ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Москва, ул. 2-я Бауманская, д.5, стр.1
+7 (495) 182-83-85, do@bmstu.ru
do.bmstu.ru

# Программа курса «Язык программирования C#»

| Номер | Название темы | Количество часов | Описание темы |
|---|---|---|---|
| 1 | Знакомство с редактором Visual Studio Code и средой выполнения .NET | 6 | Рассмотрение текстового редактора Visual Studio Code. Данный текстовый редактор доступен для всех основных операционных систем: Windows, MacOS, Linux. Роль платформы .NET. Основные алгоритмические конструкции (условные операторы, циклы, вложенные циклы). |
| 2 | Особенности реализации объектно- ориентированного программирования в .NET на примере языка C# | 6 | Вывод определения «Класса». Знакомство с ООП подходом. Более глубокое знакомство с типами значений и ссылочными типами. Проектирование класса. Структура класса и модификаторы доступа. Отдельно и подробно рассматриваются свойства класса. |
| 3 | Отношение между классами: ассоциация, композиция и агрегация | 6 | Наследование классов и альтернативные наследованию отношения между классами: ассоциация, композиция и агрегация. Разбор примеров некорректного применения наследования. |
| 4 | Отношение между классами: реализация (интерфейсы) | 6 | Отношение между классами: реализация (интерфейсы). Разбор различия в использовании абстрактных классов и интерфейсов (состояние /действие). |
| 5 | Универсальные шаблоны в .NET | 6 | Знакомство с обобщенными типами (generics), создание обобщенных методов. Обеспечение типобезопасности. |
| 6 | Делегаты и события. Анонимные методы. Лямбда-выражения. LINQ | 6 | Делегаты, события и лямбды. Как делегаты хранят ссылки на методы: через события / анонимные методы / лямбда-выражения / LINQ. Преимущества использования делегатов. |
| 7 | Введение в многопоточность. Класс Thread. Асинхронное программирование | 6 | Основной функционал для использования потоков в приложении (System Threading). Класс Thread. Разработка приложения, использующего несколько потоков, которые будут выполнять различные задачи одновременно. |
| 8 | SOLID: пять принципов объектно-ориентированного программирования | 6 | Пять основных принципов проектирования в объектно-ориентированном программировании. |
| 9 | Разработка N-tier приложений | 6 | Понятие архитектуры программного обеспечения (software architecture) как совокупность важнейших решений об организации программной системы. Проектирование трехслойного приложения. |
| 10 | ORM. Построение слоя доступа к данным (DAL) | 6 | Разделение бизнес-логики и работы с данными на уровне отдельного приложения. Проектирование слоя доступа к данным (Data access layer). Реализация паттерна Repository. Знакомство с технологией ORM (entity framework и dapper). |
| 11 | Инверсия управления. IoC | 6 | Абстрактный принцип Inversion of Control (инверсия управления) для написания слабо связанного кода. Dependency Injection (внедрение зависимостей). Реализация IoC контейнера Ninject и внедрение зависимостей. |
| 12 | Введение в .NET MAUI | 4 | Паттерн MVVM (Model-View-ViewModel). Взаимодействие XAML и C#. |

Москва, ул. 2-я Бауманская, д.5, стр.1
+7 (495) 182-83-85, do@bmstu.ru
do.bmstu.ru

| | | | |
|---|---|---|---|
| 13 | Итоговая аттестация | 2 | Зачет. |

**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ
Первый проректор –
проректор по учебной работе
МГТУ им. Н.Э. Баумана
_____Б.В. Падалкин
«17» августа 2023 г.

**Дополнительное профессиональное образование**

# ДОПОЛНИТЕЛЬНАЯ ПРОФЕССИОНАЛЬНАЯ ПРОГРАММА
# ПРОГРАММА ПОВЫШЕНИЯ КВАЛИФИКАЦИИ
# **«Язык программирования C#»**

**Регистрац. №** 05.2223.01.50

Москва, 2023

**АВТОРЫ ПРОГРАММЫ:**

Доцент каф. СМ13 _____ А.Д. Новиков

**СОГЛАСОВАНО:**

Начальник УСП _____ Т.А. Гузева

Директор
Центра дополнительного образования _____ М.В. Стоянова

## Оглавление

# 1. ОБЩАЯ ХАРАКТЕРИСТИКА ДПП

Программа подготовлена на основе:

- Федерального закона от 29 декабря 2012 года № 273-ФЗ «Об образовании в Российской Федерации»;

- требований Приказа Минобрнауки России от 01.07.2013 года № 499 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным профессиональным программам»;

- методических рекомендаций-разъяснений Минобрнауки России по разработке дополнительных профессиональных программ на основе профессиональных стандартов от 22 апреля 2015 года № ВК-1030/06.

