Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Структурное подразделение «Сибирская школа геонаук (119)»

УТВЕРЖДЕНА:
на заседании ДОТ
Протокол №29 от 10 апреля 2025 г.

Рабочая программа дисциплины

«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ / OBJECT-ORIENTED PROGRAMMING»

Направление: 09.03.02 Информационные системы и технологии

Информационные технологии в науках о Земле и окружающей среде / Information Technologies in Earth and Environmental Sciences

Квалификация: Бакалавр

Форма обучения: очная

Документ подписан простой электронной подписью Составитель программы: Ланько Анна Викторовна Дата подписания: 14.12.2025	Документ подписан простой электронной подписью Утвердил: Ланько Анна Викторовна Дата подписания: 14.12.2025	Документ подписан простой электронной подписью Согласовал: Паршин Александр Вадимович Дата подписания: 13.01.2026

Год набора – 2025

Иркутск, 2025 г.

1 Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы

1.1 Дисциплина «Объектно-ориентированное программирование / Object-Oriented Programming» обеспечивает формирование следующих компетенций с учётом индикаторов их достижения

Код, наименование компетенции	Код индикатора компетенции
ПКС-4 Способность управлять проектами в области информационных технологий, обеспечивая выполнение всех этапов проекта в рамках утвержденных параметров	ПКС-4.1, ПКС-4.2

1.2 В результате освоения дисциплины у обучающихся должны быть сформированы

Код индикатора	Содержание индикатора	Результат обучения
ПКС-4.1	Знать базовые элементы конфигурации информационной системы	**Знать** базовые элементы конфигурации информационной системы в объектно-ориентированном программировании. **Уметь** применять ООП для управления проектами ИТ, обеспечивая выполнение этапов в рамках параметров. **Владеть** навыками проектирования и реализации конфигураций ИС с использованием принципов ООП.
ПКС-4.2	Способность идентифицировать базовые элементы конфигурации информационной системы в соответствии с проектным планом	**Знать** базовые элементы конфигурации информационной системы и их идентификацию в соответствии с проектным планом. **Уметь** идентифицировать элементы конфигурации ИС на основе проектного плана в объектно-ориентированном программировании. **Владеть** методами анализа и идентификации компонентов ИС для обеспечения соответствия проектным параметрам.

2 Место дисциплины в структуре ООП

Изучение дисциплины «Объектно-ориентированное программирование / Object-Oriented Programming» базируется на результатах освоения следующих дисциплин/практик: «Программирование на языке высокого уровня / Programming in High-Level Languages»

Дисциплина является предшествующей для дисциплин/практик: «Объектно-ориентированные технологии и шаблоны проектирования /Object-Oriented Technologies and Design Patterns»

3 Объем дисциплины

Объем дисциплины составляет – 3 ЗЕТ

Вид учебной работы	**Трудоемкость в академических часах** (Один академический час соответствует 45 минутам астрономического часа)	
	Всего	Семестр № 5
Общая трудоемкость дисциплины	108	108
Аудиторные занятия, в том числе:	45	45
лекции	15	15
лабораторные работы	30	30
практические/семинарские занятия	0	0
Самостоятельная работа (в т.ч. курсовое проектирование)	36	36
Трудоемкость промежуточной аттестации	27	27
Вид промежуточной аттестации (итогового контроля по дисциплине)	Экзамен	Экзамен

4 Структура и содержание дисциплины

4.1 Сводные данные по содержанию дисциплины

Семестр № 5

№ п/п	**Наименование раздела и темы дисциплины**	**Виды контактной работы**						СРС		**Форма текущего контроля**
		Лекции		ЛР		ПЗ(СЕМ)				
		№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	
1	2	3	4	5	6	7	8	9	10	11
1	1. Введение в объектно-ориентированное программирование (ООП)	1	1					5	5	Устный опрос
2	2. Классы и объекты	2	1	1	4					Устный опрос
3	3.Конструкторы и деструкторы	3	1					2	6	Устный опрос
4	4. Наследование	4	1	2	6			3	5	Устный опрос
5	5. Полиморфизм	5	1					4	5	Устный опрос
6	6. Абстракция и абстрактные классы	6	2	3	4					Устный опрос
7	7. Инкапсуляция и модификаторы доступа	7	2					5	5	Устный опрос
8	8. Статические члены и	8	2	4	6			4	5	Устный опрос

