МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
И.о. директора института
перспективной инженерии,
канд. техн. наук, доцент
А.А. Порохня

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

для проведения текущего контроля успеваемости и промежуточной аттестации по дисциплине

Паттерны проектирования

Направление подготовки	09.03.04 Программная инженерия
Направленность (профиль)	Разработка и сопровождение программного обеспечения
Форма обучения	очная
Год начала обучения	2026
Реализуется в	6 семестре

Введение

1. Назначение: Оценочные материалы (оценочные средства) предназначены для проведения текущего контроля успеваемости и промежуточной аттестации обучающихся, и представляют собой совокупность контрольно-измерительных материалов и методов их использования для измерения уровня достижения обучающимся установленных результатов обучения.

2. ФОС является приложением к программе дисциплины «Паттерны проектирования»

3. Разработчик Потапов И.Р., ассистент межинститутской базовой кафедры департамента цифровых, робототехнических систем и электроники

4. Проведена экспертиза ФОС.

Члены экспертной группы:

Председатель: Новикова Елена Николаевна, зав. межинститутской базовой кафедрой

(ФИО, должность)

Члены комиссии:

Николаев Евгений Иванович, доцент департаменты цифровых и робототехнических систем и электроники.

(ФИО, должность)

Винокурский Д. Л., доцент департамента цифровых, робототехнических систем и электроники

(ФИО, должность)

Представитель организации-работодателя: А. В. Новиков, начальник отдела АСУТП ООО «ЕНДС-Ставрополь»

(ФИО, должность)

Экспертное заключение: Оценочные материалы (оценочные средства) по дисциплине «Паттерны проектирования» составлены в соответствии с требованиями самостоятельно устанавливаемого Федерального государственного образовательного стандарта высшего образования по направлению 09.03.04 «Программная инженерия» и позволяют оценить уровень сформированности компетенции ПК-2.

5. Срок действия ФОС определяется сроком реализации образовательной программы.

1. Описание показателей и критериев оценивания на различных этапах их формирования, описание шкал оценивания

Уровни сформированности компетенци(ий), индикатора (ов)	Дескрипторы			
	Минимальный уровень не достигнут (Неудовлетворительно) 2 балла	Минимальный уровень (удовлетворительно) 3 балла	Средний уровень (хорошо) 4 балла	Высокий уровень (отлично) 5 баллов
Компетенция: **ПК-2** – способен выполнять проектирование и разработку архитектуры программных систем среднего и крупного масштаба с применением современных паттернов и методов проектирования.				
Результаты обучения по дисциплине (модулю): *Индикатор:* ПК-$2_{ИН-3}$ – Применяет порождающие, структурные и поведенческие паттерны GoF, а также архитектурные паттерны (CQRS, Event Sourcing, Saga, Circuit Breaker)	Не знает классификацию и назначение паттернов GoF. Не может перечислить или реализовать ни один из порождающих, структурных или поведенческих паттернов. Не имеет представления об архитектурных паттернах CQRS, Event Sourcing, Saga, Circuit Breaker.	Слабые знания: с помощью наводя-щих вопросов называет отдельные паттерны (например, Singleton, Adapter), но путает их принадлежность к категориям. Может написать простейшую реализацию Singleton или Factory Method, но не объясняет выбор паттерна. Знает названия архитектурных паттернов, но не может описать их суть.	Уверенно применяет основные порождающие, структурные и поведенческие паттерны в типовых задачах. Обосновывает выбор паттерна для конкретной ситуации. Понимает назначение CQRS, Event Sourcing, Saga, Circuit Breaker, может привести примеры их использования. Однако испытывает затруднения при комбинировании нескольких паттернов или при выборе между схожими паттернами (например, Strategy vs State).	Свободно владеет всеми перечисленными паттернами GoF. Способен спроектировать систему, комбинируя порождающие, структурные и поведенческие паттерны для достижения гибкости и расширяемости. Аргументированно выбирает архитектурные паттерны (CQRS, Event Sourcing, Saga, Circuit Breaker) под контекст задачи, критически оценивает их последствия (сложность, согласованность, производительность).
Результаты обучения по дисциплине (модулю): *Индикатор* ПК-$2_{ИД-4}$ – Реализует многослойную архитектуру (Clean	Не имеет представления о многослойной архитектуре, Dependency Inversion, Clean Architecture, Hexagonal,	Знает общие принципы многослойности и Dependency Inversion, но не может реализовать чистую	Реализует базовую структуру Clean Architecture или Hexagonal в учебном проекте (слои Domain, Application,	Проектирует и реализует многослойную архитектуру любого масштаба, обеспечивая полную независимость от

