Филиал федерального государственного бюджетного образовательного учреждения высшего образования «Мурманский арктический государственный университет»
в г. Кировске Мурманской области
(филиал МАГУ в г. Кировске)

Всероссийский конкурс «Методическая разработка педагога»

Методическая разработка практического занятия по МДК 02.02 Прикладное программирование для специальности 09.02.03 на тему «Программирование паттернов. Singleton».

Специальность 09.02.03 Программирование в компьютерных системах

Автор:
преподаватель дисциплин профессионального цикла, Ломова Любовь Андреевна

2022

Оглавление

Ключевые слова

Программирование, паттерны, ООП, C#.

Аннотация

Методическая разработка практического занятия по теме: «Программирование паттернов. Singleton» составлена в соответствии с требованиями Федерального государственного образовательного стандарта СПО.

В методической разработке занятия даны обоснования развивающего обучения, способствующие формированию у студента знаний и умений по дисциплине.

На занятии используются приемы, средства и методы обучения, активизирующие мыслительную деятельность, воспитывающие у студентов устойчивый познавательный интерес, а также умение осмысливать и применять имеющиеся знания в различной практической деятельности.

Методическая разработка занятия включает себя:

- методическую разработку для преподавателя;
- методическую разработку практического занятия для студентов.

Методическая цель практического занятия – мотивация обучающихся к самостоятельному, и творческому практическому применению умений и знаний при решении задач в предполагаемых условиях.

Основная часть

1. МЕТОДИЧЕСКАЯ РАЗРАБОТКА ПРАКТИЧЕСКОГО ЗАНЯТИЯ ДЛЯ ПРЕПОДАВАТЕЛЯ

Курс 3, специальность 09.02.03 «Программирование в компьютерных системах»

Тема: «Программирование паттернов. Singleton»

Цели занятия:

1. Дидактические:

– формирование профессиональных компетенций в соответствии с требованиями ФГОС:

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

2. Развивающие:

– развивать способность осуществлять поиск информации, использовать информацию, необходимую для эффективного выполнения профессиональных задач, профессионального и личностного развития (ОК.4);

– использовать информационно-коммуникационные технологии в своей деятельности (ОК.5);

– развивать способность организовывать свою деятельность, выбирать методы и способы решения поставленных задач (ОК.2);

– развивать способность работать в команде (ОК.6);

– развивать способность принимать решение в стандартных и нестандартных ситуациях (ОК.3).

3. Воспитательные:

– воспитывать устойчивый интерес к своей будущей профессии (ОК.1);

– воспитывать чувство ответственности за результаты своей работы, работы членов команды (ОК.7).

В соответствии с требованиями ФГОС студенты

должны знать:

– основные принципы процесса разработки программного обеспечения;

– основные методы и средства эффективной разработки.

должны уметь:

– владеть основными методологиями процессов разработки программного обеспечения;

использовать приобретенные знания и умения в практической деятельности для:

– разработки алгоритма поставленной задачи и реализации его средствами автоматизированного проектирования;

– разработки кода программного продукта на основе готовой спецификации на уровне модуля;

– использования инструментальных средств на этапе отладки программного продукта.

Тип занятия: комбинированное.

Вид занятия: практическое занятие.

Методы обучения: инструктаж и самостоятельная работа, проблемный, репродуктивный.

Метод контроля знаний: демонстрация результатов работы; тестирование.

Продолжительность занятия: 180 минут.

Межпредметные связи:

– Основы программирования;

– ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем МДК 01.01 Системное программирование.

Распределение времени практического занятия представлено в таблице 1.

Таблица 1 – Хронокарта практического занятия

№	Элементы занятия	Время, мин.
1	Организационный момент	2
2	Мотивация	3
3	Контроль исходного уровня знаний	10
4	Методические инструкции	10
5	Индивидуальная работа студентов	100
6	Представление результатов работы	35
7	Тестирование	15
8	Подведение итогов занятия	3
9	Сообщение домашнего задания	2
Итого:		180

Характеристика отдельных элементов занятия:

1. Приветствие. Контроль внешнего вида студентов, отсутствующих студентов, готовности аудитории к занятию.

