

Скоков
Роман
Юрьевич
Ректор "
ВИЭПП"

2026.02.27
15:46:51
+03'00'

Приложение 2 к рабочей программе дисциплины
«Объектно-ориентированное проектирование и программирование»

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОЛЖСКИЙ ИНСТИТУТ ЭКОНОМИКИ, ПЕДАГОГИКИ И ПРАВА»

Факультет информационной безопасности и права

Фонд оценочных средств
по дисциплине
«Объектно-ориентированное проектирование и программирование»

Направление подготовки:
09.03.03 Прикладная информатика

Направленность (профиль) образовательной программы:
Прикладная информатика в юриспруденции

Уровень высшего образования:
бакалавриат

Квалификация выпускника:
«бакалавр»

Волжский 2026 г.

Содержание

1	Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы, описание показателей, критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания.....	3
2	Методические материалы и типовые контрольные задания, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы.....	12
2.1	Типовые контрольные задания, используемые для проведения входного контроля.....	12
2.2	Типовые контрольные задания, используемые для промежуточной аттестации по дисциплине.....	13
2.2.1	Примерный перечень вопросов к экзамену.....	13
2.2.2	Примерное задание на экзамен.....	14
2.3	Методические материалы и типовые контрольные задания, используемые для текущего контроля по дисциплине.....	14
2.3.1	Методические материалы, используемые для текущего контроля знаний по дисциплине.....	14
2.3.2	Вопросы, выносимые на самостоятельное изучение.....	16
2.3.3	Задания для самостоятельной работы.....	16
2.3.4	Тесты по дисциплине.....	16
2.3.5	Типовые задания.....	18
2.3.6	Задания для контрольной работы.....	19
2.3.7	Методика проведения лекции-дискуссии.....	19
2.3.8	Методика проведения интерактивного решения задач.....	20
3	Методические материалы, определяющие процедуры оценивания знаний, умений, навыков..... по дисциплине.....	21
3.1	Балльно-рейтинговая система оценки успеваемости по дисциплине.....	21

1 Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы, описание показателей, критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Перечень компетенций	Показатели оценивания компетенций	Критерии оценивания компетенций	Этапы формирования компетенций	Шкала оценивания
ПК-1. Способен осуществлять руководство разработкой программного кода	- знает принципы ООП (инкапсуляция, наследование, полиморфизм, абстракция); методологии и парадигмы проектирования; стандарты оформления кода; жизненный цикл ПО; - умеет проектировать архитектуру ПО на основе ООП; выбирать оптимальные паттерны проектирования; оценивать качество программного кода; организовывать работу команды разработчиков; - владеет инструментами проектирования (UML, диаграммы); методами рефакторинга; техниками code review; средами разработки (IDE)	- имеет базовые знания принципов ООП и методологий проектирования; - умеет на основе типовых примеров проектировать простые классы и отношения между ними; - владеет навыками применения базовых принципов ООП при написании кода	Начальный	удовлетворительно (60–74 балла)
	- знает принципы ООП (инкапсуляция, наследование, полиморфизм, абстракция); методологии и парадигмы проектирования; стандарты	- знает принципы ООП и методологии проектирования; - умеет проектировать архитектуру средних по	Основной	хорошо (75–89 баллов)

	оформления кода; жизненный цикл ПО; - умеет проектировать архитектуру ПО на основе ООП; выбирать оптимальные паттерны проектирования; оценивать качество программного кода; организовывать работу команды разработчиков; - владеет инструментами проектирования (UML, диаграммы); методами рефакторинга; техниками code review; средами разработки (IDE)	сложности приложений с применением паттернов проектирования; - владеет навыками самостоятельного применения принципов ООП и рефакторинга кода		
	- знает принципы ООП (инкапсуляция, наследование, полиморфизм, абстракция); методологии и парадигмы проектирования; стандарты оформления кода; жизненный цикл ПО; - умеет проектировать архитектуру ПО на основе ООП; выбирать оптимальные паттерны проектирования; оценивать качество программного кода; организовывать работу команды разработчиков; - владеет инструментами проектирования (UML, диаграммы);	- знает принципы ООП и методологии проектирования; - умеет проектировать архитектуру сложных приложений, оптимизировать структуру кода, руководить командой разработчиков; - владеет опытом применения передовых практик ООП и организации процесса разработки	Завершающий	отлично (90–100 баллов)

