Фонд оценочных средств для промежуточной аттестации по дисциплине

Проектирование и разработка распределенных программных систем

Направление подготовки (специальность)

09.03.04 Программная инженерия

Направленность (профиль)

Разработка программно-информационных систем

Присваиваемая квалификация (степень)
бакалавр

Форма обучения
заочная форма обучения

Год(ы) набора 2026

Челябинск 2026 г.

09.03.04 Программная инженерия профиль Разработка программно-информационных систем, дисциплина Проектирование и разработка распределенных программных систем, 2026 год набора, заочная форма обучения

Фонд оценочных средств дисциплины (модуля) одобрен и рекомендован:
Проректор по учебной работе утверждено 27.02.2026 А.А. Саламатов

Ученым советом института информационных технологий

Протокол заседания № 7 от 26.02.2026

Председатель Ученого совета
института информационных
технологий согласовано Ю.В. Петриченко

Заседанием кафедры информационных технологий и экономической информатики

Протокол заседания №7 от 26.02.2026

Заведующий кафедрой согласовано С.А. Скрипов

Автор (составитель) В.А. Мельников

Структура фондов оценочных средств соответствует приказу ректора ФГБОУ ВО «ЧелГУ» от 27 сентября 2022 № 573-1

Содержание

1. Паспорт фонда оценочных средств 3
2. Перечень формируемых компетенций 4
3. Содержание оценочных средств по дисциплине 6
   3.1. Виды оценочных средств 6
   3.2. Содержание оценочных средств 8
4. Порядок проведения и критерии оценивания промежуточной аттестации 29
   4.1. Порядок проведения промежуточной аттестации 29
   4.2. Критерии оценивания промежуточной аттестации по видам оценочных средств 29
   4.3. Результаты промежуточной аттестации и уровни сформированности компетенций 30

1. Паспорт фонда оценочных средств

Направление подготовки: 09.03.04 Программная инженерия
Направленность: Разработка программно-информационных систем
Дисциплина: Проектирование и разработка распределенных программных систем
Курс (курсы) изучения: 3
Форма промежуточной аттестации: экзамен (3 курс), курсовая работа (3 курс).
Для оценивания результатов обучения используется балльно-рейтинговая система.

2. Перечень формируемых компетенций

Изучение дисциплины «Проектирование и разработка распределенных программных систем» направлено на формирование компетенций, приведённых в 1.

Таблица 1. Результаты обучения по дисциплине.

Коды компетенции согласно ФГОС (ОПОП ВО)	Содержание компетенций согласно ФГОС (ОПОП ВО)	Индикаторы достижения компетенции согласно ОПОП	Перечень планируемых результатов обучения по дисциплине
1	2	3	4
ПК-2	Владение навыками использования различных технологий промышленной разработки программного обеспечения с применением инструментов автоматизации сборки, интеграции, тестирования и развертывания ПО	ПК-2.1. Демонстрирует знание основных принципов и технологий промышленной разработки программного обеспечения ПК-2.2. Демонстрирует умения разрабатывать программное обеспечение с применением инструментов автоматизации сборки, интеграции, тестирования и развертывания ПО ПК-2.3. Имеет практический опыт промышленной разработки программного обеспечения	Знать: основные принципы и технологии промышленной разработки распределенных программных систем Уметь: разрабатывать распределенные программные системы с применением инструментов автоматизации сборки, интеграции, тестирования и развертывания ПО Владеть: навыками промышленной разработки распределенных программных систем
ПК-3	Способность проводить тестирование компонентов программного обеспечения и оценивать качество программного обеспечения (надежность, производительность, безопасность, удобство использования)	ПК-3.1. Демонстрирует знание основ тестирования и методов оценки качества программного обеспечения ПК-3.2. Демонстрирует умения проводить тестирование, определять метрики качества программного обеспечения (надежность, производительность, безопасность, удобство использования), решать задачи автоматизации тестирования ПК-3.3. Имеет практический опыт решения задач обеспечения качества программных продуктов	Знать: основы тестирования и методы оценки качества распределенных программных систем Уметь: проводить тестирование, определять метрики качества распределенных программных систем Владеть: навыками решения задач обеспечения качества распределенных программных систем
ПК-5	Способность выполнять проектирование компонентов программного обеспечения, включая проектирование баз данных, программных интерфейсов; разрабатывать	ПК-5.1. Демонстрирует знание принципов и шаблонов проектирования программного обеспечения, баз данных, программных интерфейсов, основ	Знать: принципы и шаблоны проектирования распределенных программных систем, программных интерфейсов Уметь: выполнять проектирование компонентов

	технические спецификации на компоненты программных систем и их взаимодействие	моделирования предметной области ПК-5.2. Демонстрирует умение выполнять проектирование компонентов программного обеспечения по заданным требованиям в рамках определенной предметной области ПК-5.3. Имеет практический опыт разработки технических спецификаций на компоненты программного обеспечения и интерфейсы	распределенных программных систем по заданным требованиям в рамках определенной предметной области Владеть: навыками разработки технических спецификаций на компоненты распределенных программных систем и интерфейсы

3. Содержание оценочных средств по дисциплине

3.1.Виды оценочных средств

Таблица 2. Виды оценочных средств.

