```
//XboxConsole.h

/*****************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *****************************************************/
/
**********************************************************************
*************
 * CapstonePhase1

**********************************************************************
*************/


#ifndef XBOXCONSOLE_H
#define XBOXCONSOLE_H

#include <iostream>

using namespace std;

class XboxConsole
{
private:
    static int s_nextID;
    int m_id;
    string m_generation;
    string m_submodel;
    string m_chipset;
    int m_ramSize;
    int m_storageRating;
    int m_quantity;
    double m_price;


public:
    XboxConsole();
    XboxConsole(string, string, string, int, int, int, double);
    inline int getId() const { return m_id; }
    inline string getGeneration() const { return m_generation; }
    inline string getSubmodel() const { return m_submodel; }
    inline string getChipset() const { return m_chipset; }
    inline int getRamSize() const { return m_ramSize; }
```

```
    inline int getStorageRating() const { return
m_storageRating; }
    inline int getQuantity () const { return m_quantity; }
    inline double getPrice() const { return m_price; }

    inline void setGeneration(string generation) { m_generation
= generation; }
    inline void setSubmodel(string submodel) { m_submodel =
submodel; }
    inline void setChipset(string chipset) { m_chipset =
chipset; }
    inline void setRamSize(int ramSize) { m_ramSize = ramSize; }
    inline void setStorageRating(int storageRating)
{ m_storageRating = storageRating; }
    inline void setQuantity(int quantity) { m_quantity =
quantity; }
    inline void setPrice(double price) { m_price = price; }

    bool operator==(const XboxConsole& other);
    ostream operator<<(const XboxConsole& other);

    friend ostream& operator<<(ostream& os, const XboxConsole&
x);


};

#endif /* XboxConsole_h */




//XboxController.h

/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
**********************************************************************
*************
 * CapstonePhase1
```

```
**********************************************************************
*************/

#ifndef XBOXCONTROLLER_H
#define XBOXCONTROLLER_H

#include <iostream>

using namespace std;

class XboxController
{
    private:

        static int s_nextID;
        int m_id;
        string m_generation;
        string m_layout;
        string m_design;
        string m_color;
        int m_quantity;
        double m_price;

    public:
        XboxController();
        XboxController(string, string, string, string, int,
double);
        inline int getId() const { return m_id; }
        inline string getGeneration() const { return
m_generation; }
        inline string getLayout() const { return m_layout; }
        inline string getDesign() const { return m_design; }
        inline string getColor() const { return m_color; }
        inline int getQuantity () const { return m_quantity; }
        inline double getPrice() const { return m_price; }

        inline void setGeneration(string generation)
{ m_generation = generation; }
        inline void setLayout(string layout) { m_layout =
layout; }
        inline void setDesign(string design) { m_design =
design; }
        inline void setColor(string color) { m_color = color; }
        inline void setQuantity(int quantity) { m_quantity =
quantity; }
        inline void setPrice(double price) { m_price = price; }
```

```
        bool operator==(const XboxController& other);
        ostream operator<<(const XboxController& other);

        friend ostream& operator<<(ostream& os, const
XboxController& xc);
};

#endif /* XboxController_h */




//XboxExclusive.h


/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
**************************************************************************
*************
 * CapstonePhase1

**************************************************************************
*************/

#ifndef XBOXEXCLUSIVE_H
#define XBOXEXCLUSIVE_H

#include <iostream>

using namespace std;

class XboxExclusive
{
private:
    static int s_nextID;
    int m_id;
    string m_generation;
    string m_title;
    string m_edition;
```

```
    string m_genre;
    char m_esrbRating;
    int m_quantity;
    double m_price;


public:
    XboxExclusive();
    XboxExclusive(string, string, string, string, char, int,
double);
    inline int getId() const { return m_id; }
    inline string getGeneration() const { return m_generation; }
    inline string getTitle() const { return m_title; }
    inline string getEdition() const { return m_edition; }
    inline string getGenre() const { return m_genre; }
    inline char getEsrbRating() const { return m_esrbRating; }
    inline int getQuantity () const { return m_quantity; }
    inline double getPrice() const { return m_price; }

    inline void setGeneration(string generation) { m_generation
= generation; }
    inline void setTitle(string title) { m_title = title; }
    inline void setEdition(string edition) { m_edition =
edition; }
    inline void setGenre(string genre) { m_genre = genre; }
    inline void setEsrbRating(char esrbRating) { m_esrbRating =
esrbRating; }
    inline void setQuantity(int quantity) { m_quantity =
quantity; }
    inline void setPrice(double price) { m_price = price; }

