МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Благовещенский государственный педагогический университет»

ОСНОВНАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА
Рабочая программа дисциплины

УТВЕРЖДАЮ
декан факультета физико-математического образования и технологии
ФГБОУ ВО БГПУ
Н.В. Слесаренко
«03» сентября 2024 г.

Рабочая программа дисциплины
СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Направление подготовки

02.03.03 – МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Профиль

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Уровень высшего образования
БАКАЛАВРИАТ

Принята
на заседании кафедры информатики
и методики преподавания информатики
(протокол № 8 от «25» мая 2024 г.)

Благовещенск 2024

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА 3
2 УЧЕБНО-ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ 5
3 СОДЕРЖАНИЕ ТЕМ (РАЗДЕЛОВ) 5
4 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ (УКАЗАНИЯ) ДЛЯ СТУДЕНТОВ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ 6
5 ПРАКТИКУМ ПО ДИСЦИПЛИНЕ 8
6 ДИДАКТИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ КОНТРОЛЯ (САМОКОНТРОЛЯ) УСВОЕННОГО МАТЕРИАЛА 9
7 ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ 19
В ПРОЦЕССЕ ОБУЧЕНИЯ 19
8 ОСОБЕННОСТИ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ ИНВАЛИДАМИ И ЛИЦАМИ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ 19
9 СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ 20
10 МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ БАЗА 20
11 ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ 22

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1 Цель дисциплины: формирование у студентов представления о современных подходах и методах программирования. Обеспечения базы теоретической и практической подготовки в области проектирования, разработки, тестирования и развертывания современных программных продуктов. Формирование практических навыков применения систем контроля версий программного обеспечения.

1.2 Место дисциплины в структуре ООП: Дисциплина «Современные технологии программирования» относится к дисциплинам обязательной части блока Б1 (Б1.О.32).

Для освоения дисциплины «Современные технологии программирования» используются знания, умения и виды деятельности, формируемые в процессе изучения дисциплин «Программирование», «Технология программирования Java», «Технология разработки программного обеспечения». Дисциплина «Современные технологии программирования» в профессиональной подготовке выпускника является одной и основных дисциплин, изучение которой позволит студентам выработать современный подход к качеству и содержанию компьютерных программ.

Дисциплина «Современные технологии программирования» обеспечивает раскрытие общего круга вопросов разработки программного обеспечения, организации процесса разработки, контроля версий и развертывания. В ходе изучения дисциплины разбираются основные системы сборки, тестирования и отладки, контейнеризации и развертывания.

1.3 Дисциплина направлена на формирование следующих компетенций: ПК-4, ПК-5, ПК-7.

- **ПК-4.** Способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях, **индикаторами** достижения которой является:

- ПК-4.1 – **знает** современные технологии проектирования и производства программного продукта;
- ПК-4.2 – **умеет** использовать подобные технологии при создании программных продуктов;
- ПК-4.3 – **имеет** практический опыт применения подобных технологий.

- **ПК-5.** Способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов, **индикаторами** достижения которой является:

- ПК-5.1 **– знает** современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов, их сопровождения и администрирования;
- ПК-5.2 **– умеет** использовать подобные инструментальные средства в практической деятельности;
- ПК-5.3 **– имеет практический опыт** применения подобных инструментальных средств.

- **ПК-7.** Способен использовать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений, **индикаторами** достижения которой является:

- ПК-7.1 **– знает** основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы,

способы и средства разработки программ в рамках этих направлений;

- ПК-7.2 – **умеет** программировать в рамках этих направлений;
- ПК-7.3 – **имеет практический опыт** разработки программ в рамках этих направлений.

1.4 Перечень планируемых результатов обучения. В результате изучения дисциплины студент должен

знать:

- ключевые парадигмы современного программирования;
- необходимые инструментальные средства для разработки современных программных приложений;
- инструменты и средства тестирования и отладки;
- необходимые инструментальные средства для контейнеризации и оркестровки контейнеризированных приложений

уметь:

- решать базовые задачи автоматизации;
- уметь объяснить способы и методы решения;
- самостоятельно изучать на существующих примерах программного кода способы разработки программного обеспечения;
- покрывать программный код тестами;
- развертывать приложения;

владеть:

- базовыми алгоритмами и техниками решения учебных задач программирования современными языковыми и технологическими средствами;
- навыками работы с современными системами контроля версий.