	методами рефакторинга; техниками code review; средами разработки (IDE)			
ПК-2. Способен осуществлять руководство проверкой работоспособности компьютерного программного обеспечения	- знает методы и виды тестирования ПО; принципы модульного тестирования; инструменты автоматизации тестирования; - умеет разрабатывать тестовые сценарии для ООП-приложений; организовывать процесс тестирования; анализировать результаты тестирования и устранять дефекты; - владеет фреймворками тестирования (JUnit, NUnit); методиками отладки; инструментами профилирования и анализа производительности	- имеет базовые знания методов тестирования; - умеет на основе типовых примеров разрабатывать простые тестовые сценарии; - владеет навыками применения базовых инструментов тестирования	Начальный	удовлетворительно (60–74 балла)
	- знает методы и виды тестирования ПО; принципы модульного тестирования; инструменты автоматизации тестирования; - умеет разрабатывать тестовые сценарии для ООП-приложений; организовывать процесс тестирования; анализировать результаты тестирования и устранять дефекты;	- знает методы и виды тестирования; - умеет разрабатывать комплексные тестовые сценарии для ООП-приложений; - владеет навыками самостоятельного применения фреймворков	Основной	хорошо (75–89 баллов)

	- владеет фреймворками тестирования (JUnit, NUnit); методиками отладки; инструментами профилирования и анализа производительности	тестирования и отладки		
	- знает методы и виды тестирования ПО; принципы модульного тестирования; инструменты автоматизации тестирования; - умеет разрабатывать тестовые сценарии для ООП-приложений; организовывать процесс тестирования; анализировать результаты тестирования и устранять дефекты; - владеет фреймворками тестирования (JUnit, NUnit); методиками отладки; инструментами профилирования и анализа производительности	- знает методы и виды тестирования; - умеет организовывать процесс тестирования, анализировать результаты и руководить командой тестировщиков; - владеет опытом применения комплексных подходов к обеспечению качества ПО	Завершающий	отлично (90–100 баллов)
ПК-3. Способен руководить интеграцией программных модулей и компонентов компьютерного программного обеспечения	- знает принципы компонентной архитектуры; интерфейсы взаимодействия модулей; методы управления зависимостями; - умеет проектировать интерфейсы взаимодействия модулей; организовывать процесс интеграции; решать конфликты	- имеет базовые знания принципов компонентной архитектуры; - умеет на основе типовых примеров проектировать простые интерфейсы взаимодействия; - владеет навыками	Начальный	удовлетворительно (60–74 балла)

	версий и зависимостей; - владеет технологиями CI/CD; системами контроля версий (Git); инструментами сборки (Maven, Gradle); контейнеризацией (Docker)	применения базовых инструментов контроля версий		
	- знает принципы компонентной архитектуры; интерфейсы взаимодействия модулей; методы управления зависимостями; - умеет проектировать интерфейсы взаимодействия модулей; организовывать процесс интеграции; решать конфликты версий и зависимостей; - владеет технологиями CI/CD; системами контроля версий (Git); инструментами сборки (Maven, Gradle); контейнеризацией (Docker)	- знает принципы компонентной архитектуры; - умеет проектировать интерфейсы взаимодействия и организовывать процесс интеграции; - владеет навыками самостоятельного применения инструментов контроля версий и сборки	Основной	хорошо (75–89 баллов)
	- знает принципы компонентной архитектуры; интерфейсы взаимодействия модулей; методы управления зависимостями; - умеет проектировать интерфейсы взаимодействия модулей; организовывать процесс интеграции; решать конфликты версий и зависимостей; - владеет технологиями CI/CD;	- знает принципы компонентной архитектуры; - умеет руководить процессом интеграции, решать сложные конфликты зависимостей; - владеет опытом применения технологий CI/CD и контейнеризации	Завершающий	отлично (90–100 баллов)

	системами контроля версий (Git); инструментами сборки (Maven, Gradle); контейнеризацией (Docker)			
ПК-4. Способен руководить разработкой проектной и технической документации на компьютерное программное обеспечение	- знает стандарты оформления технической документации (IEEE, ГОСТ); виды проектной документации; нотации моделирования (UML); - умеет разрабатывать спецификации требований; создавать диаграммы классов, последовательностей, состояний; оформлять техническую документацию в соответствии со стандартами; - владеет инструментами визуального моделирования (Enterprise Architect, StarUML); методиками описания архитектуры ПО	- имеет базовые знания стандартов оформления документации; - умеет на основе типовых примеров создавать простые диаграммы классов; - владеет навыками применения базовых инструментов моделирования	Начальный	удовлетворительно (60–74 балла)
	- знает стандарты оформления технической документации (IEEE, ГОСТ); виды проектной документации; нотации моделирования (UML); - умеет разрабатывать спецификации требований; создавать диаграммы классов,	- знает стандарты оформления документации; - умеет разрабатывать спецификации и создавать комплексные диаграммы; - владеет навыками самостоятельного применения инструментов	Основной	хорошо (75–89 баллов)

