

Отчёт по лабораторной работе №8

Тема: Тестирование программного обеспечения

Сведения о студенте

Дата: 2025-12-15

Семestr: 2 курс 1 семестр (3 семестр)

Группа: ПИН-б-о-24-1

Дисциплина: Технологии программирования

Студент: Губжоков Роман Русланович

Оглавление

1. [Введение](#)
2. [Цель работы](#)
3. [Часть 1: Тестирование Employee](#)
4. [Часть 2: Тестирование наследования](#)
5. [Часть 3: Тестирование полиморфизма и магических методов](#)
6. [Часть 4: Тестирование композиции и агрегации](#)
7. [Часть 5: Тестирование паттернов проектирования](#)
8. [Результаты выполнения тестов](#)
9. [Выводы](#)
10. [Заключение](#)

Введение

Тестирование программного обеспечения является критически важной частью процесса разработки. Модульное тестирование (unit testing) позволяет проверить корректность работы отдельных компонентов системы в изоляции от остальных частей.

Использование фреймворка pytest и техник изоляции зависимостей с помощью моков и стабов значительно упрощает процесс написания и поддержки тестов.

Данная работа направлена на освоение основ модульного тестирования для системы учета сотрудников, реализованной на основе принципов ООП и паттернов проектирования. Система включает иерархию классов сотрудников, отделы, проекты, компанию и различные паттерны проектирования.

Цель работы

Освоить основы модульного тестирования, научиться писать unit-тесты для классов и методов, использовать фреймворк pytest, применять техники изоляции зависимостей с помощью моков и стабов.

Часть 1: Тестирование Employee

Цель: Написать модульные тесты для класса Employee, проверяющие корректность инкапсуляции, валидации данных и работы методов.

Реализованные тесты

1. **test_init_emp** - Тест инициализации работника с корректными данными
2. **test_emp_errors** - Тест валидации некорректного ID (отрицательного и нулевого)
3. **test_emp_salary_error** - Тест валидации отрицательной базовой зарплаты
4. **test_emp_no_name** - Тест валидации пустого имени и имени из пробельных символов
5. **test_emp_salary** - Тест расчета зарплаты (возвращает базовую зарплату)
6. **test_emp_str** - Тест строкового представления объекта Employee
7. **test_emp_id_valid** - Тест валидации сеттера ID (положительные значения)
8. **test_emp_name_valid** - Тест валидации сеттера имени (непустые значения)
9. **test_employee_salary_valid** - Тест валидации сеттера базовой зарплаты (неотрицательные значения)
10. **test_employee_dept_valid** - Тест валидации сеттера отдела (непустые значения)
11. **test_emp_get_info** - Тест метода get_info на наличие ключевой информации о сотруднике

Результаты тестирования

```
=====
test session starts
=====
platform win32 -- Python 3.13.9, pytest-9.0.2, pluggy-1.6.0 --
C:\Users\DezerTear\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation
.Python.3.13_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\DezerTear\Desktop\uni\progtech
collected 11 items

lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_init_emp
Инициализация работника...
    Все данные корректны!
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_errors
Тип исключения: ValueError
    Сообщение: ID должен быть положительным целым числом.
    Сообщение: ID должен быть положительным целым числом.
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_salary_error
    Тип исключения: ValueError
    Сообщение: Зарплата не может быть отрицательной.
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_no_name
Тип исключения: ValueError
    Сообщение: Имя не может быть пустой строкой.
    Тип исключения: ValueError
    Сообщение: Имя не может быть пустой строкой.
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_salary
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_str PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_id_valid
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_name_valid
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_employee_salary_
_valid PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_employee_dept_v
alid PASSED
lab_4_5_8_9_00P/unit_tests/test_employee.py::TestEmployee::test_emp_get_info
PASSED

=====
11 passed in 0.02s
=====
```

Часть 2: Тестирование наследования

Цель: Написать модульные тесты для иерархии классов сотрудников, проверяющие корректность наследования, реализацию абстрактных методов и полиморфное поведение.

