

Документ подписан простой электронной подписью  
Информация о владельце:  
ФИО: Таскаев Сергей Валерьевич  
Должность: Ректор  
Дата подписания: 05.09.2025 11:05:23  
Уникальный программный ключ:  
04c19ed8bfb98f4b6c977a486b9a8788b832573



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Челябинский государственный университет» (ФГБОУ ВО «ЧелГУ»)

Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю) «Проектирование и разработка распределенных программных систем» по направлению подготовки 09.03.04 «Программная инженерия» направленности «Информационные системы и интеллектуальные технологии»  
ФГБОУ ВО «ЧелГУ»

стр. 1

**Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю)  
«Проектирование и разработка распределенных программных систем»**

**Направление подготовки (специальность)  
09.03.04 «Программная инженерия»**

**Направленность (профиль)  
«Информационные системы и интеллектуальные технологии»**

**Присваиваемая квалификация  
Бакалавр**

**Форма обучения  
Очная**

**Год набора  
2025**

Челябинск, 2025 г.

**09.03.04 Программная инженерия, Информационные системы и интеллектуальные технологии, бакалавр, *Проектирование и разработка распределенных программных систем, 2025*, очная**

**Фонд оценочных средств дисциплины (модуля) одобрен и рекомендован**

Проректор по учебной работе      утверждено 24.02.2025      А.А. Саламатов

Ученым советом института информационных технологий

Протокол заседания № 6 от 20.02.2025

Председатель Ученого совета  
института информационных  
технологий

согласовано

Ю. В. Петриченко

**Заседанием кафедры информационных технологий и экономической информатики**

Протокол заседания № 6 от 20.02.2025

И. о. заведующего кафедрой

согласовано

С.А. Скрипов

Автор (составитель)

С.А. Скрипов

**Структура рабочей программы соответствует приказу ректора ФГБОУ ВО «ЧелГУ» от «13» апреля 2021 г. № 247-1**



## 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 .....	20
4.1. Порядок проведения промежуточной аттестации .....	20
4.2. Критерии оценивания промежуточной аттестации по видам оценочных средств .....	20
4.3. Результаты промежуточной аттестации и уровни сформированности компетенций.....	20



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Челябинский государственный университет» (ФГБОУ ВО «ЧелГУ»)

Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю) «Проектирование и разработка распределенных программных систем» по направлению подготовки 09.03.04 «Программная инженерия» направленности «Информационные системы и интеллектуальные технологии»  
ФГБОУ ВО «ЧелГУ»

стр. 3

## 1. Паспорт фонда оценочных средств

Направление подготовки: 09.03.04 Программная инженерия

Направленность: Информационные системы и интеллектуальные технологии

Дисциплина: Проектирование и разработка распределенных программных систем

Семестр: 6

Форма промежуточной аттестации: экзамен

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 и 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-6	Способность выполнять проектирование компонентов программного обеспечения, включая проектирование баз данных, программных	ПК-6.1. Демонстрирует знание принципов и шаблонов проектирования программного обеспечения, баз данных, программных	Знать: принципы и шаблоны проектирования распределенных программных систем, программных интерфейсов Уметь: выполнять



	интерфейсов; разрабатывать технические спецификации на компоненты программных систем и их взаимодействие	интерфейсов, основ моделирования предметной области ПК-6.2. Демонстрирует умение выполнять проектирование компонентов программного обеспечения по заданным требованиям в рамках определенной предметной области ПК-6.3. Имеет практический опыт разработки технических спецификаций на компоненты программного обеспечения и интерфейсы	проектирование компонентов 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	Тест Проверка практических заданий	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	Тест Проверка практических заданий	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	Тест Проверка практических заданий	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	Тест Проверка практических заданий	Задания теста № 1-10, 21-25, 34-93, 117-123
5	ПК-3.2. Демонстрирует умения проводить тестирование, определять метрики качества программного обеспечения (надежность,	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений	Тест Проверка практических заданий	Задания теста № 26-33, 72-116



