Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №6**

По дисциплине: “Языки программирования”

Тема: “Классы. Инкапсуляция. Наследование”**Вариант №12**

**Выполнил**: студент 2 курса группы ПО-7 Якимчик Дмитрий Вячеславович

**Проверила:** Бойко Д.О

Брест 2021

**Цель работы**: ознакомиться с принципами инкапсуляции и наследования

**Постановка задачи**

**Задание 1**

1. Определить пользовательский класс – «Автобус»

2. Определить счетчик

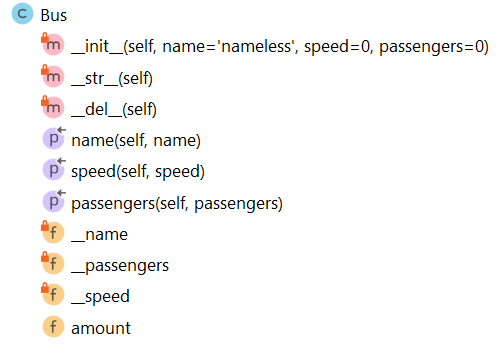
3. Определить в классе конструкторы с параметрами и без. Конструктор должен выводить сообщение о количестве объектов.

4. Определить в классе внешние компоненты-функции для получения и установки полей данных.

5. Написать демонстрационную программу, в которой объекты пользовательского класса создаются с помощью неявного использования конструкторов без параметров. 6. Показать в программе явное использование конструкторов с параметрами. Определение пользовательского класса

Поля класса «Автобус»:

* name: str название
* speed: int скорость
* passengers: int количество пассажиров



**Код программы:**

*# Вариант 12 Задание 1-18  
# Автобус*class Bus:  
 amount = 0  
 \_\_name: str  
 \_\_speed: int  
 \_\_passengers: int  
  
 def \_\_init\_\_(self, name: str = 'nameless', speed: int = 0, passengers: int = 0) -> None:  
 Bus.amount += 1  
 self.\_\_name = name  
 self.\_\_speed = speed  
 self.\_\_passengers = passengers  
 print(f'Created {self}')  
  
 def \_\_str\_\_(self) -> str:  
 return f'Bus (name: {self.\_\_name}, speed: {self.\_\_speed}, ' \  
 f'passengers: {self.passengers}; amount: {Bus.amount})'  
  
 def \_\_del\_\_(self) -> None:  
 print(f'Deleted {self}')  
  
 @property  
 def name(self) -> str:  
 return self.\_\_name  
  
 @name.setter  
 def name(self, name: str) -> None:  
 self.\_\_name = name  
  
 @property  
 def speed(self) -> int:  
 return self.\_\_speed  
  
 @speed.setter  
 def speed(self, speed: int) -> None:  
 self.\_\_speed = speed  
  
 @property  
 def passengers(self) -> int:  
 return self.\_\_passengers  
  
 @passengers.setter  
 def passengers(self, passengers: int) -> None:  
 self.\_\_passengers = passengers

**Задание 2**

Для варианта «Актер-Сотрудник-Режиссер» выполнить следующее:

1. Построить модель предметной области в соответствии со своим вариантом. Разработать классы в соответствии с моделью предметной области

3. Для каждого класса создать конструктор

4. Для каждого класса создать внешние функции установки и получения полей данных

5. Для каждого класса разработать функции, позволяющие представить на экране значения полей данных

6. Для каждого класса разработать функции, позволяющие вводить с консоли значения полей данных

7. Написать демонстрационную программу, иллюстрирующую использование разработанных классов

8. Построить диаграмму классов

9. Разработанные классы разместить в одном пакете

**Модель предметной области**

Родительским классом является работник. У него есть поля: **имя, возраст и зарплата**. От него наследуется актер с дополнительным полем **роль** и режиссер и дополнительным полем **фильм**.



**Код программы:**

*# Вариант 12 Задание 2-13:  
# Актер<-Сотрудник->Режиссер*class Employee:  
 \_name: str  
 \_age: int  
 \_salary: int  
  
 def \_\_init\_\_(self, name='nameless', age=0, salary=0) -> None:  
 self.\_name = name  
 self.\_age = age  
 self.\_salary = salary  
 print(f'Created employee: {self.\_name}, {self.\_age}, {self.\_salary}')  
  
 def \_\_str\_\_(self) -> str:  
 return f'Employee: {self.\_name}, {self.\_age}, {self.\_salary}'  
  
 def \_\_del\_\_(self) -> None:  
 print(f'Deleted {self}')  
  
