|  |  |
| --- | --- |
| 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»  «Конструкторы и деструкторы»  Вариант 6 | |
| Выполнил | Студент гр. 21-ИТ-1  Катушёнок И.В. |
| Проверил | Ассистент  Хирьянов И.Д. |
| Полоцк, 2023г. | |

**Цель работы:** Научиться описывать конструкторы и деструкторы для класса, понимать различия между конструкторами. Понимать, в каких случаях могут быть вызваны конструкторы копирования, понимать время жизни объекта.

**Ход работы**

**Задание:**

1. Дополнить и при необходимости модифицировать приложение,

разработанное согласно варианта лабораторной работы No1.

2. Определить в классах следующие конструкторы: без параметров, с

параметрами, копирования, перемещения.

3. Определить в классе деструктор.

4. В каждом конструкторе и деструкторе выдавать сообщение,

показывающее, какой именно конструктор или деструктор был вызван.

5. Дополнить основной класс методом с параметрами, внутри которого

должен создаваться объект дополнительного класса с помощью

конструктора с параметрами и добавляться в массив объектов

дополнительного класса.

6. Дополнить основной класс методом с двумя параметрами (ссылка на

объект дополнительного класса и количество), внутри которого должны

создаваться объекты дополнительного класса с помощью конструктора

копирования (создается необходимое количество копий указанного

объекта) и заноситься в массив объектов дополнительного класса.

7. Дополнить/модифицировать основное тело программы таким образом,

чтобы продемонстрировать использование всех конструкторов.

8. Сделать выводы.

**Вариант 6**

Варианты соответствуют лабораторной работе №1.

Листинг 1.1 – реализация меню программы:

class Program

{

static void Main()

{

Catalog catalog = new Catalog();

bool MenuCycle = true;

while (MenuCycle)

{

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\n[7] - make copies");

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

switch (choise)

{

case 1:

Console.WriteLine("Which constructor use to add a film?\n[1] - with parametrs\n[2] - without parametrs");

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

switch (choiseConstructor)

{

case 1:

catalog.AddToCatalogUsingConstructorWithParametrs();

break;

case 2:

catalog.AddToCatalogUsingConstructorWithoutParametrs();

break;

}

break;

case 2:

catalog.RemoveFromCatalog();

break;

case 3:

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

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

if (catalog.GetFilmList().Count == 0)

{

Console.WriteLine("Catalog is empty");

break;

}

if (position < 0 || position > catalog.GetFilmList().Count)

{

Console.WriteLine("Film with this number not found");

break;

}

Film editableFilm = catalog.GetFilmFromCatalog(position);

editableFilm.Edit();

catalog.SetFilmInCatalog(position, editableFilm);

break;

case 4:

catalog.SearchFilm();

break;

case 5:

catalog.SeeAll();

break;

case 6:

catalog.SortFilms();

break;

case 7:

int filmNumber;

while(true)

{

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

if (int.TryParse(Console.ReadLine(), out filmNumber));

if(filmNumber <= catalog.GetFilmList().Count - 1 && filmNumber >= 0) break;

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

}

Console.Write("Enter count of copies:");

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

catalog.CopyFilm(catalog.GetFilmList()[filmNumber], copiesCount);

break;

case 8:

MenuCycle = false;

break;

}

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

}

Console.ReadKey();

}

}

Листинг 1.2 – реализация основного класса:

class Catalog

{

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

public void CopyFilm(Film film, int CountOfCopies) //копирование фильмов

{

for(int i = 0; i < CountOfCopies; i++)

{

Film film1 = new Film(film);

filmList.Add(film1);

}

}

public List<Film> GetFilmList() { return filmList; }

public void SetFilmList(List<Film> List) { filmList = List; }

public Film GetFilmFromCatalog(int filmPosition) => filmList[filmPosition];

public void SetFilmInCatalog(int filmPosition, Film film) { filmList[filmPosition] = film; }

public void AddToCatalogUsingConstructorWithoutParametrs()

{

Film film = new Film();

filmList.Add(film);

}

public void AddToCatalogUsingConstructorWithParametrs()

{

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

string name;

while(true)

{

name = Console.ReadLine();

if (name != "") break;

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

}

int year;

while (true)

{

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

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

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

{

break;

}

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

}

string type;

while (true)

{

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

type = Console.ReadLine();

if (CorrectInputTests.IsOnlyLetters(type) && type != "")

{

break;

}

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

}

int rate;

while (true)

{

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

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

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

{

break;

}

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

}

string country;

while (true)

{

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

country = Console.ReadLine();

if (CorrectInputTests.IsOnlyLetters(country) && country != "")