1.5 Общая трудоемкость дисциплины «Современные технологии программирования» составляет 4 зачетные единицы (далее – ЗЕ) (144 часа).

Программа предусматривает изучение материала на лекциях и лабораторных работах. Предусмотрена самостоятельная работа студентов по темам и разделам. Проверка знаний осуществляется фронтально, индивидуально.

1.6 Объем дисциплины и виды учебной деятельности

Объем дисциплины и виды учебной деятельности (очная форма обучения)

Вид учебной работы	Всего часов	Семестр 6
Общая трудоемкость	144	144
Аудиторные занятия	60	60
Лекции	24	24
Лабораторные работы	36	36
Самостоятельная работа, написание курсовой работы	48	48
Вид контроля		защита курсовой работы
Вид итогового контроля	36	экзамен

2 УЧЕБНО-ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ

2.1 Очная форма обучения

Учебно-тематический план

№	Наименование тем (разделов)	Всего часов	Аудиторные занятия		Самостоятельная работа, написание курсовой работы
			Лекции	Лабораторные работы	
1.	Тема 1. Системы контроля версий.	24	6	8	10
2.	Тема 2. Системы автоматической сборки.	20	4	6	10
3.	Тема 3. Написание тестов.	22	6	8	8
4.	Тема 4. Контейнеризация приложений.	22	4	8	12
5.	Тема 5. Оркестровка контейнеризированных приложений.	20	4	6	8
Экзамен		36			
Защита курсовой работы					
ИТОГО		**144**	**24**	**36**	**48**

Интерактивное обучение по дисциплине

№	Наименование тем (разделов)	Вид занятия	Форма интерактивного занятия	Кол-во часов
1.	Системы контроля версий.	ЛБ	Работа в малых группах	8
2.	Контейнеризация приложений.	ЛБ	Выполнение проекта.	8
ИТОГО				**16**

3 СОДЕРЖАНИЕ ТЕМ (РАЗДЕЛОВ)

Тема 1. Системы контроля версий.

Определение системы контроля версий. Типы систем контроля версий. Хранение истории изменений. Ветвление. Практические аспекты использования. Ежедневный цикл работы. Слияние версий. Конфликты и их разрешения. Блокировки. Распределенные системы управления версиями. Сравнение систем управления версиями. Конфигурационное управление.

Тема 2. Системы автоматической сборки.

Компиляция и сборка проекта. Модель описания проекта. Соглашения по конфигурации. Жизненный цикл. Зависимости. Сравнение систем автоматической сборки. Зонтичные проекты. Монорепозитории. Мультипроектная сборка. Взаимодействие со средами разработки. Кросс-компиляция.

Тема 3. Написание тестов.

Модульное тестирование. Документирование кода. Интеграционное тестирование. Приложения модульного тестирования. Разработка через тестирование. Цикл разработки через тестирование. Fake-, mock-объекты. Рефакторинг. Покрытие кода.

Тема 4. Контейнеризация приложений.

Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Сравнительный анализ существующих решений. Программное обеспечение для эмуляции и виртуализации. Chroot. FreeBSD Jail. Docker.

Тема 5. Оркестровка контейнеризированных приложений.

Концепции. Архитектура и компоненты. Подсистема управления. Компоненты узлов. Масштабируемость. Кластеризация контейнерных приложений. Разработка и развёртывание. Распространение и конкуренция.

4 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ (УКАЗАНИЯ) ДЛЯ СТУДЕНТОВ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ

4.1 Общие методические рекомендации

В ходе изучения дисциплины достигается освоение студентами теории и практики разработки современных программных продуктов, формируется представление и навык реализации на современном языке программирования наиболее актуальных алгоритмов, навыки разработки приложений в современных интегрированных средах разработки.

Представленные материалы призваны организовать процесс изучения дисциплины «Современные технологии программирования».

Список литературы позволяет использовать материалы как для подготовки к лабораторным работам, так и для организации самостоятельной подготовки, а также расширения представлений о направлениях исследований, так или иначе связанных с дисциплиной.

