**HW 7**

**LAB 9**

**Name: Abdurasulov Mirsaid**

**ID: 12225253**

**Field: ISE**

**Course: Object oriented programming**

**Date: 14.11.2023**

**Exercise #1:**

***Source code:***

#include<iostream>

#include<iomanip>

#include<ctime>

#include<cstdlib>

using namespace std;

// Class representing an integer array

class IntArray

{

private:

int m\_len{ 0 };

int\* m\_data{ nullptr };

public:

// Constructor to initialize the array with a given length

IntArray(int len)

: m\_len{ len }

{

m\_data = new int[m\_len];

}

// Destructor to free dynamically allocated memory

~IntArray() {

if (m\_data) delete[] m\_data;

}

// Getter function to retrieve the array data

int\* getData() const {

return m\_data;

}

// Getter function to retrieve the array length

int getLength() const {

return m\_len;

}

};

// Class for handling operations on IntArray objects

class IntArrayHandler

{

private:

IntArray\* m\_array;

public:

// Constructor to initialize with an IntArray object

IntArrayHandler(IntArray\* array)

: m\_array{ array }

{

}

// Set the current IntArray object

void setIntArray(IntArray\* array) {

m\_array = array;

}

// Set the elements of the array with the given data

void setArray(int\* data, int len) {

if (m\_array && data) {

int arrayLen = m\_array->getLength();

int\* arrayData = m\_array->getData();

for (int i = 0; i < arrayLen && i < len; ++i) {

arrayData[i] = data[i];

}

}

}

// Display the elements of the array

void displayArray() const {

if (m\_array) {

int\* arrayData = m\_array->getData();

int arrayLen = m\_array->getLength();

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

cout << "[ " << setw(2) << i << "] " << arrayData[i] << endl;

}

}

}

// Calculate and display statistics of the array

void stat() const {

if (m\_array) {

int\* arrayData = m\_array->getData();

int arrayLen = m\_array->getLength();

int sum = 0;

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

sum += arrayData[i];

}

cout << "# of elements: " << arrayLen << endl;

cout << "Sum: " << sum << endl;

cout << "Average: " << fixed << setprecision(2) << static\_cast<double>(sum) / arrayLen << endl;

}

}

};

const int arSize = 20;

int main() {

int i;

int data1[arSize], data2[arSize];

IntArray ar1{ arSize }, ar2{ arSize };

srand((unsigned int)time(NULL));

for (i = 0; i < arSize; ++i) {

data1[i] = rand() % 100;

data2[i] = rand() % 100;

}

// Create IntArrayHandler objects and perform operations

IntArrayHandler handler{ &ar1 };

handler.setArray(data1, arSize);

cout << "== ar1: displayArray() ===" << endl;

handler.displayArray();

cout << "====== ar1: stat() =======" << endl;

handler.stat();

cout << "==========================" << endl;

handler.setIntArray(&ar2);

handler.setArray(data2, arSize);

cout << endl << "== ar2: displayArray() ===" << endl;

handler.displayArray();

cout << "====== ar2: stat() =======" << endl;

handler.stat();

cout << "==========================" << endl;

return 0;

}

***Result:***

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Exercise #2:**

***Source code:***

#include <iostream>

using namespace std;

class Test {

private:

static int counter;

int m\_id;

int m\_data;

public:

Test(int data)

: m\_id{ ++counter }, m\_data{ data }

{

}

// Static member function to get the number of objects created

static int NumOfObjects() {

return counter;

}

// Member function to get the ID of the object

int getID() const {

return m\_id;

}

// Member function to get the data of the object

int getData() const {

return m\_data;

}

};

// Initialize the static ID counter

int Test::counter = 0;

int main() {

cout << "# of objects: ";

cout << Test::NumOfObjects() << endl;

Test t1{ 10 }, t2{ 20 }, t3{ 30 }, t4{ 40 };

cout << "[t1] data: " << t1.getData() << ", id: " << t1.getID() << endl;

cout << "[t2] data: " << t2.getData() << ", id: " << t2.getID() << endl;

cout << "[t3] data: " << t3.getData() << ", id: " << t3.getID() << endl;

cout << "[t4] data: " << t4.getData() << ", id: " << t4.getID() << endl;

cout << "# of objects: ";

cout << Test::NumOfObjects() << endl;

return 0;

}

***Result:***

A screen shot of a computer

Description automatically generated

**Exercise #3:**

***Source code:***

#include <iostream>

using namespace std;

class Apple {

private:

string color;

const string fruit;

public:

Apple(string c, string f = "apple") : color(c), fruit(f) {};

// Getter function to retrieve the fruit name

string getName() const {

return fruit;

}

// Getter function to retrieve the color

string getColor() {

return color;

}

};

// Class representing a Banana, inheriting from Apple

class Banana : public Apple {

public:

string color; // Hides the color variable in the base class

const string fruit;

// Constructor to initialize a Banana with a default fruit "banana", a default color "yellow"

// Calls the constructor of the base class (Apple) to set the color and fruit

Banana(string f = "banana", string c = "yellow") : Apple(c, f) {};

};

int main() {

Apple a{ "red" };