2. Постановка целей и задач. Создание мотивационного пространства.

Преподаватель четко называет тему занятия, цель занятия, этапы занятия.

Совместно со студентами формируется значение и место данной темы в будущей профессии. При создании мотивационного пространства используются межпредметные связи, показывается значение данной темы при изучении профессиональных модулей.

3. Контроль исходного уровня знаний.

Используется лекционный материал по предыдущей теме «Паттерны».

Вопросы:

1) Что такое паттерн?

2) Какие группы паттернов вам известны?

3) Что является преимуществом использования паттернов проектирования?

4) Определение паттерна Одиночка.

5) Что такое порождающие паттерны, какого рода задачи они позволяют решить? (**проблематика занятия**)

4. Инструктаж к выполнению практической работы.

Преподаватель совместно со студентами разбирают предстоящую практическую работу в соответствии с методическими указаниями. Преподаватель обращает внимание на наиболее сложные моменты, на требования к оформлению программного кода, наличию комментариев и демонстрации результатов работы. Работа выполняется индивидуально.

5. Выполнение практической работы

Студенты выполняют практическую работу в соответствии с методическими указаниями и рекомендациями, данными преподавателем. Преподаватель в процессе выполнения работы консультирует студентов, направляет их при возникновении затруднений.

6. Представление результатов работы.

Студенты представляют результаты работы, демонстрируя работу созданной программы, поясняют программный код (в частности реализацию паттерна, взаимодействие с другими классами, создание экземпляров и обработку исключений) и отвечают на вопросы.

7. Тестирование.

Студенты выполняют тест по теме, подтверждая уровень знаний.

Критерий оценки:

100 % – «5» (отлично);

80 % – «4» (хорошо);

60 % – «3» (удовлетворительно);

менее 60 % – «2» (не удовлетворительно).

8. Подведение итогов занятия.

Преподаватель обобщает результаты работы, достижение целей занятия, комментирует работу на занятии отдельных студентов и всей группы в целом. Итоговые оценки выставляются интегративно с учётом вводного контроля, проделанной самостоятельной работы и тестирования.

9. Сообщение домашнего задания.

Преподаватель сообщает тему следующего занятия: «Программирование паттернов. Стратегия».

10. Сообщение домашнего задания.

Преподаватель сообщает тему следующего практического занятия: «Программирование паттерна. Facade».

2. МЕТОДИЧЕСКАЯ РАЗРАБОТКА ПРАКТИЧЕСКОГО ЗАНЯТИЯ ДЛЯ СТУДЕНТОВ

Тема: Программирование паттернов. Singleton.

Цель: освоить принципы работы с паттернами, закрепить навыки использования порождающего паттерна проектирования «Одиночка».

Ход работы:

1. Ознакомиться с кратким теоретическим материалом.
2. Выполнить индивидуальное задание:

– разработать программу алгоритма решения задачи, используя паттерн «Одиночка»;

– использовать механизм отлавливания и обработки ошибок в написанной программе;

– *реализовать графический интерфейс*;

– выполнить отладку и запуски программы с тестовыми наборами исходных данных.

3. Сформулировать выводы по проделанной работе.

Методические указания по выполнению работы

1) Краткие теоретические сведения

Паттерн (*design pattern*) в разработке программного обеспечения – повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях.

Паттерн Singleton – гарантирует, что у класса может быть только один экземпляр. В частном случае предоставляется возможность наличия, заранее определенного числа экземпляров.

В реальной жизни аналогией объекта в единственном экземпляре может служить Солнце. Как бы мы не смотрели на него, мы всегда будем видеть одно и то же Солнце – звезду, вокруг которой вращается планета Земля, так как не может быть другого экземпляра Солнца, по крайней мере, для нашей планеты.

Архитектура паттерна Singleton

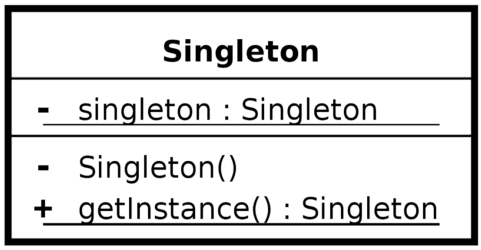


Рисунок 1 – Структура паттерна Singleton на языке UML

Участники: Singleton (Одиночка) предоставляет интерфейс (метод Instance) для получения доступа к единственному экземпляру.