	последовательностей, состояний; оформлять техническую документацию в соответствии со стандартами; - владеет инструментами визуального моделирования (Enterprise Architect, StarUML); методиками описания архитектуры ПО	моделирования		
	- знает стандарты оформления технической документации (IEEE, ГОСТ); виды проектной документации; нотации моделирования (UML); - умеет разрабатывать спецификации требований; создавать диаграммы классов, последовательностей, состояний; оформлять техническую документацию в соответствии со стандартами; - владеет инструментами визуального моделирования (Enterprise Architect, StarUML); методиками описания архитектуры ПО	- знает стандарты оформления документации; - умеет руководить разработкой полного комплекта проектной документации; - владеет опытом применения современных методик описания архитектуры ПО	Завершающий	отлично (90–100 баллов)
ПК-5. Способен руководить разработкой технических	- знает методологии анализа требований; методы формализации	- имеет базовые знания методологий анализа	Начальный	удовлетворительно

спецификаций компьютерного программного обеспечения	спецификаций; подходы к управлению требованиями; - умеет анализировать и формализовать требования заказчика; разрабатывать технические задания и спецификации; согласовывать спецификации с заинтересованными сторонами; - владеет техниками интервьюирования и анализа; инструментами управления требованиями (Jira, Confluence); методами валидации спецификаций	требований; - умеет на основе типовых примеров формализовать простые требования; - владеет навыками применения базовых техник анализа		(60–74 балла)
	- знает методологии анализа требований; методы формализации спецификаций; подходы к управлению требованиями; - умеет анализировать и формализовать требования заказчика; разрабатывать технические задания и спецификации; согласовывать спецификации с заинтересованными сторонами; - владеет техниками интервьюирования и анализа; инструментами управления	- знает методологии анализа требований; - умеет анализировать и формализовать требования, разрабатывать спецификации; - владеет навыками самостоятельного применения инструментов управления требованиями	Основной	хорошо (75–89 баллов)

	требованиями (Jira, Confluence); методами валидации спецификаций			
	- знает методологии анализа требований; методы формализации спецификаций; подходы к управлению требованиями; - умеет анализировать и формализовать требования заказчика; разрабатывать технические задания и спецификации; согласовывать спецификации с заинтересованными сторонами; - владеет техниками интервьюирования и анализа; инструментами управления требованиями (Jira, Confluence); методами валидации спецификаций	- знает методологии анализа требований; - умеет руководить процессом разработки спецификаций, согласовывать их с заинтересованными сторонами; - владеет опытом применения комплексных подходов к управлению требованиями	Завершающий	отлично (90–100 баллов)

2 Методические материалы и типовые контрольные задания, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

2.1 Типовые контрольные задания, используемые для проведения входного контроля

1. Что такое объектно-ориентированное программирование? а) Парадигма программирования, основанная на использовании функций б) Парадигма программирования, основанная на использовании объектов и классов в) Парадигма программирования, основанная на использовании процедур г) Парадигма программирования, основанная на использовании потоков данных
2. Какой из перечисленных языков программирования НЕ поддерживает ООП? а) Java б) C# в) Python г) C
3. Что такое класс в ООП? а) Экземпляр объекта б) Шаблон для создания объектов в) Метод объекта г) Свойство объекта
4. Что такое наследование в ООП? а) Возможность создания нескольких экземпляров одного класса б) Возможность одного класса получать свойства и методы другого класса в) Возможность изменения поведения метода в дочернем классе г) Возможность сокрытия внутренней реализации класса
5. Что такое полиморфизм в ООП? а) Возможность создания нескольких экземпляров одного класса б) Возможность одного класса получать свойства и методы другого класса в) Возможность объектов разных классов реагировать на один и тот же метод по-разному г) Возможность сокрытия внутренней реализации класса
6. Что такое инкапсуляция в ООП? а) Объединение данных и методов, работающих с этими данными, в одном объекте б) Возможность одного класса получать свойства и методы другого класса в) Возможность объектов разных классов реагировать на один и тот же метод по-разному г) Возможность создания абстрактных классов
7. Какой принцип ООП предполагает, что объект должен предоставлять интерфейс для взаимодействия, скрывая внутреннюю реализацию? а) Наследование б) Полиморфизм в) Инкапсуляция г) Абстракция
8. Что такое интерфейс в ООП? а) Графический интерфейс пользователя б) Контракт, определяющий набор методов, которые должен реализовать класс в) Экземпляр абстрактного класса г) Механизм наследования
9. Какой из перечисленных принципов НЕ относится к принципам SOLID? а) Принцип единственной ответственности б) Принцип открытости/закрытости в) Принцип наследования г) Принцип подстановки Барбары Лисков
10. Что такое абстрактный класс? а) Класс, который нельзя инстанцировать и содержит хотя бы один абстрактный метод б) Класс, который содержит только статические методы в) Класс, который наследуется от интерфейса г) Класс, который не содержит методов
11. Что такое агрегация в диаграммах классов UML? а) Отношение "часть-целое", при котором часть не может существовать без целого б) Отношение "часть-целое", при котором часть может существовать независимо от целого в) Отношение наследования между классами г) Отношение ассоциации между классами
12. Что такое композиция в диаграммах классов UML? а) Отношение "часть-целое", при котором часть не может существовать без целого б) Отношение "часть-целое", при котором часть может существовать независимо от целого в) Отношение наследования между классами г) Отношение ассоциации между классами
13. Какой из перечисленных паттернов проектирования относится к порождающим? а) Адаптер б) Наблюдатель в) Фабрика г) Декоратор
14. Какой из перечисленных паттернов проектирования относится к структурным? а) Строитель б) Адаптер в) Стратегия г) Команда

