МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
(РУТ (МИИТ)

Рабочая программа дисциплины (модуля),
как компонент образовательной программы
базового высшего образования
по направлению подготовки
09.03.01 Информатика и вычислительная техника,
утвержденной первым проректором РУТ (МИИТ)
Тимониным В.С.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Технологии программирования

Направление подготовки: 09.03.01 Информатика и вычислительная техника

Направленность (профиль): Технологии разработки программного обеспечения

Форма обучения: Очная

1. Общие сведения о дисциплине (модуле).

Дисциплина посвящена технологиям разработки программного обеспечения от анализа предметной области и подготовки технического задания до объектно-ориентированного проектирования, реализации, тестирования и оценки качества программного продукта. В содержании рассматриваются жизненный цикл программного обеспечения, требования, UML-моделирование, принципы SOLID, шаблоны проектирования, отладка, верификация, структурное и функциональное тестирование. На лабораторных занятиях обучающиеся последовательно проектируют программную систему выбранной предметной области, создают модели, реализуют объектную структуру, применяют шаблоны проектирования и проверяют качество полученного решения.

Целью освоения дисциплины является формирование способности применять современные технологии программирования для анализа предметной области, проектирования, реализации, тестирования и документирования программных продуктов, пригодных для практического применения.

Для достижения поставленной цели в рамках дисциплины решается комплекс задач, направленных на формирование у обучающихся способности – анализировать предметную область и требования к программному продукту, разрабатывать техническое задание и модели программной системы, применять объектно-ориентированный подход, UML, принципы SOLID и шаблоны проектирования, реализовывать программные компоненты на языке высокого уровня, проводить отладку, тестирование и оценку качества программного решения, готовить техническую документацию по результатам разработки.

2. Планируемые результаты обучения по дисциплине (модулю).

Перечень формируемых результатов освоения образовательной программы (компетенций) в результате обучения по дисциплине (модулю):

ОПК-3 - Способен использовать современные информационные технологии и программно-аппаратные средства, в том числе отечественного производства, при решении задач профессиональной деятельности;

ОПК-4 - Способен решать задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и технологий искусственного интеллекта, а также с учетом основных требований информационной безопасности;

ОПК-6 - Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения.

Обучение по дисциплине (модулю) предполагает, что по его результатам обучающийся будет:

Знать:

- место технологий программирования в программной инженерии и связь программирования с анализом, проектированием, тестированием и сопровождением программного продукта
- этапы развития технологий программирования и причины перехода от неструктурированного к структурному, модульному и объектно-ориентированному подходу
- проблемы разработки сложных информационных систем, включая рост связности, дублирование логики, неявные зависимости и трудность сопровождения
- блочно-иерархический подход к построению программных систем и принципы декомпозиции сложной предметной области
- назначение технического задания, описание предметной области, функциональных требований, ограничений и критериев приемки программного продукта
- основные модели жизненного цикла программного обеспечения и состав работ на этапах анализа, проектирования, реализации, тестирования и сопровождения
- критерии качества программного обеспечения, включая функциональную пригодность, надежность, удобство сопровождения, переносимость и защищенность
- назначение современных языков программирования и сред разработки при решении задач проектирования и реализации программных систем
- основы объектно-ориентированного подхода, включая классы, объекты, инкапсуляцию, наследование, полиморфизм и композицию
- назначение UML как языка моделирования программных систем и роль диаграмм при согласовании структуры и поведения решения
- структурные диаграммы UML, включая диаграммы классов, компонентов, пакетов, объектов и развертывания
- поведенческие диаграммы UML, включая диаграммы вариантов использования, деятельности, состояний, последовательности и коммуникации

- принципы SOLID и их применение для снижения связности, повышения расширяемости и повышения сопровождаемости программного кода

- назначение шаблонов проектирования и различия между порождающими, структурными и поведенческими шаблонами

- назначение шаблонов «Строитель», «Фабричный метод», «Абстрактная фабрика», «Адаптер» и «Репозиторий» в объектно-ориентированном проектировании

- понятие антишаблона и признаки проектных решений, усложняющих сопровождение программного продукта

- виды ошибок программного обеспечения, включая синтаксические, логические, архитектурные, интеграционные и эксплуатационные ошибки

- методы отладки, верификации, структурного и функционального тестирования программных продуктов