№ п/п	Код компетенции/ планируемые результаты обучения	Контролируемые темы/ разделы	Наименование оценочного средства для текущего контроля	Наименование оценочного средства на промежуточной аттестации/№ задания
1	ПК-2.1. Демонстрирует знание основных принципов и технологий промышленной разработки программного обеспечения Знать: основные принципы и технологии промышленной разработки распределенных программных систем	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 1-10, 21-25, 34-93, 117-123
2	ПК-2.2. Демонстрирует умения разрабатывать программное обеспечение с применением инструментов автоматизации сборки, интеграции, тестирования и развертывания ПО Уметь: разрабатывать распределенные программные системы с применением инструментов автоматизации сборки, интеграции, тестирования и развертывания ПО	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 26-33, 72-116
3	ПК-2.3. Имеет практический опыт промышленной разработки программного обеспечения Владеть: навыками промышленной разработки распределенных программных систем	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 11-20, 94-116, 124-136
4	ПК-3.1. Демонстрирует знание основ тестирования и методов оценки качества программного обеспечения Знать: основы тестирования и методы оценки качества распределенных программных систем	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 1-10, 21-25, 34-93, 117-123
5	ПК-3.2. Демонстрирует умения проводить тестирование, определять метрики качества программного обеспечения (надежность, производительность,	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 26-33, 72-116

	безопасность, удобство использования), решать задачи автоматизации тестирования Уметь: проводить тестирование, определять метрики качества распределенных программных систем			
6	ПК-3.3. Имеет практический опыт решения задач обеспечения качества программных продуктов Владеть: навыками решения задач обеспечения качества распределенных программных систем	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 11-20, 94-116, 124-136
7	ПК-5.1. Демонстрирует знание принципов и шаблонов проектирования программного обеспечения, баз данных, программных интерфейсов, основ моделирования предметной области Знать: принципы и шаблоны проектирования распределенных программных систем, программных интерфейсов	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 1-10, 21-25, 34-93, 117-123
8	ПК-5.2. Демонстрирует умение выполнять проектирование компонентов программного обеспечения по заданным требованиям в рамках определенной предметной области Уметь: выполнять проектирование компонентов распределенных программных систем по заданным требованиям в рамках определенной предметной области	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 26-33, 72-116
9	ПК-5.3. Имеет практический опыт разработки технических спецификаций на компоненты программного обеспечения и интерфейсы Владеть: навыками разработки технических спецификаций на компоненты распределенных программных систем и	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений Курсовая работа	Тест	Задания теста № 11-20, 94-116, 124-136

	интерфейсы			

Типовые задания, критерии и показатели оценивания в рамках текущего контроля представлены в рабочей программе дисциплины (модуля). Полные комплекты оценочных средств и контрольно-измерительных материалов хранятся на кафедре.

3.2.Содержание оценочных средств

База тестовых вопросов для 6 семестра

№ п/п	Формулировка вопроса	Варианты ответов (полужирным шрифтом – верные варианты)
1.	Что нельзя рассматривать как модуль программной системы?	a. Класс **b. Метод** c. Сборку\Пакет d. Пространство имен (namespace)
2.	Какой из этих видов Cohesion считается наилучшим?	a. Logical cohesion b. Temporal cohesion c. Procedural cohesion **d. Sequential cohesion**
3.	Какой из этих видов Coupling возникает при связи модулей через некоторую глобальную переменную?	a. Control coupling b. Content coupling **c. Common coupling** d. Data coupling
4.	Какой из этих видов Coupling по сути означает нарушение инкапсуляции или принципа Information Hiding ?	a. Control coupling **b. Content coupling** c. Common coupling d. Data coupling
5.	Какой из этих видов Cohesion возникает при группировке элементов в модуль по критерию того, что все элеметны решают какую-либо узкую и хорошо-описанную задачу?	a. Logical cohesion **b. Functional cohesion** c. Procedural cohesion d. Sequential cohesion
6.	К какой парадигме относится язык C#?	a. Функциональной b. Объектно-ориентированной c. Процедурной **d. Ко всем перечисленным одновременно**
7.	В какой парадигме неизменяемость данных является одним из основных концептов?	**a. Функциональной** b. Объектно-ориентированной c. Процедурной d. Во всех перечисленных
8.	Выберите верное утверждение	**a. ООП хорошо подходит для моделирования предметных областей**

		b. для ООП требуется наличие понятия "класс" в языке c. Основное применение наследования в ООП - переиспользование кода d. Отношение IS-A в ООП реализуется с помощью Агрегации
9.	Выберите верное утверждение о процедурном программировании	a. В этом подходе обязательно появляются глобальные переменые b. Одним из основных концептов является разделение данных и вычислений (процедур, функций) c. Этот подход устарел с появлением ООП и не применяется с 1980ых годов **d. Это подход, наиболее подходящий для описания низкоуровневых алгоритмов**
10.	Функция высшего порядка (High-order function) это...	**a. Функция, которая принимает другие функции в качестве своих аргументов** b. Функция, которая вызывает в процессе работы другие функции c. Функция, которая передается в другую, как ее аргумент d. Функция, которая не имеет побочных эфектов
11.	Какой паттерн используется при необходимости сделать несколько взаимозаменяемых вариантов алгоритма	a. Посредник (Mediator) **b. Стратегия (Strategy)** c. Наблюдатель (Observer) d. Посетитель (Visitor)
12.	Какой паттерн предназначен для преобразования одного интерфейса к другому виду?	a. Декоратор (Decorator) **b. Адаптер (Adapter)** c. Наблюдатель (Observer) d. Посетитель (Visitor)
13.	Какой паттерн предназначен для динамического добавления объекту новых обязанностей?	**a. Декоратор (Decorator)** b. Адаптер (Adapter) c. Наблюдатель (Observer) d. Посетитель (Visitor)
14.	Какой паттерн моделирует глобальные переменные в ООП-языках?	a. Декоратор (Decorator) b. Адаптер (Adapter)

