Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Филиал «Минский радиотехнический колледж»

Учебная дисциплина

«Инструментальное программное обеспечение»

**Инструкция**

по выполнению лабораторной работы

«Разработка и проектирование классов.

Разработка, отладка и испытание программ создания классов

и экземпляров классов»

Минск 2020 г.

# **Лабораторная работа №12 (2 часа)**

**Тема работы: «Разработка и проектирование классов.**

**Разработка, отладка и испытание программ создания классов**

**и экземпляров классов»**

**1 Цель работы**

Научиться проектировать и создавать классы для решения задач.

**2 Задание**

Спроектировать и реализовать 3 класса, которые будет содержать не менее 3 полей, одного конструктора и трех методов. Предметную область для создания классов определить самим.

**3 Оснащение работы**

Задание по варианту, ЭВМ, среда разработки **Python 3.7, IDLE**.

**4 Основные теоретические сведения**

**Объектно-ориентированное программирование**

Поле объекта класса

Сейчас самое время вспомнить, что класс сам по себе является объектом. Точно так же, как реальным объектом является лист бумаги, на котором выполнен чертеж автомобиля. До этого мы, образно выражаясь, имели дело с постройкой "автомобилей". Сейчас более пристально посмотрим на тот "чертеж", с которого все началось.  
 А мы начнем с важного момента: программный код класса выполняется. До этого мы в теле класса размещали только объявления методов, и никаких других интересных команд там не было, хотя теоретически они там возможны (другой вопрос, нужны ли они там). Самое простое, что приходит на ум -разместить в теле класса команду присваивания значения переменной. В этом случае получим поле, но не экземпляра класса, а непосредственно самого класса (или поле объекта касса).  
 О поле объекта класса можно думать как о переменной, которая "известна" только в теле класса. За пределами класса доступ к такой переменной осуществляется через явное указание класса, в котором она объявлена. Используется точечный синтаксис вида класс . переменная.  
 Если вернуться к аналогии с чертежом на листе бумаги, то поле объекта  
класса - это такая липучка с напоминанием или какой-то информацией, нацепленная на лист бумаги. Все, кто пользуются чертежом, видят и эту липучку. Но на процесс создания автомобилей по чертежу она не влияет. Это просто такой "довесочек" к важному документу, какая-то дополнительная информация.  
 С точки зрения прикладного программирования для нас на Данный момент  
важны два обстоятельства:  
• В теле класса можно присвоить значение переменной (поле класса).  
• Обращаться к такой переменной следует так: указывается имя класса, и, через точку, имя переменной

Нередко в справочной литературе можно встретить утверждение, что поля класса играют роль статических переменных (как в языках С++, Java, С#). Для тех, кто не знаком с этим термином: статическая переменная или статическое поле – это поле, общее для всех экземпляров класса. В некотором смысле можно конечно, относиться к полям класса как к статическим переменным. Но на самом деле это не совсем так - то есть, полной аналогии нет. С одной стороны, мы при определенных обстоятельствах (которые обсуждаются далее) и только для считывания значения можем обращаться к полю класса через экземпляр класса. В этом случае вместо имени класса указывается имя экземпляра класса (но и здесь не так все  
просто). Что касается присваивания значения полю класса, то через экземпляр класса сделать это крайне проблематично. Чтобы прояснить ситуацию мы далее рассмотрим несколько примеров.

В листинге представлен программный код, который в некоторой степени позволят понять разницу между полем объекта класса и полем экземпляра класса.

Class MyClass:

# Поле класса  
name= " Kлacc MyC la s s "  
# Мет од для присваивания значения  
# полю э кземпляра класса  
def set ( self, n ):  
 self.nickname=n  
# Метод для отображения значения  
# поля э кземпляра класса  
def get ( self ):  
 рrint ( " Значение пoля : " , self.nickname )  
# Конструктор  
def \_\_init\_\_ ( self , n ) :  
# Полю экземпляра класса  
# присваивается значение  
 self.set ( n )  
# Отображается сообщение  
 print ( " Coздaн экземпляр класса . " )  
# Отображается значение поля экземпляра  
 self.get ( )  
# Создается первый экземпляр класса  
green=MyClass ( " Зeлeный " )  
# Обращение к полю класса через экземпляр класса  
рrint ( " Принадлежность : " , grееn.nаmе )  
# Создается второй экземпляр класса  
red=MyClass ( " Красный " )  
# Обращение к полю класса через экземпляр класса  
рrint ( " Принадлежност ь : " , rеd.nаmе )  
# Полю класса присваивается значение

MyClass.name= " Здecь могла быть Ваша реклама ! "  
# Обращение · к полю класса через экземпляр класса  
рrint ( " Спрашивает Kpacный : " , red.name )  
# Обращение к полю класса через экземпляр класса  
рrint ( " Спрашивает Зеленый : " , grееn.nаmе )

Результат работы программы:

Создан экземпляр класса .  
Значение поля : Зеленый  
Принадлежность : Класс MyClass  
Создан экземпляр класса .  
Значение поля : Красный  
Принадлежность : Класс MyClass  
Спрашивает Красный : Здесь могла быт ь Ваша реклама !  
Спрашивает Зеленый : Здесь могла быт ь Ваша реклама !