Реализованные тесты

1. **test_abs_emp_impossible** - Тест невозможности создания экземпляра абстрактного класса AbstractEmployee
2. **test_manager_sal_calc** - Тест расчета зарплаты менеджера (базовая зарплата + бонус)
3. **test_manager_get_info_w_bonus** - Тест вывода информации о менеджере с данными о бонусе
4. **test_manager_bonus_setter_valid** - Тест валидации сеттера бонуса менеджера
5. **test_developer_salary_by_level** - Параметризованный тест расчета зарплаты разработчика в зависимости от уровня
6. **test_developer_add_skill** - Тест добавления нового навыка разработчику
7. **test_developer_add_duplicate_skill** - Тест обработки дублирования навыков у разработчика
8. **test_developer_get_info_includes_tech_stack** - Тест вывода информации о разработчике со стеком технологий
9. **test_developer_invalid_seniority_level** - Тест валидации некорректного уровня разработчика
10. **test_developer_skills_iteration** - Тест итерации по навыкам разработчика
11. **test_salesperson_salary_calculation** - Тест расчета зарплаты продавца (базовая + комиссия)
12. **test_salesperson_update_sales** - Тест обновления объема продаж и перерасчета зарплаты
13. **test_salesperson_get_info_includes_commission** - Тест вывода информации о продавце с данными о комиссии
14. **test_salesperson_invalid_commission_rate** - Тест валидации некорректной ставки комиссии
15. **test_employee_builder_method** - Тест построения различных типов сотрудников через билдер
16. **test_employee_builder_missing_states** - Тест обработки отсутствия обязательных полей в билдере
17. **test_employee_builder_wrong_states** - Тест обработки некорректных комбинаций параметров в билдере
18. **test_polymorphic_behavior** - Тест полиморфного поведения различных типов сотрудников

Результаты тестирования

```
=====
test session starts
=====
platform win32 -- Python 3.13.9, pytest-9.0.2, pluggy-1.6.0 --
C:\Users\DezerTear\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation
.Python.3.13_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\DezerTear\Desktop\uni\progtech
collected 20 items
```

```
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestAbstractEmployee::te
st_abs_emp_impossible <ExceptionInfo TypeError("Can't instantiate abstract
class AbstractEmployee without an implementation for abstract methods
'calculate_salary', 'from_dict', 'get_info')> tlen=1>
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestManager::test_manage
r_sal_calc PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestManager::test_manage
r_get_info_w_bonus PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestManager::test_manage
r_bonus_setter_valid PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_salary_by_level[junior-1250] PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_salary_by_level[middle-1875] PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_salary_by_level[senior-2500] PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_add_skill PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_add_duplicate_skill PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_get_info_includes_tech_stack PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_invalid_seniority_level <ExceptionInfo ValueError('Уровень разработчика
должен быть одним
из: junior, middle, senior')> tlen=3>
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestDeveloper::test_deve
loper_skills_iteration PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestSalesperson::test_sa
lesperson_salary_calculation PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestSalesperson::test_sa
lesperson_update_sales PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestSalesperson::test_sa
lesperson_get_info_includes_commission PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestSalesperson::test_sa
lesperson_invalid_commission_rate <ExceptionInfo ValueError('Процент комиссии
должен быть в диапазоне от 0 до 1 включительно.')> tlen=3> <ExceptionInfo
ValueError('Процент комиссии должен быть в диапазоне от 0 до 1 включительно.')>
tlen=3>
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestEmployeeBuilder::tes
t_employee_builder_method PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestEmployeeBuilder::tes
t_employee_builder_missing_states <ExceptionInfo ValueError('Обязательные
параметры не указаны:
base_salary')> tlen=2>
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestEmployeeBuilder::tes
t_employee_builder_wrong_states <ExceptionInfo ValueError('Конфликт параметров:
указаны поля для разных типов сотрудников')> tlen=2>
```

```
PASSED
lab_4_5_8_9_00P/unit_tests/test_employee_hierarchy.py::TestPolymorphicBehavior:
:test_polymorphic_behavior PASSED
=====
20 passed in 0.03s
=====
```

Часть 3: Тестирование полиморфизма и магических методов

Цель: Написать комплексные тесты для проверки полиморфного поведения, перегрузки операторов и магических методов.