{

break;

}

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

}

Film kino = new Film(name, year, type, rate, country);

filmList.Add(kino);

}

public void RemoveFromCatalog()

{

if (filmList.Count == 0)

{

Console.WriteLine("Catalog is empty");

return;

}

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

if (!(int.TryParse(Console.ReadLine(), out int position)))

{

Console.WriteLine("Incorrect film number");

return;

}

if (CorrectInputTests.InRange(0,filmList.Count - 1, position))

{

filmList.RemoveAt(position);

GC.Collect();

return;

}

else

{

Console.WriteLine("Incorrect film number");

}

}

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.TryParse(Console.ReadLine(), out int 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 (CorrectInputTests.InRange(0, 100, rate)) break;

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

}

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

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

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

{

private string \_name;

private int \_year;

private int \_rate;

private string \_type;

private string \_country;

public Film() //конструктор без параметров

{

Console.WriteLine("Constructor without parametrs is called");

SetName();

SetYear();

SetType();

SetRate();

SetCountry();

}

public Film(string name, int year, string type, int rate, string country) //конструктор с параметрами

{

Console.WriteLine("Constructor with parametrs was called");

\_name = name;

\_year = year;

\_type = type;

\_rate = rate;

\_country = country;

}

public Film(Film film) //конструктор копирования

{

\_name = film.\_name;

\_year = film.\_year;

\_type = film.\_type;

\_rate = film.\_rate;

\_country = film.\_country;

Console.WriteLine("Copy constructor was called");

}

~Film() //деструктор

{

Console.WriteLine("Destructor was called");

}

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 (CorrectInputTests.InRange(1927, 2023, year))

{

\_year = year;

break;

}

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

}

}

public void SetType()

{

while (true)

{

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

string type = Console.ReadLine();

if (CorrectInputTests.IsOnlyLetters(type) && type != "")

{

\_type = type;

break;

}

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

}

}

public void SetRate()

{

while (true)

{

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

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

if (CorrectInputTests.InRange(0,100,rate))

{

\_rate = rate;

break;

}

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

}

}

public void SetCountry()

{

while (true)

{

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

string country = Console.ReadLine();

if (CorrectInputTests.IsOnlyLetters(country) && country != "")

{

\_country = country;

break;

}

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

}

}

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);

}

Листинг 1.4 – реализация класса для проверки корректности ввода

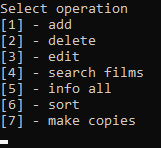
}

Рисунок 1. Начальное меню

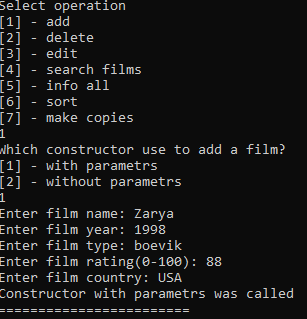


Рисунок 2. Добавление фильма с помощью конструктора с параметрами

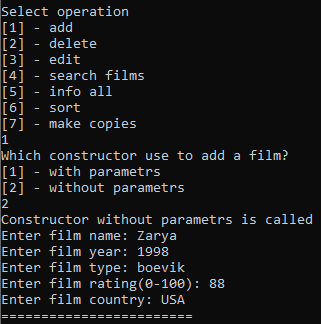


Рисунок 3. Добавление фильма с помощью конструктора без параметрами



Рисунок 4. Просмотр всех фильмов

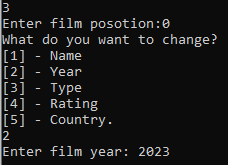


Рисунок 4. Редактирование фильма

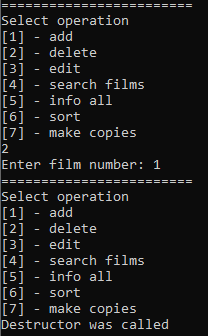


Рисунок 5. Удаление фильма

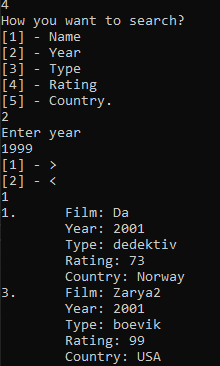


Рисунок 6. Поиск фильма

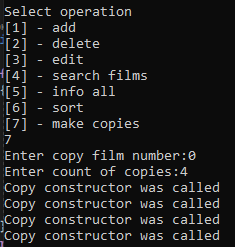


Рисунок 7. Вызов конструктора копирования

**Вывод:** В ходе данной лабораторной работы было изучено описание конструкторов и деструкторов для классов, изучено различие между конструкторами без параметров, параметрами, копирования.