		c. Наблюдатель (Observer) **d. Одиночка (Singleton)**
15.	Какая группа паттернов не была представленна в классической книге "Шаблоны проектирования" (Design Patterns)?	a. Паттерны Поведения b. Порождающие Паттерны **c. Архитектурные Паттерны** d. Структурные Паттерны
16.	Что требуется для соблюдения Liskov Substitution Principle при наследовании?	a. Обеспечение гибкости с помощью полиморфизма **b. Соблюдение контракта базового класса** c. Отсутствие наследников у класса-потомка d. Зависимость базового класса от абстракции
17.	Уменьшение цены модификации кода, благодаря лучшей расширяемости, является целью...	a. Single Responsibility Principle b. Dependency Inversion Principle **c. Open-Closed Principle** d. Liskov Substitution Principle
18.	Борьба со сложностью, которая возникает при развитии логики приложения, является целью...	**a. Single Responsibility Principle** b. Dependency Inversion Principle c. Open-Closed Principle d. Liskov Substitution Principle
19.	С помощи передачи зависимостей через конструктор, реализуется принцип...	a. Single Responsibility Principle **b. Dependency Inversion Principle** c. Open-Closed Principle d. Liskov Substitution Principle
20.	Соблюдение Interface Segregation Principle зависит от того...	a. Сколько методов содержит интерфейс класса **b. В каких сценариях используются метода класса** c. Какие обязаноси выполняет класс d. Насколько сложная логика содержится в методах класса
21.	Unit-tests не используются для	a. Документации кода **b. Увеличения продаж программного продукта**

		c. Тестирования кода d. Улучшения дизайна кода
22.	Какого вида тестирования ПО не существует?	**a. Clinical-testing** b. Unit-testing c. Auto-testing d. Integration-testing
23.	Какая характеристика не относится к юнит-тестированию?	**a. Проверяют работу всей системы** b. Тестируют отдельный метод c. Изолируют окружение с помощью "моков" d. Каждый тест "падает" по одной причине
24.	TDD отличается от обычного юнит-тестирования, тем что	a. Тест проверят взаимодействие со сторонними сервисами b. Тест использует "моки" для изоляции окружения c. Тест проверяет взаимодействие различных компонентов **d. Тест, который проверяет код, пишется до того, как этот код написан**
25.	Какой из принципов SOLID, необходим для обеспечения тестируемости кода?	a. Liskov Substitution Principle b. Interface Segregation Principle **c. Dependency Inversion Principle** d. Single Responsibility Principle
26.	Если какая-то часть метода требует отдельных пояснений, следует:	a. Написать комментарий, поясняющий блок кода **b. Вынести блок кода в отдельный метод, назвать новый метод так, чтобы вопросов не оставалось** c. Описать этот случай в технической документации d. Написать юнит-тест, который гарантирует правильное выполнение этого метода
27.	Выберите неверный ответ	**a. Комментарии в коде не требует рефакторинга** b. Если комментарий поясняет блок кода, требуется "Извлечение

		метода" c. Если комментарий поясняет назначение метода, требуется "Переименование метода" d. Если комментарий поясняет выражение, требуется "Извлечение переменной"
28.	Какой рефакторинг следует использовать, в случае, если метод обращается к данным другого объекта чаще, чем к собственным данным?	a. Извлечение метода **b. Перемещение метода** c. Удаление посредника d. Встраивание метода
29.	Какой рефакторинг не применим, в случае, если метод имеет слишком длинный список параметров?	a. Замену параметра вызовом метода. b. Передача всего объекта, вместо передачи его полей c. Создание нового класса и замена параметров передей объекта **d. Переименование метода**
30.	Что следует сделать, в случае, если в коде часто встречается пара значений в виде отдельных переменных (например координаты X и Y)?	a. Использовать паттерн "Состояние" b. Использовать рефакторинг "Введение Null-объекта" **c. Создание отдельного класса для представления этой группы данных** d. Заменить конструктор фабричным методом
31.	Что можно отнести к 'ограничениям' в разработке ПО	a. Поддержка мобильных устройств **b. Законодательство страны** c. Безопасность d. Удобство пользовательского интерфейса
32.	Какой из этих способов не подходит для полноценного описания контракта класса?	a. Комментарии b. Документация **c. Сигнатура функции**
33.	Проактивные изменения...	**a. Как правило не затрагивают внешний вид продукта (UI)** b. Независят от команды разработчиков c. Вызваны внешними изменениями d. Требуют полного

