Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю)
«Объектно-ориентированные технологии»

Направление подготовки (специальность)
02.04.02 «Фундаментальная информатика и информационные технологии»

Направленность (профиль)
«Робототехника»

Присваиваемая квалификация
Магистр

Форма обучения
Очная

Челябинск, 2025 г.

Содержание

1. Паспорт фонда оценочных средств

Направление подготовки: 02.04.02 Фундаментальная информатика и информационные технологии.
Направленность (профиль): Робототехника.
Дисциплина: Объектно-ориентированные технологии.
Семестры: 3.
Форма промежуточной аттестации: экзамен в 3 семестре.
Для оценивания результатов обучения используется балльно-рейтинговая система.

2. Перечень формируемых компетенций

Изучение дисциплины «**Объектно-ориентированные технологии**» направлено на формирование компетенций, приведённых в Таблице 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
ПК-3	Способность применять методы и средства информационных технологий при исследованиях и информационно-технологических разработках робототехнических систем, их подсистем, включая информационно-сенсорные	ПК-3.1. Демонстрирует знание имеющихся программных пакетов и нового программного обеспечения, необходимого для обработки информации в робототехнических системах, а также для их проектирования; методов проектирования и разработки программного обеспечения, необходимого для обработки информации в робототехнических системах. ПК-3.2. Демонстрирует умения проектировать и разрабатывать программное обеспечение, необходимое для обработки информации в робототехнических системах; применять методы и средства информационных технологий при выполнении научно-исследовательских или информационно-технологических проектов в области обработки информации в робототехнических системах. ПК-3.3. Имеет навыки разработки программного обеспечения, необходимого для обработки информации в робототехнических системах.	**Знать:** теоретические основы проектирования программного обеспечения с использованием объектно-ориентированных технологий в заданной предметной области. **Уметь:** использовать инструменты описания для объектного моделирования, выбирать и использовать шаблоны проектирования. **Владеть:** навыками проектирования программного обеспечения.

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
1	ПК-3 **Знать:** теоретические основы проектирования программного обеспечения с использованием объектно-ориентированных технологий в заданной предметной области. **Уметь:** использовать инструменты описания для объектного моделирования, выбирать и использовать шаблоны проектирования. **Владеть:** навыками проектирования программного обеспечения.	Инструменты объектно-ориентированного программирования	Практическая работа	Задания теста № 1-26 Вопросы для экзамена
2		Шаблоны проектирования	Практическая работа	Задания теста № 27-49 Вопросы для экзамена
3		Инструменты объектно-ориентированного программирования Шаблоны проектирования	Практическая работа	Задания теста № 50-55 Вопросы для экзамена

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

Промежуточная аттестация проводится в виде экзамена в 3 семестре.

Вопросы для экзамена:

1. Классы и экземпляры классов
2. Наследование
3. Наследование и контроль доступа
4. Приведение указателя на класс
5. Затенение полей и методов
6. Использование конструкторов
7. Деструкторы
8. Замещение виртуальных методов
9. Класс с виртуальными функциями как интерфейс
10. Виртуальный деструктор
11. Абстрактный метод

12. Язык UML
13. Шаблон проектирования «Стратегия»
14. Шаблон проектирования «Наблюдатель»
15. Шаблон проектирования «Декоратор»
16. Шаблон проектирования «Абстрактная фабрика»
17. Шаблон проектирования «Одиночка»
18. Шаблон проектирования «Команда»
19. Шаблон проектирования «Адаптер»
20. Шаблон проектирования «Шаблонный метод»

Задания практических работ:

1. Написать приложение с использованием шаблона проектирования «Стратегия».
2. Написать приложение с использованием шаблона проектирования «Наблюдатель».
3. Написать приложение с использованием шаблона проектирования «Декоратор».
4. Написать приложение с использованием шаблона проектирования «Абстрактная фабрика».
5. Написать приложение с использованием шаблона проектирования «Команда».
6. Написать приложение с использованием шаблона проектирования «Адаптер».
7. Написать приложение с использованием шаблона проектирования «Шаблонный метод».