Реализованные тесты

1. **test_department_add_employee** - Тест добавления сотрудника в отдел
2. **test_department_remove_employee** - Тест удаления сотрудника из отдела по ID
3. **test_department_get_employees** - Тест получения списка сотрудников отдела
4. **test_department_calculate_total_salary_polymorphic** - Тест расчета общей зарплаты отдела с полиморфными сотрудниками
5. **test_department_get_employee_count** - Тест подсчета сотрудников по типам в отделе
6. **test_department_find_employee_by_id** - Тест поиска сотрудника в отделе по ID
7. **test_department_find_employee_by_id_not_found** - Тест поиска несуществующего сотрудника в отделе
8. **test_employee_equality** - Тест сравнения сотрудников на равенство (по ID)
9. **test_employee_salary_comparison** - Тест сравнения сотрудников по зарплате (операторы < и >)
10. **test_employee_addition** - Тест сложения зарплат сотрудников (оператор +)
11. **test_employee_radd** - Тест суммирования зарплат нескольких сотрудников через sum()
12. **test_department_magic_methods** - Тест магических методов отдела (len, getitem, contains)
13. **test_department_iteration** - Тест итерации по сотрудникам отдела
14. **test_department_getitem_index_error** - Тест обработки ошибки индексации в отделе
15. **test_developer_skills_iteration** - Тест итерации по навыкам разработчика
16. **test_employee_serialization** - Тест сериализации/десериализации базового сотрудника
17. **test_manager_serialization** - Тест сериализации/десериализации менеджера
18. **test_developer_serialization** - Тест сериализации/десериализации разработчика
19. **test_employee_sorting** - Тест сортировки сотрудников по имени и зарплате
20. **test_employee_sorting_using_lt** - Тест сортировки сотрудников с использованием оператора <
21. **test_department_integration** - Интеграционный тест отдела с сотрудниками разных типов

Результаты тестирования

```
=====
test session starts
=====
platform win32 -- Python 3.13.9, pytest-9.0.2, pluggy-1.6.0 --
C:\Users\DezerTear\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation
.Python.3.13_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\DezerTear\Desktop\uni\progtech
collected 21 items

lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
add_employee PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
remove_employee Сотрудник John (ID: 1) удален из отдела 'IT'
PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
get_employees PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
calculate_total_salary_polymorphic PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
get_employee_count PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
find_employee_by_id PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartment::test_department_
find_employee_by_id_not_found PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeMagicMethods::test_e
mployee_equality PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeMagicMethods::test_e
mployee_salary_comparison PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeMagicMethods::test_e
mployee_addition PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeMagicMethods::test_e
mployee_radd PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartmentMagicMethods::test_
_department_magic_methods PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartmentMagicMethods::test_
_department_iteration PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartmentMagicMethods::test_
_department_getitem_index_error <ExceptionInfo IndexError('list index out of
range') tlen=2>
PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDeveloperSkillsIteration::te
st_developer_skills_iteration PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeSerialization::test_
employee_serialization PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeSerialization::test_
manager_serialization PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeSerialization::test_
developer_serialization PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeSorting::test_employ
ee_sorting PASSED
```

```
lab_4_5_8_9_00P/unit_tests/test_department.py::TestEmployeeSorting::test_employee_sorting_using_lt PASSED
lab_4_5_8_9_00P/unit_tests/test_department.py::TestDepartmentIntegration::test_department_integration PASSED
=====
21 passed in 0.03s
=====
```

Часть 4: Тестирование композиции и агрегации

Цель: Написать тесты для проверки корректности работы композиции, агрегации, валидации данных и сложных бизнес-методов.