		переписывания программной системы
34.	Выберите неверное утверждение - целью проектирования ПО является	**a. Повышение производительности системы** b. Удовлетворение всех функциональных требований c. Удовлетворение всех не-функциональных требований d. Соблюдение всех ограничений
35.	Случайная сложность (Accidental complexity) вызвана...	**a. Низкой квалификацией разработчиков** b. Сложностью предметной области c. Несоблюдением законодательства d. Требованиями к производительности
36.	Рефакторинг является примером ... изменения	a. реактивного **b. проактивного** c. случайного d. не нужного
37.	Необходимая сложность (Essential complexity) вызвана...	a. Низкой квалификацией разработчиков **b. Сложностью предметной области** c. Сложностью языка программирования d. Требованиями к производительности
38.	Дизайн, в отличии от архитектуры, включает в себя	a. Описание ответственности модуля b. Крупные компоненты и их взаимодействие между собой **c. Выбор алгоритмов и структур данных** d. Проектирование связей между компонентами
39.	Приложение, состоящее из отдельного UI(сделаного на js), серверного веб приложения и базы данных, относится к следующему архитектурному стилю	a. Monolithic b. Client\Server **c. 3-Tier** d. N-Tier
40.	Какой вид взаимодействия чаще всего не требует промежуточного звена?	a. Файлы **b. Удаленный вызов** c. БД

		d. Сообщения
41.	Модель Publisher-Subsriber, предполагает...	a. Конвейерную обработку данных **b. Получение сообщения всеми потребитерями** c. Получение одного сообщения, только одним из потребителей d. Сохранение в БД
42.	Микросервесный стиль архитектуры в первую очередь характеризуется	a. Взаимодействием через общую БД **b. Разбиением одного большого приложения на несколько полностью независимых частей** c. Выделением слоя бизнес-логики d. Необходимостью использовать балансировку нагрузки
43.	Что из этого НЕ относится к ограничениям при разработки ПО?	a. Необходимость соблюдать законы страны **b. Поддержка Retina-дисплеев** c. Бюджет проекта d. Время разработки проекта
44.	Что не входит в интерфейс функции	a. Имя функции b. Аргументы функции c. Возвращаемое значение **d. Описание поведения функции**
45.	Какой из этих языков наиболее низкоуровневый?	a. Java **b. Assembler** c. C d. C++
46.	Что из этого не является частью контракта метода?	a. предусловия b. постусловия **c. имя функции** d. инварианты класса
47.	Если менеджер продукта решил поменять дизайн главной страницы сайта, то это изменение является...	**a. Реактивным** b. Проактивным c. Случайным d. Фатальным
48.	Проектирование приложения с учетом возможного добавления какой-либо функции в будущем, в первую очередь нарушает принцип...	a. KISS **b. YAGNI** c. DRY d. Принцип наименьшего удивления

49.	Использование значения по умолчанию, для параметра, который не удалось получить из файла конфигурации нарушает принцип...	a. KISS b. YAGNI c. DRY **d. Fail Fast**
50.	К чему призывает принцип "Information Hiding" (принцип скрытия информации)?	**a. К проектированию с учетом только интерфейсов/конрактов модулей** b. К проектированию с учетом особеностей внутренней реализации модулей c. К выбору наиболее простых решений при проектировании d. К избавлению от дублирования информации при проектировании
51.	К какому виду дублирования чаще всего приводят проблемы в коммуникации между разработчиками?	a. Imposed (Вынужденное) b. Inadvertent (Неосознанное, неумышленное) c. Impatient (Нетерпеливаое) **d. Interdeveloper (Коллективное)**
52.	В чем основная идея принципа "Наименьшего удивления"?	a. Нужно обеспечивать максимальную корректность работы системы b. Нужно наиболее явным образом выражать в коде то, какую проблему этот код решает **c. Нужно проектировать программы с учетом общепринятых представлений о разработке** d. Необходимо не допускать дублированния информации
53.	Переусложение программного решения, в первую очередь нарушает принцип	a. DRY **b. KISS** c. YAGNI d. Принцип наименьшего удивления
54.	Принцип ортогональности (Orthogonality) призывает	a. Решать задачу наиболее простым способом **b. Сокращать взаимное влияние между несвязанными вещами** c. Избегать дублирования информации

		d. Решать задачу, не пытаясь предугадать ещё неизвестные требования
55.	Принцип DRY призывает	a. Решать задачу наиболее простым способом b. Сокращать взаимное влияние между несвязанными вещами **c. Избегать дублирования информации** d. Решать задачу, не пытаясь предугадать ещё неизвестные требования
56.	Принцип Intentionality нарушается если	a. Созданное ПО обладает гибкостью в тех местах, где это не требуется b. Есть дублирование функциональности **c. По написанному коду очень сложно понять, какую задачу он решает** d. Модуль в своей работе опирается на детали реализации другого модуля
57.	Дублирование вызванное особеностями инстрементов разработки (средой исполнения, языком) называется	**a. Imposed (Вынужденное)** b. Inadvertent (Неосознанное, неумышленное) c. Impatient (Нетерпеливое) d. Interdeveloper (Коллективное)
58.	Проектирование с учетом предполагаемых, но не утвержденных требований нарушает	a. DRY b. KISS **c. YAGNI** d. Intentionality
59.	Какое свойство 'принципа' отличает принцип от банальности?	a. Абстрактность **b. Опровергаемость** c. Непременимость d. Однозначность
60.	Абстрактность по отношению к прицину означает, что...	**a. Принцип не может быть применим немедленно** b. Принцип может быть оспорен c. Принцип должен скрывать детали реализации d. Принцип требует неукоснительного следования ему
61.	Content coupling возникает...	a. При взаимодействии двух модулей через глобальную