База тестовых вопросов:

п/п	Формулировка вопроса	Варианты ответов (полужирным шрифтом – верные варианты)
1.	Абстрактный класс это:	**a. Класс, имеющий хотя бы один абстрактный метод** b. Класс, предок которого имеет хотя бы один абстрактный метод c. Класс, предок которого имеет исключительно абстрактные методы d. Класс, у которого все унаследованные абстрактные методы реализованы
2.	Виртуальный деструктор необходимо использовать, если:	**a. Класс будет удаляться через интерфейс (или через указатель на предка)** b. Класс использует виртуальный конструктор c. У класса нет предков
3.	Виртуальный метод:	**a. Замещается в классе потомке** **b.** Затеняется в классе потомке **c.** Не может быть абстрактным
4.	Модификатор доступа private для полей и методов означает:	a. Доступен всем b. Доступен только методам самого класса и методам всех порожденных от него классов c. **Доступен только внутри класса**
5.	Модификатор доступа private при наследовании означает:	a. Поля и методы (public и protected) наследуются с тем же доступом b. public и protected в базовом классе становятся protected c. **Поля и методы (**public **и protected) наследуются с доступом private**

6.	Модификатор доступа protected для полей и методов означает:	a. Доступен всем b. **Доступен только методам самого класса и методам всех порожденных от него классов** c. Доступен только внутри класса
7.	Модификатор доступа protected при наследовании означает:	a. Поля и методы (public и protected) наследуются с тем же доступом b. **public и protected в базовом классе становятся protected** c. Поля и методы (public и protected) наследуются с доступом private
8.	Модификатор доступа public для полей и методов означает:	**a. Доступен всем** **b.** Доступен только методам самого класса и методам всех порожденных от него классов **c.** Доступен только внутри класса
9.	Модификатор доступа public при наследовании означает:	**a. Поля и методы (public и protected) наследуются с тем же доступом** **b.** public и protected в базовом классе становятся protected **c.** Поля и методы (public и protected) наследуются с доступом privat
10.	Особенность абстрактного класса:	**a. Нельзя создавать экземпляры абстрактного класса** **b.** Абстрактный класс не может иметь виртуальные методы **c.** Экземпляр абстрактного класса занимает больше места в памяти
11.	Особенностью метода класса является:	**a. Не может использовать поля экземпляра** **b.** Можно использовать только вместе с экземпляром класса **c.** Не имеет возвращаемого значения **d.** Объявляется с ключевым словом virtual
12.	Особенностью поля класса является:	**a. Можно использовать, даже если не создан ни один экземпляр класса** **b.** Может быть изменено только методом класса **c.** Для такого поля выделяется память для каждого экземпляра класса **d.** Объявляется с ключевым словом virtual
13.	Особенностью поля экземпляра класса является:	a. Можно использовать, даже если не создан ни один экземпляр класса b. **Можно использовать только вместе с экземпляром класса** c. Для каждого экземпляра класса память под поле не выделяется d. Объявляется с ключевым словом virtual
14.	Особенностью конструктора является:	**a. При создании экземпляра потомка будут вызваны конструкторы предков** **b.** Нельзя использовать перегруженные конструкторы **c.** Конструктор в C++ объявляется с ключевым словом const

15.	Преимущество виртуального метода:	**a. Можно использовать класспредок как интерфейс для доступа к потомкам** **b.** Виртуальный метод выполняется быстрее **c.** Виртуальный метод не замещается наследником, а затеняется
16.	Достоинство паттерна "абстрактная фабрика":	a. Нет необходимости создавать подклассы (наследников) для расширения функциональности объекта b. Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам c. Широковещательная рассылка сообщений группе объектов **d. Гарантирует сочетаемость продуктов**
17.	Достоинство паттерна "декоратор":	**a. Нет необходимости создавать подклассы (наследников) для расширения функциональности объекта** **b.** Однократное **использование** инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам **c.** Широковещательная рассылка сообщений группе объектов
18.	Достоинство паттерна "команда":	**a. Классы, выполняющие конкретные действия могут быть реализованы без учета особенностей их использования** b. Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам c. Широковещательная рассылка сообщений группе объектов
19.	Достоинство паттерна "наблюдатель":	a. Позволяет создавать группы взаимосвязанных объектов b. Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам **c**. Широковещательная рассылка сообщений группе объектов
20.	Достоинство паттерна "одиночка":	a. Позволяет создавать группы взаимосвязанных объектов b. Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам c. Широковещательная рассылка сообщений группе объектов d. Экземпляр **создается только в случае необходимости**
21.	Достоинство паттерна "стратегия":	**a. Вызов всех алгоритмов из семейства одним стандартным образом** b. Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам c. Широковещательная рассылка сообщений группе объектов