15. Какой из перечисленных паттернов проектирования относится к поведенческим? а) Одиночка б) Фасад в) Наблюдатель г) Мост
16. Что такое рефакторинг кода? а) Изменение функциональности программы б) Изменение структуры кода без изменения его внешнего поведения в) Оптимизация производительности программы г) Исправление ошибок в коде
17. Какая система контроля версий является распределенной? а) SVN б) CVS в) Git г) Perforce
18. Что такое ветка (branch) в системе контроля версий Git? а) Отдельная линия разработки б) Точка сохранения состояния репозитория в) Удаленный репозиторий г) Файл с изменениями
19. Что такое диаграмма последовательности в UML? а) Диаграмма, показывающая статические отношения между классами б) Диаграмма, показывающая взаимодействие объектов во времени в) Диаграмма, показывающая состояния объекта и переходы между ними г) Диаграмма, показывающая варианты использования системы
20. Что такое диаграмма классов в UML? а) Диаграмма, показывающая статические отношения между классами б) Диаграмма, показывающая взаимодействие объектов во времени в) Диаграмма, показывающая состояния объекта и переходы между ними г) Диаграмма, показывающая варианты использования системы

2.2 Типовые контрольные задания, используемые для промежуточной аттестации по дисциплине

2.2.1 Примерный перечень вопросов к экзамену

Теоретические вопросы

1. Эволюция парадигм программирования: процедурное, объектно-ориентированное, функциональное программирование. Преимущества и недостатки каждой парадигмы.
2. Четыре основных принципа ООП: инкапсуляция, наследование, полиморфизм, абстракция. Примеры реализации на современных языках программирования.
3. Анализ предметной области: методы выявления сущностей, атрибутов, связей. Диаграммы прецедентов (Use Case): акторы, сценарии, отношения.
4. Класс как шаблон объекта. Поля, методы, конструкторы, деструкторы. Модификаторы доступа и их роль в обеспечении инкапсуляции.
5. Наследование: базовые и производные классы, цепочка наследования, множественное наследование и его проблемы.
6. Полиморфизм: статический (перегрузка) и динамический (виртуальные методы, переопределение). Механизмы реализации полиморфизма в различных языках.
7. Интерфейсы и абстрактные классы: сравнение, сценарии применения, преимущества и недостатки каждого подхода.
8. Диаграммы классов UML: элементы нотации, типы отношений (ассоциация, агрегация, композиция, наследование, зависимость).
9. Принципы SOLID: формулировка каждого принципа, примеры нарушений и их устранения, преимущества применения.
10. Рефакторинг кода: определение, цели, основные техники (извлечение метода, замена условного оператора полиморфизмом, введение параметра и др.).
11. Порождающие паттерны проектирования: Фабрика, Абстрактная фабрика, Строитель, Одиночка, Прототип. Сценарии применения.
12. Структурные паттерны проектирования: Адаптер, Декоратор, Фасад, Компоновщик, Мост, Приспособленец, Заместитель. Сценарии применения.
13. Поведенческие паттерны проектирования: Наблюдатель, Стратегия, Команда, Состояние, Шаблонный метод, Цепочка обязанностей. Сценарии применения.

14. Диаграммы последовательности UML: объекты, сообщения, активации, фрагменты взаимодействия.
15. Диаграммы состояний UML: состояния, переходы, события, действия, псевдосостояния.
16. Системы контроля версий: централизованные (SVN) и распределенные (Git). Основные команды Git: clone, add, commit, push, pull, branch, merge.
17. Работа с ветками в Git: стратегии ветвления (Git Flow, GitHub Flow), разрешение конфликтов слияния.
18. Введение в CI/CD: концепция непрерывной интеграции и непрерывной доставки, преимущества автоматизации процессов разработки.
19. Инструменты автоматизации тестирования: фреймворки модульного тестирования (JUnit, NUnit), интеграционного тестирования.
20. Подготовка технической документации: стандарты IEEE, ГОСТ, виды документации (ТЗ, спецификации, архитектурные описания).