 @property  
 def name(self) -> str:  
 return self.\_name  
  
 @name.setter  
 def name(self, name) -> None:  
 self.\_name = name  
  
 @property  
 def age(self) -> int:  
 return self.\_age  
  
 @age.setter  
 def age(self, age) -> None:  
 self.\_age = age  
  
 @property  
 def salary(self) -> int:  
 return self.\_salary  
  
 @salary.setter  
 def salary(self, salary) -> None:  
 self.\_salary = salary  
  
  
class Actor(Employee):  
 \_role: str  
  
 def \_\_init\_\_(self, name='nameless', age=0, salary=0, role='empty') -> None:  
 super().\_\_init\_\_(name, age, salary)  
 self.\_role = role  
 prev: str = super().\_\_str\_\_()  
 print(f'Created actor{prev[prev.find(":"):]}, {self.\_role}')  
  
 def \_\_str\_\_(self) -> str:  
 prev: str = super().\_\_str\_\_()  
 return 'Actor' + prev[prev.find(':'):] + f', {self.\_role}'  
  
 @property  
 def role(self) -> str:  
 return self.\_role  
  
 @role.setter  
 def role(self, role) -> None:  
 self.\_role = role  
  
  
class Director(Employee):  
 \_film: str  
  
 def \_\_init\_\_(self, name='nameless', age=0, salary=0, film='secret') -> None:  
 super().\_\_init\_\_(name, age, salary)  
 self.\_film = film  
 prev: str = super().\_\_str\_\_()  
 print(f'Created director{prev[prev.find(":"):]}, {self.\_film}')  
  
 def \_\_str\_\_(self) -> str:  
 prev: str = super().\_\_str\_\_()  
 return 'Director' + prev[prev.find(':'):] + f', {self.\_film}'  
  
 @property  
 def film(self) -> str:  
 return self.\_film  
  
 @film.setter  
 def film(self, film) -> None:  
 self.\_film = film  
  
  
def main():  
 print('Задание 1')  
 maz = Bus('maz', 120, 50)  
 mercedes = Bus('mercedes', 150, 35)  
 maz.passengers = 55  
 mercedes.speed = 140  
 print(maz)  
 print(mercedes)  
  
 print('\nЗадание 2')  
 print('Создание экземпляров класса:')  
 employee = Employee('Random woman', 60, 600)  
 actor = Actor('Kirill', 18, 1500, 'main')  
 director = Director('Michel', 19, 3600, 'The last choice')  
 people = [employee, actor, director]  
 print('\nРезультат:', \*people, sep='\n')  
  
 print('\nИзменение с консоли:')  
 ans = input(f'Хотите ли вы поменять поля у {employee.name}? (y/n):')  
 if ans == 'y':  
 employee.name = input('employee.name: ')  
 employee.age = int(input('employee.age: '))  
 employee.salary = int(input('employee.salary: '))  
  
 ans = input(f'Хотите ли вы поменять поля у {actor.name}? (y/n):')  
 if ans == 'y':  
 actor.name = input('actor.name: ')  
 actor.age = int(input('actor.age: '))  
 actor.salary = int(input('actor.salary: '))  
 actor.role = input('actor.role: ')  
  
 ans = input(f'Хотите ли вы поменять поля у {director.name}? (y/n):')  
 if ans == 'y':  
 director.name = input('director.name: ')  
 director.age = int(input('director.age: '))  
 director.salary = int(input('director.salary: '))  
 director.film = input('director.film: ')  
  
 print('\nРезультат:', \*people, sep='\n')  
 print('\nСборщик мусора:')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результаты программы:**

"D:\Lab\_6\_YAP\venv\Scripts\python.exe D:/Lab\_6\_YAP/main.py"

Задание 1

Created Bus (name: maz, speed: 120, passengers: 50; amount: 1)

Created Bus (name: mercedes, speed: 150, passengers: 35; amount: 2)

Bus (name: maz, speed: 120, passengers: 55; amount: 2)

Bus (name: mercedes, speed: 140, passengers: 35; amount: 2)

Задание 2

Создание экземпляров класса:

Created employee: Random woman, 60, 600

Created employee: Kirill, 18, 1500

Created actor: Kirill, 18, 1500, main

Created employee: Michel, 19, 3600

Created director: Michel, 19, 3600, The last choice

Изменение:

Employee: Lida, 60, 600

Actor: Kirill, 18, 1200, second

Director: Michel, 19, 4600, The last chance

Сборщик мусора:

Deleted Bus (name: maz, speed: 120, passengers: 55; amount: 2)

Deleted Bus (name: mercedes, speed: 140, passengers: 35; amount: 2)

Deleted Director: Michel, 19, 4600, The last chance

Deleted Actor: Kirill, 18, 1200, second

Deleted Employee: Lida, 60, 600

Process finished with exit code 0

**Вывод:** в ходе выполнения я изучил принципы инкапсуляции и наследования