Реализация программы ДПП направлена на получение новой компетенции, необходимой для профессиональной деятельности.

**1.1. Цель ДПП**

Сформировать у обучающихся знания, навыки и умения в области разработки, отладки, проверки работоспособности, модификации компьютерного программного обеспечения.

**1.2. Планируемые результаты обучения**

Планируемые результаты обучения по ДПП:

- освоение профессиональных компетенций в процессе изучения перечисленных тем в учебном плане;

- успешное освоение программы повышения квалификации;

- успешное прохождение итоговой аттестации (зачет).

Обучающимся, успешно прошедшим обучение, выполнившим текущие контрольные задания и выдержавшим предусмотренное учебным планом зачет, выдается удостоверение о повышении квалификации по ДПП «Язык программирования C#».

**1.3. Дополнительные характеристики ДПП**

Характеристики новой квалификации определены в приказе Минтруда России от 20.07.2022 №424н «Об утверждении профессионального стандарта «Программист».

Вид профессиональной деятельности:

- Разработка компьютерного программного обеспечения (Код 06.001).

Трудовые функции:

- Проектирование компьютерного программного обеспечения (D/03.6).

**1.4. Перечень профессиональных компетенций в рамках имеющейся квалификации, качественное изменение которых осуществляется в результате обучения**

Получаемые компетенции базируются на основании Приказа Минобрнауки России от 19 сентября 2017 г. № 929 «Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.03.01 Информатика и вычислительная техника (уровень бакалавриата)».

Перечень компетенций:

ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения.

**1.5. Соответствие видов деятельности профессиональным компетенциям и их составляющих**

| Профессиональные компетенции | Практический опыт | Умения | Знания |
|---|---|---|---|
| Проектирование компьютерного программного обеспечения (D/03.6) | | | |
| ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения | Проектирование структур данных; Проектирование баз данных; Проектирование программных интерфейсов | Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов | Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения |

# 2. УЧЕБНЫЙ ПЛАН ДПП

## 2.1. Категория слушателей ДПП

Имеющаяся квалификация (требования к слушателям) – к освоению ДПП допускаются лица, имеющие среднее профессиональное и/или высшее образование.

## 2.2. Общая трудоёмкость программы, аудиторная и самостоятельная работа

Общая трудоемкость программы 72 академических часа, из них 64 академических часа аудиторной работы, 6 академических часов самостоятельной работы и 2 академических часов итоговой аттестации.

## 2.3. Форма обучения

Форма обучения по ДПП – очная с применением дистанционных образовательных технологий.

## 2.4. Учебный план

ДПП «Язык программирования C#» реализуется одним модулем.

| № п/п | Наименование темы, модуля | Форма контроля | Всего, час | В том числе | | | |
|---|---|---|---|---|---|---|---|
| | | | | Лекции | Практ. занятия | Самост. работа | Итоговая аттестация |
| 1. | Знакомство с редактором Visual Studio Code и средой выполнения .NET | - | 6 | 3 | 3 | - | - |
| 2. | Особенности реализации объектно-ориентированного программирования в .NET на примере языка C# | - | 6 | 3 | 3 | - | - |
| 3. | Отношение между классами: ассоциация, композиция и агрегация | - | 6 | 3 | 3 | - | - |
| 4. | Отношение между классами: реализация (интерфейсы) | Практ. задание | 6 | 2 | 2 | 2 | - |
| 5. | Универсальные шаблоны в .NET | - | 6 | 3 | 3 | - | - |
| 6. | Делегаты и события. Анонимные методы. Лямбда-выражения. LINQ | Практ. задание | 6 | 2 | 2 | 2 | - |
| 7. | Введение в многопоточность. Класс Thread. Асинхронное программирование | - | 6 | 3 | 3 | - | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8. | SOLID: пять принципов объектно-ориентированного программирования | - | 6 | 3 | 3 | - | - |
| 9. | Разработка N – tier приложений | - | 6 | 3 | 3 | - | - |
| 10. | ORM. Построение слоя доступа к данным (DAL) | Практ. задание | 6 | 2 | 2 | 2 | - |
| 11. | Инверсия управления. IoC | - | 6 | 3 | 3 | - | - |
| 12. | Введение в .NET MAUI | - | 4 | 4 | - | - | - |
| 13. | Итоговая аттестация | Зачет | 2 | - | - | - | 2 |
| | ИТОГО | - | 72 | 34 | 30 | 6 | 2 |