Практические вопросы

1. Разработать диаграмму классов для системы управления библиотекой с учетом принципов ООП и отношений между сущностями.
2. Реализовать иерархию классов для геометрических фигур с применением наследования и полиморфизма.
3. Применить паттерн "Стратегия" для реализации различных алгоритмов сортировки в едином интерфейсе.
4. Реализовать паттерн "Наблюдатель" для системы уведомлений пользователей о событиях в приложении.
5. Применить принципы SOLID для рефакторинга "запахов кода" в предоставленном фрагменте программы.
6. Создать диаграмму последовательности для сценария оформления заказа в интернет-магазине.
7. Разработать архитектуру многослойного приложения (представление, бизнес-логика, доступ к данным) с применением паттернов проектирования.
8. Настроить базовый конвейер CI/CD для автоматической сборки и тестирования проекта с использованием GitHub Actions или GitLab CI.
9. Оформить техническое задание и спецификации требований для разработки простого веб-приложения.
10. Проанализировать предоставленный код на соответствие принципам ООП и предложить улучшения с обоснованием.

2.2.2 Примерное задание на экзамен

Задание 1. Опишите принципы SOLID. Приведите примеры нарушения каждого принципа и покажите, как их можно устранить с помощью рефакторинга.

Задание 2. Разработайте архитектуру системы управления задачами (Task Management System) с применением ООП:

- Создайте диаграмму классов с указанием отношений между классами
- Реализуйте иерархию классов с применением наследования и полиморфизма
- Примените не менее двух паттернов проектирования с обоснованием выбора
- Опишите подход к тестированию ключевых компонентов системы

2.3 Методические материалы и типовые контрольные задания, используемые для текущего контроля по дисциплине

2.3.1 Методические материалы, используемые для текущего контроля знаний по дисциплине

Наименова	Краткая характеристика	Представление
-----------	------------------------	---------------

ние оценочного средства	оценочного средства	оценочного средства в фонде
Контрольный опрос	Метод оценки уровня освоения компетенций на основе непосредственного или опосредованного взаимодействия преподавателя и студента	Примерный перечень вопросов к экзамену, вопросы для самостоятельного изучения
Собеседование	Относительно свободный диалог между преподавателем и студентом на заданную тему	Вопросы для самостоятельного изучения, задания для самостоятельной работы
Тестовые задания	Эмпирико-аналитическая процедура оценки уровня освоения компетенций	Тесты по дисциплине (раздел 2.3.4)
Самостоятельное решение задач	Метод, при котором обучающиеся приобретают навыки творческого мышления и самостоятельного решения проблем	Типовые задания (раздел 2.3.5), задания для контрольной работы
Лекция-дискуссия	Оценочное средство, позволяющее включить обучающихся в процесс обсуждения спорного вопроса	Методика проведения лекции-дискуссии (раздел 2.3.7)
Интерактивное решение задач	Метод модерации, при котором при решении задач принимают участие все обучающиеся под руководством преподавателя-модератора	Методика проведения интерактивного решения задач (раздел 2.3.8), типовые задания

2.3.2 Вопросы, выносимые на самостоятельное изучение

Базовые концепции ООП

Сравните парадигмы процедурного и объектно-ориентированного программирования. В каких случаях каждая из них предпочтительнее?

Объясните концепцию инкапсуляции. Как модификаторы доступа способствуют реализации этого принципа?

В чем разница между наследованием реализации и наследованием интерфейса? Приведите примеры.

Объясните разницу между статическим и динамическим полиморфизмом. Как они реализуются в Java/C#?

Проектирование и паттерны 5. Опишите каждый из принципов SOLID с примерами кода на языке программирования по выбору. 6. В чем разница между агрегацией и композицией? Приведите примеры из реальных систем. 7. Сравните паттерны "Фабрика" и "Абстрактная фабрика". Когда следует использовать каждый из них? 8. Опишите паттерн "Декоратор" и его преимущества перед наследованием для расширения функциональности. 9. В чем разница между паттернами "Стратегия" и "Состояние"? Приведите примеры их применения.

Инструменты и методологии 10. Опишите основные команды Git и их назначение. Как организовать работу команды с использованием веток? 11. Что такое конфликт слияния в Git и как его разрешить? 12. Опишите основные элементы диаграммы последовательности UML и их назначение. 13. Какие виды диаграмм состояний существуют в UML? Приведите примеры их применения. 14. Опишите преимущества применения практик CI/CD в процессе разработки ПО.

Блок вопросов для закрепления практических навыков 15. Как выбрать оптимальные паттерны проектирования для решения конкретной задачи? 16. Как провести рефакторинг "запахов кода" (длинный метод, дублирование кода, большая сложность)? 17. Как спроектировать систему с низкой связанностью и высокой связанностью компонентов? 18. Как организовать процесс код-ревью в команде разработчиков? 19. Как оценить качество архитектуры программного обеспечения? 20. Как обеспечить соответствие разрабатываемого ПО требованиям заказчика на всех этапах жизненного цикла?

2.3.3 Задания для самостоятельной работы

Разработать диаграмму классов для системы онлайн-банкинга с указанием отношений между сущностями (клиент, счет, транзакция, карта).

Реализовать иерархию классов для системы обработки геометрических фигур с применением полиморфизма для расчета площади и периметра.

Применить паттерн "Наблюдатель" для реализации системы уведомлений в приложении социальной сети.