		переменную b. Когда один модуль меняет поведение другого, путем передачи уравляющего флага **c. При нарушении принципа Information Hiding/инкапсуляции** d. Когда один модуль взаимодействует с другим через внешний файл определенного формата
62.	Common coupling возникает...	**a. При взаимодействии двух модулей через глобальную переменную** b. Когда один модуль меняет поведение другого, путем передачи уравляющего флага c. При нарушении принципа Information Hiding/инкапсуляции d. Когда один модуль взаимодействует с другим через внешний файл определенного формата
63.	External coupling возникает...	a. При взаимодействии двух модулей через глобальную переменную b. Когда один модуль меняет поведение другого, путем передачи уравляющего флага c. При нарушении принципа Information Hiding/инкапсуляции **d. Когда один модуль взаимодействует с другим через внешний файл определенного формата**
64.	Каким в идеале должен быть coupling?	a. Не важно b. Средним c. Высоким **d. Низким**
65.	Каким в идеале должен быть cohesion?	a. Не важно b. Средним **c. Высоким** d. Низким
66.	Coincidental cohesion означает, что компоненты объеденены в модуль	a. потому что относятся к одной логической категории **b. без какой либо логики (случайно)**

		c. потому что вместе решают какую-либо хорошо определенную задачу d. потому что всегда вызываются друг за другом
67.	Procedural cohesion означает, что компоненты объеденены в модуль	a. потому что относятся к одной логической категории b. без какой либо логики (случайно) c. потому что вместе решают какую-либо хорошо определенную задачу **d. потому что всегда вызываются друг за другом**
68.	Logical cohesion означает, что компоненты объеденены в модуль	**a. потому что относятся к одной логической категории** b. без какой либо логики (случайно) c. потому что вместе решают какую-либо хорошо определенную задачу d. потому что всегда вызываются друг за другом
69.	Functional cohesion означает, что компоненты объеденены в модуль	a. потому что относятся к одной логической категории b. без какой либо логики (случайно) **c. потому что вместе решают какую-либо хорошо определенную задачу** d. потому что всегда вызываются друг за другом
70.	Функция, принимающая другие функции в качестве аргумента, называется	a. Чистой (Pure) **b. Высшего порядка (High-ordered)** c. Рекурсивной (Recursion) d. Неизменяемой (Immutable)
71.	Возможность перегрузки операторов в C#/C++ и других языках является примером	a. Overloading polymorphism **b. Parametric Polymorphism** c. Subtype Polymorphism d. Multiple inheritance
72.	Полиморфизм подтипов (Subtype polymorphism) реализуется с помощью	a. Инкапсуляции **b. Наследования интерфейса** c. Агрегации

		d. Композиции
73.	Отношение IS-A в ООП моделируется с помощью	a. Инкапсуляции **b. Наследования** c. Агрегации d. Композиции
74.	Отношение HAS-A в ООП моделируется с помощью	a. Инкапсуляции b. Наследования **c. Агрегации и Композиции** d. Абстракции

База тестовых вопросов для 7 семестра

№ п/п	Формулировка вопроса	Варианты ответов (полужирным шрифтом – верные варианты)
75.	Anemic Domain Model, отличается от Rich Domain Model тем, что	a. Классы сущностей не имеют прямых связей друг с другом b. Между классами сущностей нет отношений наследования **c. Классы сущностей не содержат методов** d. Классы сущностей не имеют getters и setters
76.	Инкапсуляция применяется для...	a. Моделирования предметных областей b. Абстрагирования путем наследования поведения **c. Абстрагирования путем скрытия деталей реализации** d. Динамического изменения поведения класса
77.	Построение программы, как последовательности вызовов процедур - это характеристика ... парадигмы	a. Функциональной b. Объектно-ориентированной **c. Процедурной** d. Всех перечисленных
78.	Для какой парадигмы необходимо наличие понятия "класс" в языке?	a. Функциональной b. Объектно-ориентированной c. Процедурной **d. Никакой**
79.	Функция, не имеющая побочных эффектов, называется	**a. Чистой (Pure)** b. Высшего порядка (High-ordered) c. Рекурсивной (Recursion) d. Неизменяемой

		(Immutable)
80.	Целью Liskov Substitution Principle является	**a. Организация правильных иерархий наследования** b. Уменьшение цены модификации кода, благодаря лучшей расширяемости c. Уменьшение количества связей между классами d. Борьба со сложностью, которая возникает при развитии логики приложения
81.	Целью Open-Closed Principle является	a. Организация правильных иерархий наследования **b. Уменьшение цены модификации кода, благодаря лучшей расширяемости** c. Уменьшение количества связей между классами d. Борьба со сложностью, которая возникает при развитии логики приложения
82.	Open-Closed Principle реализуется с помощью	a. Соблюдения контракта класса **b. Инкапсуляции и наследования** c. Разделения класса на несколько, при необходимости d. Передачи зависимостей через конструктор
83.	Dependency Inversion Principle реализуется с помощью	a. Соблюдения контракта класса b. Инкапсуляции и наследования c. Разделения класса на несколько, при необходимости **d. Передачи зависимостей через конструктор**
84.	Single Responsibility Principle реализуется с помощью	a. Соблюдения контракта класса b. Инкапсуляции и наследования **c. Разделения класса на несколько, при необходимости** d. Передачи зависимостей