22.	Недостаток паттерна "абстрактная фабрика":	a. Фабрика и создаваемый объект не совместимы b. Проблемы при реализации многопоточного приложения **c. При создании нового продукта необходимо также создавать фабрику**
23.	Недостаток паттерна "декоратор":	**a. Декоратор и его компонент не идентичны** b. Проблемы при реализации многопоточного приложения c. Сложно добавить дополнительную функциональность к декорируемому объекту
24.	Недостаток паттерна "наблюдатель":	a. Наблюдатель теряет часть функциональности b. Проблемы при реализации многопоточного приложения **c. Плохо предсказуемая рассылка сообщений**
25.	Недостаток паттерна "одиночка":	**a. Накладные расходы на синхронизацию в многопоточных приложениях** b. Создание дополнительных классов c. Объект не всегда обладает достаточной функциональностью
26.	Недостаток паттерна "стратегия":	**a. Создание дополнительных классов** b. Проблемы при реализации многопоточного приложения c. Плохо предсказуемая рассылка сообщений
27.	Паттерн "стратегия" можно использовать, если:	**a. Имеется семейство родственных алгоритмов и необходимо выбирать тот или иной алгоритм во время выполнения** **b.** Может потребоваться добавить к объекту некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта **c.** Необходимо реализовать отложенный вызов
28.	Паттерн "абстрактная фабрика" можно использовать, если:	a. Имеется семейство родственных алгоритмов и необходимо выбирать тот или иной алгоритм во время выполнения b. Может потребоваться добавить к объекту некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта c. Существует один объект, рассылающий сообщения и несколько получателей d. Должен быть ровно один экземпляр некоторого класса, легко доступный всем клиентам e. **Входящие в семейство взаимосвязанные объекты должны использоваться вместе**

29.	Паттерн "абстрактная фабрика" предназначен для:	a. Определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости b. Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом **d. Создания групп взаимосвязанных объектов**
30.	Паттерн "адаптер" предназначен для:	a. Предоставления глобальной точки доступа к единственному экземпляру класса b. Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом d. Динамического подключения дополнительного поведения к объекту e. **Организации использования функций объекта, недоступного для модификации, через специально созданный интерфейс**
31.	Паттерн "декоратор" можно использовать, если:	a. Имеется семейство родственных алгоритмов и необходимо выбирать тот или иной алгоритм во время выполнения b. **Может потребоваться добавить к объекту некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта** c. Существует один объект, рассылающий сообщения и несколько получателей d. Должен быть ровно один экземпляр некоторого класса, легко доступный всем клиентам
32.	Паттерн "декоратор" предназначен для:	a. Определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости b. Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом **d. Динамического подключения дополнительного поведения к объекту**

33.	Паттерн "команда" можно использовать, если:	a. Имеется семейство родственных алгоритмов и необходимо выбирать тот или иной алгоритм во время выполнения b. Может потребоваться добавить к объекту некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта c. Существует один объект, рассылающий сообщения и несколько получателей d. **Необходимо реализовать отложенный вызов**
34.	Паттерн "команда" предназначен для:	a. Определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости b. Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом d. **Инкапсуляции всей информации, необходимой для дальнейшего (отложенного) вызова метода**
35.	Паттерн "наблюдатель" можно использовать, если:	a. Имеется семейство родственных алгоритмов и необходимо выбирать тот или иной алгоритм во время выполнения b. Может потребоваться добавить к объекту некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта c. **Существует один объект, рассылающий сообщения и несколько получателей**
36.	Паттерн "наблюдатель" предназначен для:	a. Определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости **b. Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии** c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом
37.	Паттерн "одиночка" можно использовать, если:	a. Имеется семейство родственных алгоритмов и необходимо выбирать тот или иной алгоритм во время выполнения b. Может потребоваться добавить к объекту некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта c. Существует один объект, рассылающий сообщения и несколько получателей