- назначение модульных тестов, тестовых данных, граничных случаев и регрессионной проверки при изменении программного кода

- требования к технической документации программного продукта, включая описание требований, моделей, архитектуры, алгоритмов, тестов и ограничений применения

Уметь:

- уметь анализировать предметную область при помощи текстового описания, таблиц сущностей и диаграмм переходов в условиях неполных исходных требований

- уметь разрабатывать техническое задание при помощи ГОСТ 19.201 и шаблона требований в условиях фиксированной предметной области и заданных ограничений

- уметь проектировать структуру программной системы при помощи UML-диаграмм классов, компонентов и пакетов в условиях объектно-ориентированной разработки

- уметь моделировать поведение программной системы при помощи UML-диаграмм вариантов использования, деятельности и последовательности в условиях согласования пользовательских сценариев

- уметь реализовывать объектную модель при помощи Java или Python в условиях разделения ответственности между классами и модулями

- уметь применять принцип единственной ответственности при помощи реорганизации классов в условиях выявленного смешения обязанностей

- уметь применять принцип открытости и закрытости при помощи интерфейсов, абстрактных классов и расширяемых компонентов в условиях добавления новой функциональности без изменения проверенного кода

- уметь применять принципы подстановки, разделения интерфейсов и инверсии зависимостей при помощи объектно-ориентированных средств языка в условиях снижения связности программных компонентов

- уметь выбирать и реализовывать шаблоны «Строитель», «Фабричный метод», «Абстрактная фабрика», «Адаптер» и «Репозиторий» при помощи Java или Python в условиях типовых задач создания, преобразования и хранения объектов

- уметь выявлять антишаблоны проектирования при помощи анализа связности, дублирования и чрезмерной ответственности классов в условиях сопровождения существующего кода

- уметь выполнять отладку программного кода при помощи средств среды разработки и журналирования в условиях воспроизведения логических и интеграционных ошибок

- уметь разрабатывать модульные и функциональные тесты при помощи JUnit или pytest в условиях проверки основных, ошибочных и граничных сценариев

- уметь оценивать качество программного продукта при помощи критериев ISO/IEC 25010 и результатов тестирования в условиях подготовки технического заключения

- уметь готовить техническую документацию при помощи структуры технического задания, UML-моделей, описания архитектуры и протокола испытаний в условиях воспроизводимой передачи результата разработки

Владеть:

- навыком анализа предметной области и формализации требований к программному продукту

- навыком разработки технического задания и комплекта проектных моделей программной системы

- навыком построения UML-диаграмм структуры и поведения программного продукта

- навыком реализации объектно-ориентированной структуры программы с соблюдением принципов SOLID

- навыком применения шаблонов проектирования при создании расширяемых программных компонентов

- навыком отладки, модульного и функционального тестирования программного продукта

- навыком оценки качества программного решения и подготовки технической документации

3. Объем дисциплины (модуля).

3.1. Общая трудоемкость дисциплины (модуля).

Общая трудоемкость дисциплины (модуля) составляет 3 з.е. (108 академических часа(ов).

3.2. Объем дисциплины (модуля) в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

Тип учебных занятий	Количество часов	
	Всего	Семестр №4
Контактная работа при проведении учебных занятий (всего):	64	64
В том числе:		
Занятия лекционного типа	32	32
Занятия семинарского типа	32	32

3.3. Объем дисциплины (модуля) в форме самостоятельной работы обучающихся, а также в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении промежуточной аттестации составляет 44 академических часа (ов).

3.4. При обучении по индивидуальному учебному плану, в том числе при ускоренном обучении, объем дисциплины (модуля) может быть реализован полностью в форме самостоятельной работы обучающихся, а также в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении промежуточной аттестации.

4. Содержание дисциплины (модуля).

4.1. Занятия лекционного типа.