		через конструктор
85.	Соблюдение Interface Segregation Principle зависит от	a. Возможностей потенциального расширения функционала класса b. Контракта класса **c. Клиентов класса** d. Ответственности класса
86.	Соблюдение Single Responsibility Principle зависит от	a. Возможностей потенциального расширения функционала класса b. Контракта класса c. Клиентов класса **d. Ответственности класса**
87.	Целью Single Responsibility Principle является	a. Организация правильных иерархий наследования b. Уменьшение цены модификации кода, благодаря лучшей расширяемости c. Уменьшение количества связей между классами **d. Борьба со сложностью, которая возникает при развитии логики приложения**
88.	Паттерн Наблюдатель (Observer) нужен для	**a. Возможности следить и реагировать на события, происходящие в другом объекте** b. Возможности заменять алгоритмы прямо во время выполнения программы c. Гарантии того, что у класса будет только один экземпляр d. Согласования несовместимых интерфейсов
89.	Паттерн Стратегия (Strategy) нужен для	a. Возможности следить и реагировать на события, происходящие в другом объекте **b. Возможности заменять алгоритмы прямо во время выполнения программы** c. Гарантии того, что у класса будет только один экземпляр d. Согласования несовместимых интерфейсов

90.	Паттерн позволяющий динамически добавлять объектам новую функциональность, называется	a. Шаблонный метод (Template Method) **b. Декоратор (Decorator)** c. Фабричный метод (Factory Method) d. Фасад (Facade)
91.	Паттерн предоставляющий простой интерфейс к сложной системе классов, называется	a. Шаблонный метод (Template Method) b. Наблюдатель (Observer) c. Фабричный метод (Factory Method) **d. Фасад (Facade)**
92.	Паттерн предоставляющий общий алгоритм и позволяющий менять отдельные его шаги, через наследование, называется	**a. Шаблонный метод (Template Method)** b. Декоратор (Decorator) c. Стратегия (Strategy) d. Фасад (Facade)
93.	Паттерн Адаптер (Adapter) нужен для	a. Возможности следить и реагировать на события, происходящие в другом объекте b. Возможности заменять алгоритмы прямо во время выполнения программы c. Гарантии того, что у класса будет только один экземпляр **d. Согласования несовместимых интерфейсов**
94.	Какой вид тестов проверяет взаимодействие ПО с внешними системами?	a. Это проверяется всеми видами тестами b. Unit-tests c. Auto-tests **d. Integration-tests**
95.	Какой вид программных тестов проверяет корректность работы системы с т.з. пользователя, включая UI?	a. Это проверяется всеми видами тестами b. Unit-tests **c. Auto-tests** d. Integration-tests
96.	При каком виде тестирования есть необходимость использования заглушек (Mock'ов и Stub'ов)?	a. Это требуется во всех видах тестирования **b. Unit-tests** c. Auto-tests d. Integration-tests
97.	Mock'и и Stub'ы используются для...	a. Добавления новой функциональности **b. Изоляции тестов от**

		внешнего окружения c. Уменьшения связности модулей d. Проверки взаимодействия с внешними системами
98.	Какая характеристика относится к юнит-тестированию?	a. Тесты проверяют ошибки интеграции b. Тесты проверяют проблемы производительности c. Тесты гарантируют правильность работы ПО **d. Каждый тест "падает" по одной причине**
99.	Написание теста, до создания кода, называется?	a. Auto-testing b. Unit-testing c. DDD **d. TDD**
100.	Осознанное "завязывание" проекта на конкретную БД является примером	a. Неумышленного долговременного технического долга b. Умышленного кратковременного технического долга c. Неумышленного кратковременного технического долга **d. Умышленного долговременного технического долга**
101.	Написание "плохого" кода из-за спешки перед релизом, скорее всего является примером	a. Неумышленного долговременного технического долга **b. Умышленного кратковременного технического долга** c. Неумышленного кратковременного технического долга d. Умышленного долговременного технического долга
102.	Наиболее вероятным рефакторингом, в случае если метод одного класса работает с большим количеством полей другого класса, является	**a. Перемещение метода** b. Перемещение поля c. Извлечение класса d. Замена алгоритма
103.	Наиболее вероятным рефакторингом, в случае если есть некоторый массив в разных ячейках которого хранятся ращличные по смыслу данные, является	**a. Замена поля-массива объектом** b. Перемещение поля c. Инкапсуляция поля

		d. Перемещение метода
104.	Наиболее вероятным рефакторингом, в случае если есть сложное для понимания выражение, является	a. Извлечение метода b. Замена алгоритма **c. Извлечение переменной** d. Встраивание переменной
105.	Случай, когда вместо отдельного типа имеется набор чисел или строк, который определяет список допустимых значений для какой-то сущности, называется	a. Магическая строкой\числом **b. Кодирование типа** c. Null-объекта d. Делегированием
106.	Случай, когда объект вместо вызова своего метода перенаправляет вызов другому объекту, называется	a. Магическая строкой\числом b. Кодирование типа c. Null-объекта **d. Делегированием**
107.	Архитектурный стиль, при котором все приложение представляет собой один инсполняемый файл, называется	**a. Monolithic** b. Client\Server c. 3-Tier d. N-Tier
108.	Приложение, состоящее из отдельного UI и База данных с логикой в хранимых процедурах, относится к следующему архитектурному стилю	a. Monolithic **b. Client\Server** c. 3-Tier d. N-Tier
109.	Выберите неверное утверждение. Цель архитектурного проектирования - это	**a. Повышение надежности и безопасности ПО** b. Упрощение дизайна через его разбиение на функциональные области c. Снижение рисков связанных с выбранным техническим решением d. Разрешение компромиса между противоречивыми требованиями разных сторон
110.	Какое решение однозначно относится к архитектурному	**a. Невозможно сказать точно** b. Выбор базы данных c. Выбор языка разработки d. Выбор css-фреймворка
111.	Архитектура, в отличии от дизайна, включает в себя	a. Описание ответственности модуля **b. Крупные компоненты и их взаимодействие между собой** c. Выбор алгоритмов и структур данных d. Проектирование связей