	вложенные классы									
9	9. Обработка исключений в ООП	9	2	5	4			1	5	Устный опрос
10	10. Шаблоны проектирования в ООП	10	2	6	6					Устный опрос
	Промежуточная аттестация								27	Экзамен
	Всего		15		30				63	

4.2 **Краткое содержание разделов и тем занятий**

Семестр № 5

№	Тема	Краткое содержание
1	1. Введение в объектно-ориентированное программирование (ООП)	Основные понятия ООП, парадигмы программирования, преимущества ООП для информационных систем (ИС), обзор языков ООП (C++, Java, Python).
2	2. Классы и объекты	Определение классов, создание объектов, атрибуты и методы, инкапсуляция данных в контексте конфигурации ИС.
3	3.Конструкторы и деструкторы	Инициализация объектов, конструкторы по умолчанию и с параметрами, управление жизненным циклом объектов в проектах ИТ.
4	4. Наследование	Иерархии классов, базовые и производные классы, переопределение методов, применение в модульной конфигурации ИС.
5	5. Полиморфизм	Статический и динамический полиморфизм, виртуальные функции, интерфейсы, роль в гибкой архитектуре ИС.
6	6. Абстракция и абстрактные классы	Абстрактные классы и методы, чистые виртуальные функции, проектирование шаблонов для идентификации элементов ИС.
7	7. Инкапсуляция и модификаторы доступа	Public, private, protected, геттеры и сеттеры, обеспечение безопасности данных в конфигурации ИС.
8	8. Статические члены и вложенные классы	Статические поля и методы, вложенные и локальные классы, их использование в управлении ИТ-проектами.
9	9. Обработка исключений в ООП	Иерархия исключений, try-catch-файналли, пользовательские исключения, надежность конфигурации ИС.
10	10. Шаблоны проектирования в ООП	Singleton, Factory, Observer; применение для идентификации и управления базовыми элементами конфигурации ИС в соответствии с проектным планом.

4.**3 Перечень лабораторных работ**

Семестр № 5

№	Наименование лабораторной работы	Кол-во академических часов
1	Лабораторная работа №1. Создание классов и объектов в Python	4
2	Лабораторная работа №2. Реализация наследования для моделирования компонентов ИС	6
3	Лабораторная работа №3. Разработка абстрактных классов для проектных компонентов	4
4	Лабораторная работа №4. Применение статических членов в модели ИС	6
5	Лабораторная работа №5. Система обработки исключений для элементов ИС	4
6	Лабораторная работа №6. Реализация шаблона Factory для конфигурации ИС	6

4.4 Перечень практических занятий

Практических занятий не предусмотрено

4.5 Самостоятельная работа

Семестр № 5

№	Вид СРС	Кол-во академических часов
1	Оформление отчетов по лабораторным и практическим работам	5
2	Подготовка к зачёту	6
3	Подготовка к практическим занятиям (лабораторным работам)	5
4	Подготовка к сдаче и защите отчетов	10
5	Проработка разделов теоретического материала	10

В ходе проведения занятий по дисциплине используются следующие интерактивные методы обучения: работы в малых группах

5 Перечень учебно-методического обеспечения дисциплины

5.1 Методические указания для обучающихся по освоению дисциплины

5.1.1 Методические указания для обучающихся по лабораторным работам:

Методические указания по лабораторным работам
Лабораторная работа №1. Создание классов и объектов в Python
Цель работы: Освоить создание классов и объектов в объектно-ориентированном программировании (ООП) для моделирования базовых элементов конфигурации информационной системы (ИС).
Теоретическая справка:
Класс в ООП представляет шаблон для создания объектов, объединяющий данные (атрибуты) и операции над ними (методы). В Python классы определяются ключевым словом class, а объекты создаются вызовом класса как функции. Инкапсуляция

обеспечивает сокрытие внутренней реализации, что критично для конфигурации ИС, где компоненты (базы данных, серверы, клиенты) должны взаимодействовать через четкие интерфейсы. Атрибуты делятся на экземплярные (принадлежат конкретному объекту) и классовые (общие для всех объектов). Метод __init__ выступает конструктором, инициализирующим объект. В контексте ИТ-проектов классы моделируют элементы конфигурации: класс Server может хранить IP-адрес, порт и методы подключения; класс Database — схему данных и операции CRUD (Create, Read, Update, Delete). Это позволяет идентифицировать базовые элементы ИС в соответствии с проектным планом, обеспечивая модульность и повторное использование кода. Синтаксис Python упрощает ООП: self ссылается на экземпляр, методы начинаются с def внутри класса.
Пример базового класса:

```
class Component:
    def __init__(self, name, version):
        self.name = name
        self.version = version
    def get_info(self):
        return f"{self.name} v{self.version}"
```

Ход выполнения работы:
1. Откройте IDE (PyCharm, VS Code) и создайте новый файл lab1.py.
2. Определите класс ISComponent с атрибутами: name (название), type (тип: 'server', 'db', 'client'), status ('active', 'inactive').
3. Реализуйте метод __init__ для инициализации атрибутов и метод get_status() возвращающий строку с информацией о статусе.
4. Создайте три объекта: сервер, базу данных, клиент; выведите их статусы.
5. Добавьте метод activate(), меняющий статус на 'active' и возвращающий подтверждение.
6. Протестируйте активацию объектов и выведите результаты.
7. Сохраните код, запустите и зафиксируйте вывод в отчете.

Ожидаемый результат:
Работающий скрипт, создающий и управляющий объектами элементов ИС с выводом:

```
Server 'WebServer' status: active
Database 'UserDB' status: inactive
Client 'AdminClient' status: active
Server activated successfully
```

Контрольные вопросы:
1. В чем разница между атрибутами экземпляра и класса?
2. Зачем нужен параметр self в методах класса?
3. Как инкапсуляция помогает в конфигурации ИС?

Лабораторная работа №2. Реализация наследования для моделирования компонентов ИС
Цель работы: Научиться использовать наследование для создания иерархии классов, моделирующих компоненты конфигурации информационной системы.
Теоретическая справка:
Наследование позволяет создавать производные классы на базе базовых, наследуя атрибуты и методы, что упрощает проектирование сложных ИС. В Python наследование указывается в скобках после класса: class Derived(Base). Производный класс может переопределять методы (полиморфизм) или расширять их через super(). Иерархия отражает структуру ИС: базовый класс NetworkComponent для всех сетевых элементов, производные — WebServer, LoadBalancer. Это обеспечивает идентификацию элементов по

проектному плану: общие свойства (IP, порт) наследуются, специфические (балансировка нагрузки) добавляются. Множественное наследование (от нескольких базовых) требует разрешения конфликтов через MRO (Method Resolution Order). В ИТ-проектах наследование снижает дублирование кода на 30-50%, повышая поддерживаемость. super().init() вызывает конструктор родителя.
Пример:

```
class NetworkComponent:
    def __init__(self, ip):
        self.ip = ip

class WebServer(NetworkComponent):
    def __init__(self, ip, port):
        super().__init__(ip)
        self.port = port
```

Ход выполнения работы:
1. Создайте файл lab2.py.
2. Определите базовый класс ISComponent с атрибутами id, version и методом describe().
3. Создайте производный класс Server(ISComponent) с дополнительными атрибутами ip, port и переопределенным describe().
4. Создайте еще один производный класс Database(ISComponent) с атрибутами capacity, engine.
5. В основном коде создайте по одному объекту каждого класса, вызовите describe() для каждого.
6. Добавьте метод connect() в базовый класс и переопределите его в производных.
7. Протестируйте подключение и сохраните результаты.