## 3. КАЛЕНДАРНЫЙ УЧЕБНЫЙ ГРАФИК

| № п/п | Наименование темы, модуля | 1 день | 2 день | 3 день | 4 день | 5 день | 6 день | 7 день | 8 день | 9 день |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Знакомство с редактором Visual Studio Code и средой выполнения .NET | | | | | | | | | |
| 2 | Особенности реализации объектно-ориентированного программирования в .NET на примере языка C# | | | | | | | | | |
| 3 | Отношение между классами: ассоциация, композиция и агрегация | | | | | | | | | |
| 4 | Отношение между классами: реализация (интерфейсы) | | | | | | | | | |
| 5 | Универсальные шаблоны в .NET | | | | | | | | | |
| 6 | Делегаты и события. Анонимные методы. Лямбда-выражения. LINQ | | | | | | | | | |
| 7 | Введение в многопоточность. Класс Thread. Асинхронное программирование | | | | | | | | | |
| 8 | SOLID: пять принципов объектно-ориентированного программирования | | | | | | | | | |
| 9 | Разработка N – tier приложений | | | | | | | | | |
| 10 | ORM. Построение слоя доступа к данным (DAL) | | | | | | | | | |
| 11 | Инверсия управления. IoC | | | | | | | | | |
| 12 | Введение в .NET MAUI | | | | | | | | | |
| 13 | Итоговая аттестация | | | | | | | | | |

Минимальный срок освоения ДПП – 9 дней.

# 4. РАБОЧАЯ ПРОГРАММА ДПП

## 4.1. Рабочая программа модуля «Язык программирования C#»

4.1.1. Цель изучения модуля: сформировать у обучающихся знания, навыки и умения в области разработки, отладки, проверки работоспособности, модификации компьютерного программного обеспечения.

4.1.2. Задачи изучения модуля:

1. Получить знание о базовых классах платформы .NET.

2. Освоить основные алгоритмических конструкциях языка C#.

3. Сформировать навыки разработки и отладки приложений в редакторе кода Visual Studio Code.

4. Владеть принципами объектно-ориентированного программирования: абстракцию, инкапсуляцию, наследование, полиморфизм.

5. Уметь проектировать классы и выстраивать «общение» между объектами.

6. Разоботать принципы организации работы с базой данный средствами ORM.

7. Знакомство с визуальным программирвоанием на примере MAUI-приложений.

8. Изучить архитектурные MV-паттерны, паттерны GoF и принципы SOLID.

4.1.3. Планируемые результаты обучения

Процесс изучения раздела направлен на формирование следующих компетенций

| Код компетенции | Перечень планируемых результатов обучения по модулю | Формы и методы обучения, способствующие формированию и развитию компетенции |
|---|---|---|
| ОПК-8 | **Знать:**<br>Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения.<br>**Уметь:**<br>Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов.<br>**Владеть:**<br>Проектирование структур данных;<br>Проектирование баз данных;<br>Проектирование программных интерфейсов. | Формы обучения:<br>Фронтальная.<br>Методы обучения:<br>Лекция;<br>Практическая работа;<br>Самостоятельная работа. |

4.1.4 Содержание курса

**Тема 1. Знакомство с редактором Visual Studio Code и средой выполнения .NET (6 часов)**

Лекции (3 часа). Рассмотрение текстового редактора Visual Studio Code. Данный текстовый редактор доступен для всех основных операционных систем: Windows, MacOS, Linux. Роль платформы .NET.

Практические занятия (3 часа). Основные алгоритмические конструкции (условные операторы, циклы, вложенные циклы).

**Тема 2. Особенности реализации объектно-ориентированного программирования в .NET на примере языка C# (6 часов)**

Лекции (3 часа). Вывод определения «Класса». Знакомство с ООП подходом, показывая, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов. Более глубокое знакомство с типами значений и ссылочными типами. Структура класса и модификаторы доступа. Отдельно и подробно рассматриваются свойства класса.

Практические занятия (3 часа). Проектирование класса. Подробно рассматриваются определения свойств: свойства только для чтения и записи; вычисляемые свойства; автоматические свойства. Через проблемные ситуации объясняются важность модификаторов доступа.

**Тема 3. Отношение между классами: ассоциация, композиция и агрегация (6 часов)**

Лекции (3 часа). Рассмотрено наследование классов и альтернативные наследованию отношения между классами: ассоциация, композиция и агрегация.

Практические занятия (3 часа). Разбор примеров, в которых наследование смотрится неуместно. Примеры больших иерархий, которые в долгосрочной перспективе не дадут наращивать функциональность проекта. Решение озвученных проблемы через альтернативы наследованию.

**Тема 4. Отношение между классами: реализация (интерфейсы) (6 часов)**

Лекции (2 часа). Отношение между классами: реализация (интерфейсы).

Практические занятия (2 часа). Разбор различия в использовании абстрактных классов и интерфейсов (состояние / действие).

Самостоятельная работа (2 часа). Выполнение практического задания.

