Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

ЛАБОРАТОРНАЯ РАБОТА №9

Применение делегатов и событий

Студент:
Швалов Даниил Андреевич
Факультет ИКТ
Группа: *К32211*
Преподаватель:
Иванов Сергей Евгеньевич

Санкт-Петербург — 2023

Упражнение 1. Использование делегата при вызове метода

В проект был добавлен класс `Operation`, который содержит статический метод `PrintTitle` для отображения информации о книге.

```
using System;

namespace MyClass
{
    public class Operation
    {
        public static void PrintTitle(Book b)
        {
            b.Show();
        }
    }
}
```

В классе `Book` был объявлен делегат `ProcessBookDelegate`, свойство `ReturnSrok`, а также метод `ProcessPaperbackBooks`.

```
using System;

namespace MyClass
{
    public class Book : Item
    {
        private String author;
        private String title;
        private String publisher;
        private int pages;
        private int year;
        private static double price = 9;

        public Book() { }

        static Book()
        {
            price = 10;
        }

        public Book(
            String author,
            String title,
            String publisher,
            int pages,
            int year,
            long invNumber,
```

```
    bool taken
)
    : base(invNumber, taken)
{
    this.author = author;
    this.title = title;
    this.publisher = publisher;
    this.pages = pages;
    this.year = year;
}

public Book(String author, String title)
{
    this.author = author;
    this.title = title;
}

public void SetBook(
    String author,
    String title,
    String publisher,
    int pages,
    int year
)
{
    this.author = author;
    this.title = title;
    this.publisher = publisher;
    this.pages = pages;
    this.year = year;
}

public static void SetPrice(double price)
{
    Book.price = price;
}

public override void Show()
{
    Console.WriteLine("Книга:");
    Console.WriteLine("Автор: {0}", author);
    Console.WriteLine("Название: {0}", title);
    Console.WriteLine("Год издания: {0}", year);
    Console.WriteLine("Страницы: {0}", pages);
    Console.WriteLine("Стоимость аренды: {0}", Book.price);
    base.Show();
}

public double PriceBook(int s)
{
    double cust = s * price;
    return cust;
```

```
        }

        public bool ReturnSrok { get; set; }

        public override void Return()
        {
            taken = ReturnSrok;
        }

        public delegate void ProcessBookDelegate(Book book);

        public void ProcessPaperbackBooks(ProcessBookDelegate processBook)
        {
            if (ReturnSrok)
            {
                processBook(this);
            }
        }
    }
}
```

В класс `Program` был добавлен код для проверки программы:

```
namespace MyClass;

class Program
{
    static void Main(string[] args)
    {
        Book b1 = new Book(
            "Толстой Л.Н.",
            "Анна Каренина",
            "Знание",
            1204,
            2014,
            103,
            true
        );
        Book b2 = new Book(
            "Неш Т",
            "Программирование для  профессионалов",
            "Вильямс",
            1200,
            2014,
            108,
            true
        );

        b1.ReturnSrok = true;
        b2.ReturnSrok = false;
```

```
        Console.WriteLine("Книги возвращены в срок:");
        b1.ProcessPaperbackBooks(Operation.PrintTitle);
        b2.ProcessPaperbackBooks(Operation.PrintTitle);
    }
}
```

Остальные классы остались без изменений.

```
using System;

namespace MyClass
{
    public abstract class Item : IComparable
    {
        protected long invNumber;
        protected bool taken;

        public Item(long invNumber, bool taken)
        {
            this.invNumber = invNumber;
            this.taken = taken;
        }

        public Item()
        {
            this.taken = true;
        }

        public bool IsAvaliable()
        {
            return taken;
        }

        public long GetInvNumber()
        {
            return invNumber;
        }

        private void Take()
        {
            taken = false;
        }

        public abstract void Return();

        public void TakeItem()
        {
            if (IsAvaliable())
            {
```

```
                Take();
            }
        }

        public virtual void Show()
        {
            Console.WriteLine("Состояние единицы хранения:");
            Console.WriteLine("Инвентарный номер: {0}", invNumber);
            Console.WriteLine("Наличие: {0}", taken);
        }

        int IComparable.CompareTo(object obj)
        {
            Item it = (Item)obj;
            if (invNumber == it.invNumber)
            {
                return 0;
            }
            else if (invNumber > it.invNumber)
            {
                return 1;
            }
            else
            {
                return -1;
            }
        }
    }
}
```

```
using System;

namespace MyClass
{
    public interface IPubs
    {
        void Subs();
        bool IfSubs { get; set; }
    }
}
```

На рис. 1.1 представлен пример работы программы.

```
Книги возвращены в срок:
Книга:
Автор: Толстой Л.Н.
Название: Анна Каренина
Год издания: 2014
Страницы: 1204
Стоимость аренды: 10
Состояние единицы хранения:
Инвентарный номер: 103
Наличие: True
```

