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, отладка и испытание программ с применением

наследования классов»

Минск 2020 г.

# **Лабораторная работа №14**

**Тема работы: «Разработка, отладка и испытание программ с применением наследования классов»**

**1 Цель работы**

Научить писать программы с применением наследования и множественного наследования.

**2 Задание**

К классу из лабораторной работы №13 добавить 3 класс которые будут является потомками уже имеющегося класса.

**3 Оснащение работы**

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

**4 Основные теоретические сведения**

**Наследование**

Наследование в объектно-ориентированном программировании очень похоже на наследование в реальной жизни, где ребенок наследует те или иные характеристики его родителей в дополнение к его собственным характеристикам.

В объектно-ориентированном программировании, наследование означает отношение IS-A. Например, болид – это транспорт. Наследование это одна из самых удивительных концепций объектно-ориентированного программирования, так как оно подразумевает **повторное использование**.

Основная идея наследования в объектно-ориентированном программировании заключается в том, что **класс может наследовать характеристики другого класса**. Класс, который наследует другой класс, называется **дочерним классом** или производным классом, и класс, который дает наследие, называется **родительским**, или основным.

Рассмотрим на очень простой пример наследования. Выполним следующий скрипт:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # Создание класса Vehicle  class Vehicle:      def vehicle\_method(self):          print("Это родительский метод из класса Vehicle")    # Создание класса Car, который наследует Vehicle  class Car(Vehicle):      def car\_method(self):          print("Это метод из дочернего класса") |

В скрипте выше мы создаем два класса: Vehicle и Car, который наследует класс Vehicle. Чтобы наследовать класс, вам нужно только вписать название родительского класса внутри скобок, которая следует за названием дочернего класса. Класс Vehicle содержит метод vehicle\_method(), а дочерний класс содержит метод car\_method(). Однако, так как класс Car наследует класс Vehicle, он также наследует и метод vehicle\_method().

Рассмотрим это на практике и выполним следующий скрипт:

|  |  |
| --- | --- |
| 1  2 | car\_a = Car()  car\_a.vehicle\_method() # Вызываем метод родительского класса |

В этом скрипте мы создали объект класса Car вызывали метод vehicle\_method() при помощи объекта класса Car. Вы можете обратить внимание на то, что класс Car не содержит ни одного метода vehicle\_method(), но так как он унаследовал класс Vehicle, который содержит vehicle\_method(), класс Car также будет использовать его. Выдача выглядит следующим образом:

|  |  |
| --- | --- |
| 1 | Это родительский метод из класса Vehicle |

**Множественное наследование Python**

В Python **родительский класс** может иметь несколько **дочерних**, и, аналогично, **дочерний класс** может иметь несколько **родительских классов**. Давайте рассмотрим первый сценарий. Выполним следующий скрипт:

|  |  |
| --- | --- |
|  | # создаем класс Vehicle  class Vehicle:      def vehicle\_method(self):          print("Это родительский метод из класса Vehicle")    # создаем класс Car, который наследует Vehicle  class Car(Vehicle):      def car\_method(self):          print("Это дочерний метод из класса Car")    # создаем класс Cycle, который наследует Vehicle  class Cycle(Vehicle):      def cycleMethod(self):          print("Это дочерний метод из класса Cycle") |

В этом скрипте, родительский класс Vehicle наследуется двумя дочерними классами — Car и Cycle. Оба дочерних класса будут иметь доступ к vehicle\_method() родительского класса. Запустите следующий скрипт, чтобы увидеть это лично:

|  |  |
| --- | --- |
| 1  2  3  4 | car\_a = Car()  car\_a.vehicle\_method() # вызов метода родительского класса  car\_b = Cycle()  car\_b.vehicle\_method() # вызов метода родительского класса |

В выдаче вы увидите выдачу метода vehicle\_method() дважды, как показано ниже:

|  |  |
| --- | --- |
| 1  2 | Это родительский метод из класса Vehicle  Это родительский метод из класса Vehicle |

Вы можете видеть, как **родительский класс** наследуется двумя **дочерними классами**. Таким же образом, дочерний класс может иметь несколько родительских. Посмотрим на пример:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class Camera:      def camera\_method(self):          print("Это родительский метод из класса Camera")    class Radio:      def radio\_method(self):          print("Это родительский метод из класса Radio")    class CellPhone(Camera, Radio):       def cell\_phone\_method(self):          print("Это дочерний метод из класса CellPhone") |

В скрипте выше мы создали три класса: Camera, Radio, и CellPhone. Классы Camera и Radio наследуются классом CellPhone. Это значит, что класс CellPhone будет иметь доступ к методам классов Camera и Radio. Следующий скрипт подтверждает это:

|  |  |
| --- | --- |
| 1  2  3 | cell\_phone\_a = CellPhone()  cell\_phone\_a.camera\_method()  cell\_phone\_a.radio\_method() |

Выдача будет выглядеть следующим образом:

|  |  |
| --- | --- |
| 1  2 | Это родительский метод из класса Camera  Это родительский метод из класса Radio |

**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. Что дает наследование классов?
2. Как организовать наследование классов в языке Питон?

**8 Рекомендуемая литература**

**Плас, Дж. В.** Python для сложных задач. Наука о данных и машинное обучение / Дж.В. Плас. – СПб: Питер, 2018.

**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.