4.2 Методические рекомендации по подготовке к лекциям

Приступая к изучению дисциплины «Современные технологии программирования» студент должен иметь представление о современных подходах к разработке программного обеспечения, о практическом применении получаемых в рамках курса знаний.

Самостоятельная подготовка к лекциям происходит до посещения занятий путем чтения рекомендованной литературы и выполнения задач, полученных на практических занятиях. Выполнение многих задач требует больше времени, чем отведено для работы в аудиториях университета.

Посещение лекции и активное участие в интерактивных формах обучения является еще одной формой самостоятельной работы студента. Конспектирование ключевых мыслей и программного кода не является обязательным компонентом такой работы, но рекомендуется, так как повышает эффективность выполнения заданий в ходе практических занятий.

Важной частью самостоятельной работы является периодическое повторение пройденного материала, что способствует более глубокому усвоению знаний и упрощает продвижение по пути освоения последующих тем.

4.3 Методические рекомендации по подготовке к лабораторным занятиям

Целью лабораторных занятий является закрепление теоретического материала лекций и выработка умения использования информационных и других ресурсов, предоставляемых университетом.

Подготовка к лабораторным работам предполагает изучение теоретического материала по указанной теме, с использованием конспектов лекций и дополнительной литературы. При необходимости можно обращаться за консультацией к преподавателю.

В процессе подготовки к занятиям рекомендуется взаимное обсуждение материала, во время которого закрепляются знания, а также приобретается практика в изложении и разъяснении полученных знаний, развивается речь.

В случае появления каких-либо вопросов следует обращаться к преподавателю в часы его консультаций.

Для проведения практических занятий используются компьютеры, оснащенные ОС Windows 7 и выше, ОС Linux, Java Development Kit, NetBeans, Git, Maven, Gradle, Docker система электронного образования университета. Возможно использование проектора или интерактивной доски.

4.4 Методические рекомендации к самостоятельной работе студентов

Самостоятельная работа студентов при изучении дисциплины «Современные технологии программирования» организуется с целью формирования профессиональных компетенций, понимаемых как способность применять знания, умения и личностные качества для успешной деятельности в определенной области, в том числе:

– формирования умений по поиску и использованию различных источников информации;

– качественного освоения и систематизации полученных теоретических знаний, их углубления и расширения по применению на уровне межпредметных связей;

– формирования умения применять полученные знания на практике;

– развития познавательных способностей студентов, формирования самостоятельности мышления;

– развития активности, творческой инициативы, самостоятельности, ответственности и организованности;

– формирования способностей к саморазвитию;

В ходе изучения дисциплины «Технология программирования Java» предлагается выполнить различные виды самостоятельной работы:

– выполнение индивидуальных заданий лабораторных работ;

– подготовка к аудиторным занятиям;

– изучение отдельных тем (вопросов) дисциплины в соответствии с учебно-тематическим планом, составление конспектов;

– подготовка ко всем видам контрольных испытаний.

В методических указаниях излагается порядок выполнения лабораторных работ. При выполнении работ используются Java Development Kit, NetBeans, Git, Maven, Gradle, Docker.

К зачету по лабораторной работе предъявляется проекты, размещенные в открытом репозитории систем контроля версий.

4.5 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине:

1. Оценочные средства.
2. Задания.
3. Список тем для собеседования на экзамене.
4. Список литературы и информационных ресурсов.

4.6 Рекомендации по написанию курсовой работы

Курсовая работа – одна из обязательных форм учебно-исследовательской работы студента, выполняемая в пределах часов, отводимых на самостоятельное изучение дисциплины в соответствии с ФГОС ВО. Курсовая работа выполняется в соответствии с Положением о курсовой работе (проекте) в ФГБОУ ВО «БГПУ».