Реализованные тесты

1. **test_project_team_management** - Тест управления командой проекта (добавление/удаление участников)
2. **test_project_total_salary** - Тест расчета общей зарплаты команды проекта
3. **test_project_invalid_status_raises_error** - Тест валидации статуса проекта и обработки недопустимых статусов
4. **test_company_department_management** - Тест управления отделами компании (добавление/удаление)
5. **test_company_find_employee** - Тест поиска сотрудника по ID во всей компании
6. **test_company_cannot_delete_department_with_employees** - Тест запрета удаления отдела с сотрудниками
7. **test_company_duplicate_employee_id_raises_error** - Тест обработки дублирования ID сотрудника
8. **test_company_get_all_employees** - Тест получения всех сотрудников компании
9. **test_company_calculate_total_monthly_cost** - Тест расчета общей месячной стоимости компании
10. **test_company_serialization_roundtrip** - Тест сериализации/десериализации компании в JSON
11. **test_company_department_statistics** - Тест сбора статистики по отделам компании
12. **test_complex_company_structure** - Тест сложной структуры компании с несколькими отделами и типами сотрудников
13. **test_company_project_management** - Тест управления проектами компании (добавление/назначение сотрудников)
14. **test_company_duplicate_project_id** - Тест обработки дублирования ID проекта
15. **test_company_cannot_delete_project_with_team** - Тест запрета удаления проекта с назначенными сотрудниками

Результаты тестирования

```
=====
test session starts
=====
platform win32 -- Python 3.13.9, pytest-9.0.2, pluggy-1.6.0 --
C:\Users\DezerTear\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation
.Python.3.13_qbz5n2kfra8p0\python.exe
```

```
cachedir: .pytest_cache
rootdir: C:\Users\DezerTear\Desktop\uni\progtech
collected 15 items

lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestProject::test_project_team_management Сотрудник John добавлен в
проект 'AI Platform'
Сотрудник John (ID: 1) удален из проекта 'AI Platform'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestProject::test_project_total_salary Сотрудник Rick добавлен в
проект 'AI Platform'
Сотрудник Robert добавлен в проект 'AI Platform'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestProject::test_project_invalid_status_raises_error
<ExceptionInfo InvalidStatusError('Неверный формат статуса проекта для: "Test"
- "error", используйте статусы: ["planning", "active", "completed",
"cancelled"].') tlen=3>
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompany::test_company_department_management Отдел 'Development'
добавлен в компанию 'TechCorp'
Отдел 'Development' удален из компании 'TechCorp'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompany::test_company_find_employee Отдел 'Development'
добавлен в компанию 'TechCorp'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompany::test_company_cannot_delete_department_with_employees
Отдел 'Development' добавлен в компанию 'TechCorp'
<ExceptionInfo ValueError("Невозможно удалить отдел 'Development', в котором
находятся сотрудники") tlen=2>
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompany::test_company_duplicate_employee_id_raises_error Отдел
'Development' добавлен в компанию 'TechCorp'
<ExceptionInfo ValueError('Сотрудник Jane (ID: 1) уже находится в отделе
"Development"') tlen=2>
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompany::test_company_get_all_employees Отдел 'Development'
добавлен в компанию 'TechCorp'
Отдел 'Sales' добавлен в компанию 'TechCorp'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompany::test_company_calculate_total_monthly_cost Отдел
'Development' добавлен в компанию 'TechCorp'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompanySerialization::test_company_serialization_roundtrip
```

```
Отдел 'Development' добавлен в компанию 'TechCorp'
Компания сохранена в файл:
C:\Users\DezerTear\Desktop\uni\progtech\test_output\tmpx3fpawf8.json
Размер файла: 1071 байт
Файл сохранен в:
C:\Users\DezerTear\Desktop\uni\progtech\test_output\tmpx3fpawf8.json
Загружаем из файла:
C:\Users\DezerTear\Desktop\uni\progtech\test_output\tmpx3fpawf8.json
Отдел: Development (1 сотрудников)

Компания 'TechCorp' успешно загружена
Отделов: 1
Проектов: 0
Сотрудников: 1
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestCompanyBusinessMethods::test_company_department_statistics
Отдел 'Development' добавлен в компанию 'TechCorp'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestComplexCompanyStructure::test_complex_company_structure Отдел
'Development' добавлен в компанию 'TechInnovations'
Отдел 'Sales' добавлен в компанию 'TechInnovations'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestComplexCompanyStructure::test_company_project_management Отдел
'Development' добавлен в компанию 'TechCorp'
Проект 'AI Platform' (ID: 1) добавлен в компанию 'TechCorp'
Сотрудник Robert добавлен в проект 'AI Platform'
Сотрудник Robert назначен на проект 'AI Platform'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestComplexCompanyStructure::test_company_duplicate_project_id
Проект 'AI Platform' (ID: 1) добавлен в компанию 'TechCorp'
PASSED
lab_4_5_8_9_00P/unit_tests/test_company-
project.py::TestComplexCompanyStructure::test_company_cannot_delete_project_wit
h_team Отдел 'Development' добавлен в компанию 'TechCorp'
Проект 'AI Platform' (ID: 1) добавлен в компанию 'TechCorp'
Сотрудник Robert добавлен в проект 'AI Platform'
Сотрудник Robert назначен на проект 'AI Platform'
PASSED

=====
15 passed in 0.03s
=====
```