**Тема 5. Универсальные шаблоны в .NET (6 часов)**

Лекции (3 часа). Знакомство с обобщенными типами (generics), а также создание обобщенных методов. Показывается, что ранее уже применяли их (List<>). Чтобы разобраться в особенности данного явления, рассматривается проблема, которая могла возникнуть до появления обобщенных типов. Через проблемный пример раскрывается важность generics.

Практические занятия (3 часа). Продемонстрированы примеры кода, где generics помогают обеспечить типобезопасность. Решение задачи без применения generics и показаны проблемы в случае неправильного использования спроектированного класса. Повторное написание программы с применением generics класса.

**Тема 6. Делегаты и события. Анонимные методы. Лямбда-выражения. LINQ (6 часов)**

Лекции (2 часа). Делегаты, события и лямбды. Как делегаты хранят ссылки на методы: через события / анонимные методы / лямбда-выражения / LINQ показана истинная сила делегатов.

Практические занятия (2 часа). Постановка проблемы, которая показывает, что на момент написания программы мы можем не знать, какой именно код будет выполняться в определенном месте. Показано, как делегаты позволяют делегировать определение действия из класса во внешний код.

Продемонстрирована связь делегатов с анонимными методами. Показана элегантность лямбда-выражений, которые представляют упрощенную запись анонимных методов. Введены понятия метода расширения и выведена демонстрация примеров языка запросов к источнику данных (LINQ).

Самостоятельная работа (2 часа). Выполнение практического задания.

**Тема 7. Универсальные шаблоны в .NET (6 часов)**

Лекции (3 часа). Изучение основного функционала для использования потоков в приложении (System.Threading). Подробно рассматривается класс, представляющий отдельный поток – класс Thread.

Практические занятия (3 часа). Разработка приложения, использующего несколько потоков, которые будут выполнять различные задачи одновременно.

**Тема 8. Универсальные шаблоны в .NET (6 часов)**

Лекции (3 часа). Рассмотрены пять основных принципов проектирования в объектно-ориентированном программировании.

Практические занятия (3 часа). Разбор двух принципов, которые понадобятся для последующих тем:

Open closed Principle – принцип открытости-закрытости. Классы должны быть открыты для расширения, но закрыты для изменения;

Dependency Inversion Principle – принцип инверсии зависимостей. Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те, и другие должны зависеть от абстракции. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

**Тема 9. Универсальные шаблоны в .NET (6 часов)**

Лекции (3 часа). Рассмотрено понятие архитектуры программного обеспечения (software architecture) как совокупность важнейших решений об организации программной системы.

Практические занятия (3 часа). Проектирование трехслойного приложения. Создание проекта со следующими слоями:

1) View (exe проект), который отображает список сотрудников. View представлен двумя вариантами: WinFormApp и ComsoleApp;

2) BL (Class Library), класс в котором есть функции: «заполни список»; «верни текущее состояние списка сотрудников»; «отредактируй список»;

3) Model (Class Library), хранящая доменные объекты (сотрудник).

**Тема 10. Библиотека PyQt для создания desktop-приложений (6 часов)**

Лекции (2 часа). Разбор проблемы разделения бизнес-логики и работы с данными на уровне отдельного приложения, которую решает широко известный архитектурный шаблон Data Access Layer (DAL). Для того, чтобы этот шаблон можно было масштабировать до уровня всего предприятия, необходимо дополнить его рядом архитектурных принципов, которые рассматриваются в данной лекции.

Практические занятия (2 часа). Проектирование слоя доступа к данным (Data access layer). Реализуем паттерн Repository. Знакомство с технологией ORM (entity framework и dapper).

Самостоятельная работа (2 часа). Выполнение практического задания.

**Тема 11. Инверсия управления. IoC (6 часов)**

Лекции (3 часа). Рассматривается абстрактный принцип Inversion of Control (инверсия управления) для написания слабо связанного кода, а также одна из реализаций этого принципа – Dependency Injection (внедрение зависимостей).

Практические занятия (3 часа). Реализация IoC контейнера Ninject и внедрение зависимостей (Dependency Injection), предназначенных для того, чтобы «вживлять» разные реализации классов Repository в проект.

**Тема 12. Введение в .NET MAUI (4 часа)**

Лекции (4 часа). Разбирается паттерн MVVM (Model-View-ViewModel).

- Показано, как он позволяет отделить логику приложения от визуальной части.

- Рассмотрен декларативный язык разметки XAML, на котором пишут .NET MAUI приложения.

- На примере .NET MAUI рассмотрено взаимодействие XAML и C#.

4.1.5. Оценочное средство для текущего контроля (формулировка практических заданий):

*Тема 4. Формулировка практического задания:*

1. Рассчитать и вывести на экран площадь и периметр прямоугольника стороны которого запросить у пользователя (ввод с клавиатуры).