№ п/п	Тематика лекционных занятий / краткое содержание
1	Технологии программирования в программной инженерии Рассматриваемые вопросы: - понятие технологии программирования и ее место в создании программного обеспечения; - связь анализа, проектирования, реализации, тестирования и сопровождения; - требования к программному продукту, пригодному для практического применения.
2	Эволюция подходов к разработке программного обеспечения Рассматриваемые вопросы:

№ п/п	Тематика лекционных занятий / краткое содержание
	- неструктурированное, структурное, модульное и объектно-ориентированное программирование; - причины усложнения программных систем и способы управления сложностью; - роль языков программирования и сред разработки в организации процесса создания программ.
3	Жизненный цикл программного обеспечения Рассматриваемые вопросы: - модели жизненного цикла и их применимость к различным типам задач; - процессы анализа требований, проектирования, реализации, проверки и сопровождения; - результаты этапов жизненного цикла и их связь с технической документацией.
4	Техническое задание и описание предметной области Рассматриваемые вопросы: - назначение технического задания и структура требований к программному продукту; - описание предметной области, действующих лиц, сущностей и ограничений; - критерии приемки и проверяемость требований.
5	Качество программного обеспечения Рассматриваемые вопросы: - характеристики качества программного продукта по ISO/IEC 25010; - функциональная пригодность, надежность, сопровождаемость, переносимость и защищенность; - связь качества с архитектурными решениями, тестированием и документацией.
6	Объектно-ориентированное проектирование программных систем Рассматриваемые вопросы: - классы, объекты, инкапсуляция, наследование, полиморфизм и композиция; - ответственность класса и границы объекта предметной области; - переход от описания предметной области к объектной модели.
7	UML как язык моделирования программных систем Рассматриваемые вопросы: - назначение UML и роль моделей в проектировании программного обеспечения; - диаграммы структуры и диаграммы поведения; - согласование диаграмм с требованиями и программной реализацией.
8	Структурные диаграммы UML Рассматриваемые вопросы: - диаграммы классов, объектов, компонентов и пакетов; - связи, зависимости, обобщение, агрегация и композиция; - правила чтения и проверки структурных моделей.
9	Поведенческие диаграммы UML Рассматриваемые вопросы: - диаграммы вариантов использования, деятельности и состояний; - диаграммы последовательности и коммуникации; - моделирование пользовательских сценариев и взаимодействия объектов.
10	Принципы SOLID в объектно-ориентированном проектировании Рассматриваемые вопросы: - принцип единственной ответственности и принцип открытости и закрытости; - принцип подстановки, разделение интерфейсов и инверсия зависимостей; - признаки нарушения принципов и способы исправления проектных решений.
11	Шаблоны проектирования программного обеспечения Рассматриваемые вопросы: - назначение шаблонов проектирования и условия их применения; - порождающие, структурные и поведенческие шаблоны; - различие между шаблоном проектирования и антишаблоном.
12	Порождающие шаблоны проектирования Рассматриваемые вопросы:

№ п/п	Тематика лекционных занятий / краткое содержание
	- шаблон «Строитель» и пошаговое создание сложного объекта; - шаблон «Фабричный метод» и выбор создаваемого объекта через обобщенный интерфейс; - шаблон «Абстрактная фабрика» и создание согласованных семейств объектов.
13	Структурные шаблоны и организация доступа к данным Рассматриваемые вопросы: - шаблон «Адаптер» и согласование несовместимых интерфейсов; - шаблон «Репозиторий» и отделение хранения данных от прикладной логики; - влияние структурных шаблонов на сопровождаемость программного кода.
14	Ошибки, отладка и верификация программ Рассматриваемые вопросы: - виды ошибок программного обеспечения и причины их возникновения; - методы воспроизведения, локализации и исправления дефектов; - назначение верификации и связь отладки с тестированием.
15	Структурное и функциональное тестирование программных продуктов Рассматриваемые вопросы: - модульное, интеграционное, структурное и функциональное тестирование; - тестовые данные, граничные случаи и проверка ошибочных сценариев; - регрессионная проверка при изменении программного кода.
16	Сопровождение и документирование программного продукта Рассматриваемые вопросы: - техническая документация как средство передачи и сопровождения программного решения; - оценка соответствия реализации требованиям, моделям и результатам тестирования; - подготовка заключения о качестве и ограничениях применения программного продукта.

4.2. Занятия семинарского типа.