Architecture, Hexagonal, PCMEF), обеспечивая независимость от фреймворков через Dependency Inversion	PCMEF. Не может объяснить, как отделить бизнес-логику от фреймворков.	архитектуру в коде. При попытке создания слоёв допускает жёсткие связи с фреймворками (например, логика внутри контроллера). Может написать простой пример с одним-двумя слоями, но не обеспечивает тестируемость.	Infrastructure, Presentation). Применяет Dependency Inversion через интерфейсы. Способен провести рефакторинг простого монолита для повышения тестируемости. Однако адаптация под масштаб системы (например, перераспределение слоёв для микросервисов) вызывает сложности.	фреймворков и внешних сервисов. Грамотно адаптирует Clean Architecture / Hexagonal под требования (например, добавляет CQRS на уровне Application). Проводит рефакторинг легаси-кода, внедряя Dependency Inversion и изолируя тестируемые модули. Демонстрирует понимание компромиссов между строгостью архитектуры и прагматизмом.
Результаты обучения по дисциплине (модулю): *Индикатор:* ПК-2$_{ИД-5}$ – Применяет стратегический и тактический DDD: выделяет ограниченные контексты (Bounded Contexts), проектирует агрегаты (Aggregate), value objects, доменные события и репозитории	Не знает понятий DDD (Bounded Context, Aggregate, Value Object, Domain Event, Repository). Не может назвать анти-паттерны God Object, Lava Flow, Inner Platform. Не понимает сущности технического долга.	Даёт формальные определения ограниченного контекста и агрегата, но не может выделить их в реальной предметной области. Пытается проектировать Value Object или репозиторий, но допускает концептуальные ошибки (например, хранит ссылки между агрегатами напрямую). Распознаёт очевидные случаи God Object, но	Самостоятельно выделяет ограниченные контексты в домене средней сложности (3–5 контекстов). Проектирует агрегаты с корректными границами согласованности, использует Value Objects и доменные события. Применяет репозитории для доступа к агрегатам. Идентифицирует технический долг и предлагает плановую замену паттернов	Глубокое владение стратегическим и тактическим DDD. Выделяет контексты с учётом отношений (Customer/Supplier, Conformist, Anti-Corruption Layer). Проектирует событийно-ориентированные агрегаты, эволюционирует модель через Domain Events. Проводит оценку технического долга с метриками (например,

		не способен предложить рефакторинг.	(например, замена Transaction Script на DDD). Устраняет простые анти-паттерны (например, разбивает God Object).	цикломатическая сложность, связность). Рефакторит код, заменяя анти-паттерны на паттерны DDD и внедряя Eventual Consistency. Обучает команду принципам DDD.
Результаты обучения по дисциплине (модулю): *Индикатор:* ПК-2$_{ИД-7}$ – Разрабатывает архитектуру интеграции решений с использованием API-шлюзов, брокеров сообщений (Kafka, RabbitMQ) и паттернов интеграции	Не знает назначения API-шлюзов, брокеров сообщений. Не может описать Event Bus, Saga, CQRS на монолите, Event Sourcing. Не имеет опыта работы с интеграционными паттернами.	Понимает роль брокера сообщений (например, очереди) для асинхронной передачи данных, но не может спроектировать Saga или восстановление состояния через Event Sourcing. Может настроить простой обмен событиями через RabbitMQ (публикация/подписка) без гарантий доставки. CQRS на монолите реализует только как разделение Command и Query, но не обеспечивает синхронизацию.	Разрабатывает схему интеграции с использованием API-шлюза для маршрутизации и аутентификации. Настраивает Kafka/RabbitMQ с обработкой ошибок и at-least-once delivery. Реализует Saga типа «хореография» для компенсирующих транзакций. Применяет Event Sourcing с сохранением событий и восстановлением состояния. Внедряет Event Bus внутри монолита для слабой связанности модулей. Однако испытывает трудности с выбором между choreography и orchestration для Saga, а также с оптимизацией восстановления	Проектирует отказоустойчивую интеграционную архитектуру на основе API-шлюза (например, Kong, NGINX) и брокеров сообщений с exactly-once семантикой и идемпотентностью. Реализует сложные сценарии: Saga orchestration с компенсирующими действиями, CQRS на монолите с синхронизацией через Domain Events, Event Sourcing с Snapshot’ами и индексированием. Проводит анализ компромиссов между синхронным и асинхронным взаимодействием, выбирает оптимальные паттерны

			при большом количестве событий.	интеграции для заданных нефункциональных требований (масштабируемость, консистентность, доступность).