*Задача на ввод/вывод информации, арифметические операции.*

2. Найдите сумму «1+2+3+…+n», где n вводится пользователем с клавиатуры.

*Задача на цикл «For…».*

3. Даны числа от 35 до 87. Вывести на консоль те из них, которые при делении на 7 дают остаток 1, 2 или 5.

*Задача на цикл «for (int 1 = начальное значение; i<конечное значение; i++)», а также на условный оператор, так как нам нужно выводить не все числа из указанного диапазона, а только те, которые отвечают определенному условию (if (a … || …) {Console.WriteLine(i);}, а именно делятся на 7 и в остатке дают определенной число. Проверить остаток числа от деления можно командой «%» (пример: int a = 50 % 7; // a = 1).*

4. Представьте, что на складе имеется определённое количество ящиков с яблоками. Когда подъезжает машина для погрузки, попросить пользователя ввести с клавиатуры

число – количество ящиков, которые готова забрать машина. Цикл должен работать до тех пор, пока все ящики не отгрузят со склада. Предусмотреть тот случай, когда пользователь введёт количество ящиков больше, чем есть на складе (сообщить о том, что столько нет и отгрузить сколько есть).

*Задача на цикл «while...». Организовать цикл пока ящики есть (пока переменная отвечающая за ящики больше 0), в цикле вычитать из этой переменной то количество, которое запросил пользователь с клавиатуры.*

*Тема 6. Формулировка практического задания:*

Кейс: «Руководство Почты доложилось правительству об успешной цифровой трансформации!

Воодушевившись их успехом, Московский зоопарк, принимает решение провести цифровизацию своих бизнес-процессов и обращается к вам с просьбой написать ERP-систему.»

**Со слов заказчика записано:**

«Мы собираемся сделать жизнь наших животных счастливой! Счастливое животное – сытое и здоровое! Нам важно, что бы они получали еду вовремя, а также, согласно внутреннему регламенту, проходили медосмотр при приеме в наш зоопарк» В ходе работы продакт-менеджера были выявлены детали, которые помогли архитектору проекта спроектировать архитектуру доменной модели классов, которую он и озвучил на кик-офф встрече по проекту: «В связи с тем, что ERP-система в будущем будет развиваться, и в конечном счете производить учет не только животных, но и людей, а также вещей подлежащих инвентаризации, я набросал следующую доменную модель для нашего проекта. interface IAlive – для определения принадлежности наших типов к категории «живых».

Интерфейс будет наделять реализуемые его типы свойством int Health { get; set; } interface IInventory – для определения принадлежности наших типов к категории «инвентаризационная вещь». Интерфейс будет наделять реализуемые его типы

свойством int Number { get; set; }

В ходе беседы мы выяснили, что инвентаризации подлежат не только предметы зоопарка, но и животные (но не сотрудники). У хищных животных нужно хранить информацию о том, что они едят, а у травоядных информацию об уровне их доброты.

Основываясь на этом, примите решение о иерархии и реализации следующих классов и интерфейсов:

- class Animal
- class Herbo

- class Predator
- class Monkey
- class Rabbit
- class Tiger
- class Wolf
- class Manager
- class Thing
- class Table
- class Computer
- interface IAlive
- interface IInventory

Также в разговоре с управляющим зоопарка мы определили ряд бизнес-процессов, которые подлежат цифровизации в первой версии ERP-системы: 1. Присваивание инвентаризационного номера животному осуществляется менеджером, но после того, как животное, желающее попасть в зоопарк, пройдет ветеринарный контроль в Ветеринарной клинике. 2. Ветеринарная клиника должна следить за прибывающими в зоопарк животными и незамедлительно производить осмотр вновь прибывших. В ходе осмотра нужно принимать одно из двух решений: принять животное в зоопарк или отказаться от данного животного. 3. Менеджер должен получать информацию о принятом на баланс в зоопарк животном, присваивать ему инвентаризационный номер и быть готовым накормить животное, когда последнее об этом «сообщит». 4. Животное сообщает о необходимости его покормить, если уровень его здоровья падает ниже 0. 5. В случае если животное перестает числиться в зоопарке, менеджер должен перестать его подкармливать, даже если животное будет приходить и просить.

Исходя их представленных бизнес процессов, были наспех набросаны классы:

- class Hospital
- class Zoopark

Класс Zoopark должен хранить информацию о животных, которые поступают в зоопарк - ObservableCollection<Animal> Animals, а также о животных уже принятых в зоопарк

- ObservableCollection<Animal> WantingAnimals. Коллекция WantingAnimals отслеживается

Ветеринарной клиникой с целью проверить здоровье вновь прибывших животных и решить, что с ними делать: ставить или не ставить на баланс (добавлять или нет в Animals),

а за коллекцией Animals следит менеджер, который назначает номера добавленным в зоопарк животным, а также начинает слушать их на предмет сытости.