Учебно-методическое обеспечение самостоятельной работы студентов по дисциплине

Очная форма обучения

№	Наименование раздела (темы)	Формы/виды самостоятельной работы	Количество часов, в соответствии с учебно-тематическим планом
1.	Тема 1. Системы контроля версий.	Проработка теоретического материала по конспектам лекций. Сбор материала для выполнения курсовой работы. Подготовка текста курсовой работы. Чтение рекомендованной литературы.	10
2.	Тема 2. Системы автоматической сборки.	Проработка теоретического материала по конспектам лекций. Сбор материала для выполнения курсовой работы. Подготовка текста курсовой работы.	10
3.	Тема 3. Написание тестов.	Проработка теоретического материала по конспектам лекций. Сбор материала для выполнения курсовой работы. Подготовка текста курсовой работы.	8
4.	Тема 4. Контейнеризация приложений.	Проработка теоретического материала по конспектам лекций. Сбор материала для выполнения курсовой работы. Подготовка текста курсовой работы.	12
5.	Тема 5. Оркестровка контейнеризированных приложений.	Проработка теоретического материала по конспектам лекций. Сбор материала для выполнения курсовой работы. Подготовка текста курсовой работы.	8
	ИТОГО		**48**

5 ПРАКТИКУМ ПО ДИСЦИПЛИНЕ

Системы контроля версий.
Лабораторная работа №1 «Управлением репозиторием git»: 2 часа
Лабораторная работа №2 «Управление ветками проекта»: 4 часа
Лабораторная работа №3 «Pull request и рецензирование кода»: 2 часа

Системы автоматической сборки.
Лабораторная работа №4 «Работа с Maven и Gradle»: 2 часа
Лабораторная работа №5 «Кросс-компиляция cmake»: 4 часа

Написание тестов.
Лабораторная работа №6 «Модульное тестирование»: 4 часа
Лабораторная работа №7 «Интеграционное тестирование»: 4 часа

Контейнеризация приложений.
Лабораторная работа №8 «Контейнеризация Docker»: 4 часа
Лабораторная работа №9 «Управление контейнерами docker-compose»: 4 часа

Оркестровка контейнеризированных приложений.
Лабораторная работа №1 «Основы Kubernetes»: 2 часа
Лабораторная работа №1 «Кластеризация»: 4 часа

6 ДИДАКТИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ КОНТРОЛЯ (САМОКОНТРОЛЯ) УСВОЕННОГО МАТЕРИАЛА

6.1 Оценочные средства, показатели и критерии оценивания компетенций

Индекс компетенции	Оценочное средство	Показатели оценивания	Критерии оценивания сформированности компетенций
ПК-4, ПК-5, ПК-7	Проблемная задача	Низкий – до 60 баллов (неудовлетворительно)	Работа студента не засчитывается если: 1. студент обнаруживает неумение выполнять решения большей части задания, допускает грубые ошибки в решении задач, беспорядочно и неуверенно излагает материал.
		Пороговый – 61-75 баллов (удовлетворительно)	Студент обнаруживает знание формул и понимание основных методов решения задач, но: 1. излагает решения неполно и допускает неточности в вычислениях; не умеет рационально решать задачи.
		Базовый – 76-84 баллов (хорошо)	Студент выполняет работу полностью, обнаруживает понимание материала, но: 1. допускает некоторые вычислительные ошибки; 2. небрежно оформляет решения; демонстрирует решения задач только в рамках алгоритмов, изученных на занятиях.
		Высокий – 85-100 баллов (отлично)	Студент получает высокий балл, если: 1. выполняет задание в полном объеме;

			2. обнаруживает понимание материала; 3. использует рациональные способы решения задач; демонстрирует умение пользоваться дополнительными источниками знаний.
ПК-4, ПК-5, ПК-7	Лабораторная работа	Низкий (неудовлетворительно)	Лабораторная работа студенту не засчитывается если студент: 1. допустил число ошибок и недочетов превосходящее норму, при которой пересекается пороговый показатель; 2. или если правильно выполнил менее половины работы.
		Пороговый (удовлетворительно)	Если студент правильно выполнил не менее половины работы или допустил: 1. не более двух грубых ошибок; 2. или не более одной грубой и одной негрубой ошибки и одного недочета; 3. или не более двух-трех негрубых ошибок; 4. или одной негрубой ошибки и трех недочетов; 5. или при отсутствии ошибок, но при наличии четырех-пяти недочетов.
		Базовый (хорошо)	Если студент выполнил работу полностью, но допустил в ней: 1. не более одной негрубой ошибки и одного недочета; 2. или не более двух недочетов.
		Высокий (отлично)	Если студент: 1. выполнил работу без ошибок и недочетов; 2. допустил не более одного недочета.
ПК-2	Курсовая работа	Низкий (неудовлетворительно)	• Не выполнены требования к оформлению работ, согласно нормоконтролю. • Не раскрыто основное содержание учебного материала. • Курсовая работа не допущена научным руководителем к защите. • Не сформированы компетенции, умения и навыки.
		Пороговый (удовлетворительно)	• Выполнены требования к оформлению работ, согласно нормоконтролю. • Не полно раскрыто содержание материала работы, но точно используется терминология; нарушена определенная логическая последовательность.