Banana b;

cout << "My " << a.getName() << " is " << a.getColor() << ".\n";

cout << "My " << b.getName() << " is " << b.getColor() << ".\n";

return 0;

}

***Result:***



**Exercise #4:**

***Source code:***

#include <iostream>

using namespace std;

class Apple {

private:

string color;

string fruit;

public:

Apple(string c, string f = "apple") : color(c), fruit(f) {};

string getName() const {

return fruit;

}

string getColor() const {

return color;

}

};

class Banana : public Apple {

public:

Banana(string f = "banana", string c = "yellow") : Apple(c, f) {};

};

class RedBanana : public Banana {

public:

// Constructor for RedBanana, inheriting from Banana

// Calls the Banana constructor to set the fruit and color

RedBanana(string f = "red banana", string c = "red") : Banana(f, c) {};

};

int main() {

Apple a{ "red" };

Banana b;

RedBanana c;

cout << "My " << a.getName() << " is " << a.getColor() << ".\n";

cout << "My " << b.getName() << " is " << b.getColor() << ".\n";

cout << "My " << c.getName() << " is " << c.getColor() << ".\n";

return 0;

}

***Result:***

A black background with white text

Description automatically generated

**Exercise #5:**

***Source code:***

#include <iostream>

using namespace std;

// Class representing a physical book

class Book {

private:

string title;

string isbn; // ISBN (International Standard Book Number) of the book

double price;

public:

// Constructor to initialize the Book object

Book(string t, string i, double p) : title(t), isbn(i), price(p) {};

// Member function to display information about the book

void ShowBookInfo() {

cout << "Title: " << title << endl;

cout << "ISBN: " << isbn << endl;

cout << "Price(USD): " << price << endl;

}

};

// Class representing an electronic book (eBook), inheriting from Book

class EBook : public Book {

private:

string drmkey; // DRM

string format; // Format of the eBook (default is "Kindle")

public:

EBook(string t, string i, double p, string d, string f = "Kindle") : Book(t, i, p), drmkey(d), format(f) {};

// Member function to display information about the eBook, including Book information

void ShowEBookInfo() {

ShowBookInfo(); // Call the ShowBookInfo() function from the base class

cout << "DRMkey: " << drmkey << endl;

cout << "Format: " << format << endl;

}

};

int main() {

Book book("Modern C++ Programming Cookbook", "1800208987", 49.99);

book.ShowBookInfo();

cout << endl;

EBook ebook("Modern C++ Programming Cookbook(ebook)", "1800208987", 34.99, "dkb34x!@\*~");

ebook.ShowEBookInfo();

return 0;

}

***Result:***

A screen shot of a computer program

Description automatically generated

**Exercise #6:**

***Source code:***

#include <iostream>

using namespace std;

// Class representing a physical book

class Book {

private:

string title;

string isbn; // ISBN (International Standard Book Number) of the book

double price;

public:

// Constructor to initialize the Book object

Book(string t, string i, double p) : title(t), isbn(i), price(p) {};

// Member function to display information about the book

void ShowBookInfo() {

cout << "Title: " << title << endl;

cout << "ISBN: " << isbn << endl;

cout << "Price(USD): " << price << endl;

}

};

// Class representing an electronic book (eBook), inheriting from Book

class EBook : public Book {

private:

string drmkey; // DRM

string format; // Format of the eBook (default is "Kindle")

public:

EBook(string t, string i, double p, string d, string f = "Kindle") : Book(t, i, p), drmkey(d), format(f) {};

// Member function to display information about the eBook, including Book information

void ShowEBookInfo() {

ShowBookInfo(); // Call the ShowBookInfo() function from the base class

cout << "DRMkey: " << drmkey << endl;

cout << "Format: " << format << endl;

}

};

const int MAX\_BOOK = 100;

class EBookLibrary {

private:

EBook\* m\_books[MAX\_BOOK]; // Array to store pointers to EBook objects

int m\_cnt; // Current count of stored EBook objects

public:

EBookLibrary() : m\_cnt(0) {};

// Destructor to free dynamically allocated memory

~EBookLibrary() {

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

delete m\_books[i];

}

};

// Add Ebook to library

void AddBook(EBook\* book) {

if (m\_cnt < MAX\_BOOK) {

m\_books[m\_cnt++] = book;

}

else {

cout << "EBookLibrary is full. Cannot add more books." << endl;

}

}

// Show all info of ebooks

void ShowAllBooks(void) {

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

m\_books[i]->ShowEBookInfo();

cout << endl;

}

};

};

int main() {

EBookLibrary elib;

elib.AddBook(new EBook{ "Book1", "1234567890", 10.99, "AAAAAA", "ePub" });

elib.AddBook(new EBook{ "Book2", "2345678901", 20.99, "BBBBBB" });

elib.AddBook(new EBook{ "Book3", "3456789012", 30.99, "CCCCCC", "ePub" });

elib.AddBook(new EBook{ "Book4", "4567890123", 40.99, "DDDDDD" });

elib.ShowAllBooks();

return 0;

}

***Result:***

A screen shot of a computer

Description automatically generated