Ожидаемый результат:
Вывод описаний и подключений:
Server ID:001 v1.0, IP:192.168.1.1:8080 connected
Database ID:002 v2.1, Capacity:500GB, MySQL connected

Контрольные вопросы:
1. Что делает super() в наследовании?
2. Как наследование помогает идентифицировать элементы ИС?
3. В чем преимущество иерархического наследования?

Лабораторная работа №3. Разработка абстрактных классов для проектных компонентов
Цель работы: Освоить абстрактные классы для задания интерфейсов элементов конфигурации ИС в соответствии с проектным планом.
Теоретическая справка:
Абстрактные классы (Abstract Base Classes, ABC) определяют интерфейс, который должны реализовать производные классы, используя модуль abc. Декоратор @abstractmethod делает метод обязательным для реализации. Это обеспечивает единообразие в ИС: абстрактный ConfigurableComponent требует метод configure(plan), где plan — проектный план. В Python: from abc import ABC, abstractmethod. Абстрактные классы нельзя инстанцировать, что предотвращает неполную реализацию. В ИТ-проектах ABC гарантируют соответствие элементов плану: все компоненты имеют validate_plan(). Это ключ к идентификации базовых элементов — контракт определяет минимальный набор методов.
Пример:

```
from abc import ABC, abstractmethod
class Configurable(ABC):
    @abstractmethod
    def configure(self, plan):
        pass
```

Ход выполнения работы:
1. Создайте lab3.py, импортируйте abc.
2. Определите абстрактный класс ISModule(ABC) с абстрактными методами configure(plan) и validate().
3. Реализуйте производный AuthModule(ISModule) с логикой аутентификации.
4. Реализуйте StorageModule(ISModule) с проверкой плана на объем.
5. Создайте объекты, передайте тестовый план (словарь {'budget':1000, 'users':100}), вызовите методы.
6. Добавьте is_valid_plan(plan) в базовый класс.
7. Протестируйте и отладьте.

Ожидаемый результат:
AuthModule configured for 100 users. Valid: True
StorageModule requires 200GB, plan budget exceeded. Valid: False

Контрольные вопросы:
1. Почему абстрактный класс нельзя инстанцировать?
2. Как ABC обеспечивает соответствие проектному плану?
3. Зачем нужны абстрактные методы?

Лабораторная работа №4. Применение статических членов в модели ИС
Цель работы: Научиться использовать статические члены для управления глобальными ресурсами конфигурации ИС.
Теоретическая справка:
Статические члены (staticmethod, classmethod) принадлежат классу, а не экземпляру. @staticmethod не принимает self/cls, @classmethod — cls для работы с классом. Статические поля — через ClassName.var. В ИС статика моделирует реестры: ComponentRegistry отслеживает все экземпляры. get_instance_count() — статический метод. Это упрощает идентификацию элементов: центральный счетчик компонентов по плану. В многопоточных ИТ-проектах статика требует блокировок.
Пример:

```
class Registry:
    count = 0
    @staticmethod
    def get_count():
        return Registry.count
```

Ход выполнения работы:
1. Файл lab4.py.
2. Класс ISRegistry со статическим полем total_components = 0.
3. Статический метод get_total() и @classmethod def register(cls, name).
4. В __init__ каждого компонента инкремент total_components.
5. Создайте 5 компонентов разных типов, выводите счетчик после каждого.
6. Добавьте @staticmethod def validate_limit(max_count).
7. Тестируйте превышение лимита.

Ожидаемый результат:

Registered: Server1, Total: 1
Registered: DB1, Total: 5
Limit 5 exceeded!

Контрольные вопросы:
1. Разница между @staticmethod и @classmethod?
2. Когда использовать статику в ИС?
3. Как статика помогает в управлении проектами?

