

case 7:

cout<<"Bye\n";

break;

}

cout << endl;

} while (userChoice != 7);

system("PAUSE");

return EXIT\_SUCCESS;

}

**<VoiceAssistant.h>**

#ifndef VOICEASSISTANT\_H

#define VOICEASSISTANT\_H

#include <string>

#include <iostream>

#include "TechProduct.h"

using namespace std;

class VoiceAssistant:public TechProduct

{

private:

string m\_searchEngine;

string m\_color;

double m\_weight;

public:

static const int MAX\_INVENTORY=100;

VoiceAssistant();

VoiceAssistant(string brand, string model, string searchEngine, string color, double weight, double price, int quantity);

inline string getSearchEngine() const { return m\_searchEngine; }

inline string getColor() const { return m\_color; }

inline double getWeight() const { return m\_weight;}

void update();

inline void setSearchEngine(string searchEngine) { m\_searchEngine = searchEngine; }

inline void setColor(string color) { m\_color = color; }

inline void setWeight(double weight) { m\_weight = weight; }

bool operator==(const VoiceAssistant& other);

void print();

};

#endif

**<VoiceAssistant.cpp>**

#include "VoiceAssistant.h"

VoiceAssistant::VoiceAssistant() : TechProduct()

{

m\_searchEngine="";

m\_color="";

m\_weight=0.0;

}

VoiceAssistant::VoiceAssistant(string brand, string model, string searchEngine, string color, double weight, double price, int quantity):TechProduct::TechProduct(brand,model,price,quantity)

{

m\_searchEngine=searchEngine;

m\_color=color;

m\_weight=weight;

}

bool VoiceAssistant::operator==(const VoiceAssistant& other) //compares only brand, model, color and price. Those should be enough to identify the same model of Voice Assistant.

{

return m\_brand == other.m\_brand && m\_model == other.m\_model && m\_color == other.m\_color

&& m\_price == other.m\_price;

}

void VoiceAssistant::print()

{

cout << "Voice Assistant [ID#" << m\_id << ", Brand=" << m\_brand << ", Model=" << m\_model << ", Search Engine=" << m\_searchEngine

<< ", Color=" << m\_color << ", Weight=" << m\_weight << " pounds, Quantity=" << m\_quantity

<< ", Price=$" << m\_price << "]";

}

void VoiceAssistant::update(){

string brand, model, searchEngine, color;

double weight, price;

int quantity;

cin.ignore(INT\_MAX, '\n');

cout << "\nEnter Updated Brand: ";

getline(cin, brand);

cout << "Enter Updated Model: ";

getline(cin, model);

cout << "Enter Updated Search Engine: ";

getline(cin, searchEngine);

cout << "Enter Updated Color: ";

cin >> color;

cout << "Enter Updated weight: ";

cin >> weight;

cout << "Enter Updated Quantity: ";

cin >> quantity;

cout << "Enter Updated Price $";

cin >> price;

setBrand(brand);

setModel(model);

setSearchEngine(searchEngine);

setWeight(weight);

setColor(color);

setPrice(price);

setQuantity(quantity);

}

**<TechProductList.h>**

#ifndef TECHPRODUCTLIST\_H

#define TECHPRODUCTLIST\_H

#include "TechProduct.h"

#include <iostream>

#include <vector>

using namespace std;

class TechProductList

{

private:

int m\_count;

const static int SIZE=100;

TechProduct \*m\_list[SIZE];

public:

TechProductList();

bool addTechProduct(TechProduct \*d);

bool removeTechProduct(int id);

bool updateTechProduct(int id);

void print();

};

#endif

**<TechProductList.cpp>**

#include "TechProductList.h"

#include <iostream>

using namespace std;

TechProductList::TechProductList(){

m\_count=0;

}

bool TechProductList::addTechProduct(TechProduct \*d){

if(m\_count<SIZE){

m\_count++;

m\_list[m\_count-1]=d;

return true;

}

else{

return false;