			• В работе представлена практическая часть, выполненная самостоятельно; не показано умение иллюстрировать теоретические положения конкретными примерами и их применение в новой ситуации. • При защите работы продемонстрирована сформированность компетенций, умений и навыков, допущены недочёты при освещении основного содержания курсовой работы, получены ответы не на все вопросы комиссии.
		Базовый (хорошо)	• Выполнены все требования к оформлению работ, согласно нормоконтролю. • Полно раскрыто содержание материала; материал изложен грамотно, в определенной логической последовательности; точно используется терминология. • В работе представлена практическая часть, выполненная самостоятельно; показано умение иллюстрировать теоретические положения конкретными примерами. • При защите работы продемонстрирована сформированность компетенций, умений и навыков, допущены один – два недочёта при освещении основного содержания курсовой работы, получены ответы не на все вопросы комиссии.
		Высокий (отлично)	• Выполнены все требования к оформлению работ, согласно нормоконтролю. • Полно раскрыто содержание материала; материал изложен грамотно, в определенной логической последовательности; точно используется терминология. • В работе представлена практическая часть, выполненная самостоятельно; показано умение иллюстрировать теоретические положения конкретными примерами и их применение в новой ситуации. • При защите работы продемонстрирована сформированность и устойчивость компетенций, умений и навыков; получены полные ответы на вопросы комиссии.

6.2 Промежуточная аттестация студентов по дисциплине

Промежуточная аттестация является проверкой всех знаний, навыков и умений студентов, приобретённых в процессе изучения дисциплины. Формой промежуточной аттестации по дисциплине является экзамен.

Для оценивания результатов освоения дисциплины применяется следующие критерии оценивания.

Критерии оценивания устного ответа на экзамене

Оценка 5 (отлично) ставится, если:

- полно раскрыто содержание вопросов в объеме программы и рекомендованной литературы;
- четко и правильно даны определения и раскрыто содержание концептуальных понятий, закономерностей, корректно использованы научные термины;
- для доказательства использованы различные теоретические знания, выводы из наблюдений и опытов;
- ответ самостоятельный, исчерпывающий, без наводящих дополнительных вопросов, с опорой на знания, приобретенные в процессе специализации по выбранному направлению информатики.

Оценка 4 (хорошо) ставится, если:

- раскрыто основное содержание вопросов;
- в основном правильно даны определения понятий и использованы научные термины;
- ответ самостоятельный;
- определения понятий неполные, допущены нарушения последовательности изложения, небольшие неточности при использовании научных терминов или в выводах и обобщениях, исправляемые по дополнительным вопросам экзаменаторов.

Оценка 3 (удовлетворительно) ставится, если:

- усвоено основное содержание учебного материала, но изложено фрагментарно, не всегда последовательно;
- определение понятий недостаточно четкое;
- не использованы в качестве доказательства выводы из наблюдений и опытов или допущены ошибки при их изложении;
- допущены ошибки и неточности в использовании научной терминологии, определении понятий.

Оценка 2 (неудовлетворительно) ставится, если:

- ответ неправильный, не раскрыто основное содержание программного материала;
- не даны ответы на вспомогательные вопросы экзаменаторов;
- допущены грубые ошибки в определении понятий, при использовании терминологии.