Провести рефакторинг предоставленного фрагмента кода с нарушением принципов SOLID, обосновав каждое изменение.

Создать репозиторий Git для учебного проекта, настроить ветвление по модели Git Flow и выполнить несколько операций слияния.

Разработать диаграмму последовательности для сценария авторизации пользователя в веб-приложении.

Подготовить техническое задание для разработки простого мобильного приложения "Список задач" с указанием функциональных и нефункциональных требований.

Реализовать паттерн "Стратегия" для поддержки различных способов оплаты в интернет-магазине.

2.3.4 Тесты по дисциплине

Что такое объектно-ориентированное программирование? а) Парадигма программирования, основанная на использовании функций б) Парадигма программирования, основанная на использовании объектов и классов в) Парадигма программирования, основанная на использовании процедур г) Парадигма программирования, основанная на использовании потоков данных

Какой из перечисленных языков программирования НЕ поддерживает ООП?

а) Java б) C# в) Python г) C

Что такое класс в ООП?

а) Экземпляр объекта б) Шаблон для создания объектов в) Метод объекта г) Свойство объекта

Что такое наследование в ООП?

а) Возможность создания нескольких экземпляров одного класса б) Возможность одного класса получать свойства и методы другого класса в) Возможность изменения

поведения метода в дочернем классе г) Возможность сокрытия внутренней реализации класса

Что такое полиморфизм в ООП?

а) Возможность создания нескольких экземпляров одного класса б) Возможность одного класса получать свойства и методы другого класса в) Возможность объектов разных классов реагировать на один и тот же метод по-разному г) Возможность сокрытия внутренней реализации класса

Что такое инкапсуляция в ООП?

а) Объединение данных и методов, работающих с этими данными, в одном объекте б) Возможность одного класса получать свойства и методы другого класса в) Возможность объектов разных классов реагировать на один и тот же метод по-разному г) Возможность создания абстрактных классов

Какой принцип ООП предполагает, что объект должен предоставлять интерфейс для взаимодействия, скрывая внутреннюю реализацию?

а) Наследование б) Полиморфизм в) Инкапсуляция г) Абстракция

Что такое интерфейс в ООП?

а) Графический интерфейс пользователя б) Контракт, определяющий набор методов, которые должен реализовать класс в) Экземпляр абстрактного класса г) Механизм наследования

Какой из перечисленных принципов НЕ относится к принципам SOLID?

а) Принцип единственной ответственности б) Принцип открытости/закрытости в) Принцип наследования г) Принцип подстановки Барбары Лисков

Что такое абстрактный класс?

а) Класс, который нельзя инстанцировать и содержит хотя бы один абстрактный метод б) Класс, который содержит только статические методы в) Класс, который наследуется от интерфейса г) Класс, который не содержит методов

Что такое агрегация в диаграммах классов UML?

а) Отношение "часть-целое", при котором часть не может существовать без целого б) Отношение "часть-целое", при котором часть может существовать независимо от целого в) Отношение наследования между классами г) Отношение ассоциации между классами

Что такое композиция в диаграммах классов UML?

а) Отношение "часть-целое", при котором часть не может существовать без целого б) Отношение "часть-целое", при котором часть может существовать независимо от целого в) Отношение наследования между классами г) Отношение ассоциации между классами

Какой из перечисленных паттернов проектирования относится к порождающим?

а) Адаптер б) Наблюдатель в) Фабрика г) Декоратор

Какой из перечисленных паттернов проектирования относится к структурным?

а) Строитель б) Адаптер в) Стратегия г) Команда

Какой из перечисленных паттернов проектирования относится к поведенческим?

а) Одиночка б) Фасад в) Наблюдатель г) Мост

Что такое рефакторинг кода?

а) Изменение функциональности программы б) Изменение структуры кода без изменения его внешнего поведения в) Оптимизация производительности программы г) Исправление ошибок в коде

Какая система контроля версий является распределенной?

а) SVN б) CVS в) Git г) Perforce

Что такое ветка (branch) в системе контроля версий Git?

а) Отдельная линия разработки б) Точка сохранения состояния репозитория в) Удаленный репозиторий г) Файл с изменениями

Что такое диаграмма последовательности в UML?

а) Диаграмма, показывающая статические отношения между классами б) Диаграмма, показывающая взаимодействие объектов во времени в) Диаграмма, показывающая состояния объекта и переходы между ними г) Диаграмма, показывающая варианты использования системы

Что такое диаграмма классов в UML?

а) Диаграмма, показывающая статические отношения между классами б) Диаграмма, показывающая взаимодействие объектов во времени в) Диаграмма, показывающая состояния объекта и переходы между ними г) Диаграмма, показывающая варианты использования системы

Какой принцип SOLID гласит: "Программные сущности должны быть открыты для расширения, но закрыты для модификации"?