Класс Hospital имеет агрегацию с классом Zoopark. Связь класса Manager с классом Zoopark не определена на этапе проектирования и должна быть реализована на ваше усмотрение.

Объекты класса Animal должны сообщать о том, что они голодные через событие, аргументами которого должен быть объект класса class StarveEventArgs: EventArgs.

Аргумент события представлен свойством: public int CurrentHealth {get;}. Менеджер должен дать столько единиц корма, сколько не хватает до 100% здоровья, то есть кормление - увеличение здоровья до 100%.

Уже после того как вы спроектировали класс Zoopark, было принято решение о расширении его возможностей – добавлении метода Print (bool showPredators, bool showHerbos), который должен выводить на консоль информацию о животных (либо только хищников, либо только травоядных, либо и тех, и других) в зависимости от переданных логических флагов.

Так как класс Zoopark уже используется в различных частях проекта, было принято не добавлять данный метод, а реализовать его через механизм методов расширения (extension methods), в котором фильтрацию животных осуществить средствами LINQ to Objects.

*Тема 10. Формулировка практического задания:*

Кейс: «Руководство почты решило повысить производительность своих сотрудников и обратилось к вам с целью цифровой трансформации!»

**Со слов заказчика записано:**

*«Мы собираемся раз и навсегда ликвидировать очереди в наших почтовых отделениях! Нам необходима информационная система, которая будет выдавать начальнику почтового отделения информацию о всех его сотрудниках (кассиры, операторы и почтальоны).*

*Начальника почты будет интересовать следующая информация: ФИО; должностные обязанности (которые зависят от должности); статус загрузки (занят делом или свободен).*

*Но так как нам важно разгрузить очередь, то начальнику почты не следует считаться с текущим статусом загрузки сотрудника, то есть все сотрудники должны быть «не заняты» и это не обсуждается, данное поле мы вводим только согласно требованиям закона, просим это понять и не задавать вопросов...».*

**В ходе работы продакт-менеджера были выявлены детали, которые помогли архитектору проекта спроектировать архитектуру доменной модели классов, которую он и озвучил на кик-офф встрече по проекту:** *«Доменная модель нашего проекта будет включать базовый класс Employee, который включает следующие свойства и методы:*

- *Имя сотрудника: авто-свойство string Name {get; set;}*
- *Статус сотрудника: проинициализированное в классе свойство, доступное только для чтения bool IsBusy {get;} = false;*
- *Должностные обязанности: виртуальный/абстрактный (подумайте) метод, который в наследниках будет переопределен ... void OfficialDuties () ...;*

*У класса Employee будет три класса-наследника: Cashier; Operator; Postman.*

*Каждый из наследников в обязательном порядке обязан переопределить поведение базового виртуального метода OfficialDuties (обязанности выводить в виде текста).*

*При этом код нашей создаваемой информационной системы не рассчитан на работу с объектами Employee. Исключить возможность создания таковых. Также мы не предполагаем расширения функционала, а, следовательно, не должно быть никаких подвидов, озвученных выше должностей. Исключить возможность дальнейшего расширения иерархии классов.*

*Доменная модель нашей программы также будет иметь класс, описывающий почтовое отделение с его сотрудниками - PostOffice, который имеет ассоциацию в виде композиции*

*с коллекцией объектов Employee: List<Employee> Employees, а также метод void Poll(),*

*который произведет опрос всех сотрудников данного почтового отделения, то есть запустит цикл по коллекции Employees и для каждого объекта в данной коллекции выведет имя и статус (свойства базового класса Employee), а также должностные обязанности каждого сотрудника (переопределенный метод в наследниках)»*

**Со слов безумно-богатого заказчика, которому нельзя отказать, вновь записано:** *«Когда почтальоны узнали о введении данной системы они создали профсоюз! И профсоюз добился того, что при выводе информации о статусе загруженности почтальона, система должна выдавать не фиктивную информацию, а соответствующую действительности.»* **Архитектор проекта выдал следующее техническое задание на доработку.**

Вам предстоит исправить доменную модель классов таким образом, чтобы существующий метод Poll класса PostOffice исключительно для почтальонов стал выводить

реальный статус их загрузки (свойство IsBusy), а не фиктивный («свободен») как для всех остальных сотрудников.

Для этого вам потребуется внести следующие изменения в доменную модель: 1. Добавьте (не переопределите, а именно перекройте) в классе Postman авто-свойство IsBusy {get; set;} которое позволит хранить реальный статус загрузки конкретно- взятого почтальона. 2. Внесите изменения в метод Poll класса PostOffice, которые позволят, идентифицировав почтальона в коллекции List<Employee> Employees, вызвать именно его свойство авто-свойство IsBusy {get; set;}, а не то свойство доступное только для чтения из базового класса - IsBusy {get;}.