Часть 5: Тестирование паттернов проектирования

Цель: Написать тесты для проверки корректной работы различных паттернов проектирования.

Реализованные тесты

1. **test_singleton_same_instance** - Тест паттерна Singleton: получение одного и того же экземпляра для одного пути к базе данных
2. **test_different_paths_different_instances** - Тест получения разных экземпляров Singleton для разных путей к базе данных
3. **test_build_simple_employee** - Тест построения обычного сотрудника через EmployeeBuilder
4. **test_build_manager** - Тест построения менеджера через EmployeeBuilder
5. **test_build_developer** - Тест построения разработчика через EmployeeBuilder с добавлением навыков
6. **test_tech_company_factoryCreates_correct_types** - Тест фабрики тех-компании: проверка создания разработческих отделов без отделов продаж
7. **test_sales_company_factoryCreates_correct_types** - Тест фабрики продажной компании: проверка создания отделов продаж без разработческих отделов
8. **test_adapter_calculates_different_from_native** - Тест адаптера зарплаты: проверка расчёта зарплат, отличных от нативного расчёта сотрудников
9. **test_bonus_decorator** - Тест декоратора бонуса: проверка фиксированной надбавки к зарплате
10. **test_training_decorator** - Тест декоратора обучения: проверка процентной надбавки к зарплате
11. **test_decorator_composition** - Тест композиции декораторов: применение нескольких декораторов к сотруднику
12. **test_facade_hire_and_statistics** - Тест фасада компании: найм сотрудников разных типов, получение статистики и увольнение