Рис. 1.1: Пример работы программы

Упражнение 2. Работа с событиями

В класс `Book` было добавлено событие `RetSrok`, изменен принцип работы свойства `ReturnSrok`, а также добавлен метод `ToString`:

```
using System;

namespace MyClass
{
    public class Book : Item
    {
        private String author;
        private String title;
        private String publisher;
        private int pages;
        private int year;
        private static double price = 9;
        private bool returnSrok = false;

        public Book() { }

        static Book()
        {
            price = 10;
        }

        public Book(
            String author,
            String title,
            String publisher,
            int pages,
            int year,
```

```
    long invNumber,
    bool taken
)
    : base(invNumber, taken)
{
    this.author = author;
    this.title = title;
    this.publisher = publisher;
    this.pages = pages;
    this.year = year;
}

public Book(String author, String title)
{
    this.author = author;
    this.title = title;
}

public void SetBook(
    String author,
    String title,
    String publisher,
    int pages,
    int year
)
{
    this.author = author;
    this.title = title;
    this.publisher = publisher;
    this.pages = pages;
    this.year = year;
}

public static void SetPrice(double price)
{
    Book.price = price;
}

public override void Show()
{
    Console.WriteLine("Книга:");
    Console.WriteLine("Автор: {0}", author);
    Console.WriteLine("Название: {0}", title);
    Console.WriteLine("Год издания: {0}", year);
    Console.WriteLine("Страницы: {0}", pages);
    Console.WriteLine("Стоимость аренды: {0}", Book.price);
    base.Show();
}

public double PriceBook(int s)
{
    double cust = s * price;
```

```
            return cust;
        }

        public bool ReturnSrok
        {
            get { return returnSrok; }
            set
            {
                returnSrok = value;
                if (ReturnSrok)
                {
                    RetSrok(this);
                }
            }
        }

        public override void Return()
        {
            taken = ReturnSrok;
        }

        public delegate void ProcessBookDelegate(Book book);

        public void ProcessPaperbackBooks(ProcessBookDelegate processBook)
        {
            if (ReturnSrok)
            {
                processBook(this);
            }
        }

        public static event ProcessBookDelegate RetSrok;

        public override string ToString()
        {
            return title + ", " + author + " Инв. номер " + invNumber;
        }
    }
}
```

В класс `Operation` был добавлен метод обработчик события `MetodObrabotchik`:

```
using System;

namespace MyClass
{
    public class Operation
    {
        public static void PrintTitle(Book b)
        {
```

```
            b.Show();
        }

        public static void MetodObrabotchik(Book b)
        {
            Console.WriteLine("Книга {0} сдана в срок.", b.ToString());
        }
    }
}
```

В класс `Program` был добавлен код, проверяющий новую функциональность:

```
namespace MyClass;

class Program
{
    static void Main(string[] args)
    {
        Book b1 = new Book(
            "Толстой Л.Н.",
            "Анна Каренина",
            "Знание",
            1204,
            2014,
            103,
            true
        );
        Book b2 = new Book(
            "Неш Т",
            "Программирование для  профессионалов",
            "Вильямс",
            1200,
            2014,
            108,
            true
        );

        Book.RetSrok += new Book.ProcessBookDelegate(
            Operation.MetodObrabotchik
        );

        Console.WriteLine("Книги возвращены в срок:");

        b1.ReturnSrok = true;
        b2.ReturnSrok = false;

        Console.WriteLine("Книги возвращены в срок:");
        b2.ReturnSrok = true;
    }
}
```

Все остальные классы остались без изменения:

```
using System;

namespace MyClass
{
    public abstract class Item : IComparable
    {
        protected long invNumber;
        protected bool taken;

        public Item(long invNumber, bool taken)
        {
            this.invNumber = invNumber;
            this.taken = taken;
        }

        public Item()
        {
            this.taken = true;
        }

        public bool IsAvaliable()
        {
            return taken;
        }

        public long GetInvNumber()
        {
            return invNumber;
        }

        private void Take()
        {
            taken = false;
        }

        public abstract void Return();

        public void TakeItem()
        {
            if (IsAvaliable())
            {
                Take();
            }
        }

        public virtual void Show()
        {
            Console.WriteLine("Состояние единицы хранения:");
            Console.WriteLine("Инвентарный номер: {0}", invNumber);
```

```
            Console.WriteLine("Наличие: {0}", taken);
        }

        int IComparable.CompareTo(object obj)
        {
            Item it = (Item)obj;
            if (invNumber == it.invNumber)
            {
                return 0;
            }
            else if (invNumber > it.invNumber)
            {
                return 1;
            }
            else
            {
                return -1;
            }
        }
    }
}
```

```
using System;

namespace MyClass
{
    public interface IPubs
    {
        void Subs();
        bool IfSubs { get; set; }
    }
}
```