а) Принцип единственной ответственности б) Принцип открытости/закрытости в) Принцип подстановки Лисков г) Принцип разделения интерфейса

Какой паттерн проектирования позволяет создавать семейства связанных объектов без привязки к конкретным классам?

а) Фабрика б) Абстрактная фабрика в) Строитель г) Прототип

Какой паттерн проектирования позволяет добавлять новую функциональность объекту динамически?

а) Адаптер б) Декоратор в) Фасад г) Заместитель

Какой паттерн проектирования определяет семейство алгоритмов, инкапсулирует каждый из них и делает их взаимозаменяемыми?

а) Стратегия б) Команда в) Состояние г) Наблюдатель

Что такое "запах кода"?

а) Синтаксическая ошибка в коде б) Признак потенциальной проблемы в структуре кода в) Ошибка времени выполнения г) Предупреждение компилятора

2.3.5 Типовые задания

Задача 1: Разработка архитектуры системы управления библиотекой

Описание задачи: Разработать архитектуру системы управления библиотекой с применением принципов ООП и паттернов проектирования.

Задание:

1. Проведите анализ предметной области: выделите основные сущности (книга, читатель, библиотекарь, выдача, возврат).
2. Создайте диаграмму классов с указанием атрибутов, методов и отношений между классами.
3. Реализуйте иерархию классов с применением наследования (например, для различных типов изданий).
4. Примените паттерн "Фасад" для упрощения взаимодействия с подсистемами библиотеки.
5. Примените паттерн "Наблюдатель" для уведомления читателей о возврате книг.
6. Разработайте набор модульных тестов для ключевых операций системы.

Критерии оценки:

1. Полнота и корректность анализа предметной области
2. Качество диаграммы классов и соответствие принципам ООП
3. Обоснованность выбора и корректность применения паттернов проектирования
4. Качество реализации и соответствие принципам SOLID
5. Полнота и качество тестового покрытия

Задача 2: Рефакторинг устаревшего кода

1. Описание задачи: Провести рефакторинг фрагмента кода с множеством "запахов кода" для приведения его в соответствие с принципами ООП и SOLID.
2. Задание:
3. Проанализируйте предоставленный код и выявите "запахи кода" (длинные методы, дублирование, большая сложность и т.д.).
4. Определите нарушения принципов SOLID.
5. Разработайте план рефакторинга с указанием последовательности изменений.
6. Проведите рефакторинг с применением соответствующих техник (извлечение метода, замена условного оператора полиморфизмом и др.).
7. Напишите модульные тесты для проверки сохранения функциональности после рефакторинга.
8. Подготовьте отчет с обоснованием каждого изменения.
9. Критерии оценки:
10. Глубина анализа исходного кода и выявление проблем
11. Обоснованность плана рефакторинга
12. Качество проведенного рефакторинга и соответствие принципам ООП
13. Сохранение функциональности после рефакторинга (подтверждено тестами)
14. Качество документирования изменений

2.3.6 Задания для контрольной работы**2.3.6.1 Задание к теоретической части контрольной работы**

Задание 1: Опишите принципы SOLID. Для каждого принципа:

- Приведите формулировку
- Покажите пример нарушения на языке программирования по выбору
- Предложите решение проблемы с обоснованием

Задание 2: Сравните паттерны проектирования "Стратегия" и "Состояние":

- Опишите назначение каждого паттерна
- Приведите структурные диаграммы
- Покажите реализацию на примере
- Укажите ключевые различия и сценарии применения

2.3.6.2 Задание к практической части контрольной работы

Задача 1: Реализация паттерна "Декоратор"

Разработать систему оформления текста с применением паттерна "Декоратор":

1. Создайте базовый интерфейс TextComponent с методом render()
2. Реализуйте конкретный компонент SimpleText
3. Создайте абстрактный декоратор TextDecorator
4. Реализуйте конкретные декораторы: BoldDecorator, ItalicDecorator, UnderlineDecorator
5. Продемонстрируйте возможность комбинирования декораторов

Задача 2: Работа с системой контроля версий Git

1. Создайте локальный репозиторий Git
2. Создайте файл README.md и зафиксируйте изменения
3. Создайте ветку feature/login и переключитесь на нее
4. Добавьте файл auth.js с простой реализацией аутентификации
5. Зафиксируйте изменения в ветке feature/login
6. Вернитесь в основную ветку и выполните слияние с веткой feature/login
7. Разрешите конфликт слияния (искусственно созданный)
8. Подготовьте отчет о выполненных операциях с указанием команд Git

2.3.7 Методика проведения лекции-дискуссии

Цель: Достижение определенного мнения по обсуждаемой проблеме путем осознания участниками своих мнений, суждений, оценок по теме дискуссии; уточнения взаимных позиций в споре; выработки уважительного отношения к другому мнению.

