|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №2 по курсу «Объектно-ориентированные технологии программирования и стандарты проектирования»  «Создание классов и объектов»  Вариант 6 | |
| Выполнил | Студент гр. 21-ИТ-1  Катушёнок И.В. |
| Проверил | Ассистент  Хирьянов И.Д. |
| Полоцк, 2023г. | |

**Цель работы:** Научиться создавать иерархию классов, создавать объекты базовых классов иклассов-наследников, вызывать методы базового класса и класса-наследника,понимать влияние модификаторов доступа при наследовании, пониматьпорядок вызова конструкторов и деструкторов при наследовании.

**Ход работы**

**Задание:**

1. Дополнить и при необходимости модифицировать приложение,

разработанное согласно варианта лабораторной работы No2.

2. Создать несколько классов-наследников для дополнительного класса

(т.е. дополнительный класс станет базовым классом для новых

создаваемых классов). Каждый класс-наследник должен включать в себя

атрибуты, отличающие его от других классов.

3. Определить в новых классах конструкторы и деструктор.

4. В каждом конструкторе и деструкторе выдавать сообщение,

показывающее, какой именно конструктор или деструктор и какого

класса был вызван.

5. Определить в новых классах методы для ввода и вывода атрибутов как

самого класса-наследника, так и базового класса. Метод вывода должен

иметь то же имя, что и метод вывода в базовом классе.

6. Дополнить основное тело программы созданием объектов базового

класса, объектов классов-наследников, инициализацией атрибутов этих

объектов и выводом на экран этих объектов, не занося их в массив

объектов в основном классе из лабораторной No1 (в этой лабораторной

работе этот класс не задействован).

7. Создать массив объектов, тип массива – базовый класс. Поместить в

массив как объекты базового, так и объекты классов-наследников.

8. В цикле пройти по всем элементам созданного в п.7 массива, вызвать

метод вывода данных на экран.

9. Создать массив объектов, тип массива – один из классов-наследников.

Поместить в массив объекты этого класса-наследника.

10.В цикле пройти по всем элементам созданного в п.9 массива, вызвать

метод вывода данных на экран.

11.Сделать выводы.

**Вариант 6**

Варианты соответствуют лабораторной работе №1.

Листинг 1.1 – реализация меню программы:

class Program

{

public bool IsOnlyLetters(string str)

{

foreach (char c in str)

{

if (Char.IsNumber(c)) return false;

if (c == '-') return false;

if (c == '.') return false;

if (c == ',') return false;

if (c == ';') return false;

if (c == '/') return false;

if (c == ':') return false;

}

return true;

}

static void Main()

{

Catalog catalog = new Catalog();

while (true)

{

Console.WriteLine("Select operation");

Console.WriteLine("[1] - add\n[2] - delete\n[3] - edit\n[4] - search films\n[5] - info all\n[6] - sort");

int choise = Convert.ToInt32(Console.ReadLine());

switch (choise)

{

case 1:

Film kino = new Film();

catalog.AddToCatalog(kino);

break;

case 2:

catalog.RemoveFromCatalog();

break;

case 3:

Console.Write("Enter film posotion:");

int.TryParse(Console.ReadLine(), out int position);

List<Film> temp = catalog.GetFilmList();

temp[position].Edit();

catalog.SetFilmList(temp);

break;

case 4:

catalog.SearchFilm();

break;

case 5:

catalog.SeeAll();

break;

case 6:

catalog.SortFilms();

break;

}

Console.WriteLine("========================");

}

}

}

Листинг 1.2 – реализация основного класса:

class Catalog

{

List<Film> filmList = new List<Film>();

public List<Film> GetFilmList()

{

return filmList;

}

public void SetFilmList(List<Film> List)

{

filmList = List;

}

public void AddToCatalog(Film film)

{

filmList.Add(film);

}

public void RemoveFromCatalog()

{

Console.Write("Enter film number: ");

int.TryParse(Console.ReadLine(), out int position);

filmList.RemoveAt(position);

}

public int SearchFilmPosition(string name)

{

int counter = 0;

foreach (Film film in filmList)

{

if (film.GetName() == name) return counter;

}

return -1;

}

public int SearchFilmPosition(int year)

{

int counter = 0;

foreach (Film film in filmList)

{

if (film.GetYear() == year) return counter;

}

return -1;

}

public void GetInfoFromPosition(int filmPosition)

{

filmList[filmPosition].SeeInfo();

}

public void SearchFilm()

{

Console.WriteLine("How you want to search?");

Console.WriteLine("[1] - Name\n[2] - Year\n[3] - Type\n[4] - Rating\n[5] - Country.");

int choise = 0;

int counter = 0;

int.TryParse(Console.ReadLine(), out choise);

switch (choise)

{

case 1:

SearchName();

break;

case 2:

SearchYear();

break;

case 3:

SearchType();

break;

case 4:

SeatchRate();

break;

case 5:

SearchCounrty();

break;

}

}

public void SearchCounrty()

{

Console.WriteLine("Enter country");

string Country = Console.ReadLine();

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetCountry() == Country)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SearchType()

{

Console.WriteLine("Enter type");

string Type = Console.ReadLine();

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetType() == Type)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SearchName()

{

Console.WriteLine("Enter name");

string Name = Console.ReadLine();

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetName().IndexOf(Name) != -1)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SearchYear()

{

Console.WriteLine("Enter year");

int.TryParse(Console.ReadLine(), out int year);

Console.WriteLine("[1] - >\n[2] - <");

int.TryParse(Console.ReadLine(), out int choise);