    bool operator==(const XboxExclusive& other);
    ostream operator<<(const XboxExclusive& other);

    friend ostream& operator<<(ostream& os, const XboxExclusive&
xe);

};


#endif /* XboxExclusive_h */
```

```
//XboxConsoleList.h

/*************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *************************************************/
/
*************************************************************************
*************
 * CapstonePhase1

*************************************************************************
*************/

#ifndef XBOXCONSOLELIST_H
#define XBOXCONSOLELIST_H
#include "XboxConsole.h"
#include <iostream>

class XboxConsoleList
{
private:
    int m_count;
    const static int SIZE = 100;
    XboxConsole m_list[SIZE];

public:
    XboxConsoleList() { m_count = 0; }
    bool addXbox(XboxConsole xb);
    bool removeXbox(int id);
    bool updateConsole(int id, string generation, string
submodel, string chipset, int ramSize, int storageRating, int
quantity, double price);
    inline int getCount() const { return m_count; }

    friend ostream& operator<<(ostream& os, const
XboxConsoleList& xb);
};

#endif /* XboxConsoleList_h */
```

```
//XboxControllerList.h



/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
**************************************************************************
*************
 * CapstonePhase1

**************************************************************************
*************/

#ifndef XBOXCONTROLLERLIST_H
#define XBOXCONTROLLERLIST_H

#include "XboxController.h"
#include <iostream>

class XboxControllerList
{
    private:
        int m_count;
        const static int SIZE = 100;
        XboxController m_list[SIZE];

    public:
        XboxControllerList() { m_count = 0; }
        bool addXboxController(XboxController xc);
        bool removeXboxController(int id);
        bool updateController(int id, string generation, string
layout, string design, string color, int quantity, double
price);
        inline int getCount() const { return m_count; }

        friend ostream& operator<<(ostream& os, const
XboxControllerList& xc);
    };
```

```
#endif /* XboxControllerList_h */




//XboxExclusiveList.h



/*************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *************************************************/
/
**********************************************************************
*************
 * CapstonePhase1

**********************************************************************
*************/

#ifndef XBOXEXCLUSIVELIST_H
#define XBOXEXCLUSIVELIST_H

#include "XboxExclusive.h"
#include <iostream>

class XboxExclusiveList
{
private:
    int m_count;
    const static int SIZE = 100;
    XboxExclusive m_list[SIZE];

public:
    XboxExclusiveList() { m_count = 0; }
    bool addXboxExclusive(XboxExclusive xe);
    bool removeXboxExclusive(int id);
    bool updateExclusive(int id, string generation, string
title, string edition, string genre, char esrbRating, int
quantity, double price);
```

```
    inline int getCount() const { return m_count; }

    friend ostream& operator<<(ostream& os, const
XboxExclusiveList& xe);
};

#endif /* XboxExclusiveList_h */




//XboxConsole.cpp


/*************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *************************************************/
/
**************************************************************************
*************
 * CapstonePhase1

**************************************************************************
*************/

#include "XboxConsole.h"
#include <string>

int XboxConsole::s_nextID = 10000;

XboxConsole::XboxConsole()
{
    m_id = s_nextID;
    m_generation = "One";
    m_submodel = "X";
    m_chipset = "Evolved Jaguar";
    m_ramSize = 12;
    m_storageRating = 500;
    m_quantity = 1;
    m_price = 400;
}
```

```
XboxConsole::XboxConsole(string generation, string submodel,
string chipset, int ramSize, int storageRating, int quantity,
double price)
{
    m_id = s_nextID++;
    m_generation = generation;
    m_submodel = submodel;
    m_chipset = chipset;
    m_ramSize = ramSize;
    m_storageRating = storageRating;
    m_quantity = quantity;
    m_price = price;
}

bool XboxConsole::operator==(const XboxConsole& other)
{
    return m_generation == other.m_generation && m_submodel ==
other.m_submodel && m_chipset == other.m_chipset && m_ramSize ==
other.m_ramSize && m_storageRating == other.m_storageRating &&
m_quantity == other.m_quantity && m_price == other.m_price;
}

ostream& operator<<(ostream& os, const XboxConsole& x)
{
    os << "Xbox [ID#" << x.m_id << ", Generation=" <<
x.m_generation << ", Submodel=" << x.m_submodel << ", Chipset="
<< x.m_chipset
    << ", RAM Size=" << x.m_ramSize << " GB, Storage Rating=" <<
x.m_storageRating << " GB, Quantity=" << x.m_quantity
    << ", Price=$" << x.m_price << "]";
    return os;
}
```

```
//XboxController.cpp
```

```
/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
```

```
 ***************************************************/
/
*****************************************************************
*************
 * CapstonePhase1

*****************************************************************
*************/
//

#include "XboxController.h"
#include <string>

int XboxController::s_nextID = 10000;

XboxController::XboxController()
{
    m_id = s_nextID;
    m_generation = "One";