	производительность, 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	Тест Проверка практических заданий	Задания теста № 11-20, 94-116, 124-136
7	ПК-6.1. 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	Тест Проверка практических заданий	Задания теста № 1-10, 21-25, 34-93, 117-123
8	ПК-6.2. 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	Тест Проверка практических заданий	Задания теста № 26-33, 72-116
9	ПК-6.3. Имеет практический опыт разработки технических спецификаций на компоненты программного обеспечения и интерфейсы Владеть: навыками разработки технических спецификаций на	Основы проектирования распределенных приложений Подходы для разработки распределенных приложений	Тест Проверка практических заданий	Задания теста № 11-20, 94-116, 124-136



компоненты распределенных программных систем и 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

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



8.	Выберите верное утверждение	<b>a. ООП хорошо подходит для моделирования предметных областей</b> 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 переменные 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 <b>d. Это подход, наиболее подходящий для описания низкоуровневых алгоритмов</b>
10.	Функция высшего порядка (High-order function) это...	<b>a. Функция, которая принимает другие функции в качестве своих аргументов</b> b. Функция, которая вызывает в процессе работы другие функции c. Функция, которая передается в другую, как ее аргумент d. Функция, которая не имеет побочных эффектов
11.	Какой паттерн используется при необходимости сделать несколько взаимозаменяемых вариантов алгоритма	a. Посредник (Mediator) <b>b. Стратегия (Strategy)</b> c. Наблюдатель (Observer) d. Посетитель (Visitor)
12.	Какой паттерн предназначен для преобразования одного интерфейса к другому виду?	a. Декоратор (Decorator) <b>b. Адаптер (Adapter)</b> c. Наблюдатель (Observer) d. Посетитель (Visitor)
13.	Какой паттерн предназначен для динамического добавления объекту новых обязанностей?	<b>a. Декоратор (Decorator)</b> 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) <b>d. Одиночка (Singleton)</b>
15.	Какая группа паттернов не была представлена в классической книге "Шаблоны проектирования" (Design Patterns)?	a. Паттерны Поведения b. Порождающие Паттерны <b>c. Архитектурные Паттерны</b> d. Структурные Паттерны
16.	Что требуется для соблюдения Liskov Substitution Principle при наследовании?	a. Обеспечение гибкости с помощью полиморфизма <b>b. Соблюдение контракта базового класса</b> 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 <b>c. Open-Closed Principle</b> d. Liskov Substitution Principle
18.	Борьба со сложностью, которая возникает при развитии логики приложения, является целью...	<b>a. Single Responsibility Principle</b> 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>b. Dependency Inversion Principle</b> c. Open-Closed Principle d. Liskov Substitution Principle
20.	Соблюдение Interface Segregation Principle зависит от того...	a. Сколько методов содержит интерфейс класса <b>b. В каких сценариях используются метода класса</b> c. Какие обязанности выполняет класс d. Насколько сложная логика содержится в методах класса



21.	Unit-tests не используются для	a. Документации кода <b>b. Увеличения продаж программного продукта</b> c. Тестирования кода d. Улучшения дизайна кода
22.	Какого вида тестирования ПО не существует?	<b>a. Clinical-testing</b> b. Unit-testing c. Auto-testing d. Integration-testing
23.	Какая характеристика не относится к юнит-тестированию?	<b>a. Проверяют работу всей системы</b> 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 проверяет взаимодействие со сторонними сервисами b. Тест использует "моки" для изоляции окружения c. Тест проверяет взаимодействие различных компонентов <b>d. Тест, который проверяет код, пишется до того, как этот код написан</b>
25.	Какой из принципов SOLID, необходим для обеспечения тестируемости кода?	a. Liskov Substitution Principle b. Interface Segregation Principle <b>c. Dependency Inversion Principle</b> d. Single Responsibility Principle
26.	Если какая-то часть метода требует отдельных пояснений, следует:	a. Написать комментарий, поясняющий блок кода <b>b. Вынести блок кода в отдельный метод, назвать новый метод так, чтобы вопросов не оставалось</b> c. Описать этот случай в технической документации d. Написать юнит-тест, который гарантирует правильное выполнение этого метода
27.	Выберите неверный ответ	<b>a. Комментарии в коде не требуют рефакторинга</b>