На рис. 2.1 представлен пример работы программы.

```
Книги возвращены в срок:
Книга Анна Каренина, Толстой Л.Н. Инв. номер 103 сдана в срок.
Книги возвращены в срок:
Книга Программирование для  профессионалов, Неш Т Инв. номер 108 сдана в срок.
```

Рис. 2.1: Пример работы программы

Упражнение 3. Реализация события

В класс `IgralnayaKost` были добавлены делегат `ProcessIgralnayaKostDelegate` и событие `MaxPoints`:

```
using System;

namespace Igra
{
    public class IgralnayaKost
    {
        Random r;

        public IgralnayaKost()
        {
            r = new Random();
        }

        public int Random()
        {
            const int maxPoint = 6;
            int points = r.Next(maxPoint) + 1;

            if (points == maxPoint)
            {
                MaxPoints(this);
            }

            return points;
        }

        public delegate void ProcessIgralnayaKostDelegate(IgralnayaKost book);

        public static event ProcessIgralnayaKostDelegate MaxPoints;
    }
}
```

В класс `Gamer` был добавлен метод обработчик `MaxPointsHandler`. Также в конструктор была добавлена подписка на событие `MaxPoints`:

```
using System;

namespace Igra
{
    public class Gamer
    {
        private string name;
        private IgralnayaKost seans;

        public Gamer(string name)
        {
            this.name = name;
            seans = new IgralnayaKost();
            IgralnayaKost.MaxPoints +=
```

```
                new IgralnayaKost.ProcessIgralnayaKostDelegate(
                    MaxPointsHandler
                );
        }

        public int SeansGame()
        {
            return seans.Random();
        }

        public override string ToString()
        {
            return name;
        }

        private static void MaxPointsHandler(IgralnayaKost igralnayaKost)
        {
            Console.WriteLine("Выпало максимальное количество очков");
        }
    }
}
```

В класс `Program` был добавлен код, проверяющий работу программы:

```
namespace Igra;

class Program
{
    static void Main(string[] args)
    {
        Gamer g = new Gamer("Niko");

        while (true)
        {
            int points = g.SeansGame();
            if (points == 6)
            {
                break;
            }
            Console.WriteLine("Выпало {0} очков", points);
        }
    }
}
```

На рис. 3.1 представлен пример работы программы.

```
Выпало 1 очков
Выпало 2 очков
Выпало 5 очков
Выпало 5 очков
Выпало максимальное количество очков
```

Рис. 3.1: Пример работы программы

Упражнение 4. Иерархия классов учебного центра

Абстрактный класс `Person` хранит в себе фамилию и дату рождения, а также имеет два метода: виртуальный метод `Show` и обычный метод `GetAge`:

```
namespace LearningCenter
{
    public abstract class Person
    {
        public string Surname { get; set; }
        public DateTime BirthDate { get; set; }

        public Person(string surname, DateTime birthDate)
        {
            Surname = surname;
            BirthDate = birthDate;
        }

        public virtual void Show()
        {
            Console.WriteLine("Фамилия: {0}", Surname);
            Console.WriteLine("Дата рождения: {0:dd.MM.yyyy}", BirthDate);
        }

        public int GetAge()
        {
            DateTime today = DateTime.Today;
            int age = today.Year - BirthDate.Year;
            if (BirthDate.Date > today.AddYears(-age))
            {
                age -;
            }
            return age;
        }
    }
}
```

Интерфейс `IEmployee` содержит метод для получения зарплаты. Этот интерфейс будет применен для реализации классов администратора, преподавателя и менеджера, т. е. сотрудников учебного центра.

```
namespace LearningCenter
{
    public interface IEmployee
    {
        decimal GetSalary();
    }
}
```