		между компонентами
112.	Какой вид взаимодействия относится к синхронному?	a. Файлы **b. Удаленный вызов** c. БД d. Сообщения
113.	Выберите верное утверждение. Взаимодействие UI и Application Layer при строгом разделении на слои	a. UI может обращаться к Application Layer, Application Layer может вызывать функции UI **b. UI может обращаться к Application Layer, Application Layer не может вызывать функции UI** c. UI не может обращаться к Application Layer, Application Layer может вызывать функции UI d. UI не может обращаться к Application Layer, Application Layer не может вызывать функции UI
114.	DDD-style отличается от Layered architecture, тем что	a. Нет разделения на слои b. Слой базы данных не зависит от других слоев **c. Слой бизнес-логики не зависит от других слоев** d. Есть возможность полностью заменить, например, слой UI
115.	Сообщение от отправителя, доставляется только одному потребителю, соответсвует паттерну	a. Observer b. State machine **c. Producer-Consumer** d. Publisher-Subsriber
116.	CAP теорема не оперирует понятием	a. Доступности (Avalability) b. Согласованности (Consistency) **c. Атомарности (Atomicity)** d. Устойчивости к разделению (Partition Tolerance)
117.	Какой из способов взаимодействия является синхронным	- Файлы **+ Удаленный вызов** - Сообщения - БД
118.	К недостаткам удаленный вызова можно отнести	- Асинхронную модель выполнения - Синхронную модель выполнения **+ Прямую зависимость одного приложения от**

		другого - Наличие промежуточных звеньев
119.	Какая из этих характеристик используется в формулировке CAP теоремы	**+ Consistency** - Capacity - Agility - Programability
120.	Согласно CAP теореме, система способная обработать любой запрос к любому узлу системы обладает свойством	- Безотказности - Надежности **- Доступности** - Стабильности
121.	Стуктура программной системы называется:	- Архитектурой - Дизайном **+ Может быть и тем и тем, в зависисмости от уровня детализации**
122.	Приложение с UI, работающее в связке с БД, является примерном архитектурного стиля:	- Монолитного **+ Клиент-серверного** - Трех-звенного
123.	Монолитное приложение отличается тем, что:	- Его разработка ведется в одном репозитории - Его невозможно масштабировать **+ Отсутствует взаимодействие между узлами** - Его невозможно перенести на другой сервер
124.	Кооперативная многозадачность реализуется с помощью паттерна	- Visitor **+ Reactor** - Factory - Builder
125.	Кооперативная многозадачность лучше всего подходит для	**+ Операций ввода\вывода** - Операций рассчета - Ни для чего, это устаревшая модель многозадачности - Операций, требующих большого количества оперативной памяти
126.	Кооперативная многозадачность хуже всего подходит для	- Операций ввода\вывода **- Операций рассчета** - Ни для чего, это устаревшая модель многозадачности - Операций, требующих большого количества оперативной памяти
127.	Кооперативная многозадачность, чаще всего реализуется в языках высокого уровня с помощью:	- Потоков **+ Функций обратного вызова** **+ "Зеленых потоков"** - Использования примитивов синхронизации (например концструкции lock)

128.	Для возможности горизонтального масштабирования, система должна удовлетворять концепциям	**+ Stateless** **+ Shared Nothing** - Immutability - Pure calculation
129.	Какой закон можно сформулировать фразой «Добавление новых узлов в систему, со временем начинает приносить все меньший прирост производительности»	**- Закон Амдала** - Закон больших чисел - Закон конечного масштабирования - Закон Дейкстры
130.	Отношение успешного (НЕ cache miss) числа запросов и общего количества запросов к кэшу называется	- Коэфицентом Амдала **- HitRatio** - Capacity Ratio - Коэфицентом доступности
131.	Какой вид формулы шардинга позволяет управлять местонахождением каждой записи в кластере вручную?	- Consistency Hashing **- Table Function** - Остаток от деление на количество серверов
132.	Какой вид функции шардинга НЕ позволяет проводить решардинг с работающим кластером	- Consistency Hashing - Table Function **+ Остаток от деление на количество серверов**
133.	БД основанные на LSM деревях отличаются тем, что	- Обладают лучшей надежностью - Лучше обеспечивают согласованность данных - Быстрее на чтение **+ Быстрее на запись**
134.	Какие алгоритмы относятся к алгоритмам выбора лидера в распределенных системах:	- Алгоритм Дейкстры **+ Алгоритм Paxos** - Алгоритм Хаос **+ Алгоритм RAFT**
135.	При каких видах репликации может происходить нарушение консистентности данных?	- Сихронная - Асинхронная **- В обоих видах**
136.	Линеаризуемость является	**+ Видом consistency** - Видом availability - Видом Partition Tolerance