    m_layout = "Elite";
    m_design = "Standard";
    m_color = "Red";
    m_quantity = 1;
    m_price = 60;
}

XboxController::XboxController(string generation, string layout,
string design, string color, int quantity, double price)
{
    m_id = s_nextID++;
    m_generation = generation;
    m_layout = layout;
    m_design = design;
    m_color = color;
    m_quantity = quantity;
    m_price = price;
}

bool XboxController::operator==(const XboxController& other)
{
    return m_generation == other.m_generation && m_layout ==
other.m_layout && m_design == other.m_design && m_color ==
other.m_color && m_quantity == other.m_quantity && m_price ==
other.m_price;
}

ostream& operator<<(ostream& os, const XboxController& xc)
```

```
{
    os << "Xbox Controller[ID#" << xc.m_id << ", Generation=" <<
xc.m_generation << ", Layout=" << xc.m_layout << ", Design=" <<
xc.m_design
    << ", Color=" << xc.m_color << ", Quantity=" <<
xc.m_quantity
    << ", Price=$" << xc.m_price << "]";
    return os;
}
```

```
//XboxExclusive.cpp
```

```
/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
*************************************************************************
*************
 * CapstonePhase1

*************************************************************************
*************/

#include "XboxExclusive.h"
#include <string>

int XboxExclusive::s_nextID = 10000;

XboxExclusive::XboxExclusive()
{
    m_id = s_nextID;
    m_generation = "360";
    m_title = "Halo 3";
    m_edition = "Standard";
    m_genre = "Sci-Fi";
```

```
    m_esrbRating = 'M';
    m_quantity = 1;
    m_price = 60;
}

XboxExclusive::XboxExclusive(string generation, string title,
string edition, string genre, char esrbRating, int quantity,
double price)
{
    m_id = s_nextID++;
    m_generation = generation;
    m_title = title;
    m_edition = edition;
    m_genre = genre;
    m_esrbRating = esrbRating;
    m_quantity = quantity;
    m_price = price;
}

bool XboxExclusive::operator==(const XboxExclusive& other)
{
    return m_generation == other.m_generation && m_title ==
other.m_title && m_edition == other.m_edition && m_genre ==
other.m_genre && m_esrbRating == other.m_esrbRating &&
m_quantity == other.m_quantity && m_price == other.m_price;
}

ostream& operator<<(ostream& os, const XboxExclusive& xe)
{
    os << "Xbox Exclusive[ID#" << xe.m_id << ", Generation=" <<
xe.m_generation << ", Title=" << xe.m_title << ", Edition=" <<
xe.m_edition
    << ", Genre=" << xe.m_genre << " , ESRB Rating=" <<
xe.m_esrbRating << " , Quantity=" << xe.m_quantity
    << ", Price=$" << xe.m_price << "]";
    return os;
}









//XboxConsoleList.cpp

/*********************************************************
 * AUTHOR: Kyle Stephan Harris
```

```
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 ************************************************************/
/
*************************************************************************
*************
 * CapstonePhase1

*************************************************************************
*************/


#include "XboxConsoleList.h"

using namespace std;

bool XboxConsoleList::addXbox(XboxConsole xb)
{
    if(m_count >= SIZE - 1)
    {
        return false;
    }
    m_list[m_count++] = xb;
    return true;
}
bool XboxConsoleList::removeXbox(int id)
{
    for(int i = 0; i < m_count; i++)
    {
        if(m_list[i].getId() == id)
        {
            for(int j = i; j < m_count; j++)
            {
                m_list[j] = m_list[j+1];
            }
            m_count--;
            return true;
        }
    }
    return false;
}

bool XboxConsoleList::updateConsole(int id, string generation,
string submodel, string chipset, int ramSize, int storageRating,
int quantity, double price)
```

```
{
    for(int i = 0; i < m_count; i++)
    {
        if(id == m_list[i].getId())
        {
            m_list[i].setGeneration(generation);
            m_list[i].setSubmodel(submodel);
            m_list[i].setChipset(chipset);
            m_list[i].setRamSize(ramSize);
            m_list[i].setStorageRating(storageRating);
            m_list[i].setQuantity(quantity);
            m_list[i].setPrice(price);
            return true;
        }
    }
    return false;
}

ostream& operator<<(ostream& os, const XboxConsoleList& xb)
{
    os << "~~~Current Inventory of Xbox Consoles~~~\n\n";
    for(int i = 0; i < xb.m_count; i++)
    {
        os << xb.m_list[i] << endl;
    }
    return os;
}
```

```
//XboxControllerList.cpp


/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
**********************************************************************
*************
 * CapstonePhase1
```

```
**********************************************************************
*************/

#include "XboxControllerList.h"

using namespace std;

bool XboxControllerList::addXboxController(XboxController xc)
{
    if(m_count >= SIZE - 1)
    {
        return false;
    }
    m_list[m_count++] = xc;
    return true;
}
bool XboxControllerList::removeXboxController(int id)
{
    for(int i = 0; i < m_count; i++)
    {
        if(m_list[i].getId() == id)
        {
            for(int j = i; j < m_count; j++)
            {
                m_list[j] = m_list[j+1];
            }
            m_count--;
            return true;
        }
    }
    return false;
}

bool XboxControllerList::updateController(int id, string