Лабораторная работа №5. Система обработки исключений для элементов ИС
Цель работы: Разработать иерархию исключений для обеспечения надежности конфигурации ИС.
Теоретическая справка:
Обработка исключений (try-except-finally-else) предотвращает крах системы. В ООП создаются пользовательские исключения через наследование Exception. Иерархия: ISConfigError → InvalidPlanError, ResourceLimitError. В ИС это критично: try: component.configure(plan) except InvalidPlanError. raise передает исключение. finally гарантирует очистку ресурсов. В проектах ИТ снижает downtime на 40%.
Пример:
```
class ConfigError(Exception):
    pass
class InvalidPlan(ConfigError):
    pass
```

Ход выполнения работы:
1. lab5.py.
2. Создайте ISConfigError(Exception), BudgetExceeded(ISConfigError), InvalidType(ISConfigError).
3. Класс Configurator с методом deploy(plan), бросающим исключения при ошибках.
4. В основном коде: try-deploy 3 планов, обработайте каждое исключение.
5. Добавьте finally: print("Cleanup").
6. Логируйте ошибки в файл.
7. Тестируйте.

Ожидаемый результат:
Deployment failed: BudgetExceeded - Need 2000, got 1000
Cleanup completed
All deployments logged

Контрольные вопросы:
1. Иерархия исключений — зачем?
2. Разница try-except-else-finally?
3. Как исключения обеспечивают надежность ИС?

Лабораторная работа №6. Реализация шаблона Factory для конфигурации ИС
Цель работы: Применить шаблон Factory для динамического создания элементов ИС по проектному плану.
Теоретическая справка:
Factory Method — порождающий шаблон ООП, делегирующий создание объектов подклассам. Абстрактный ISFactory с create_component(type, plan). Конкретные фабрики: ServerFactory, DBFactory. Позволяет идентифицировать и создавать элементы по плану без if-else цепочек. В ИС: factory.create('server', {'ip':'192.168.1.1'}). Расширяемо: новая

фабрика — новый тип без изменений клиента. В Python часто с dict-маппингом.
Пример:

```
class ComponentFactory:
   @staticmethod
   def create(type_, **kwargs):
      if type_ == 'server': return Server(**kwargs)
```

Ход выполнения работы:
1. lab6.py.
2. Абстрактный ComponentFactory(ABC) с @abstractmethod create_component(type, plan).
3. ServerFactory(ComponentFactory), DatabaseFactory.
4. Каждый создает соответствующий компонент, валидирует план.
5. ProjectManager использует фабрику по типу из плана.
6. Тест: план {'components': [('server', {'port':80}), ('db', {'size':100})]}.
7. Выводите созданные компоненты.

Ожидаемый результат:

```
Created Server on port 80 from ServerFactory
Created Database 100GB from DatabaseFactory
Configuration matches plan
```

Контрольные вопросы:
1. Преимущества Factory над прямым new?
2. Как шаблон помогает в ИТ-проектах?
3. Когда использовать Factory Method?

Хотите добавить примеры кода в приложения, скорректировать под конкретный язык или расширить контрольные вопросы?

5.1.2 Методические указания для обучающихся по самостоятельной работе:

Рекомендации по самостоятельной работе:
1. Рекомендации по самостоятельной подготовке к лабораторным работам
- Изучите теоретический материал по теме лабораторной работы.

Ознакомьтесь с учебниками, лекциями и дополнительными источниками, чтобы понимать цели и задачи работы, основные понятия и методы, используемые в лабораторном задании1.
- Внимательно ознакомьтесь с методическими указаниями и требованиями к лабораторной работе. Обратите внимание на последовательность выполнения этапов, используемое программное обеспечение, форматы исходных и выходных данных, требования к визуализации и анализу результатов.
- Подготовьте исходные данные. Проверьте наличие всех необходимых файлов, убедитесь в их корректности (форматы, структура, отсутствие ошибок и пропусков данных).
- Освойте необходимые функции и инструменты программного обеспечения.