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetYear() > year && choise == 1)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

if (filmList[i].GetYear() < year && choise == 2)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SeatchRate()

{

Console.WriteLine("Enter search rating(0-100)");

int rate;

while(true)

{

int.TryParse(Console.ReadLine(), out rate);

if (rate < 100 && rate > 0) break;

Console.WriteLine("Incorrect input, try again");

}

Console.WriteLine("[1] - <, [2] - >.");

int chose = Convert.ToInt32(Console.ReadLine());

while(true)

{

int.TryParse(Console.ReadLine(), out chose);

if (chose == 1 || chose == 2) break;

Console.WriteLine("Incorrect input, try again");

}

foreach (Film film in filmList)

{

if(film.GetRate() > rate && chose == 1) film.SeeInfo();

if(film.GetRate() > rate && chose == 2) film.SeeInfo();

}

}

public void SeeAll()

{

Console.WriteLine("========================");

int counter = 0;

for (int i = 0; i < filmList.Count; i++)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

Console.WriteLine(" ");

}

}

public void SortFilms()

{

int choise = 0;

Console.WriteLine("How you want to sort?");

Console.WriteLine("[1] - Year\n[2] - Rating.");

try

{

choise = Convert.ToInt32(Console.ReadLine());

}

catch

{

Console.WriteLine("Incorrect input");

return;

}

switch (choise)

{

case 1:

for (int i = 0; i < filmList.Count; i++)

for (int j = 0; j < filmList.Count; j++)

{

if (filmList[i].GetYear() < filmList[j].GetYear())

{

Film temp = filmList[i];

filmList[i] = filmList[j];

filmList[j] = temp;

}

}

break;

case 2:

for (int i = 0; i < filmList.Count; i++)

for (int j = 0; j < filmList.Count; j++)

{

if (filmList[i].GetRate() < filmList[j].GetRate())

{

Film temp = filmList[i];

filmList[i] = filmList[j];

filmList[j] = temp;

}

}

break;

}

}

}

Листинг 1.3 – реализация дополнительного класса:

class Film

{

Program h = new Program();

private string Name;

private int Year;

private int Rate;

private string Type;

private string Country;

public string GetName() => Name;

public int GetYear() => Year;

public int GetRate() => Rate;

public string GetType() => Type;

public string GetCountry() => Country;

public void SetName()

{

Console.Write("Enter film name: ");

Name = Console.ReadLine();

}

public void SetYear()

{

while (true)

{

Console.Write("Enter film year: ");

int.TryParse(Console.ReadLine(), out int year);

if (year <= 2023 && year >= 1927)

{

Year = year;

break;

}

Console.WriteLine("Incorrect input, try again");

}

}

public void SetType()

{

while(true)

{

Console.Write("Enter film type: ");

string type = Console.ReadLine();

if (h.IsOnlyLetters(type) == true)

{

Type = type;

break;

}

Console.WriteLine("Incorrect input, try again");

}

}

public void SetRate()

{

while(true)

{

Console.Write("Enter film rating(0-100): ");

bool check = int.TryParse(Console.ReadLine(), out int rate);

if (rate <= 100 && rate >= 0 && check == true)

{

Rate = rate;

break;

}

Console.WriteLine("Incorrect input, try again");

}

}

public void SetCountry()

{

while (true)

{

Console.Write("Enter film country: ");

string country = Console.ReadLine();

if (h.IsOnlyLetters(country) == true)

{

Country = country;

break;

}

Console.WriteLine("Incorrect input, try again");

}

}

public Film()

{

SetName();

SetYear();

SetType();

SetRate();

SetCountry();

}

public void Edit()

{

Console.WriteLine("What do you want to change?");

Console.WriteLine("[1] - Name\n[2] - Year\n[3] - Type\n[4] - Rating\n[5] - Country.");

int edit = Convert.ToInt32(Console.ReadLine());

switch (edit)

{

case 1:

SetName();

break;

case 2:

SetYear();

break;

case 3:

SetType();

break;

case 4:

SetRate();

break;

case 5:

SetCountry();

break;

}

}

public void SeeInfo()

{

Console.WriteLine("\tFilm: " + Name);

Console.WriteLine("\tYear: " + Year);

Console.WriteLine("\tType: " + Type);

Console.WriteLine("\tRating: " + Rate);

Console.WriteLine("\tCountry: " + Country);

}

}

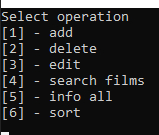


Рисунок 1. Начальное меню

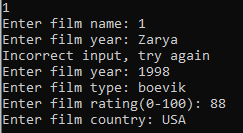


Рисунок 2. Добавление фильма



Рисунок 3. Просмотр всех фильмов

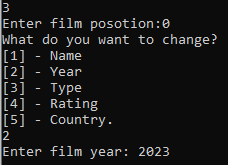


Рисунок 4. Редактирование фильма

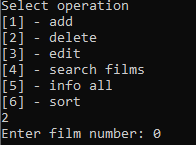


Рисунок 5. Удаление фильма

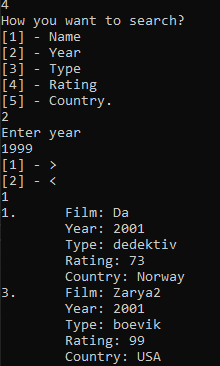


Рисунок 6. Поиск фильма

**Вывод:** В ходе данный лабораторной работе было изучено как создавать классы и объекты этих классов, вызывать методы и обращаться к данным через методы этих объектов, передавать объекты других классов через методы, понимать влияние модификаторов доступа на видимость данных и методов. А также применены перезагрузка методов на практике.