generation, string layout, string design, string color, int
quantity, double price)
{
    for(int i = 0; i < m_count; i++)
    {
        if(id == m_list[i].getId())
        {
            m_list[i].setGeneration(generation);
            m_list[i].setLayout(layout);
            m_list[i].setDesign(design);
            m_list[i].setColor(color);
            m_list[i].setQuantity(quantity);
```

```
            m_list[i].setPrice(price);
            return true;
        }
    }
    return false;
}

ostream& operator<<(ostream& os, const XboxControllerList& xc)
{
    os << "~~~Current Inventory of Xbox Controllers~~~\n\n";
    for(int i = 0; i < xc.m_count; i++)
    {
        os << xc.m_list[i] << endl;
    }
    return os;
}




//XboxExclusiveList.cpp


/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
**************************************************************************
*************
 * CapstonePhase1

**************************************************************************
*************/

#include "XboxExclusiveList.h"

using namespace std;

bool XboxExclusiveList::addXboxExclusive(XboxExclusive xe)
{
    if(m_count >= SIZE - 1)
```

```
    {
        return false;
    }
    m_list[m_count++] = xe;
    return true;
}
bool XboxExclusiveList::removeXboxExclusive(int id)
{
    for(int i = 0; i < m_count; i++)
    {
        if(m_list[i].getId() == id)
        {
            for(int j = i; j < m_count; j++)
            {
                m_list[j] = m_list[j+1];
            }
            m_count--;
            return true;
        }
    }
    return false;
}

bool XboxExclusiveList::updateExclusive(int id, string
generation, string title, string edition, string genre, char
esrbRating, int quantity, double price)
{
    for(int i = 0; i < m_count; i++)
    {
        if(id == m_list[i].getId())
        {
            m_list[i].setGeneration(generation);
            m_list[i].setTitle(title);
            m_list[i].setEdition(edition);
            m_list[i].setGenre(genre);
            m_list[i].setEsrbRating(esrbRating);
            m_list[i].setQuantity(quantity);
            m_list[i].setPrice(price);
            return true;
        }
    }
    return false;
}

ostream& operator<<(ostream& os, const XboxExclusiveList& xe)
{
    os << "~~~Current Inventory of Xbox Exclusives~~~\n\n";
```

```
    for(int i = 0; i < xe.m_count; i++)
    {
        os << xe.m_list[i] << endl;
    }
    return os;
}









//Project05_CapstonePhase1 (main)


/*********************************************************
 * AUTHOR: Kyle Stephan Harris
 * COURSE: CS 150: C++ Programming 1
 * SECTION: TTh 11:00-12:50
 * PROJECT: 05
 * LAST MODIFIED: 11/30/18
 *********************************************************/
/
*************************************************************************
*************
 * CapstonePhase1

*************************************************************************
*************/

#include <cstdlib>
#include <iostream>
#include "XboxConsole.h"
#include "XboxConsoleList.h"
#include "XboxController.h"
#include "XboxControllerList.h"
#include "XboxExclusive.h"
#include "XboxExclusiveList.h"

int main(int argc, char * argv[])
{
    XboxConsoleList x1;
    string generation, submodel, chipset;
    int id, ramSize, storageRating, quantity;
```

```
    double price;
    int choice = 0;
    int mainChoice = 0;

    //string generation,
    string layout, design, color;
    //double price;
    //int id, quantity;
    XboxControllerList controllerList;

    //string generation,
    string title, edition, genre;
    char esrbRating;
    //int id, quantity;
    //double price;
    XboxExclusiveList exclusiveList;

    do
    {
        cout <<
"**********************************************************************
*********" << endl;
        cout << "**
**" << endl;
        cout << "**                              WELCOME TO THE
**" << endl;
        cout << "**                            XBOX INVENTORY APP
**" << endl;
        cout << "**
**" << endl;
        cout <<
"**********************************************************************
*********" << endl;
        cout << "** Please make a choice from the following
options:                    **" << endl;
        cout << "** 1)  Xbox Console Inventory
**" << endl;
        cout << "** 2)  Xbox Controller Inventory
**" << endl;
        cout << "** 3)  Xbox Exclusive Inventory
**" << endl;
        cout << "** 4)  Exit
**" << endl;
        cout <<
"**********************************************************************
*********" << endl;
        cout << ">> ";
```

```
        cin >> mainChoice;
        cin.ignore(INT_MAX, '\n');

        switch(mainChoice)
        {
            case 1:
            {
                do
                {
                    cout <<
"*****************************************************************
*********" << endl;
                    cout << "** Please make a choice from the
following options:                    **" << endl;
                    cout << "** 1)  Add a new Xbox to Inventory
**" << endl;
                    cout << "** 2)  Remove an Xbox from
Inventory                                  **" << endl;
                    cout << "** 3)  Update an existing Xbox
Console                                  **" << endl;
                    cout << "** 4)  Display all Xbox Consoles in