Повторите работу с теми модулями и инструментами, которые будут использоваться в лабораторной работе.
- Планируйте время. Разделите выполнение работы на этапы: подготовка данных, выполнение анализа, оформление визуализации, написание отчета.

2. Рекомендации по оформлению отчетов по лабораторным работам
- Структурируйте отчет по стандартной схеме:
- Титульный лист (название работы, ФИО, группа, дата)

- Цель работы
- Краткое описание исходных данных
- Описание используемых методов и программного обеспечения
- Последовательное изложение этапов работы с иллюстрациями (скриншотами, графиками, картами)
- Анализ полученных результатов (выявленные особенности, сравнение с теорией, интерпретация)
- Выводы и рекомендации
- Список использованных источников
- Используйте качественные иллюстрации. Все графические материалы должны быть четкими, снабжены подписями, масштабами, легендами и пояснениями.
- Формулируйте выводы по существу. Кратко и ясно отражайте основные результаты работы, выявленные закономерности, достоинства и ограничения применяемых методов.
- Оформляйте отчет в соответствии с требованиями ДОТ. Соблюдайте стандарты оформления текста, таблиц, рисунков и ссылок на источники.

3. Рекомендации по самостоятельной проработке отдельных разделов тем

- Изучайте рекомендованную литературу и дополнительные источники. Используйте учебники, статьи, электронные ресурсы, профессиональные базы данных и справочные материалы, указанные в рабочей программе дисциплины1.
- Выполняйте конспектирование ключевых понятий и алгоритмов. Составляйте краткие записи по основным определениям, алгоритмам, этапам работы с ПО, особенностям визуализации и анализа данных.
- Практикуйтесь в самостоятельном выполнении типовых заданий. Решайте задачи, связанные с обработкой и визуализацией геолого-геофизических данных, используя различные программные средства.
- Формулируйте вопросы и уточнения для обсуждения на занятиях. Записывайте непонятные моменты, чтобы получить разъяснения у преподавателя или в ходе дискуссии.
- Анализируйте примеры из практики. Изучайте реальные кейсы решения задач геофизики, сравнивайте разные подходы и делайте выводы о целесообразности их применения.

4. Общие рекомендации

- Развивайте навыки поиска и критического анализа информации. Пользуйтесь современными информационными ресурсами, анализируйте достоверность и актуальность найденных данных.
- Акцентируйте внимание на интеграции знаний и умений. Старайтесь связывать теоретические знания с практическими задачами, анализируйте, как выбранные методы и технологии влияют на качество и достоверность графического представления информации.
- Соблюдайте академическую честность. Все результаты, представленные в отчетах, должны быть получены самостоятельно, с обязательным указанием источников заимствованных данных и иллюстраций.

6 Фонд оценочных средств для контроля текущей успеваемости и проведения промежуточной аттестации по дисциплине

6.1 Оценочные средства для проведения текущего контроля

6.1.1 семестр 5 | Устный опрос

Описание процедуры.

Опрос может проводиться:
Фронтально — в форме беседы с группой, когда вопросы задаются всей группе, а ответы даются по очереди или по желанию.
Индивидуально — каждый студент отвечает на один или несколько вопросов, давая развернутый, связный ответ, часто с примерами и пояснениями.
Комбинированно — сочетаются оба подхода, а также используются дополнительные методы (например, письменные карточки, рецензирование ответов товарищей)

Критерии оценивания.