Согласно описанию, были созданы классы Administrator, Student, Teacher и Manager:

```
namespace LearningCenter
{
    public class Administrator : Person, IEmployee
    {
        public Administrator(
            string surname,
            DateTime birthDate,
            string laboratory
        )
            : base(surname, birthDate)
        {
            Laboratory = laboratory;
        }

        public string Laboratory { get; set; }

        public override void Show()
        {
            Console.WriteLine("Администратор");
            base.Show();
            Console.WriteLine("Лаборатория: {0}", Laboratory);
            Console.WriteLine("Заработная плата: {0}", GetSalary());
        }

        public decimal GetSalary()
        {
            return 60000;
        }
    }
}
```

```
namespace LearningCenter
{
    public class Student : Person
    {
        public Student(
```

```
            string surname,
            DateTime birthDate,
            string faculty,
            int course
        )
            : base(surname, birthDate)
        {
            Faculty = faculty;
            Course = course;
        }

        public string Faculty { get; set; }
        public int Course { get; set; }

        public override void Show()
        {
            Console.WriteLine("Студент");
            base.Show();
            Console.WriteLine("Факультет: {0}", Faculty);
            Console.WriteLine("Курс: {0}", Course);
        }
    }
}
```

```
namespace LearningCenter
{
    public class Teacher : Person
    {
        public Teacher(
            string surname,
            DateTime birthDate,
            string faculty,
            string position,
            int workExperience
        )
            : base(surname, birthDate)
        {
            Faculty = faculty;
            Position = position;
            WorkExperience = workExperience;
        }

        public string Faculty { get; set; }
        public string Position { get; set; }
        public int WorkExperience { get; set; }

        public override void Show()
        {
            Console.WriteLine("Преподаватель");
            base.Show();
```

```
            Console.WriteLine("Факультет: {0}", Faculty);
            Console.WriteLine("Должность: {0}", Position);
            Console.WriteLine("Стаж: {0}", WorkExperience);
            Console.WriteLine("Заработная плата: {0}", GetSalary());
        }

        public decimal GetSalary()
        {
            return 80000;
        }
    }
}
```

```
namespace LearningCenter
{
    public class Manager : Person, IEmployee
    {
        public Manager(
            string surname,
            DateTime birthDate,
            string faculty,
            string position
        )
            : base(surname, birthDate)
        {
            Faculty = faculty;
            Position = position;
        }

        public string Faculty { get; set; }
        public string Position { get; set; }

        public override void Show()
        {
            Console.WriteLine("Менеджер");
            base.Show();
            Console.WriteLine("Факультет: {0}", Faculty);
            Console.WriteLine("Должность: {0}", Position);
            Console.WriteLine("Заработная плата: {0}", GetSalary());
        }

        public decimal GetSalary()
        {
            return 65000;
        }
    }
}
```

В класс `Program` был добавлен код, проверяющий работу программу, а также вы-

водящий список персон, возраст которых лежит в заданном пользователем диапазоне:

```
using System;
using System.Globalization;

namespace LearningCenter;

class Program
{
    static void Main(string[] args)
    {
        Administrator admin = new Administrator(
            "Петров",
            DateTime.ParseExact(
                "15.04.1985",
                "dd.MM.yyyy",
                CultureInfo.InvariantCulture
            ),
            "Лаборатория физики"
        );

        Student student = new Student(
            "Сидоров",
            DateTime.ParseExact(
                "21.06.2003",
                "dd.MM.yyyy",
                CultureInfo.InvariantCulture
            ),
            "ИКТ",
            2
        );

        Teacher teacher = new Teacher(
            "Попов",
            DateTime.ParseExact(
                "05.02.1975",
                "dd.MM.yyyy",
                CultureInfo.InvariantCulture
            ),
            "ИКТ",
            "Старший преподаватель",
            15
        );

        Manager manager = new Manager(
            "Коновалов",
            DateTime.ParseExact(
                "17.09.1983",
                "dd.MM.yyyy",
                CultureInfo.InvariantCulture
```

```
            ),
            "ИКТ",
            "Менеджер по закупкам"
        );

        Person[] persons = new Person[4];
        persons[0] = admin;
        persons[1] = student;
        persons[2] = teacher;
        persons[3] = manager;

        foreach (Person p in persons)
        {
            p.Show();
            Console.WriteLine("Возраст: {0}", p.GetAge());
        }

        Console.Write("Введите минимальный возраст: ");
        int minAge = int.Parse(Console.ReadLine());
        Console.Write("Введите максимальный возраст: ");
        int maxAge = int.Parse(Console.ReadLine());

        foreach (Person p in persons)
        {
            if (minAge ⩽ p.GetAge() && p.GetAge() ⩽ maxAge)
            {
                Console.WriteLine("{0}: {1}", p.Surname, p.GetAge());
            }
        }
    }
}
```

На рис. 4.1 представлен пример работы программы.

```
Администратор
Фамилия: Петров
Дата рождения: 15.04.1985
Лаборатория: Лаборатория физики
Заработная плата: 60000
Возраст: 37
Студент
Фамилия: Сидоров
Дата рождения: 21.06.2003
Факультет: ИКТ
Курс: 2
Возраст: 19
Преподаватель
Фамилия: Попов
Дата рождения: 05.02.1975
Факультет: ИКТ
Должность: Старший преподаватель
Стаж: 15
Заработная плата: 80000
Возраст: 48
Менеджер
Фамилия: Коновалов
Дата рождения: 17.09.1983
Факультет: ИКТ
Должность: Менеджер по закупкам
Заработная плата: 65000
Возраст: 39
Введите минимальный возраст: 20
Введите максимальный возраст: 40
Петров: 37
Коновалов: 39
```

Рис. 4.1: Пример работы программы