Паттерн Singleton рекомендуется использовать, когда:

– В системе должен быть только один экземпляр некоторого класса, или в частном случае заранее определенное пользователем количество экземпляров (два, три и т.д.).

– Требуется организовать расширение класса единственного экземпляра через использование механизма наследования.

Реализация паттерна проектирования Singleton на языке C#, листинг 1:

```
Ссылок: 4
class Singleton // паттерн
{
    private static Singleton instance; //закрытое статическое поле
    ссылка: 1
    private Singleton() { } // закрытий конструктор класса
    Ссылок: 0
    public static Singleton getInstance() // статичный метод для получения уникального экземпляра класса
    {
        if (instance == null) //
            instance = new Singleton();
        return instance;
    }
}
```

Листинг 1 – Паттерн Singleton

– Класс Singleton окончательный – его нельзя расширить.

– Его конструктор закрытый – никакой метод не может создать экземпляр этого класса.

– Единственный экземпляр s класса Singleton – статический, он создается внутри класса.

Возможность расширения через наследование: если класс Singleton не является статическим или герметизированным / запечатанным (sealed), то от него возможно наследование, что позволит расширить существующую функциональность.

Возможность наличия переменного числа экземпляров: Singleton позволяет создавать фиксированное число экземпляров класса Singleton, например, только два или три и т.д.

2) Пример

Задача: на каждом компьютере нужно одномоментно запустить только одну операционную систему. Реализовать решение с помощью паттерна «Одиночка».

Могут использоваться два класса – операционная система (OS, это и есть паттерн «Одиночка») и компьютер (обычный пользовательский класс, имеющий множество экземпляров), листинг1.

```
namespace Patterns.Singleton
{
    Ссылок: 6
    class OS // паттерн
    {
        private static OS instance; //закрытое статическое поле
        Ссылок: 2
        public  string Name { get; private set; } // свойство для обращения к закрытому полю name
        ссылка: 1
        private OS(string name)  // закрытий конструктор класса
        {
            this.Name = name;
        }
        ссылка: 1
        public static OS getInstance(string name) // статичный метод для получения уникального экземпляра класса
        {
            if (instance == null)
                instance = new OS(name);
            return instance;
        }
    }
    Ссылок: 2
    class Computer
    {
        Ссылок: 2
        public OS OS { get; set; }
        ссылка: 1
        public void Launch(string osName)
        {
            OS = OS.getInstance(osName); // вызов метода для создания одного экземпляра класса OS
            Console.WriteLine("Вызван метод Launch.");
        }

    }
```

Листинг 1 – Описание классов «Computer», «OS»

Создание экземпляров классов и организация взаимодействия представлена в листинге 2.

```
Ссылок: 0
class Program
{
    Ссылок: 0
    static void Main(string[] args)
    {
        Computer comp = new Computer();
        comp.Launch("Windows 10");
        Console.WriteLine(comp.OS.Name);
        Console.ReadKey();
    }
}
```

Листинг 2 – Вызов объектов классов

Критерии оценки практической работы:

«5» (отлично): выполнены все задания практической работы, студент ориентируется в исходном коде, четко и без ошибок ответил на все контрольные вопросы.

«4» (хорошо): выполнены все задания практической работы; студент ответил на все контрольные вопросы с замечаниями; исходный код содержит незначительные ошибки.

«3» (удовлетворительно): выполнены все задания практической работы с замечаниями; студент не уверено ориентируется в программном коде; исходный код содержит незначительные ошибки; ответил на контрольные вопросы с замечаниями.

«2» (не зачтено): не выполнены или выполнены неправильно задания практической работы; студент не ориентируется в структуре программы; не владеет теоретическим материалом в должном объеме.

Критерии оценки программы и тестов:

– правильность работы программы и тестов (в соответствии со спецификациями);

– качество кода (определяется перечнем типичных ошибок, комментируемостью, структурируемостью, правильным именованием переменных, методов, классов).