		d. Должен быть ровно один экземпляр некоторого класса, легко доступный всем клиентам
38.	Паттерн "одиночка" предназначен для:	**a. Предоставления глобальной точки доступа к единственному экземпляру класса** **b.** Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии **c.** Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом **d.** Динамического подключения дополнительного поведения к объекту
39.	Паттерн "стратегия" предназначен для:	**a. Определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости** b. Инкапсуляции информации, необходимой для дальнейшего (отложенного) вызова метода c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом
40.	Паттерн "шаблонный метод" предназначен для:	a. Определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости b. Определения зависимости типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии **c. Определения основы алгоритма таким образом, чтобы позволить наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом** **d.** Динамического подключения дополнительного поведения к объекту
41.	class test1{public: static int Stop(){return 15;}}; Stop -это:	a. Поле класса b. Поле экземпляра класса c. **Метод класса** d. Метод экземпляра класса
42.	class test1{public: int Stop(){return 15;}}; Stop -это:	a. Поле класса b. Поле экземпляра класса c. Метод класса d. **Метод экземпляра класса**

43.	class parent{public:int x;};class child :private parent{int y;void Func1(){x=3;};}; Метод Func1:	**a. Имеет доступ к полю x, программа откомпилируется** **b.** Не имеет доступ к полю x, так как используется модификатор доступа private при наследовании **c.** Может получить доступ к полю x, но только через приведение типов
44.	class parent{public:int x;};class child :public parent{public:int x;}; В этом примере:	**a. Экземпляр класса child имеет два поля x** **b.** Ошибка, поле x уже объявлено в предке **c.** c. Нет ошибки, но объявление поля x в потомке не имеет смысла
45.	class parent{private:int x;};class child :public parent{public:int y;}; В этом примере:	**a. Экземпляр класса child содержит поле x, но не имеет к нему доступа** b. Экземпляр класса child не содержит поле x, так как оно объявлено в предке как private c. Ошибка, в данном случае нельзя использовать модификатор public при наследовании
46.	class test{public: static double fuel;}; fuel - это:	**a. Поле класса** **b.** Поле экземпляра класса **c.** Метод класса **d.** Метод экземпляра класса
47.	class test{public:static int x;}; int test::x=0; int _tmain(int argc, _TCHAR* argv[]){ test a,b; a.x=10; b.x=15; printf("%d",a.x);return 0;} Что программа выведет на экран?	**a. 15** **b.** 10 **c.** 0 **d.** 25
48.	class test{public: double fuel;}; fuel - это:	a. Поле класса b. **Поле экземпляра класса** c. Метод класса d. Метод экземпляра класса
49.	class test{public:int x;}; int _tmain(int argc, _TCHAR* argv[]){test a,b;a.x=10;b.x=15;printf("%d",a.x);return 0;} Что программа выведет на экран?	a. 15 b. **10** c. 0 d. 25