Задание для курсовой работы

В ходе изучения курса, обучающиеся будут выполнять проект по разработке программной системы. Проект предполагает реализацию 5 ключевых этапов, соответствующих классическому жизненному циклу программного проекта:

1. Инициирование проекта
2. Сбор и анализ требований
3. Проектирование
4. Реализация
5. Тестирование и внедрение

В ходе выполнения работы над проектом должна быть разработана сама программная система и комплект документов. Комплект документов может быть изменен в зависимости от выбранной методологии разработки и управления проектом. В конце семестра проходит открытая защита проектов, в ходе которой необходимо продемонстрировать и защитить полученное решение.

Студент, как правило, выбирает задание на разработку системы самостоятельно, но система

должна отвечать следующим общим требованиям:

- Клиент-серверное приложение
- Работа с базой данных
- Графический интерфейс пользователя (возможны также варианты: веб-интерфейс, интерфейс для мобильных устройств)

Идеальный вариант: наличие реального заказчика, который испытывает потребность в данной программной системе.

4. Порядок проведения и критерии оценивания промежуточной аттестации

4.1.Порядок проведения промежуточной аттестации

Курс 3, Семестр 6:

Экзамен проводится в виде тестирования. Студент должен ответить на вопросы закрытого типа, которые предполагают выбор вариантов ответа, а также на вопросы открытого типа, которые не предполагают вариантов ответа, правильный ответ требуется написать самостоятельно. Всего 20 тестовых вопросов. Продолжительность теста – 35 минут.

Курсовая работа оценивается через процедуру защиты. На защиту студент представляет:

1. Развернутое задание.

2. Пояснительную записку на 35 – 40 страниц в электронном/отпечатанном виде, содержащую аннотацию, введение, основную часть с иллюстрациями, заключение, библиографию, приложения.

3. Презентацию проекта на 15 - 20 слайдах.

Защита курсового проекта проводится в комиссии, состоящей не менее, чем из двух преподавателей. На защите студент в течение 5 – 7 минут докладывает об основных результатах, полученных в работе, отвечает на вопросы членов комиссии.

4.2.Критерии оценивания промежуточной аттестации по видам оценочных средств

4.2.1. Критерии оценивания теста

Тест формируется в системе электронного обучения MOODLE.

Максимальный балл за тест — 100 баллов.

Оценка	Отлично/ Зачтено	Хорошо/ зачтено	Удовлетворительно/зачтено	Неудовлетворительно/ незачтено
Баллы	100-86 баллов	85-76 баллов	75-60 баллов	59-0 баллов
Уровень освоения проверяемых компетенций	высокий	средний	базовый	недостаточный

4.2.1. Критерии оценивания курсовой работы

«отлично»

1) проект реализован;
2) сложность реализации: высокая
3) проектирование: грамотно применены архитектурные паттерны
4) внедрение: проект успешно внедрен в пилотную среду
5) защита проекта: грамотно выстроена презентация, даны ответы на вопросы комиссии

«хорошо»

1) проект реализован;
2) сложность реализации: средняя
3) проектирование: грамотно применены архитектурные паттерны, возможно,

некоторые архитектурные решения не обоснованы и спорны

4) внедрение: проект частично внедрен в пилотную среду

5) защита проекта: логично выстроена презентация, даны ответы на большую часть вопросов комиссии

«удовлетворительно»

1) проект реализован частично;

2) сложность реализации: низкая-средняя

3) проектирование: имелись попытки применить архитектурные паттерны, решения не обоснованы

4) внедрение: проект не внедрен в пилотную среду

5) защита проекта: даны ответы лишь на часть вопросов комиссии

«неудовлетворительно» проект не реализован

4.3.Результаты промежуточной аттестации и уровни сформированности компетенций

Для получения «удовлетворительно» обучающийся должен выполнить итоговый контрольный тест как минимум на 60%.

Для получения «хорошо» обучающийся должен выполнить итоговый контрольный тест как минимум на 76%.

Для получения «отлично» обучающийся должен выполнить итоговый контрольный тест как минимум на 86%.;

Особенности проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья обозначены в рабочей программе дисциплины (модуля).

Уровни сформированности компетенций определяется следующим образом:

1. Высокий уровень сформированности компетенций соответствует оценке отлично:

- предполагает формирование компетенций на высоком уровне;
- знание теоретических разделов изучаемой дисциплины на уровне не ниже оценки отлично;
- студент умеет применять на практике знания, полученные в рамках изучения дисциплины
- формируются навыки использования теоретических и практических разделов дисциплины для решения задач профессиональной деятельности;

2. Средний уровень соответствует оценке хорошо:

- предполагает формирование компетенций на среднем уровне;
- знание теоретических разделов изучаемой дисциплины на уровне не ниже оценки хорошо;
- студент умеет применять знания, полученные в рамках изучения дисциплины, для решения задач профессиональной деятельности;

3. Базовый уровень соответствует оценке удовлетворительно:

- предполагает формирование компетенций на базовом уровне;

- знание теоретических разделов изучаемой дисциплины на уровне не ниже оценки удовлетворительно;

4. Недостаточный уровень соответствует оценке неудовлетворительно.