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



		<p>с. Вызваны внешними изменениями d. Требуют полного переписывания программной системы</p>
34.	Выберите неверное утверждение - целью проектирования ПО является	<p><b>a. Повышение производительности системы</b> b. Удовлетворение всех функциональных требований c. Удовлетворение всех нефункциональных требований d. Соблюдение всех ограничений</p>
35.	Случайная сложность (Accidental complexity) вызвана...	<p><b>a. Низкой квалификацией разработчиков</b> b. Сложностью предметной области c. Несоблюдением законодательства d. Требованиями к производительности</p>
36.	Рефакторинг является примером ... изменения	<p>a. реактивного <b>b. проактивного</b> c. случайного d. не нужного</p>
37.	Необходимая сложность (Essential complexity) вызвана...	<p>a. Низкой квалификацией разработчиков <b>b. Сложностью предметной области</b> c. Сложностью языка программирования d. Требованиями к производительности</p>
38.	Дизайн, в отличие от архитектуры, включает в себя	<p>a. Описание ответственности модуля b. Крупные компоненты и их взаимодействие между собой <b>c. Выбор алгоритмов и структур данных</b> d. Проектирование связей между компонентами</p>
39.	Приложение, состоящее из отдельного UI(сделаного на js), серверного веб приложения и базы данных, относится к следующему архитектурному стилю	<p>a. Monolithic b. Client\Server <b>c. 3-Tier</b> d. N-Tier</p>



40.	Какой вид взаимодействия чаще всего не требует промежуточного звена?	a. Файлы <b>b. Удаленный вызов</b> c. БД d. Сообщения
41.	Модель Publisher-Subscriber, предполагает...	a. Конвейерную обработку данных <b>b. Получение сообщения всеми потребителями</b> 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>b. Разбиением одного большого приложения на несколько полностью независимых частей</b> 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>b. Поддержка Retina-дисплеев</b> c. Бюджет проекта d. Время разработки проекта
44.	Что не входит в интерфейс функции	a. Имя функции b. Аргументы функции c. Возвращаемое значение <b>d. Описание поведения функции</b>
45.	Какой из этих языков наиболее низкоуровневый?	a. Java <b>b. Assembler</b> c. C d. C++
46.	Что из этого не является частью контракта метода?	a. предусловия b. постусловия <b>c. имя функции</b> d. инварианты класса
47.	Если менеджер продукта решил поменять дизайн главной страницы сайта, то это изменение является...	<b>a. Реактивным</b> b. Проактивным c. Случайным d. Фатальным
48.	Проектирование приложения с учетом возможного добавления	a. KISS



	какой-либо функции в будущем, в первую очередь нарушает принцип...	<b>b. YAGNI</b> 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 <b>d. Fail Fast</b>
50.	К чему призывает принцип "Information Hiding" (принцип скрытия информации)?	<b>a. К проектированию с учетом только интерфейсов/контрактов модулей</b> b. К проектированию с учетом особенностей 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 (Нетерпеливое) <b>d. Interdeveloper (Коллективное)</b>
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 <b>c. Нужно проектировать программы с учетом общепринятых представлений о разработке</b> d. Необходимо не допускать дублирования информации
53.	Переусложнение программного решения, в первую очередь нарушает принцип	a. DRY <b>b. KISS</b> c. YAGNI d. Принцип наименьшего удивления
54.	Принцип ортогональности (Orthogonality) призывает	a. Решать задачу наиболее простым способом