50.	Программа обрабатывает массив чисел. При обработке массив должен пройти через ряд фильтров, каждый из которых модифицирует числа в массиве. Набор фильтров может меняться, но в каждом конкретном случае фильтры должны применяться в строго определенном порядке. Выберите наиболее подходящий паттерн:	a. Адаптер b. Наблюдатель c. Одиночка d. **Декоратор**
51.	В вашей программе реализованы разнородные классы, выполняющие ряд полезных действий. Вызов этих действий может происходить из разных частей программы, при этом некоторые действия будут вызываться вместе по очереди (в виде макроса). Выберите наиболее подходящий паттерн:	a. Стратегия b. Наблюдатель c. Одиночка d. **Команда**
52.	В программе-симуляторе танкового сражения, которую вы проектируете, игрок сможет выделить группу танков и командовать одновременно всей выделенной группой. Выберите наиболее подходящий паттерн для реализации такой функциональности:	a. Стратегия b. **Наблюдатель** c. Одиночка d. Декоратор
53.	В Вашей программе реализована абстрактная фабрика, особым образом создающая объекты на основе статистики. Статистика хранится внутри объекта фабрики, и поэтому должны выполняться два условия. В процессе выполнения программы должен использоваться единственный экземпляр фабрики. Если фабрика пока не используется, создавать ее не нужно. Выберите наиболее подходящий паттерн для выполнения этих условий:	a. Стратегия b. Наблюдатель c. **Одиночка** d. Декоратор
54.	В программе-симуляторе, которую вы проектируете будут: пассажирские сверхзвуковые самолеты истребители крылатые ракеты пассажирские дозвуковые самолеты в дальнейшем возможно будут добавлены другие летательные аппараты Алгоритм полета пассажирских сверхзвуковых самолетов и истребителей один и тот же, отличаются только входные параметры, например, масса и мощность двигателя. Для повторного использования этого алгоритма можно использовать паттерн:	**a. Стратегия** **b.** Наблюдатель **c.** Одиночка
55.	В проектируемой Вами игре для персонажа будут создаваться дополнительные объекты (амуниция, оружие, спутники помощники и т.д.). При этом для каждого типа персонажей это будут свои объекты (например, для рыцаря нельзя создавать лук, а для лучника нельзя создавать меч). Выберите наиболее подходящий паттерн:	a. Стратегия b. Наблюдатель c. Одиночка d. **Абстрактная фабрика**

4. Порядок проведения и критерии оценивания промежуточной аттестации

4.1. Порядок проведения промежуточной аттестации

Промежуточная аттестация проходит в форме экзамена.

Экзамен проводится в виде тестирования. Студент должен ответить на вопросы закрытого типа, которые предполагают выбор вариантов ответа. Всего 20 тестовых вопросов. Продолжительность теста – 35 минут.

4.2. Критерии оценивания промежуточной аттестации по видам оценочных средств

Экзамен проводится в виде тестирования. Всего 20 тестовых вопросов. Продолжительность теста – 35 минут. Тест формируется в системе электронного обучения MOODLE. Максимальный балл за тест – 100 баллов.

4.3. Результаты промежуточной аттестации и уровни сформированности компетенций

Итоговая оценка выставляется по 100-балльной шкале, исходя из полученной суммы баллов за вопросы:

От 0 до 59 баллов – «неудовлетворительно».

От 60 до 74 баллов – «удовлетворительно».

От 75 до 89 баллов – «хорошо».

От 90 баллов – «отлично».

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. Продвинутый уровень сформированности компетенций соответствует оценке «отлично»:

Обучающийся владеет знаниями предмета в полном объеме учебной программы, достаточно глубоко осмысливает дисциплину; самостоятельно, в логической последовательности и исчерпывающе отвечает на все вопросы, подчеркивает при этом самое существенное, умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделять в нем главное: устанавливать причинно-следственные связи; четко формирует ответы.

2. Базовый уровень соответствует оценке «хорошо»:

Обучающийся владеет знаниями дисциплины почти в полном объеме программы (имеются пробелы знаний только в некоторых, особенно сложных разделах); самостоятельно и отчасти при наводящих вопросах дает полноценные ответы на вопросы; не всегда выделяет наиболее существенное, не допускает вместе с тем серьезных ошибок в ответах.

3. Пороговый уровень соответствует оценке «удовлетворительно»:

Обучающийся владеет основным объемом знаний по дисциплине; проявляет затруднения в самостоятельных ответах, оперирует неточными формулировками; в процессе ответов допускает ошибки по существу вопросов.

4. Низкий уровень соответствует оценке «неудовлетворительно»:

Обучающийся не освоил обязательного минимума знаний предмета, не способен ответить на вопросы даже при дополнительных наводящих вопросах экзаменатора.

02.04.02 Фундаментальная информатика и информационные технологии Направленность (профиль) Робототехника, ФОС Объектно-ориентированные технологии, 2025 г.н., очная форма обучения

Проректор по учебной работе утверждено 24.02.25 А.А. Саламатов

Ученым советом математического факультета

Протокол заседания № 6 от 20.02.2025

Председатель Ученого совета
математического факультета согласовано Е.А. Сбродова

Заседанием кафедры вычислительной механики и информационных технологий

Протокол заседания № 6 от 30.01.2025

Заведующий кафедрой согласовано М.В. Плеханова

Автор (составитель) А.С. Авилович

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