

Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«Финансовый университет при Правительстве Российской Федерации»  
Канашский филиал Финуниверситета

Методические рекомендации для студентов  
по выполнению самостоятельной работы  
по дисциплине ОП.07 Основы алгоритмизации и программирования  
по специальности 09.02.08 Интеллектуальные интегрированные системы

2025 г.

Организация-разработчик: ФГОБУ ВО «Финансовый университет при Правительстве Российской Федерации» Канашский филиал Финуниверситета

Разработчик(и):

Николаева И.В.- преподаватель ВКК Канашского филиала Финуниверситета

Рабочая программа дисциплины рассмотрена и рекомендована к утверждению на заседании предметной (цикловой) комиссии интеллектуальных интегрированных систем

Протокол от «29» мар 20 25 г. № 1

Председатель предметно (цикловой) комиссии:  /Славкина А.И./

## Пояснительная записка

Методические указания (рекомендации) для студентов по выполнению самостоятельной работы по дисциплине разработаны на основе требований Федерального государственного образовательного стандарта по специальности (специальностям) среднего профессионального образования по специальности 09.02.08 Интеллектуальные интегрированные системы.

Самостоятельная работа дисциплине ОП.07 Основы алгоритмизации и программирования проводится с целью:

- систематизации и закрепления полученных теоретических знаний и практических умений;
- формирования общих и профессиональных компетенций;
- углубления и расширения теоретических знаний;
- формирования умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развития познавательных способностей и активности: творческой инициативы, самостоятельности, ответственности и организованности;
- формирования самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развития исследовательских умений.

Самостоятельная работа по дисциплине ОП.07 Основы алгоритмизации и программирования включает задания по выполнению практических заданий, составлению конспектов, решению ситуационных задач и др.

Самостоятельная работа по дисциплине ОП.07 Основы алгоритмизации и программирования является внеаудиторной и обязательна для всех студентов. Внеаудиторная самостоятельная работа - планируемая учебная, учебно-исследовательская работа студентов, выполняемая вне занятий по заданию и при управлении преподавателем, но без его непосредственного участия.

Критериями оценки результатов внеаудиторной самостоятельной работы являются:

- уровень освоения учебного материала;
- умения использовать теоретические знания при выполнении практических задач;
- сформированность общеучебных умений;
- обоснованность и четкость изложения ответа;
- оформление материала в соответствии с требованиями.

**САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТА**  
**ПО ДИСЦИПЛИНЕ ОП.07 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ**  
**СПЕЦИАЛЬНОСТИ 09.02.08 ИНТЕЛЛЕКТУАЛЬНЫЕ ИНТЕГРИРОВАННЫЕ**  
**СИСТЕМЫ**

Тема	Кол-во часов	Вид работы
1	2	3
<b>РАЗДЕЛ 2 ЗНАКОМСТВО С БАЗОВЫМИ КОНСТРУКЦИЯМИ ИЗУЧАЕМОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ</b>		
Тема 2.4 Программирование циклов	2	Выполнение отчетов по практическим работам
Тема 2.5 Процедуры и функции	1	Выполнение отчетов по практическим работам
<b>РАЗДЕЛ 3 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ</b>		
Тема 3.3 Создание объектов (экземпляров) класса	2	Функции доступа и инкапсуляция

Самостоятельная работа 1

Тема 2.4 Программирование циклов

Задание 1. Выполнение отчетов по практическим работам

Самостоятельная работа 2

Тема 2.5 Процедуры и функции

Задание 1. Выполнение отчетов по практическим работам

Самостоятельная работа 3

Тема 3.3 Создание объектов (экземпляров) класса

Задание 1. Функции доступа и инкапсуляция

Описание задачи: Вам необходимо создать класс BankAccount, который представляет банковский счет клиента. Этот класс должен содержать приватные поля для хранения баланса счета и номера счета. Необходимо реализовать функции доступа для получения и изменения этих полей, соблюдая принципы инкапсуляции.

Требования:

Класс BankAccount должен содержать следующие члены:

Приватные поля:

accountNumber: Номер банковского счета (целочисленный тип).

balance: Текущий баланс счета (тип с плавающей точкой).

Конструктор, принимающий номер счета и начальный баланс.

Методы доступа:

Метод getAccountNumber() для получения номера счета.

Метод getBalance() для получения текущего баланса.

Метод deposit(amount) для внесения денег на счет (увеличивающий баланс).

Метод withdraw(amount) для снятия денег со счета (уменьшающий баланс). Метод должен проверять, достаточно ли средств на счете перед снятием.

Инкапсуляция: Все данные-члены должны быть приватными, а доступ к ним должен осуществляться только через соответствующие методы.

Дополнительно: Реализуйте проверку корректности ввода значений в методах deposit и withdraw. Сумма должна быть положительным числом.

Тестирование: Напишите простую программу, которая демонстрирует создание объекта класса BankAccount, внесение и снятие средств, а также получение текущей информации о счете.

Пример кода на языке Python:

class BankAccount:

```
class BankAccount:
    def __init__(self, account_number, initial_balance):
        self.__accountNumber = account_number
        self.__balance = initial_balance

    # Функция доступа для получения номера счета
    def getAccountNumber(self):
        return self.__accountNumber

    # Функция доступа для получения текущего баланса
    def getBalance(self):
        return self.__balance

    # Метод для внесения денег на счет
    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
            print(f"{amount} внесено на счет. Новый баланс: {self.__balance}")
        else:
            print("Сумма депозита должна быть положительной.")

    # Метод для снятия денег со счета
    def withdraw(self, amount):
        if amount > 0 and amount <= self.__balance:
            self.__balance -= amount
            print(f"{amount} снято со счета. Новый баланс: {self.__balance}")
        elif amount > self.__balance:
            print("Недостаточно средств на счету.")
        else:
            print("Сумма снятия должна быть положительной.")
```

```
# Пример использования класса BankAccount
account = BankAccount(123456789, 1000)

print(f"Счет №{account.getAccountNumber()}, Баланс: {account.getBalance()}")

account.deposit(500)
account.withdraw(200)

print(f"Текущий баланс: {account.getBalance()}")
```

Ожидаемый результат:

После выполнения задания студент должен продемонстрировать понимание следующих концепций:

Инкапсуляция данных посредством объявления приватных членов класса.

Использование методов доступа для чтения и изменения приватных данных.

Принцип скрытия внутренней реализации класса от внешнего мира.

### **Список использованной литературы**

1. Трофимов, В. В. Основы алгоритмизации и программирования: учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская; под редакцией В. В. Трофимова. — Москва: Издательство Юрайт, 2023. — 137 с. — (Профессиональное образование). — ISBN 978-5-534-07321-8. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/515434>