Inventory                               **" << endl;
                    cout << "** 5)  Exit
**" << endl;
                    cout <<
"*****************************************************************
*********" << endl;
                    cout << ">> ";
                    cin >> choice;
                    cin.ignore(INT_MAX, '\n');
                    switch (choice)
                    {
                        case 1:
                        {
                            cout << "Enter Xbox Generation: ";
                            getline(cin, generation);
                            cout << "Enter Submodel: ";
                            getline(cin, submodel);
                            cout << "Enter Chipset: ";
                            getline(cin, chipset);
                            cout << "Enter RAM Size: ";
                            cin >> ramSize;
                            cout << "Enter Storage Rating: ";
                            cin >> storageRating;
                            cout << "Enter Quantity: ";
                            cin >> quantity;
                            cout << "Enter Price $";
```

```
                            cin >> price;


                            XboxConsole xb(generation, submodel,
chipset, ramSize, storageRating, quantity, price);
                            if(x1.addXbox(xb))
                            {
                                cout << "~~~Xbox added
successfully!~~~" << endl;
                            }
                            else
                            {
                                cout << "Inventory full, please
try again after removing." << endl;
                            }

                            cout << endl;
                            break;
                        }
                        case 2:
                        {

                            if(x1.getCount() == 0)
                            {
                                cout << "There is nothing to
remove!\n\n";
                            }
                            else
                            {
                                cout << x1 << endl;
                                cout << "\nWhich ID# would you
like to remove? (or -1 to cancel) >> ";
                                cin >> id;
                                if(x1.removeXbox(id))
                                {
                                    cout << "~~~Xbox removed
successfully!~~~" << endl;
                                }
                                else
                                {
                                    cout << "That ID does not
exist." << endl;
                                }
                            }

                            break;
                        }
```

```
                        case 3:
                        {
                            cout << x1 << endl;
                            cout << "\nWhich ID# would you like
to update? (or -1 to cancel) >> ";
                            cin >> id;
                            if (id == -1) break;
                            cin.ignore(INT_MAX, '\n');
                            cout << "Enter updated Xbox
Generation: ";
                            getline(cin, generation);
                            cout << "Enter updated Submodel: ";
                            getline(cin, submodel);
                            cout << "Enter updated Chipset: ";
                            getline(cin, chipset);
                            cout << "Enter updated RAM Size: ";
                            cin >> ramSize;
                            cout << "Enter updated Storage
Rating: ";
                            cin >> storageRating;
                            cout << "Enter updated Quantity: ";
                            cin >> quantity;
                            cout << "Enter updated Price $";
                            cin >> price;

                            if(x1.updateConsole(id, generation,
submodel, chipset, ramSize, storageRating, quantity, price))
                            {
                                cout << "\n~~~Xbox Console
Update Successful~~~\n";
                            }
                            else
                            {
                                cout << "\n~~~Failed to Update
Xbox Console~~~\n";
                            }

                        break;
                        }
                        case 4:
                        {
                            if(x1.getCount() == 0)
                            {
                                cout << "~~~Current Inventory of
Xbox Consoles~~~\n\nEmpty\n\n";
                            }
                            else
```

```
                            {
                                cout << x1 << endl;
                            }
                            break;
                        }
                        case 5:
                            break;

                    }

                }while (choice != 5);

            break;
            }

            case 2:
            {


                int userChoice = 0;

                do {
                    cout <<
"*************************************************************
*********" << endl;
                    cout << "** Please make a choice from the