полнота и правильность ответа;
понимание и осознанность материала;
логичность и последовательность изложения;
корректность терминологии;
способность отвечать на уточняющие вопросы

6.2 Оценочные средства для проведения промежуточной аттестации

6.2.1 Критерии и средства (методы) оценивания индикаторов достижения компетенции в рамках промежуточной аттестации

Индикатор достижения компетенции	Критерии оценивания	Средства (методы) оценивания промежуточной аттестации
ПКС-4.1	устный опрос	полнота и правильность ответа; понимание и осознанность материала; логичность и последовательность изложения; корректность терминологии; способность отвечать на уточняющие вопросы
ПКС-4.2	устный опрос	полнота и правильность ответа; понимание и осознанность материала; логичность и последовательность изложения; корректность

		терминологии; способность отвечать на уточняющие вопросы

6.2.2 Типовые оценочные средства промежуточной аттестации

6.2.2.1 Семестр 5, Типовые оценочные средства для проведения экзамена по дисциплине

6.2.2.1.1 Описание процедуры

Экзамен сдается в период экзаменационной сессии, предусмотренной учебным планом и календарным учебным графиком.
Студенты допускаются к сдаче экзамена по дисциплине при выполнении всех запланированных форм текущего контроля согласно рабочей программе дисциплины.

6.2.2.1.2 Критерии оценивания

Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Ответ полный, логичный и структурированный, раскрывает все теоретические вопросы билета. Приведены корректные определения, пояснения, примеры и ссылки на нормативные документы (при необходимости). Практическое задание выполнено полностью, расчеты верны, использованы правильные методы и обоснования. Ответ демонстрирует глубокое понимание материала,	Ответ в целом полный, но есть незначительные неточности или упущены отдельные детали. Теоретические вопросы раскрыты, приведены основные определения и примеры. Практическое задание выполнено правильно, но возможны несущественные ошибки или недостаточно подробные пояснения. Понимание материала хорошее, умение применять знания продемонстрирова	Ответ частичный, раскрывает основные положения, но есть существенные пробелы или ошибки в теории. Некоторые определения отсутствуют или даны неверно, примеры не приведены либо не соответствуют вопросу. Практическое задание выполнено частично, есть ошибки в расчетах или не все этапы решения отражены. Понимание материала поверхностное, самостоятельность ограничена.	Ответ не раскрывает основные вопросы билета, содержит грубые ошибки или существенные пробелы. Теоретические положения изложены неверно или отсутствуют. Практическое задание не выполнено либо выполнено неправильно, расчеты отсутствуют или неверны. Материал не усвоен, самостоятельность отсутствует.

самостоятельность мышления и умение применять знания на практике.	но.		

7 Основная учебная литература

1. Аршинский В. Л. Объектно-ориентированное программирование : электронный курс / В. Л. Аршинский, 2019

2. Барков И. А. Объектно-ориентированное программирование : учебник для вузов / И. А. Барков, 2023. - 700.

8 Дополнительная учебная литература и справочная

1. Синтес А. Освой самостоятельно объектно-ориентированное программирование за 21 день : [пер. с англ.] / Антони Синтес, 2002. - 671.

2. Бадд Тимоти. Объектно-ориентированное программирование в действии: Пер. с англ. / Тимоти Бадд, 1997. - 460.

9 Ресурсы сети Интернет

1. http://library.istu.edu/
2. https://e.lanbook.com/

10 Профессиональные базы данных

1. http://new.fips.ru/
2. http://www1.fips.ru/

11 Перечень информационных технологий, лицензионных и свободно распространяемых специализированных программных средств, информационных справочных систем

1. Лицензионное программное обеспечение Системное программное обеспечение

2. Лицензионное программное обеспечение Пакет прикладных офисных программ

3. Лицензионное программное обеспечение Интернет-браузер

12 Материально-техническое обеспечение дисциплины

1. Учебная аудитория для проведения лекционных занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации. Оснащение: комплект учебной мебели, рабочее место преподавателя, доска. Мультимедийное оборудование (в том числе переносное): мультимедийный проектор, экран, акустическая система, компьютер с выходом в интернет.

2. Учебная аудитория для проведения лабораторных/практических (семинарских) занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации. Оснащение: комплект учебной мебели, рабочее место преподавателя, доска. Мультимедийное оборудование (в том числе переносное): мультимедийный проектор, экран, акустическая система, компьютер с выходом в интернет.