6.2 Промежуточная аттестация студентов по дисциплине

Промежуточная аттестация является проверкой всех знаний, навыков и умений студентов, приобретённых в процессе изучения дисциплины. Формой промежуточной аттестации по дисциплине является экзамен, защита курсовой работы.

Для оценивания результатов освоения дисциплины применяется следующие критерии оценивания.

Критерии оценивания курсовой работы

Оценка 5 (отлично) ставится, если:

Во введении приводится обоснование выбора конкретной темы, полностью раскрыта актуальность её в научной отрасли, чётко определены грамотно поставлены задачи и цель курсовой работы. Основная часть работы демонстрирует большое количество прочитанных автором работ. Критически прочитаны источники: вся необходимая информация проанализирована, вычленена, логически структурирована. Присутствуют выводы и грамотные обобщения. В заключении сделаны логичные выводы, а собственное отношение выражено чётко. Автор курсовой работы грамотно демонстрирует осознание возможности применения исследуемых теорий, методов на практике. Приложение содержит цитаты и таблицы, иллюстрации и диаграммы: все необходимые материалы.

Курсовая работа написана в стиле академического письма (использован науный стиль изложения материала). Автор адекватно применял терминологию, правильно оформил ссылки. Оформление работы соответствует требованиям нормоконтроля, библиография, приложения оформлены на отличном уровне. Объём работы заключается в пределах от 20 до 30 страниц.

Оценка 4 (хорошо) ставится, если:

Во введении содержит некоторую нечёткость формулировок. В основной её части не всегда проводится критический анализ, отсутствует авторское отношение к изученному материалу. В заключении неадекватно использована терминология, наблюдаются незначительные ошибки в стиле, многие цитаты грамотно оформлены. Допущены незначительные неточности в оформлении библиографии, приложений.

Оценка 3 (удовлетворительно) ставится, если:

Во введении содержит лишь попытку обоснования выбора темы и актуальности, отсутствуют чёткие формулировки. Расплывчато определены задачи и цели. Основное содержание – пересказ чужих идей, нарушена логика изложения, автор попытался сформулировать выводы. В заключении автор попытался сделать обобщения, собственного отношения к работе практически не проявил. В приложении допущено несколько грубых ошибок. Не выдержан стиль требуемого академического письма по проекту в целом, часто неверно употребляются научные термины, ссылки оформлены неграмотно, наблюдается плагиат.

Оценка 2 (неудовлетворительно) ставится, если:

Во введении не содержит обоснования темы, нет актуализации темы. Не обозначены и цели, задачи проекта. Внутренняя логика всего изложения проекта слабая. Нет критического осмысления прочитанного, как и собственного мнения. Нет обобщений, выводов. В заключении не приведены грамотные выводы. Приложения либо вовсе нет, либо оно недостаточно. В работе наблюдается отсутствие ссылок, плагиат, не выдержан стиль, неверное использование терминологии. По оформлению наблюдается ряд недочётов: не соблюдены основные требования нормоконтроля, а библиография с приложениями содержат много ошибок.

6.3 Типовые контрольные задания или иные материалы, необходимые для оценки результатов освоения дисциплины

Примеры проблемных задач

Задача 1

Для представленных проектов проведите оценку оптимального количества разработчиков. Каким образом можно распределить распределение между ними задач.

1. Игра пятнашки.
2. Игра морской бой с возможностью игры по сети.
3. Desktop приложение учета контингента сотрудников для отдела кадров.
4. Web приложение учета контингента сотрудников для отдела кадров.
5. Web приложение файлового хранилища.
6. Система управления отопления и вентиляции овощехранилища.

Как поменяются роли разработчиков если из проекта убрать одного человека?
Как поменяются роли если из проекта убрать половину участников?

Задача 2

Для представленных задач приведите порядок и название коммитов:

1. Реализация сортировки пузырьком.
2. Подключение и получение информации из базы данных.
3. Реализация отображения списка сотрудников в desktop приложении.
4. Разработка простейшей сетевой программы обмена текстовыми сообщениями.

Примеры лабораторных работ

Лабораторная работа №1 «Управлением репозиторием git»

Цель работы: изучить основы работы с системой контроля версий git.