}

}

bool TechProductList::removeTechProduct(int id){

int index;

bool found=false;

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

if(m\_list[i]->getId()==id){

index=i;

found=true;

break;

}

}

if(found){

m\_count--;

for(int i=index;i<SIZE-1;i++){

m\_list[i]=m\_list[i+1];

}

}

return found;

}

bool TechProductList::updateTechProduct(int id){

string brand, model, color, search;

double weight, price;

int quantity;

for(int i=0;i<m\_count;i++){

if(m\_list[i]->getId()==id){

m\_list[i]->update();

return true;

}

}

return false;

}

void TechProductList::print(){

for(int i=0;i<m\_count;i++){

m\_list[i]->print();

}

}

**<TechProduct.h>**

#ifndef TECHPRODUCT\_H

#define TECHPRODUCT\_H

#include <iostream>

using namespace std;

class TechProduct

{

protected:

static int s\_nextId;

int m\_id;

string m\_brand;

string m\_model;

int m\_quantity;

double m\_price;

public:

TechProduct();

TechProduct(string brand, string model, double price, int quantity);

static const int MAX\_INVENTORY=100;

inline int getId() const { return m\_id; }

inline string getBrand() const { return m\_brand; }

inline string getModel() const { return m\_model; }

inline double getPrice() const { return m\_price; }

inline int getQuantity() const { return m\_quantity; }

inline void setBrand(string brand) { m\_brand = brand; }

inline void setModel(string model) { m\_model = model; }

inline void setPrice(double price) { m\_price = price; }

inline void setQuantity(int quantity) { m\_quantity = quantity; }

virtual void update()=0;

virtual void print()=0;

};

#endif

**<TechProduct.cpp>**

#include "TechProduct.h"

#include <string>

#include <iostream>

int TechProduct::s\_nextId = 10000;

TechProduct::TechProduct()

{

m\_id = ++s\_nextId;

m\_brand = "";

m\_model = "";

m\_quantity=0;

m\_price=0;

}

TechProduct::TechProduct(string brand, string model, double price=0, int quantity=0)

{

m\_id = ++s\_nextId;

m\_brand = brand;

m\_model = model;

m\_quantity=quantity;

m\_price=price;

}

**<Laptop.h>**

#include "TechProduct.h"

#ifndef LAPTOPS\_H

#define LAPTOPS\_H

#include <iostream>

using namespace std;

class Laptops : public TechProduct

{

private:

static int inventorySize;

static int t\_nextId;

int m\_id;

string m\_model;

string m\_brand;

string m\_cpu;

int m\_quantity;

double m\_price;

int m\_storage;

public:

Laptops();

Laptops(string model, string brand, string cpu, int quantity, double price, int storage);

inline int getId() const { return m\_id; }

inline string getBrand() const { return m\_brand; }

inline string getModel() const { return m\_model; }

inline string getCpu() const { return m\_cpu; }

inline int getQuantity() const { return m\_quantity; }

inline double getPrice() const { return m\_price; }

inline int getStorage() const { return m\_storage; }

inline void setBrand(string brand) { m\_brand = brand; }

inline void setModel(string model) { m\_model = model; }

inline void setCpu(string cpu) { m\_cpu = cpu; }

inline void setQuantity(int quantity) { m\_quantity = quantity; }

inline void setPrice(double price) { m\_price = price; }

inline void setStorage(int storage) { m\_storage = storage; }

bool operator==(const Laptops& other);

ostream operator<<(const Laptops& other);

friend ostream& operator<<(ostream& os, const Laptops& t);

void print();

void update();

};

#endif

**<Laptop.cpp>**

#include "Laptops.h"

#include <string>

int Laptops::inventorySize=100;

int Laptops::t\_nextId=10000;

Laptops::Laptops() : TechProduct()

{

m\_cpu="";

m\_storage=0;

}

Laptops::Laptops(string model, string brand, string cpu, int quantity, double price, int storage) : TechProduct(brand, model, price, quantity)

{

m\_cpu=cpu;

m\_storage=storage;