following options:                    **" << endl;
                    cout << "** 1)  Add a new Xbox Controller to
Inventory                           **" << endl;
                    cout << "** 2)  Remove a Xbox Controller
from Inventory                        **" << endl;
                    cout << "** 3)  Update an existing Xbox
Controller                            **" << endl;
                    cout << "** 4)  Display all Xbox Controllers
in Inventory                        **" << endl;
                    cout << "** 5)  Exit
**" << endl;
                    cout <<
"*************************************************************
*********" << endl;
                    cout << ">> ";
                    cin >> userChoice;
                    cin.ignore(INT_MAX, '\n');
                    switch (userChoice)
                    {
                        case 1:
                        {
```

```
                            cout << "Enter Xbox Generation: ";
                            getline(cin, generation);
                            cout << "Enter Layout: ";
                            getline(cin, layout);
                            cout << "Enter Design: ";
                            getline(cin, design);
                            cout << "Enter Color: ";
                            getline(cin, color);
                            cout << "Enter Quantity: ";
                            cin >> quantity;
                            cout << "Enter Price $";
                            cin >> price;

                            XboxController xc(generation,
layout, design, color, quantity, price);
                            if
(controllerList.addXboxController(xc))
                                cout << "~~~Xbox Controller
added successfully!~~~" << endl;
                            else
                                cout << "~~~Failed to add Xbox
Controller to Inventory~~~" << endl;


                            cout << endl;
                            break;
                        }
                        case 2:
                        {
                            cout << controllerList << endl;
                            cout << "\nWhich ID# would you like
to remove? (or -1 to cancel) >> ";
                            cin >> id;
                            if (id == -1) break;
                            if
(controllerList.removeXboxController(id))
                                cout << "~~~Xbox Controller
removed successfully!~~~" << endl;
                            else
                                cout << "~~~Failed to remove
Xbox Controller from Inventory~~~" << endl;
                            break;
                        }

                        case 3:
                        {
                            cout << controllerList << endl;
```

```
                            cout << "\nWhich ID# would you like
to update? (or -1 to cancel) >> ";
                            cin >> id;
                            if (id == -1) break;
                            cin.ignore(INT_MAX, '\n');
                            cout << "Enter Updated Xbox
Generation: ";
                            getline(cin, generation);
                            cout << "Enter Updated Layout: ";
                            getline(cin, layout);
                            cout << "Enter Updated Design: ";
                            getline(cin, design);
                            cout << "Enter Updated Color: ";
                            getline(cin, color);
                            cout << "Enter Updated Quantity: ";
                            cin >> quantity;
                            cout << "Enter Updated Price $";
                            cin >> price;


if(controllerList.updateController(id, generation, layout,
design, color, quantity, price))
                            {
                                cout << "\n~~~Xbox Controller
Update Successful~~~\n";
                            }
                            else
                            {
                                cout << "\n~~~Failed to Update
Xbox Controller~~~\n";
                            }

                            break;
                        }
                        case 4:
                        {
                            if(controllerList.getCount() == 0)
                            {
                                cout << "~~~Current Inventory of
Xbox Controllers~~~\n\nEmpty\n\n";
                            }
                            cout << controllerList << endl;
                            break;
                        }
                        case 5:
                            break;
                        default:
```

```
                            cout << "Choice not recognized,
please drop in again." << endl;
                    }

                } while (userChoice != 5);
                cout << endl;
            break;
            }

            case 3:
            {


                int userChoice = 0;

                do {
                    cout <<
"**********************************************************************
*********" << endl;
                    cout << "** Please make a choice from the
following options:                      **" << endl;
                    cout << "** 1)  Add a new Xbox Exclusive to