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОВЕРКИ УРОВНЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Номер задания	Правильный ответ	Содержание вопроса	Компетенция
1.	2	Какой паттерн проектирования относится к порождающим? 1) Decorator, 2) Factory Method, 3) Strategy, 4) Observer	ПК-2
2.	1,3	Какие анти-паттерны характерны для legacy-кода с «божественным объектом»? 1) God Object, 2) Lava Flow, 3) длинный метод, 4) Inner Platform	ПК-2
3.	3	Какой паттерн позволяет динамически добавлять объекту новые обязанности без наследования? 1) Proxy, 2) Composite, 3) Decorator, 4) Adapter	ПК-2
4.	2	Для решения проблемы «телескопических конструкторов» чаще всего применяют: 1) Factory Method, 2) Builder, 3) Prototype, 4) Singleton	ПК-2
5.	1,4	Какие паттерны относятся к структурным? 1) Composite, 2) Command, 3) State, 4) Proxy	ПК-2

6.	3	Какой паттерн позволяет менять поведение объекта при изменении его внутреннего состояния? 1) Strategy, 2) Command, 3) State, 4) Observer	ПК-2
7.	2	В чём основное отличие паттерна Strategy от State? 1) Strategy изменяет поведение автоматически, State – по запросу клиента, 2) State сам управляет переходами, Strategy – клиент, 3) Strategy применяется только к алгоритмам, State – к интерфейсам, 4) различий нет	ПК-2
8.	1,3	Какие паттерны используются для инкапсуляции запросов и поддержки отмены действий? 1) Command, 2) Chain of Responsibility, 3) MacroCommand, 4) Observer	ПК-2
9.	4	Паттерн Chain of Responsibility предназначен для: 1) построения дерева объектов, 2) добавления поведения во время выполнения, 3) создания семейств взаимосвязанных объектов, 4) передачи запроса по цепочке обработчиков	ПК-2
10.	2	Какой принцип Clean Architecture нарушает код, где бизнес-логика содержит прямые SQL-запросы? 1) принцип подстановки Барбары Лисков, 2) принцип инверсии зависимостей, 3) принцип разделения интерфейсов, 4) принцип единственной ответственности	ПК-2
11.	1,3	Какие компоненты являются частью паттерна Repository? 1) интерфейс репозитория, 2) Unit of Work, 3) конкретная реализация (in-memory, Jdbc), 4) Event Bus	ПК-2

12.	2	Паттерн Unit of Work предназначен для: 1) кэширования данных, 2) отслеживания изменений и групповой фиксации, 3) отложенной загрузки, 4) логирования запросов	ПК-2
13.	3	Какой тип Test Double используется для проверки взаимодействий (был ли вызван метод с нужными аргументами)? 1) Stub, 2) Fake, 3) Mock, 4) Dummy	ПК-2
14.	1,4	Какие утверждения о паттерне Observer верны? 1) обеспечивает слабую связанность издателя и подписчиков, 2) подписчики должны знать друг о друге, 3) используется только в GUI, 4) Event Bus является вариацией Observer	ПК-2
15.	4	CQRS (Command Query Responsibility Segregation) предполагает: 1) использование одной модели для команд и запросов, 2) обязательное использование Event Sourcing, 3) разделение на микросервисы, 4) разделение моделей записи и чтения	ПК-2
16.	3	В чём преимущество CQRS при сложных запросах? 1) упрощается бизнес-логика команд, 2) уменьшается количество классов, 3) можно оптимизировать read model под конкретные запросы, 4) повышается производительность команд	ПК-2
17.	2	Event Sourcing хранит: 1) только текущее состояние агрегата, 2) последовательность всех событий, приведших к текущему состоянию, 3) снапшоты вместо событий, 4) только последние 100 событий	ПК-2
18.	1	Снапшот (snapshot) в Event Sourcing нужен для:	ПК-2