Проанализируем особенности программного кода. В теле класса MyClass, кроме объявления двух методов и конструктора, есть команда nare= " Kлacc MyClass " , которой фактически создается поле name класса MyClass, и этому полю присваивается текстовое значение " Класс MyClass ". Метод set( ) предназначен для присваивания значения полю экземпляра класса n i c kname. С помощью метода get ( ) значение поля nickname экземпляра класса отображается в окне вывода. Также в классе описан конструктор. При выполнении кода конструктора полю nickname экземпляра класса присваивается значение аргумента, переданного конструктору, отображается сообщение о создании экземпляра класса и значение поля nickname. На этом код клacca MyClass в принципе исчерпан. Далее в программе идут команды, в которых создаются экземпляры класса и выполняются различные манипуляции с полем name класса MyClass.  
Первый экземпляр (переменная green) класса MyClass создается командой green=MyClass ( " Зеленый " ) . Для этого экземпляра поле nickname, очевидно, получает значение " Зеленый " . При вызове конструктора в окне вывода отображаются сообщения с текстом " Со здан экземпляр класса . " и " Значение поля : Зеленый " . Сразу после создания экземпляра green следует команда  
print ( " Принадлежность : " , green.name ). В ней обращение (инструкция green.name) к полю name объекта класса MyClass выполняется через экземпляр класса. В резульате nоявляется сообщение с текстом " Принадлежность : Класс MyClass " . Таким образом, значением инструкции green.name является текст " Класс MyClass " - то, есть это значение поля name класса MyClass . Если бы мы вместо инструкции green.name использовали MyClass.name, получили бы точно такой же результат. В данной конкретной ситуации инструкции green.name и MyClass.name эквиваленты с точки зрения результата (но это не означает, что так будет всегда)

Второй экземпляр (переменная red) класса MyClas s создается командой  
red=MyClass ( " Красный" ) . Поле nicknare экземпляра red получает значение " Красный " .·При вызове конструктора в окне вывода отображается текст " Со здан экземпляр класса . " и " Значение поля : Красный ". При Обращении к полю name класса MyClass через экземпляр класса в команде print ( " Принадлежность : " , red.name ) результатом инструкции red.name будет текстовое значение поля name класса MyClass , о чем свидетельствует сообщение Принадлежность : Класс MyClass в окне вывода.  
 Как и в предыдущем случае, обращение к полю класса через экземпляр класса дает такой же результат, как это было бы при использовании инструкции  
MyClass.name. Еще один вывод состоит в том, что какой бы экземпляр класса мы ни использовали, если мы обращаемся через экземпляр к полю· класса, получаем один и тот же результат - значение поля класса.

Чтобы проверить наши подозрения, командам MyClass.name= "  
Здесь могла быть Ваша реклама ! " присваиваем новое значение полю name класса MyClass. Затем выполняются команды print ( " Спрашивает Красный : " , red.name ) и print ( " Спрашивает Зеленый : " , green.nare ) , в которых обращение к полю класса выполняется через экземпляры класса. В результате выполнения этих команд в окне вывода появляются соответственно такие сообщения: Спрашивает Красный :  
Здесь могла быт ь Ваша реклама ! и Спрашивает Зеленый :  
Здесь могла быт ь Ваша реклама ! . Вывод простой: обе инструкции green.name и red.name на самом деле возвращают значение поля name класса МуСlаss .

**5 Порядок выполнения работы**

1. Выделить ключевые моменты задачи.

2. Построить алгоритм решения задачи.

3. Запрограммировать полученный алгоритм.

4. Провести тестирование полученной программы.

**6 Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7 Контрольные вопросы и задания**

1. Перечислите составные части класса?
2. Как создать объект класса?

**8 Рекомендуемая литература**

**Прохоренок, Н.А.** Python 3. Самое необходимое / Н.А Прохоренок, В.А. Дронов – СПб.: БВХ-Петербург, 2016.

**Федоров, Д.Ю.** Основы программирования на примере языка Python / Д.Ю. Федоров. – М.: Юрайт, 2018.