Бауманский учебный центр «Специалист»

Программа курса Python: паттерны проектирования (порождающие, структурные и поведенческие)

Код курса: **ПИТПАТ**

Тема	Ак. часов

Модуль 1. Продвинутая работа с ООП
 	
• Особенности ООП в Python:
  ◦ Дандер методы (__str__, __repr__, __call__)
  ◦ Свойства (@property) — управление доступом к атрибутам, валидация данных.
• Абстрактные классы (ABC) vs. Утиная типизация.
• Композиция или Наследование: почему наследование часто приводит к хрупкому коду.
	2

Модуль 2. Фундамент — Принципы SOLID **Тема 2.1: SRP (Принцип единственной ответственности)** • Почему один класс — одна причина для изменений • Рефакторинг «божественных объектов» • Пример: Разделение логики заказа и логики сохранения в файл/базу данных **Тема 2.2: OCP (Принцип открытости/закрытости)** • Классы открыты для расширения, но закрыты для модификации • Стратегия расширения через наследование и композицию • Пример: Модуль расчёта доставки (Почта России, СДЭК) — легко добавить новый, не трогая старый код **Тема 2.3: LSP (Принцип подстановки Барбары Лисков)** • Как не сломать код при использовании наследников • Пример: Нарушение LSP при наследовании классов «Счёт» > «Кредитный счёт» (нельзя снимать больше остатка) **Тема 2.4: ISP (Принцип разделения интерфейсов)** • «Толстые» интерфейсы — зло • Разделение на абстрактные классы • Пример: IT-компания со стажёрами, разработчиками и тимлидами — разные наборы методов **Тема 2.5: DIP (Принцип инверсии зависимостей)** • Зависимости на абстракциях, а не на конкретике • Пример: Модуль сохранения данных зависит от абстрактного *Хранилище*, а не от конкретного *ФайловоеХранилище* или *БДХранилище*	**6**

Модуль 3. Порождающие паттерны **Тема 3.1: Одиночка (Singleton)** • Гарантия единственного экземпляра • Реализация через __new__ • Пример: Пул соединений с локальной БД (SQLite) или менеджер конфигурации **Тема 3.2: Фабричный метод (Factory Method)** • Виртуальный конструктор • Пример: Кофемашина (создание разных напитков), кастомный валидатор **Тема 3.3: Абстрактная фабрика (Abstract Factory)** • Фабрика фабрик. Создание семейств взаимосвязанных объектов • Пример: GUI-библиотека (кнопки, чекбоксы, окна) для разных стилей оформления (Windows, macOS) **Тема 3.4: Строитель (Builder)** • Пошаговое создание сложного объекта • Сравнение с «телескопическим конструктором» • Пример: Сборка персонализированной email-рассылки (локальной)	**7**
Модуль 4. Структурные паттерны **Тема 4.1: Адаптер (Adapter)** • Обёртка для несовместимых интерфейсов • Пример: Адаптер для работы с разными форматами файлов (JSON, CSV, XML) под единый интерфейс **Тема 4.2: Мост (Bridge)** • Разделение абстракции и реализации • Пример: Отправка уведомлений разных типов (простое, срочное, с кнопкой) через разные каналы (Email, SMS, Telegram) — всё локально **Тема 4.3: Компоновщик (Composite)** • Древовидные структуры и единый интерфейс для листьев и узлов • Пример: Файловая система (Файл vs Папка), графический редактор (группы фигур) **Тема 4.4: Декоратор (Decorator)** • Динамическое добавление обязанностей без создания подклассов • Пример: Расчёт стоимости кофе с добавками (молоко, сахар, сироп) **Тема 4.5: Заместитель (Proxy)** • Ленивая инициализация, контроль доступа, логирование • Пример: Валидация входных данных перед передачей в основной обработчик (проверка типов, форматов, обязательных полей)	**7**

Модуль 5. Поведенческие паттерны **Тема 5.1: Цепочка обязанностей (Chain of Responsibility)** • Передача запроса по цепочке обработчиков • Пример: Система валидации формы регистрации (проверка логина, пароля, email) **Тема 5.2: Команда (Command)** • Инкапсуляция запроса в объект • Пример: Выполнение массовых операций в текстовом редакторе (копировать, вставить, удалить) с поддержкой отмены **Тема 5.3: Итератор (Iterator)** • Последовательный доступ к элементам агрегата без раскрытия структуры • Пример: Построчное чтение больших лог-файлов (локальных) **Тема 5.4: Посредник (Mediator)** • Уменьшение связанности классов • Пример: Логика сложной формы регистрации (поля влияют друг на друга) **Тема 5.5: Наблюдатель (Observer)** • Система издатель-подписчик • Пример: Система уведомлений внутри приложения (подписчики получают уведомление о событии) **Тема 5.6: Состояние (State)** • Изменение поведения объекта при изменении состояния • Пример: Статусы заказа (Новый > Подтверждён > Отгружен > В пути > Доставлен) **Тема 5.7: Стратегия (Strategy)** • Взаимозаменяемые алгоритмы • Пример: Алгоритмы сжатия данных (ZIP, RAR, GZIP) или расчёта налогов (ОСН, УСН) **Тема 5.8: Шаблонный метод (Template Method)** • Скелет алгоритма в базовом классе, конкретные шаги — в подклассах • Пример: Загрузчик данных из разных источников (CSV, JSON, XML) — единый алгоритм, разные реализации шагов	**7**
Модуль 6. Итоговая работа • Разработка локального проекта с применением изученных паттернов. • Рефакторинг легаси-кода с использованием принципов SOLID. • Защита проекта с обоснованием выбора паттернов.	**3**
Аудиторная нагрузка в классе с преподавателем	**32 ак.ч. + 32 ак.ч. самостоятельно**

По окончании обучения на курсе проводится итоговая аттестация. Аттестация проводится в виде теста на последнем занятии или на основании оценок практических работ, выполняемых во время обучения на курсе.

Контакты центра

Для частных лиц
+7 (495) 232-32-16

Для организаций
+7 (495) 780-48-44

E-mail
info@specialist.ru