		1) ускорения восстановления состояния агрегата, 2) уменьшения количества событий в хранилище, 3) обеспечения безопасности, 4) реализации CQRS	
19.	3	В паттерне Saga хореография (choreography) подразумевает: 1) наличие центрального координатора, 2) двухфазную фиксацию, 3) обмен событиями между сервисами без центрального управления, 4) синхронные вызовы	ПК-2
20.	2	Компенсирующая транзакция в Saga – это: 1) повтор неудачной операции, 2) действие, отменяющее эффект предыдущей локальной транзакции, 3) сохранение промежуточного состояния, 4) блокировка ресурсов	ПК-2
21.	1	Какое состояние отсутствует в классическом Circuit Breaker? 1) PENDING, 2) CLOSED, 3) OPEN, 4) HALF-OPEN	ПК-2
22.	4	При переходе Circuit Breaker из OPEN в HALF-OPEN происходит: 1) немедленное восстановление всех вызовов, 2) увеличение порога ошибок, 3) сброс таймера, 4) отправка пробных вызовов для проверки доступности сервиса	ПК-2
23.	2	Паттерн Object Mother используется для: 1) внедрения зависимостей, 2) централизованного создания тестовых объектов с предустановками, 3) реализации одиночки, 4) построения сложных объектов по шагам	ПК-2
24.	1,3	Какие паттерны помогают устранить длинные цепочки if-else? 1) Strategy, 2) Decorator, 3) State,	ПК-2

		4) Builder	
25.	2	Анти-паттерн «Lava Flow» характеризуется: 1) классом, который делает слишком много, 2) наличием мёртвого, непонятного кода, который боятся удалить, 3) глубокой иерархией наследования, 4) дублированием кода	ПК-2
26.	3	Какое из утверждений о паттерне Factory Method верно? 1) создаёт семейства продуктов, 2) всегда требует абстрактную фабрику, 3) определяет интерфейс создания объекта, оставляя подклассам решение о том, какой класс инстанцировать, 4) является структурным паттерном	ПК-2
27.	2	Паттерн Composite позволяет: 1) динамически изменять алгоритм, 2) единообразно работать с простыми и составными объектами, 3) добавлять новое поведение без наследования, 4) разделять создание и использование объекта	ПК-2
28.	1,4	Какие паттерны относятся к поведенческим? 1) Chain of Responsibility, 2) Proxy, 3) Builder, 4) Command	ПК-2
29.	3	Retry с экспоненциальной задержкой применяется для: 1) мгновенного повторения сбоящих вызовов, 2) защиты от лавинной нагрузки, 3) увеличения шансов на успех при временных сбоях с щадящей нагрузкой на систему, 4) ускорения ответа	ПК-2
30.	2,4	Какие паттерны тестируемости помогают создавать тестовые объекты? 1) Test Double, 2) Object Mother, 3) Dependency Injection,	ПК-2

		4) Test Data Builder	
31.	4	Какой паттерн позволяет отделить абстракцию от реализации, чтобы они могли изменяться независимо? 1) Adapter, 2) Facade, 3) Decorator, 4) Bridge	ПК-2
32.	1	Паттерн Facade предназначен для: 1) предоставления упрощённого интерфейса к сложной подсистеме, 2) преобразования интерфейса одного класса в интерфейс другого, 3) динамического добавления обязанностей, 4) создания цепочки обработчиков	ПК-2
33.	3	Какое из утверждений о паттерне Singleton является верным? 1) всегда нарушает принцип единственной ответственности, 2) легко тестируется, 3) может быть потокобезопасным с двойной проверкой блокировки, 4) не может быть реализован на Java	ПК-2
34.	2,4	Какие паттерны позволяют отделить бизнес-логику от инфраструктурных деталей? 1) Singleton, 2) Ports & Adapters, 3) Template Method, 4) Dependency Injection	ПК-2
35.	1	В чём основное отличие Mock от Stub? 1) Mock проверяет взаимодействия (verify), Stub только возвращает предопределённые ответы, 2) Stub работает только с базами данных, 3) Mock не может возвращать значения, 4) различий нет	ПК-2
36.	3	Какой тип Test Double является полноценной рабочей реализацией, но упрощённой (например, in-memory БД)? 1) Mock,	ПК-2