Inventory                              **" << endl;
                    cout << "** 2)  Remove an Xbox Exclusive
from Inventory                           **" << endl;
                    cout << "** 3)  Update an existing Xbox
Exclusive                                **" << endl;
                    cout << "** 4)  Display all Xbox Exclusives
in Inventory                           **" << endl;
                    cout << "** 5)  Exit
**" << endl;
                    cout <<
"**********************************************************************
*********" << endl;
                    cout << ">> ";
                    cin >> userChoice;
                    cin.ignore(INT_MAX, '\n');
                    switch (userChoice)
                    {
                        case 1:
                        {
                            cout << "Enter Xbox Generation: ";
                            getline(cin, generation);
                            cout << "Enter Title: ";
                            getline(cin, title);
                            cout << "Enter Edition: ";
                            getline(cin, edition);
```

```
                            cout << "Enter Genre: ";
                            getline(cin, genre);
                            cout << "Enter ESRB Rating: ";
                            cin >> esrbRating;
                            cout << "Enter Quantity: ";
                            cin >> quantity;
                            cout << "Enter Price $";
                            cin >> price;

                            XboxExclusive xe(generation, title,
edition, genre, esrbRating, quantity, price);
                            if
(exclusiveList.addXboxExclusive(xe))
                                cout << "~~~Xbox exclusive added
successfully!~~~" << endl;
                            else
                                cout << "~~~Failed to add xbox
exclusive to Inventory~~~" << endl;


                            cout << endl;
                            break;
                        }
                        case 2:
                            cout << exclusiveList << endl;
                            cout << "\nWhich ID# would you like
to remove? (or -1 to cancel) >> ";
                            cin >> id;
                            if (id == -1) break;
                            if
(exclusiveList.removeXboxExclusive(id))
                                cout << "~~~Xbox exclusive
removed successfully!~~~" << endl;
                            else
                                cout << "~~~Failed to remove
xbox exclusive from Inventory~~~" << endl;
                            break;

                        case 3:
                        {
                            cout << exclusiveList << endl;
                            cout << "\nWhich ID# would you like
to update? (or -1 to cancel) >> ";
                            cin >> id;
                            if (id == -1) break;
                            cin.ignore(INT_MAX, '\n');
                            cout << "Enter updated Generation:
```

```
";
                            getline(cin, generation);
                            cout << "Enter updated Title: ";
                            getline(cin, title);
                            cout << "Enter updated Edition: ";
                            getline(cin, edition);
                            cout << "Enter updated Genre: ";
                            getline(cin, genre);
                            cout << "Enter updated ESRB Rating:
";
                            cin >> esrbRating;
                            cout << "Enter updated Quantity: ";
                            cin >> quantity;
                            cout << "Enter updated Price $";
                            cin >> price;

                            if(exclusiveList.updateExclusive(id,
generation, title, edition, genre, esrbRating, quantity, price))
                            {
                                cout << "\n~~~Xbox Exclusive
Update Successful~~~\n";
                            }
                            else
                            {
                                cout << "\n~~~Failed to Update
Xbox Exclusive~~~\n";
                            }

                            break;
                        }
                        case 4:
                        {
                            if(exclusiveList.getCount() == 0)
                            {
                                cout << "~~~Current Inventory of
Xbox Exclusives~~~\n\nEmpty\n\n";
                            }
                            cout << exclusiveList << endl;
                            break;
                        }
                        case 5:
                            break;
                        default:
                            cout << "Choice not recognized,
please drop in again." << endl;
                    }
```

```
                } while (userChoice != 5);
                cout << endl;

            break;
            }

            case 4:
                cout << "Thank you for using Xbox Inventory!";
            break;

            default:
                cout << "Please use a valid choice";
        }

    }while(mainChoice != 4);

    cout << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```