3. ДИДАКТИЧЕСКИЙ МАТЕРИАЛ ПО ТЕМЕ

1. Повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста – это …

а) паттерн

б) класс

в) абстрактный класс

г) форма (окно), страница

2. Что является преимуществом использования паттернов проектирования?

а) они предоставляют проверенные техники решения задач

б) они упрощают разработку и поддержку пользовательских интерфейсов

в) они предоставляют механизмы для тестирования модулей системы

г) они уменьшают количество проектной документации

д) они снижают затраты на разработку, т.к. уже реализованы и их можно использовать без изменений

3. Паттерн проектирования ... обеспечивает существование одного экземпляра некоторого класса и предоставляет единую точку доступа к нему.

а) Singleton

б) Adapter

в) Bridge

г) Proxy

д) Decorator

4. Поведенческий паттерн проектирования, который позволяет передавать запросы последовательно по цепочке обработчиков. Каждый последующий обработчик решает, может ли он обработать запрос сам и стоит ли передавать запрос дальше по цепи. Какой это паттерн?

а) цепочка обязанностей (chain of responsibility)

б) компоновщик (composite)

в) наблюдатель (observer)

5. Паттерн Strategy имеет следующие преимущества:

а) позволяет переключаться между алгоритмами во время выполнения программы

б) отделяет алгоритм от класса, в котором он используется

в) гарантирует использование единственной стратегии во время выполнения программы

г) предоставляет разделение между моделью и представлением

6. Назначением какого паттерна проектирования является предоставление удобного интерфейса к громоздкому и сложному API?

а) Facade

б) Factory Method

в) Iterator

г) Decorator

д) Strategy

7. Какие из нижеперечисленных паттернов НЕ являются порождающими паттернами проектирования? *

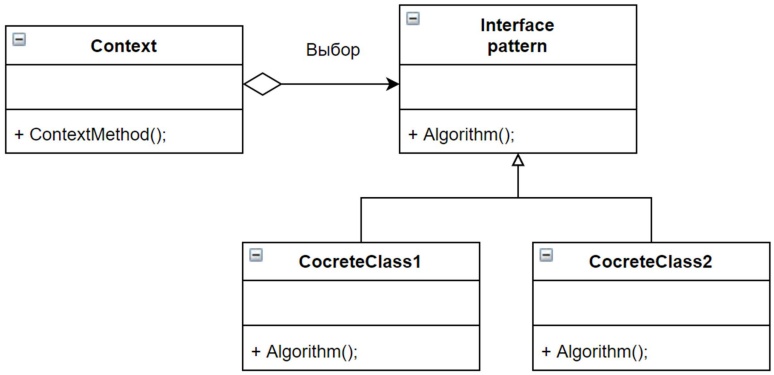
а) Facade (Фасад)

б) Visitor (Посетитель)

в) Singleton (Одиночка)

г) Abstract Factory (Абстрактная фабрика)

8. Какой паттерн проектирования реализован в данной схеме UML?



а) Стратегия

б) Одиночка

в) Фасад

г) Итератор

9. Абстрактное описание элементов дизайна задачи проектирования и способа ее решения с помощью обобщенного набора классов – это …

а) элемент шаблона проектирования «решение»

б) элемент шаблона проектирования «задача»

в) элемент шаблона проектирования «имя»

г) сам шаблон (паттерн)

10. Именованный перечень сигнатур открытых методов, который может быть унаследован классом или интерфейсом – это …

а) интерфейс

б) абстрактный класс

в) паттерн

Ключи:

Вопрос	Ответ	Вопрос	Ответ
1	а	6	а
2	а	7*	а, б
3	а	8	а
4	а	9	а
5*	а, б	10	а

Критерии оценки теста:

100 % – «5» (отлично);

80 % – «4» (хорошо);

60 % – «3» (удовлетворительно);

менее 60 % – «2» (не удовлетворительно).

СПИСОК ЛИТЕРАТУРЫ

1) Основы паттернов проектирования. Введение в паттерны проектирования [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/patterns/1.1.php

2) Тепляков С. Паттерны проектирования на платформе .NET. – СПб.: Питер, 2015. – 320 с.: ил.