Лабораторные работы

№ п/п	Наименование лабораторных работ / краткое содержание
1	Анализ предметной области программного продукта Студент выбирает предметную область и описывает ее основные сущности, действующих лиц, действия и ограничения. На основе описания формируется перечень функций будущего программного решения. Результат фиксируется в виде исходного описания предметной области и набора проверяемых требований.
2	Разработка технического задания Студент оформляет назначение программного продукта, состав функций, входные и выходные данные, ограничения и критерии приемки. Требования приводятся к проверяемой форме. Подготовленный документ используется как основа дальнейшего проектирования.
3	Моделирование пользовательских сценариев средствами UML Студент строит диаграмму вариантов использования для выбранной предметной области. Для основных сценариев уточняются участники, цели, предусловия и ожидаемые результаты. Модель проверяется на полноту относительно требований.
4	Моделирование процессов предметной области средствами UML Студент строит диаграмму деятельности для одного основного процесса. В модели фиксируются ветвления, повторения, условия переходов и конечные состояния. Диаграмма сопоставляется с ранее заданными требованиями.
5	Проектирование структуры классов средствами UML Студент строит диаграмму классов с атрибутами, операциями и связями между сущностями. Для

№ п/п	Наименование лабораторных работ / краткое содержание
	связей указываются зависимости, обобщение, агрегация или композиция. Модель проверяется на отсутствие избыточных и не связанных классов.
6	Проектирование компонентов программной системы Студент строит диаграмму компонентов или пакетов для разбиения решения на логические части. Для каждого компонента определяется ответственность и связи с другими компонентами. Полученная структура используется при создании исходного кода.
7	Реализация объектной модели программного продукта Студент создает проект в среде разработки и реализует основные классы выбранной предметной области. Для классов задаются поля, методы, конструкторы и базовые проверки допустимости данных. Реализация сопоставляется с диаграммой классов.
8	Применение принципа единственной ответственности Студент анализирует реализованные классы и выявляет смешение обязанностей. Избыточные обязанности выносятся в отдельные классы или службы. После изменения выполняется повторная проверка связности объектной структуры.
9	Применение принципов открытости и подстановки Студент выделяет расширяемые точки программы через интерфейсы или абстрактные классы. Добавляется новый вариант поведения без изменения проверенной части кода. Проверяется корректность подстановки производных объектов в существующие сценарии.
10	Применение принципов разделения интерфейсов и инверсии зависимостей Студент разделяет крупные интерфейсы на специализированные контракты. Зависимости классов направляются на абстракции, а не на конкретные реализации. Полученная структура проверяется на снижение связности компонентов.
11	Реализация порождающих шаблонов проектирования Студент применяет шаблон «Строитель» для создания сложного объекта и шаблон «Фабричный метод» или «Абстрактная фабрика» для создания семейства объектов. В коде выделяется общий интерфейс создаваемых объектов. Результат проверяется на возможность расширения набора создаваемых объектов.
12	Реализация структурных шаблонов проектирования Студент применяет шаблон «Адаптер» для согласования интерфейса существующего компонента с требованиями программы. Затем выделяется слой «Репозиторий» для операций хранения и поиска объектов. Прикладная логика отделяется от способа хранения данных.
13	Выявление и исправление антишаблонов Студент анализирует программный код на дублирование, чрезмерную ответственность классов и неявные зависимости. Выявленные недостатки устраняются путем перераспределения обязанностей и уточнения связей. После исправления обновляются соответствующие проектные модели.
14	Отладка программного продукта Студент воспроизводит логические и интеграционные ошибки в реализованной программе. Для поиска причин используются точки останова, просмотр значений переменных и журналирование. Исправления фиксируются с указанием причины дефекта и способа проверки.
15	Структурное и функциональное тестирование программного продукта Студент разрабатывает набор модульных и функциональных тестов для основных сценариев. В тесты включаются нормальные, ошибочные и граничные данные. Результаты выполнения тестов сопоставляются с требованиями технического задания.
16	Оценка качества и подготовка технической документации Студент оценивает программное решение по критериям качества, требованиям, моделям и результатам тестирования. В документацию включаются описание предметной области, техническое задание, UML-модели, сведения о реализации и протокол испытаний. Итоговый комплект проверяется на полноту и согласованность.

4.3. Самостоятельная работа обучающихся.

№ п/п	Вид самостоятельной работы
1	Изучение рекомендованной литературы.
2	Подготовка к лабораторным работам.
3	Выполнение курсовой работы.
4	Подготовка к промежуточной аттестации.
5	Подготовка к текущему контролю.