Архитектор оставил это требование без комментариев, но сказал, что так нужно для других модулей системы, в которых уже используется именно свойство IsBusy у сотрудников, а, следовательно, создание другого свойства, например, IsRealBusy просто недопустимо.

Продемонстрируйте работу вашей доменной модели на коллекции из трех сотрудников.

# 5. УСЛОВИЯ РЕАЛИЗАЦИИ ДПП

## 5.1. Организационные условия реализации ДПП

| Наименование аудитории | Вид занятия | Наименование оборудования, программного обеспечения |
|---|---|---|
| Аудитория для проведения лекций/практических занятий | Лекции | ПК с доступом в Интернет и возможностью просмотра файлов в формате *.ppt, *.pptx, *.pdf, проектор/телевизор/монитор, Visual Studio Code и среда выполнения .NET. |
| Аудитория для проведения лекций/ практических занятий | Практические занятия | ПК с доступом в Интернет и возможностью просмотра файлов в формате *.ppt, *.pptx, *.pdf, проектор/телевизор/монитор, Visual Studio Code и среда выполнения .NET. |
| Коворкинги, учебные залы и т.д. | Самостоятельная работа | ПК с доступом в Интернет и возможностью просмотра файлов в формате *.ppt, *.pptx, *.pdf, проектор/телевизор/монитор, Visual Studio Code и среда выполнения .NET. |
| Аудитория для проведения лекций/ практических занятий | Итоговая аттестация | ПК с доступом в Интернет и возможностью просмотра файлов в формате *.ppt, *.pptx, *.pdf, проектор/телевизор/монитор, Visual Studio Code и среда выполнения .NET. |

## 5.2. Педагогические условия реализации ДПП

Реализация программы обеспечивается преподавательским составом, удовлетворяющим следующим условиям:

- наличие высшего профессионального образования, соответствующее профилю программы, из числа штатных преподавателей, или привлеченных на условиях почасовой оплаты труда;

- значительный опыт практической деятельности в соответствующей сфере из числа штатных преподавателей или привлеченных на условиях почасовой оплаты труда

## 5.3. Учебно-методическое обеспечение ДПП

Основная литература:


1. Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.
2. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. / Рихтер Дж. – Санкт-Петербург: Питер, 2018. – 896 с.
3. Шаблоны проектирования информационных систем : учеб.-метод. пособие / сост.: С. А. Виденин. – Красноярск : Сиб. федер. ун-т, 2018. – 148 с. https://cloud.mail.ru/public/iABH/QMfmBQk8A.


Дополнительная литература:


1. Язык программирования C# 7 и платформы .NET и .NET Core; Автор: Эндрю Троелсен, Филипп Джепикс ; Год: 2019. — 1333 с.

Интернет-источники:

1. https://metanit.com/sharp/tutorial/.
2. https://professorweb.ru/my/csharp/charp_theory/level1/index.php.
3. https://refactoring.guru/.

**5.4. Методические рекомендации**

ДПП построена по тематическому принципу, каждый раздел представляет собой логически завершенный материал.

Преподавание программы основано на личностно-ориентированной технологии образования, сочетающей два равноправных аспекта этого процесса: обучение и учение. Личностно-ориентированный подход развивается при участии слушателей в активной работе на практических занятиях. Личностно-ориентированный подход направлен, в первую очередь, на развитие индивидуальных способностей обучающихся, создание условий для развития творческой активности слушателя и разработке инновационных идей, а также на развитие самостоятельности мышления при решении учебных задач разными способами, нахождение рационального варианта решения, сравнения и оценки нескольких вариантов их решения и т.п. Это способствует формированию приемов умственной деятельности по восприятию новой информации, ее запоминанию и осознанию, созданию образов для сложных понятий и процессов, приобретению навыков поиска решений в условиях неопределенности.

Практические занятия проводятся для приобретения навыков решения практических задач в предметной области модуля. Задания, выполняемые на практических занятиях, выполняются с использованием активных и интерактивных методов обучения.

Самостоятельная работа слушателей предназначена для проработки дополнительной литературы. Результаты практических заданий слушателей учитываются на итоговой аттестации.

При изучении курса предусмотрены следующие методы организации и осуществления учебно-познавательной деятельности:

- объяснительно-иллюстративный метод;
- репродуктивный метод;
- частично-поисковый метод.

# 6. ФОРМЫ ИТОГОВОЙ АТТЕСТАЦИИ ДПП

Итоговая аттестация проводится в форме зачета для проверки сформированности компетенций, полученных в рамках ДПП.