Теоретические сведения:

Система управления версиями (Version Control System, VCS) – программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Виды VCS:

- локальные системы контроля версий;
- централизованные системы контроля версий;
- децентрализованные системы контроля версий.

Локальные системы контроля версий. Многие люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию (возможно даже, директорию с отметкой по времени, если они достаточно сообразительны). Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели. Для того, чтобы решить эту проблему, программисты давным-давно разработали локальные СКВ с простой базой данных, которая хранит записи о всех изменениях в файлах, осуществляя тем самым контроль ревизий.

Централизованные системы контроля версий. Следующая серьёзная проблема, с которой сталкиваются люди, – это необходимость взаимодействовать с другими разработчиками. Для того, чтобы разобраться с ней, были разработаны централизованные системы контроля версий (ЦСКВ). Такие системы, как: CVS, Subversion и Perforce, имеют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. Применение ЦСКВ являлось стандартом на протяжении многих лет.

Децентрализованные системы контроля версий (ДСКВ). В ДСКВ (таких как Git, Mercurial, Bazaar или Darcs), клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени): они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.

Более того, многие ДСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно, в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах.

Git (произносится «гит») – распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

Система спроектирована как набор программ, специально разработанных с учётом их использования в сценариях. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей). Ряд сервисов предоставляют хостинг для git-репозиториев, среди наиболее известных – GitHub, SourceForge, Gitea, Bitbucket, GitLab.

Задание.

1. Создайте в IDE проект, который выводит в консоль «Hello World!».
2. Создать репозиторий на GitHub. (можно использовать GitLab или Bitbucket).
3. Синхронизируйте локальный репозиторий, с созданным на GitHub.
4. Создать файл readme.md, в котором будет содержаться описание разрабатываемого проекта. Затем закоммитить изменения в readme файле в git репозиторий
5. Отправить локальные изменения на внешний репозиторий.
6. Предоставить доступ к удаленному репозиторию ещё одному пользователю.
7. Удалите локальные файлы проекта и склонируйте удаленный репозиторий. Запустите проект.

Примечание:
Туториал по использованию git смотреть на сайте https://githowto.com/ru.

Контрольные вопросы:

1. Для чего нужны системы контроля версий.
2. Для чего нужны commit, push, pull.
3. Для чего нужны ветки (branch).
4. Для чего нужен pull request.

Лабораторная работа №6 «Модульное тестирование»: 4 часа

Цель работы: изучить основы модульного тестирования Java приложений.

Теоретические сведения:

Модульное тестирование, или юнит-тестирование (англ. unit testing) – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок. В данной работе мы будем использовать юнит-тесты для проверки функциональных требований программы.

Для использования юнит-тестов будем использовать JUnit. JUnit – библиотека для модульного тестирования программного обеспечения на языке Java. Пример юнит-теста, проверяющего равенство 2+2=4, приведен на рисунке 6.1.

Для модульного тестирования необходимо использовать драйверы и заглушки. Unit (Элемент) – наименьший компонент, который можно скомпилировать. Драйверы – модули тестов, которые запускают тестируемый элемент. Заглушки – заменяют недостающие компоненты, которые вызываются элементом и выполняют следующие действия:

- возвращаются к элементу, не выполняя никаких других действий;
- отображают трассировочное сообщение и иногда предлагают тестеру продолжить тестирование;
- возвращают постоянное значение или предлагают тестеру самому ввести возвращаемое значение;
- осуществляют упрощенную реализацию недостающей компоненты;
- имитируют исключительные или аварийные условия.

Предварительная подготовка к работе. Так как алгоритм реализован на языке Java, то юнит-тесты следует писать с использованием библиотеки JUnit (). Допускается использовать любую удобную среду разработки (IDEA / NetBeans / др.).

```
1. import org.junit.Test;
2. import junit.framework.Assert;
3.
4. public class MathTest {
5.
6.     @Test
7.     public void testEquals() {
8.         Assert.assertEquals(4, 2 + 2);
9.         Assert.assertTrue(4 == 2 + 2);
10.    }
11.
12.    @Test
13.    public void testNotEquals() {
14.        Assert.assertFalse(5 == 2 + 2);
15.    }
16. }
```

Рисунок 6.1 – Листинг кода примера модульного теста

Для упрощения задачи, предлагается выполнить интеграционное тестирование по восходящему подходу. При использовании этого метода подразумевается, что сначала тестируются все программные модули, входящие в состав системы, и только затем они объединяются для интеграционного тестирования. При таком подходе значительно упрощается локализация ошибок: если модули протестированы по отдельности, то ошибка при их совместной работе есть проблема их интерфейса.