Результаты тестирования

```
=====
test session starts
=====
platform win32 -- Python 3.13.9, pytest-9.0.2, pluggy-1.6.0 --
C:\Users\DezerTear\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation
.Python.3.13_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\DezerTear\Desktop\uni\progtech
collected 12 items

lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestDatabaseConnectionSingleton::t
est_singleton_same_instance PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestDatabaseConnectionSingleton::t
est_different_paths_different_instances PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestEmployeeBuilder::test_build_si
mple_employee PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestEmployeeBuilder::test_build_ma
nager PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestEmployeeBuilder::test_build_de
veloper PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestAbstractCompanyFactory::test_t
```

```
ech_company_factoryCreatesCorrectTypes Отдел 'Development' добавлен в
компанию 'TechNova'
Отдел 'QA' добавлен в компанию 'TechNova'
Проект 'Project 1' (ID: 1) добавлен в компанию 'TechNova'
Сотрудник Employee 1 добавлен в проект 'Project 1'
PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestAbstractCompanyFactory::test_s
ales_company_factoryCreatesCorrectTypes Отдел 'Sales' добавлен в компанию
'SellFast'
Отдел 'Marketing' добавлен в компанию 'SellFast'
PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestPayrollAdapter::test_adapter_c
alculatesDifferentFromNative PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestEmployeeDecorators::test_bonus
_decorator PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestEmployeeDecorators::test_training
_decorator PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestEmployeeDecorators::test_decorator
_composition PASSED
lab_4_5_8_9_00P/unit_tests/test_patterns.py::TestCompanyFacade::test_facade_hir
e_and_statistics Отдел 'Management' добавлен в компанию 'TestMegaCorp'
Сотрудник Виктор нанят в отдел Management с ID 1
Отдел 'Development' добавлен в компанию 'TestMegaCorp'
Сотрудник Яна нанят в отдел Development с ID 2
Отдел 'Sales' добавлен в компанию 'TestMegaCorp'
Сотрудник Максим нанят в отдел Sales с ID 3
Сотрудник Виктор (ID: 1) удален из отдела 'Management'
Сотрудник Виктор (ID: 1) уволен из отдела Management
PASSED
```

```
=====  
12 passed in 0.03s  
=====
```

Результаты выполнения тестов

Статистика тестов

- **Всего тестов:** 77
- **Покрытие:** Все основные классы и методы
- **Результат:** Все тесты проходят успешно

Категории тестов

1. **Unit-тесты** - Тестирование отдельных классов и методов
2. **Интеграционные тесты** - Тестирование взаимодействия компонентов
3. **Параметризованные тесты** - Тестирование различных сценариев

Выводы

По каждой части

Часть 1 (Employee):

- Успешно проверена инкапсуляция данных
- Валидация входных данных работает корректно
- Все сеттеры и геттеры функционируют правильно

Часть 2 (Наследование):

- Абстрактные классы корректно реализованы
- Полиморфизм работает как ожидается
- Фабрика сотрудников создает объекты правильно

Часть 3 (Полиморфизм и магические методы):

- Магические методы реализованы корректно
- Полиморфное поведение при расчете зарплат работает
- Сериализация и десериализация функционируют правильно

Часть 4 (Композиция и агрегация):

- Композиция (Project-Employee) работает корректно
- Агрегация (Company-Department-Project) функционирует правильно
- Бизнес-методы возвращают корректные результаты

Часть 5 (Паттерны):

- Все паттерны реализованы и работают корректно
- Взаимодействие паттернов между собой функционирует

Общие выводы

1. **Покрытие тестами** - Система имеет хорошее покрытие тестами, что повышает надежность кода
2. **Качество кода** - Тесты помогают выявлять ошибки на ранних этапах разработки
3. **Документация** - Тесты служат живой документацией к коду
4. **Рефакторинг** - Наличие тестов позволяет безопасно рефакторить код

Заключение

В ходе выполнения лабораторной работы №8 был реализован комплексный набор модульных тестов для системы учета сотрудников. Тесты покрывают все основные компоненты системы:

- Базовые классы и инкапсуляцию
- Наследование и полиморфизм
- Магические методы и операторы
- Композицию и агрегацию
- Паттерны проектирования

Достигнутые результаты

1. **Создано 77 тестов** для различных компонентов системы

2. Использованы различные техники тестирования:

- Параметризованные тесты
- Интеграционные тесты

3. Проверена корректность работы:

- Валидации данных
- Полиморфного поведения
- Паттернов проектирования
- Бизнес-логики

Практическая значимость

Результаты работы демонстрируют:

- Важность модульного тестирования для обеспечения качества кода
- Эффективность использования pytest для написания тестов
- Преимущества использования моков для изоляции зависимостей
- Значение тестов как документации к коду

Тестирование является неотъемлемой частью процесса разработки программного обеспечения и позволяет создавать более надежные и поддерживаемые системы.