4.4. Примерный перечень тем курсовых работ

Проектирование приложения с предметной областью «Блог».

Проектирование приложения с предметной областью «Университет».

Проектирование приложения с предметной областью «Автодилер».

Проектирование приложения с предметной областью «Интернет-магазин».

Проектирование приложения с предметной областью «Каршеринг».

Проектирование приложения с предметной областью «Система совместной подготовки документов».

Проектирование приложения с предметной областью «Система тестирования».

Проектирование приложения с предметной областью «Система обмена сообщениями».

Проектирование приложения с предметной областью «Научно-исследовательский институт».

Проектирование приложения с предметной областью «Магазин и склад».

Проектирование приложения с предметной областью «Железная дорога».

Проектирование приложения с предметной областью «Аэропорт».

Проектирование приложения с предметной областью «Банк».

Проектирование приложения с предметной областью «Библиотека».

Проектирование приложения с предметной областью «Авиабилеты».

5. Перечень изданий, которые рекомендуется использовать при освоении дисциплины (модуля).

№ п/п	Библиографическое описание	Место доступа

1	Зубкова, Т. М. Технология разработки программного обеспечения : учебное пособие / Т. М. Зубкова. — Санкт-Петербург : Лань, 2019. — 324 с. — ISBN 978-5-8114-3842-6. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/122176 (дата обращения: 19.06.2026)
2	Буч, Р. Г. Язык UML. Руководство пользователя : учебное пособие / Р. Г. Буч, И. Якобсон ; перевод с английского Н. Мухина. — 3-е изд., эл. — Москва : ДМК Пресс, 2022. — 495 с. — ISBN 978-5-89818-247-2. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/519390 (дата обращения: 19.06.2026)
3	Кручинин, В. В. Технологии программирования : учебное пособие / В. В. Кручинин. — Москва : ТУСУР, 2013. — 271 с. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/110371 (дата обращения: 19.06.2026)

6. Перечень современных профессиональных баз данных и информационных справочных систем, которые могут использоваться при освоении дисциплины (модуля).

ЭБС Лань – https://e.lanbook.com/.

Образовательная платформа Юрайт – https://urait.ru/.

Единый реестр российских программ для ЭВМ и баз данных – https://reestr.digital.gov.ru/reestr/.

Профессиональные стандарты и квалификации, справочная информация КонсультантПлюс – https://www.consultant.ru/document/cons_doc_LAW_157436/.

ГОСТ Р ИСО/МЭК 12207-2010, сведения о процессах жизненного цикла программных средств – https://docs.cntd.ru/document/1200080952.

ГОСТ 19.201-78, требования к содержанию технического задания – https://docs.cntd.ru/document/1200007652.

Документация Java SE – https://docs.oracle.com/en/java/javase/21/docs/api/index.html.

Документация Python – https://docs.python.org/3/.

Документация JUnit – https://docs.junit.org/current/user-guide/.

Документация pytest – https://docs.pytest.org/.

Документация Git – https://git-scm.com/doc.

Документация PlantUML – https://plantuml.com/ru/.

7. Перечень лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства, необходимого для освоения дисциплины (модуля).

Операционные системы – Astra Linux, ALT Linux, РЕД ОС, Debian GNU/Linux.

Среды разработки – IntelliJ IDEA Community Edition, Eclipse IDE, Visual Studio Code.

Языки и сборка – Java, Python, Maven или Gradle.

Моделирование и документация – PlantUML, diagrams.net.

Проверка и сопровождение – JUnit, pytest, Git.

8. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю).

Учебные аудитории для проведения учебных занятий, оснащенные компьютерной техникой и наборами демонстрационного оборудования.

Для лабораторных занятий – наличие персональных компьютеров вычислительного класса.

9. Форма промежуточной аттестации:

Курсовая работа в 4 семестре.

Экзамен в 4 семестре.

10. Оценочные материалы.

Оценочные материалы, применяемые при проведении промежуточной аттестации, разрабатываются в соответствии с локальным нормативным актом РУТ (МИИТ).

Авторы:

ассистент кафедры «Цифровые технологии управления транспортными процессами» Д.А. Заманов

Согласовано:

Заведующий кафедрой ЦТУТП В.Е. Нутович

Председатель учебно-методической комиссии Н.А. Андриянова