}

bool Laptops::operator==(const Laptops& other)

{

return m\_model==other.m\_model && m\_brand==other.m\_brand && m\_cpu==other.m\_cpu

&& m\_quantity==other.m\_quantity && m\_price==other.m\_price && m\_storage==other.m\_storage;

}

ostream& operator<<(ostream& os, const Laptops& t)

{

os << "Laptops [ID#" << t.m\_id << ", Brand=" << t.m\_brand << ", Model=" << t.m\_model << ", CPU name= " << t.m\_cpu

<< ", Storage(GB)=" << t.m\_storage << ", Quantity= " << t.m\_quantity << ", Price=$" << t.m\_price << "]";

return os;

}

void Laptops::print()

{

cout << "Laptops [ID#" << m\_id << ", Brand=" << m\_brand << ", Model=" << m\_model << ", CPU name= " << m\_cpu

<< ", Storage(GB)=" << m\_storage << ", Quantity= " << m\_quantity << ", Price=$" << m\_price << "]";;

}

void Laptops::update()

{

string model, brand, cpu;

int quantity, storage;

double price;

cout << "Enter Updated Brand: ";

getline(cin, brand);

cout << "Enter Updated Model: ";

getline(cin, model);

cout << "Enter Updated CPU description: ";

getline(cin, cpu);

cout << "Enter Updated Storage(Ohms): ";

cin >> storage;

cout << "Enter Updated Quantity: ";

cin >> quantity;

cout << "Enter Updated Price $";

cin >> price;

setBrand(brand);

setModel(model);

setCpu(cpu);

setStorage(storage);

setQuantity(quantity);

setPrice(price);}

**<Headphones.h>**

#include "TechProduct.h"

#ifndef HEADPHONES\_H

#define HEADPHONES\_H

#include <iostream>

using namespace std;

class Headphones : public TechProduct

{

private:

static int inventorySize;

static int t\_nextId;

int m\_id;

string m\_model;

string m\_brand;

string m\_driverSize;

int m\_quantity;

double m\_price;

int m\_impedance;

public:

Headphones();

Headphones(string model, string brand, string driverSize, int quantity, double price, int impedance);

inline int getId() const { return m\_id; }

inline string getBrand() const { return m\_brand; }

inline string getModel() const { return m\_model; }

inline string getDriverSize() const { return m\_driverSize; }

inline int getQuantity() const { return m\_quantity; }

inline double getPrice() const { return m\_price; }

inline int getImpedance() const { return m\_impedance; }

inline void setBrand(string brand) { m\_brand = brand; }

inline void setModel(string model) { m\_model = model; }

inline void setDriverSize(string driverSize) { m\_driverSize = driverSize; }

inline void setQuantity(int quantity) { m\_quantity = quantity; }

inline void setPrice(double price) { m\_price = price; }

inline void setImpedance(int impedance) { m\_impedance = impedance; }

bool operator==(const Headphones& other);

ostream operator<<(const Headphones& other);

friend ostream& operator<<(ostream& os, const Headphones& t);

void print();

void update();

};

#endif

**<Headphones.cpp>**

#include "Headphones.h"

#include <string>

Headphones::Headphones() : TechProduct()

{

m\_driverSize="";

m\_impedance=0;

}

Headphones::Headphones(string model, string brand, string driverSize, int quantity, double price, int impedance) : TechProduct(brand, model, price, quantity)

{

m\_driverSize=driverSize;

m\_impedance=impedance;

}

bool Headphones::operator==(const Headphones& other)

{

return

m\_model == other.getModel() &&

m\_brand == other.getBrand() &&

m\_quantity == other.getQuantity() &&

m\_price == other.getPrice() &&

m\_driverSize == other.m\_driverSize &&

m\_impedance == other.m\_impedance;

}

ostream& operator<<(ostream& os, const Headphones& t)

{

os << "Headphones [ID#" << t.m\_id << ", Brand=" << t.m\_brand << ", Model=" << t.m\_model << ", Driver Size= " << t.m\_driverSize