Задачи:

- Развитие умений анализировать проблему или проблемную ситуацию
- Поиск и разработка перспективных идей
- Развитие способности продуцировать множество решений
- Нахождение оптимального решения среди нескольких альтернатив

Методика проведения:

- Назначение секретаря лекции-дискуссии, его инструктаж по выполняемым функциям
- Объявление критериев оценки
- Свободный обмен мнениями в промежутках между логически оформленными разделами сообщения учебного материала
- Подведение итогов дискуссии и оценка участников дискуссии по материалам, подготовленным секретарем, переход к информационной лекции

Критерии оценки: секретарь лекции-дискуссии – 0,5–1,5 баллов; участие в дискуссии – 0,5–1 балл.

2.3.8. Методика проведения интерактивного решения задач

Цель: Проверка навыков решения профессиональных задач и развитие мыслительных операций обучающихся, направленных на достижение результатов при решении профессиональных задач.

Задачи:

1. Проверка навыков применения обучающимися ранее усвоенных знаний при решении профессиональных задач
2. Формирование навыков совместной деятельности подчиненных (обучающихся) и руководителя (преподавателя)
3. Овладение обучающимися знаниями и общими принципами решения проблемных профессиональных задач
4. Развитие навыков активной интеллектуальной деятельности
5. Развитие коммуникативных навыков (навыков общения)
6. Развитие навыков обучающихся по выработке собственной позиции по ходу решения профессиональных задач

Методика проведения интерактивного решения задач: Первый этап «подготовка проекта решения задач». Преподавателем разрабатывается проект хода решения задачи с началом или фрагментами решения.

Второй этап «организационный»:

1. Объявление темы и цели решения задачи
2. Назначение секретаря и его инструктаж по выполняемым функциям
3. Объявление критериев оценки

Третий этап «интеллектуальный»:

1. Объявление условий решения задач
2. Индивидуальное решение задачи обучающимися, исходя из собственного мнения
3. Высказывание обучающимися мнений по ходу решения задач
4. Обсуждение результатов и методики индивидуального решения задач обучающимися и принятие плана верного хода решения

Четвертый этап «подведение итогов решения задачи»:

1. Формулирование вывода решения задачи обучающимися
2. Подведение итога интерактивного решения задачи преподавателем
3. Оценка преподавателем обучающихся по материалам, подготовленным секретарем

Критерии оценки: секретарь – 1 балл; участие в решении задач – 1–3 балла.

3 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков по дисциплине

3.1 Балльно-рейтинговая система оценки успеваемости по дисциплине

В целях оперативного контроля уровня усвоения материала учебной дисциплины и стимулирования активной учебной деятельности студентов (очной формы обучения) используется балльно-рейтинговая система оценки успеваемости. В соответствии с этой системой оценки студенту в ходе изучения дисциплины предоставляется возможность набрать не менее 60 баллов за текущую работу в семестре и до 40 баллов на экзамене (итоговый контроль). Результирующая оценка по дисциплине складывается из суммы баллов текущего и итогового контроля.

К экзамену допускаются студенты, посетившие все лекции и практические занятия, прошедшие промежуточную аттестацию, с рейтингом не менее 30 баллов.

Баллы за текущую работу в семестре по дисциплине складываются из следующих видов деятельности студента:

Балльно-рейтинговая система по дисциплине «Объектно-ориентированное проектирование и программирование»

Виды контроля	Максимальная сумма баллов на выполнение одного вида задания	Всего возможных за семестр работ	Итого максимальная сумма баллов
Присутствие на лекционных и практических занятиях	0,12	72	9
Работа на практических занятиях	1,0	36	36
СРС, в т.ч.:			55
– написание конспектов	0,1	10	1
– защита рефератов	0,3	7	2
– выступление с докладом	0,2	5	1
– самостоятельное решение задач	1	12	12
– написание эссе	0,2	4	8
– контроль СРС (контрольные опросы, тестирование, решение кейсов и др. виды контроля)	0,8	10	8
– выполнение лабораторных работ	2	10	20
– участие в интерактивных занятиях	1	3	3
Итого за семестр	–	–	100
Экзамен	40,0	1	40
Итого по дисциплине	–	–	140

Примечание: Максимальная итоговая оценка – 100 баллов (баллы свыше 100 учитываются как 100).

Текущий контроль по лекционному материалу проводит лектор. Текущий контроль по практическим занятиям проводит преподаватель, проводивший эти занятия.

Балльно-рейтинговая система предусматривает поощрение студентов за работу в семестре проставлением «премиальных» баллов.

Нормы выставления дополнительных баллов по дисциплине «Объектно-ориентированное проектирование и программирование»

Виды работ	Максимальная сумма баллов
Решение дополнительных задач повышенной сложности	5
Участие в олимпиадах, конкурсах, конференциях в рамках предмета дисциплины	5
Публикация научных работ по тематике дисциплины	5
Итого	15

Шкала итоговой оценки

Количество баллов	Оценка
91–100	«отлично»
76–90	«хорошо»
60–75	«удовлетворительно»
0–59	«неудовлетворительно»