		<b>b. Сокращать взаимное влияние между несвязанными вещами</b> 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 <b>c. Избегать дублирования информации</b> 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 <b>c. По написанному коду очень сложно понять, какую задачу он решает</b> d. Модуль в своей работе опирается на детали реализации другого модуля
57.	Дублирование вызванное особенностями инструментов разработки (средой исполнения, языком) называется	<b>a. Imposed (Вынужденное)</b> b. Inadvertent (Неосознанное, неумышленное) c. Impatient (Нетерпеливое) d. Interdeveloper (Коллективное)
58.	Проектирование с учетом предполагаемых, но не утвержденных требований нарушает	a. DRY b. KISS <b>c. YAGNI</b> d. Intentionality
59.	Какое свойство 'принципа' отличает принцип от банальности?	a. Абстрактность <b>b. Опровергаемость</b> c. Непременимость d. Однозначность
60.	Абстрактность по отношению к принципу означает, что...	<b>a. Принцип не может быть применим немедленно</b> 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 управляющего флага <b>с. При нарушении принципа Information Hiding/инкапсуляции</b> d. Когда один модуль взаимодействует с другим через внешний файл определенного формата
62.	Common coupling возникает...	<b>a. При взаимодействии двух модулей через глобальную переменную</b> b. Когда один модуль меняет поведение другого, путем передачи управляющего 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 управляющего флага c. При нарушении принципа Information Hiding/инкапсуляции <b>d. Когда один модуль взаимодействует с другим через внешний файл определенного формата</b>
64.	Каким в идеале должен быть coupling?	a. Не важно b. Средним c. Высоким <b>d. Низким</b>
65.	Каким в идеале должен быть cohesion?	a. Не важно b. Средним



		<b>с. Высоким</b> <b>d. Низким</b>
66.	Coincidental cohesion означает, что компоненты объединены в модуль	a. потому что относятся к одной логической категории <b>b. без какой либо логики (случайно)</b> c. потому что вместе решают какую-либо хорошо определенную задачу d. потому что всегда вызываются друг за другом
67.	Procedural cohesion означает, что компоненты объединены 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 <b>d. потому что всегда вызываются друг за другом</b>
68.	Logical cohesion означает, что компоненты объединены в модуль	<b>a. потому что относятся к одной логической категории</b> 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 объединены в модуль	a. потому что относятся к одной логической категории b. без какой либо логики (случайно) <b>c. потому что вместе решают какую-либо хорошо определенную задачу</b> d. потому что всегда вызываются друг за другом
70.	Функция, принимающая другие функции в качестве аргумента, называется	a. Чистой (Pure) <b>b. Высшего порядка (High-ordered)</b> c. Рекурсивной (Recursion) d. Неизменяемой (Immutable)
71.	Возможность перегрузки операторов в C#/C++ и других языках является примером	a. Overloading polymorphism <b>b. Parametric Polymorphism</b>



		c. Subtype Polymorphism d. Multiple inheritance
72.	Полиморфизм подтипов (Subtype polymorphism) реализуется с помощью	a. Инкапсуляции <b>b. Наследования интерфейса</b> c. Агрегации d. Композиции
73.	Отношение IS-A в ООП моделируется с помощью	a. Инкапсуляции <b>b. Наследования</b> c. Агрегации d. Композиции
74.	Отношение HAS-A в ООП моделируется с помощью	a. Инкапсуляции b. Наследования <b>c. Агрегации и Композиции</b> d. Абстракции



## 4. Порядок проведения и критерии оценивания промежуточной аттестации

### 4.1. Порядок проведения промежуточной аттестации

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.

### 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/ Не зачтено
Баллы	100-86 баллов	85-76 баллов	75-60 баллов	59-0 баллов
Уровень освоения проверяемых компетенций	высокий	средний	базовый	недостаточный

### 4.3. Результаты промежуточной аттестации и уровни сформированности компетенций

Для получения «удовлетворительно» обучающийся должен защитить все практические задания для текущей аттестации и выполнить итоговый контрольный тест как минимум на 60%.

Для получения «хорошо» обучающийся должен защитить все практические задания для текущей аттестации и выполнить итоговый контрольный тест как минимум на 76%.

Для получения «отлично» обучающийся должен защитить все практические задания для текущей аттестации и выполнить итоговый контрольный тест как минимум на 86%.

Особенности проведения 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.