		2) Stub, 3) Fake, 4) Spy	
37.	1	Паттерн Proxy отличается от Decorator тем, что: 1) Proxy контролирует доступ к объекту, часто создавая его лениво, а Decorator добавляет поведение, 2) Proxy не реализует тот же интерфейс, 3) Decorator не может быть вложенным, 4) Proxy изменяет интерфейс	ПК-2
38.	2,4	Какие утверждения о паттерне Template Method верны? 1) является структурным паттерном, 2) определяет скелет алгоритма, оставляя реализацию шагов подклассам, 3) позволяет менять алгоритм во время выполнения, 4) использует принцип «Голливуда» (не вызывай нас, мы вызовем тебя)	ПК-2
39.	3	Паттерн Visitor предназначен для: 1) упрощения иерархии классов, 2) кэширования результатов операций, 3) добавления новых операций к существующим классам без их изменения, 4) управления состояниями конечного автомата	ПК-2
40.	1	Какая из перечисленных проблем решается паттерном Adapter? 1) несовместимость интерфейсов, 2) необходимость создания единственного экземпляра класса, 3) построение древовидных структур, 4) необходимость отмены действий	ПК-2
41.	2	Принцип инверсии зависимостей (DIP) в Clean Architecture требует, чтобы: 1) модули верхнего уровня зависели от модулей нижнего уровня, 2) абстракции не зависят от деталей, а детали зависят от абстракций, 3) использовались только интерфейсы, 4) все классы были final	ПК-2
42.	1,3	Какие компоненты являются частью шестиугольной архитектуры (Ports & Adapters)? 1) доменная модель,	ПК-2

		2) UI-фреймворк, 3) порты (интерфейсы), 4) схема БД	
43.	4	В паттерне Saga оркестрация (orchestration) предполагает: 1) полное отсутствие координатора, 2) каждый сервис самостоятельно решает, что делать, 3) обязательное использование брокера сообщений, 4) наличие центрального компонента, управляющего шагами Saga	ПК-2
44.	2	Для чего в Event Sourcing используется версия агрегата (aggregate version)? 1) для ускорения чтения, 2) для оптимистичной блокировки при конкурентной записи, 3) для шифрования событий, 4) для логирования	ПК-2
45.	1,4	Какие утверждения о Test Data Builder верны? 1) использует Fluent API для установки полей, 2) не может иметь значений по умолчанию, 3) применим только для immutable объектов, 4) улучшает читаемость тестов	ПК-2
46.	3	Паттерн Double-Checked Locking обычно применяется при реализации: 1) Abstract Factory, 2) Prototype, 3) потокобезопасного Singleton, 4) Object Pool	ПК-2
47.	2	Анти-паттерн «Inner Platform» характеризуется: 1) чрезмерной глубиной наследования, 2) попыткой воспроизвести внутри приложения возможности уже используемой платформы, 3) наличием «мёртвого» кода, 4) классом, который делает слишком много	ПК-2
48.	1	Макрокоманда (MacroCommand) в паттерне Command – это: 1) команда, содержащая список других команд и выполняющая их последовательно, 2) команда с поддержкой undo/redo,	ПК-2

		3) команда, которая выполняется в отдельном потоке, 4) команда с параметрами	
49.	3,4	Какие паттерны часто используются вместе с Command для реализации отмены действий? 1) Strategy, 2) Observer, 3) Memento, 4) история команд (стек)	ПК-2
50.	2	Паттерн Interpreter относится к: 1) порождающим паттернам, 2) поведенческим паттернам, 3) структурным паттернам, 4) архитектурным паттернам	ПК-2
51.	1,4	Какие проблемы решает внедрение Event Bus по сравнению с прямыми вызовами? 1) снижение связанности, 2) увеличение производительности, 3) упрощение отладки, 4) возможность легко добавлять новых подписчиков	ПК-2
52.	3	В CQRS синхронизация между Command и Query моделями обычно реализуется через: 1) прямые SQL-запросы, 2) двухфазную фиксацию, 3) доменные события, 4) общую транзакцию	ПК-2
53.	2	Какое из утверждений о паттерне Mediator верно? 1) является разновидностью Composite, 2) уменьшает связанность между взаимодействующими объектами, инкапсулируя их взаимодействие, 3) используется только в GUI, 4) заменяет Observer	ПК-2
54.	1,3	Какие характеристики свойственны хорошему снапшоту (snapshot) в Event Sourcing?	ПК-2