Зачет проводится в формате выполнения задания. Результатом зачета служат правильные ответы на вопросы билета.

По результатам итоговой аттестации обучающемуся выставляется оценка «ЗАЧТЕНО/НЕ ЗАЧТЕНО»:

Оценка «ЗАЧТЕНО» выставляется обучающемуся, который:

- разработана информационная система «Управление задачами»;
- продемонстрировал необходимые систематизированные знания и достаточную степень владения принципами предметной области программы, понимание их особенностей и взаимосвязь между ними в течение всего срока обучения по ДПП.

Оценка «НЕ ЗАЧТЕНО» ставятся обучающемуся, который:

- не разработана информационная система «Управление задачами»;
- имеет крайне слабые теоретические и практические знания, обнаруживает неспособность к построению самостоятельных заключений.

# 7. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ИТОГОВОЙ АТТЕСТАЦИИ

## 7.1. Паспорт комплекта оценочных средств

| Предметы оценивания | Объекты оценивания | Показатели оценки |
|---|---|---|
| ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения | Ответы на вопросы | Разработана информационная система «Управление задачами» |

## 7.2. Комплект оценочных средств

7.2.1. Темы для подготовки к зачету:

1. Концепция типа данных. Значимые и ссылочные типы.
2. Понятие исключительной ситуации. Стандартные исключения .NET.
3. Основные понятия ООП.
4. ТемОписание класса. Модификаторы доступа private и public.a для подготовки к итоговому контролю 4.
5. Методы. Перегрузка методов. Способы передачи параметров.
6. Обобщения (generics). Ограничения на параметры типов.
7. Альтернатива наследованию: композиция и агрегация.
8. Лямбды в c#, создание лямбда-выражений и их использование.
9. Многопоточность. Параллельное программирование.
10. Пять принципов объектно-ориентированного программирования.

7.2.2. Формулировка задачи для проведения зачёта:

**Требования к приложению**

Необходимо разработать систему управления задачами (аналог «Todoist» – список дел и таск-менеджер). Система должна позволять создавать, удалять, редактировать список задач. Решение должно представлять собой Rich-клиент приложение (толстый клиент).

**Общее описание процесса**

Стандартная схема работы с системой выглядит следующим образом:

1. Задачи заносится в систему.
2. Задачи могут быть отредактированы.
3. Задачи могут быть удалены.
4. Предоставлять возможность просматривать информацию по задачам.

Каждая задача, отслеживаемая системой, характеризуется следующим набором атрибутов:

- Наименование задачи
- Описание задачи
- Список исполнителей (простое текстовое поле, ссылочность не нужна)
- Дата регистрации задачи в системе
- Статус задачи: Назначена, Выполняется, Приостановлена, Завершена (число статусов для системы фиксировано и сами статусы неизменны).
- Плановая трудоёмкость задачи
- Фактическое время выполнения
- Дата завершения

**Функциональные требования**

| Идентификатор | Описание требования |
|---|---|
| FR001 | Система должна позволять создавать задачи. |
| FR002 | Просмотр и редактирование всех полей задачи. |
| FR003 | Удаление задач. |
| FR004 | Просмотр списка задач. |
| FR005 | Необходимо предусмотреть следующую модификацию со статусами задачи: статус «Завершена» может быть присвоен только после статусов «Выполняется» и «Приостановлена». |

**Требования к интерфейсу пользователя**

| Идентификатор | Описание требования |
|---|---|
| UI001 | Задачи, должны быть представлены в виде плоского списка. |
| UI002 | Вывод аннотации задачи реализуется в соответствии с его представлением |

Больше ограничений на интерфейс пользователя нет. Внешний вид приложений, состав форм и граф перехода реализуется кандидатом в соответствии с его представлением о задаче.

**Среда выполнения, инструменты, технологии**

**База данных**

Все данные приложения должны быть размещены в БД MS SQL Server. Для доступа к БД используется ORM (Entity Framework / Dapper)

**Клиентская часть**

- Клиентская часть .Net приложение с интерфейсом построенном на WPF / MAUI.
- В качестве языка реализации следует выбрать C#.
- Платформа: .Net 7.0

**Требования к реализации**

**Основные**

1. Необходимо использовать ОО подходы при построении системы.
2. Необходимо придерживаться единой схемы обработки исключений.
3. Использовать ресурсы для хранения UI-строк
4. Именование типов и переменных должно быть выдержано в единой схеме, рекомендованной Microsoft для .Net приложений.
5. UI должен быть построен с учётом интересов пользователя. Пользоваться им должно быть удобно, а лучше и интуитивно понятно.

**Дополнительные**

1. Построение трёхслойной системы будет дополнительным плюсом (MVVM)