<< ", Impedance(Ohms)=" << t.m\_impedance << ", Quantity= " << t.m\_quantity << ", Price=$" << t.m\_price << "]";

return os;

}

void Headphones::print()

{

cout << "Headphones [ID#" << m\_id << ", Brand=" << m\_brand << ", Model=" << m\_model << ", Driver Size= " << m\_driverSize

<< ", Impedance(Ohms)=" << m\_impedance << ", Quantity= " << m\_quantity << ", Price=$" << m\_price << "]";

}

void Headphones::update()

{

string model, brand, driverSize;

int quantity, impedance;

double price;

cout << "Enter Updated Brand: ";

getline(cin,brand);

cout << "Enter Updated Model: ";

getline(cin, model);

cout << "Enter Updated Driver Size (small, medium, large): ";

getline(cin, driverSize);

cout << "Enter Updated Impedance(Ohms): ";

cin >> impedance;

cout << "Enter Updated Quantity: ";

cin >> quantity;

cout << "Enter Updated Price $";

cin >> price;

setBrand(brand);

setModel(model);

setDriverSize(driverSize);

setImpedance(impedance);

setQuantity(quantity);

setPrice(price);

}

**<Drone.h>**

#ifndef DRONE\_H

#define DRONE\_H

#include <iostream>

#include "TechProduct.h"

using namespace std;

class Drone:public TechProduct

{

private:

double m\_timePerCharge;

string m\_camera;

double m\_signalDiameter;

public:

Drone();

Drone(string brand, string model, double time, double diameter, string camera, double price, int quantity);

inline double getTimePerCharge() const { return m\_timePerCharge; }

inline string getCamera() const { return m\_camera; }

inline double getSignalDiameter() const { return m\_signalDiameter; }

virtual void update();

inline void setTimePerCharge(double time) { m\_timePerCharge = time; }

inline void setCamera(string camera) { m\_camera = camera; }

inline void setSignalDiameter(double diameter) { m\_signalDiameter = diameter; }

bool operator==(const Drone& other);

virtual void print();

};

#endif

**<Drone.cpp>**

#include "Drone.h"

#include <string>

Drone::Drone():TechProduct()

{

m\_timePerCharge=0;

m\_camera="";

m\_signalDiameter=0;

}

Drone::Drone(string brand, string model, double time, double diameter, string camera, double price=0, int quantity=0):TechProduct::TechProduct(brand, model, price, quantity)

{

m\_timePerCharge=time;

m\_camera=camera;

m\_signalDiameter=diameter;

}

bool Drone::operator==(const Drone& other) //compares only brand, model, camera, time per charge and price. Those should be enough to identify the same model of Drone.

{

return m\_brand == other.m\_brand && m\_model == other.m\_model && m\_camera == other.m\_camera && m\_timePerCharge == other.m\_timePerCharge

&& m\_price == other.m\_price;

}

void Drone::print()

{

cout << "Drone [ID#" << m\_id << ", Brand=" << m\_brand << ", Model=" << m\_model << ", Time per Charge=" << m\_timePerCharge

<< " min, Camera=" << m\_camera << ", Signal diameter=" << m\_signalDiameter << " miles, Quantity=" << m\_quantity

<< ", Price=$" << m\_price << "]";

}

void Drone::update(){

string brand, model, camera;

double diameter, price, time;

int quantity;

cin.ignore(INT\_MAX, '\n');

cout << "\nEnter Updated Brand: ";

getline(cin, brand);

cout << "Enter Updated Model: ";

getline(cin, model);

cout << "Enter Updated camera: ";

getline(cin, camera);

cout << "Enter Updated Lifespan per Charge: ";

cin >> time;

cout << "Enter Updated Diameter of Signal Reception: ";

cin >> diameter;

cout << "Enter Updated Quantity: ";

cin >> quantity;

cout << "Enter Updated Price $";

cin >> price;

setBrand(brand);

setModel(model);

setCamera(camera);

setTimePerCharge(time);

setSignalDiameter(diameter);

setPrice(price);

setQuantity(quantity);