		1) содержит полное состояние агрегата, 2) хранится вместо событий, 3) имеет версию (номер последнего применённого события), 4) должен быть зашифрован	
55.	4	Какой паттерн позволяет динамически выбирать алгоритм обработки во время выполнения без изменения кода клиента? 1) Template Method, 2) State, 3) Command, 4) Strategy	ПК-2
56.	2	Паттерн Iterator используется для: 1) сортировки коллекций, 2) последовательного доступа к элементам агрегата без раскрытия его внутреннего представления, 3) фильтрации данных, 4) кэширования результатов обхода	ПК-2
57.	3	В чём отличие паттерна Flyweight от других структурных паттернов? 1) используется для построения деревьев, 2) преобразует интерфейс, 3) разделяет общее состояние между множеством объектов для экономии памяти, 4) добавляет обязанности динамически	ПК-2
58.	1,2	Какие принципы SOLID напрямую связаны с паттерном Dependency Injection? 1) Dependency Inversion Principle, 2) Single Responsibility Principle, 3) Liskov Substitution Principle, 4) Interface Segregation Principle	ПК-2
59.	4	Анти-паттерн «Stairway to Heaven» (Лестница в небо) – это: 1) божественный объект, 2) лавовый поток, 3) дублирование кода, 4) чрезмерно длинная цепочка вызовов или наследования	ПК-2
60.	2	Какое требование предъявляется к компенсирующим транзакциям в Saga? 1) должны быть быстрее основных,	ПК-2

		2) должны быть идемпотентными, 3) должны использовать двухфазную фиксацию, 4) не могут быть отменены	

2. Описание шкалы оценивания

Оценка успеваемости студентов по дисциплине осуществляется в ходе текущего контроля и промежуточной аттестации. Для проверки уровня сформированности компетенций используется совокупность оценочных средств, позволяющих оценить знания, умения и навыки, приобретённые студентами в процессе изучения дисциплины.

Основными формами текущего контроля являются:

– выполнение и защита лабораторных работ;

– устные опросы (собеседования) по темам занятий;

– проверка индивидуальных заданий;

– тестирование по отдельным разделам дисциплины.

Промежуточная аттестация проводится в форме экзамена и включает выполнение практического задания и/или ответы на теоретические вопросы.

3. Критерии оценивания компетенций заданий открытого типа

Оценка **«отлично»** выставляется студенту, если имеются результаты полного демонстрирования умений применять паттерны проектирования (GoF, архитектурные, интеграционные), проводить рефакторинг legacy-кода, обосновывать выбор конкретных паттернов с анализом компромиссов (trade-offs) и технического долга. Представленный отчёт не содержит ошибок, все задания выполнены верно, код соответствует принципам чистой архитектуры и покрыт тестами. Компетенция ПК-2 освоена на высоком уровне.

Оценка **«хорошо»** выставляется студенту, если имеются результаты демонстрирования умений применять основные паттерны проектирования, проводить рефакторинг и обосновывать выбор паттернов. Представленный отчёт содержит незначительные ошибки, отсутствуют некоторые определения, выводы или фрагменты кода, но это не влияет на общее понимание решения. Компетенция ПК-2 освоена на среднем уровне.

Оценка **«удовлетворительно»** выставляется студенту, если имеются результаты демонстрирования базовых умений применять паттерны, но допущены ошибки в выборе или реализации, недостаточно проработан анализ компромиссов. Представленный отчёт содержит ошибки, не позволяющие в полной мере оценить качество разработки, ключевые определения и выводы представлены не в полном объёме. Компетенция ПК-2 освоена на минимальном уровне.

Оценка **«неудовлетворительно»** выставляется студенту, если отсутствуют результаты демонстрирования умений применять паттерны проектирования, рефакторинг не проведён, выбор паттернов не обоснован. Представленный отчёт содержит много грубых ошибок, отсутствуют ключевые определения, выводы и работающий код. Компетенция ПК-2 не сформирована.

4. Критерии оценивания компетенций заданий закрытого типа

Оценка «отлично» выставляется студенту, если он правильно ответил на 88-100% от общего количества заданий. Компетенция ПК-2 освоена на высоком уровне.

Оценка «хорошо» выставляется студенту, если он правильно ответил на 72-87% от общего количества заданий. Компетенция ПК-2 освоена на среднем уровне.

Оценка «удовлетворительно» выставляется студенту, если он правильно ответил на 53-71% от общего количества заданий. Компетенция ПК-2 освоена на минимальном уровне.

Оценка «неудовлетворительно» выставляется студенту, выставляется студенту, если он правильно ответил на менее 53% от общего количества заданий. Компетенция ПК-8 не сформирована.