}

**<Camera.h>**

#ifndef CAMERA\_H

#define CAMERA\_H

#include <iostream>

#include "TechProduct.h"

using namespace std;

class Camera:public TechProduct

{

private:

string m\_aperture;

public:

Camera();

Camera(string brand, string model, double price, int quantity, string aperture);

inline string getAperture() const { return m\_aperture; }

void update();

inline void setAperture(string aperture) { m\_aperture = aperture; }

bool operator==(const Camera& other);

void print();

};

#endif

**<Camera.cpp>**#include "Camera.h"

#include <string>

Camera::Camera():TechProduct()

{

m\_aperture = "";

}

Camera::Camera(string model, string brand, double price = 0.0, int quantity = 0, string aperture = ""):TechProduct::TechProduct(brand, model, price, quantity)

{

m\_aperture = aperture;

}

bool Camera::operator==(const Camera& other)

{

return m\_model == other.m\_model && m\_brand == other.m\_brand && m\_price == other.m\_price && m\_quantity == other.m\_quantity && m\_aperture == other.m\_aperture;

}

void Camera::print()

{

cout << "Camera[ID#" << m\_id << ", Camera Model = " << m\_model << ", Brand Name = " << m\_brand << ", price =" << m\_price

<< ", item quantity =" << m\_quantity << ", aperture =" << m\_aperture << "]";

}

void Camera::update(){

string brand, model, aperture;

double price;

int quantity;

cin.ignore(INT\_MAX, '\n');

cout << "\nEnter Updated Brand: ";

getline(cin, brand);

cout << "Enter Updated Model: ";

getline(cin, model);

cout << "Enter Updated Quantity: ";

cin >> quantity;

cout << "Enter Updated Aperture";

cin >> aperture;

cout << "Enter Updated Price $";

cin >> price;

setBrand(brand);

setModel(model);

setQuantity(quantity);

setAperture(aperture);

setPrice(price);

}

**<Battery.h>**

#ifndef BATTERY\_H

#define BATTERY\_H

#include "TechProduct.h"

#include <iostream>

using namespace std;

class Battery:public TechProduct

{

private:

int m\_capacity;

public:

Battery();

Battery(string brand, string model, double price, int quantity, int capacity);

inline int getCapacity() const { return m\_capacity; }

void update();

inline void setCapacity(int capacity) { m\_capacity = capacity; }

bool operator==(const Battery& other);

virtual void print();

};

#endif

**<Battery.cpp>**

#include "Battery.h"

#include <string>

Battery::Battery():TechProduct()

{

m\_capacity = 0;

}

Battery::Battery(string brand, string model, double price = 0.0 , int quantity=0, int capacity = 0 ):TechProduct(brand, model, price, quantity)

{

m\_capacity = capacity;

}

bool Battery::operator==(const Battery& other)

{

return m\_model == other.m\_model && m\_brand == other.m\_brand && m\_price == other.m\_price && m\_quantity == other.m\_quantity && m\_capacity == other.m\_capacity;

}

void Battery::print()

{

cout << "Battery[ID#" << m\_id << ", Battery Model = " << m\_model << ", Brand Name = " << m\_brand << ", price =" << m\_price

<< ", item quantity =" << m\_quantity << "Capacity =" << m\_capacity << "]";

}

void Battery::update(){

string brand, model;

double price;

int quantity, capacity;

cin.ignore(INT\_MAX, '\n');

cout << "\nEnter Updated Brand: ";

getline(cin, brand);

cout << "Enter Updated Model: ";

getline(cin, model);

cout << "Enter Updated Quantity: ";

cin >> quantity;

cout << "Enter Updated Capacity";

cin >> capacity;

cout << "Enter Updated Price $";

cin >> price;

setBrand(brand);

setModel(model);

setQuantity(quantity);

setCapacity(capacity);

setPrice(price);

}