Несмотря на то, что интеграционные тесты не являются юнит-тестами в классическом понимании, при выполнении задач лабораторной работы необходимо использовать библиотеку JUnit и написать модуль юнит-тестирования вручную (используя Java-аннотации @Test и документацию библиотеки JUnit).

JUnit:

Таблица 6.1 – Аннотации JUnit

Аннотация	Описание
@Test public void method()	Помечает метод как тестовый
@Before public void method()	Метод будет выполняться перед каждым тестом, может быть использован для подготовки тестового окружения (инициализации, чтения данных)
@After public void method()	Метод будет выполняться после каждого теста. Может использоваться для отчистки окружения, удаления временных данных.

Таблица 6.2 – Тестовые методы

Сигнатура	Описание
fail(String)	Вызывает сбой. Может быть использован, чтобы удостовериться, что определенная часть кода не достигнута или пока тестовый метод не реализован.
assertTrue(true) / assertTrue(false)	Постоянно будет true / false. Может быть использован, чтобы предопределить результат теста, пока он не реализован.
assertsEquals([String message], expected, actual)	Проверяет, равны ли две величины. Важно: для массивов проверяется ссылка а не содержимое.
assertsEquals([String message], expected, actual, tolerance)	Проверяет, совпадают ли величины float и double
assertNull([message], object)	Проверяет, что объект null.
assertNotNull([message], object)	Проверяет, что объект не null.
assertSame([String], expected, actual)	Проверяет, что обе переменные ссылаются на один объект.
assertNotSame([String], expected, actual)	Проверяет, что переменные ссылаются на разные объекты.
assertTrue([message], boolean condition)	Проверяет условие на истинность.

Задание:

1. Откройте выбранную IDE и создайте проект, реализующий математические функции сложения, умножения, деления, извлечения корня.
2. Подключите к проекту библиотеку JUnit.

3. Создайте каркас для юнит-тестов (Например, в IDE Eclipse можно выбрать нужный класс, открыть контекстное меню, и выбрать New->Junit test case, в появившемся диалоговом окне выбрать методы, для которых понадобятся юнит-тесты).
4. Создайте юнит-тест согласно описанным требованиям.
5. Отладьте и запустите юнит-тест.
6. Оцените результаты выполнения юнит-тестирования и сделайте соответствующие выводы.

Примечание:
Для работы создается отдельный репозиторий с файлом readme.md, в котором описывается задание и указывается ФИО и группа исполнителя.

Контрольные вопросы:

1. Для чего нужно тестировать программное обеспечение?
2. В чем различие отладки и тестирования программ?
3. Какие существуют разновидности тестирования?
4. Каковы задачи модульного тестирования?

Примерные темы курсовых работ

1. Разработка Telegram-бота предоставляющего справочную информацию о университете.
2. Разработка приложения морской бой с возможностью сетевой игры.
3. Разработка системы статистического анализа программного кода.
4. Разработка проекта мобильного приложения Велотрекер.
5. Разработка программной системы стеганографического встраивания информации в цифровое изображение.
6. Разработка программной системы стеганографического встраивания информации в цифровое аудио.
7. Разработка программного комплекса для обработки данных на выбранную тематику.
8. Разработка программной системы встраивания цифрового водяного знака в цифровое изображение.

Вопросы к экзамену

1. Системы контроля версий: определение, методы хранение изменений.
2. Типы систем контроля версий. Сравнение различных систем.
3. Системы контроля версий: ежедневный цикл работы.
4. Системы контроля версий: слияние версий, конфликты и их разрешения.
5. Распределенные системы контроля версий.
6. Системы автоматической сборки проекта: жизненные циклы.
7. Системы автоматической сборки проекта: зависимости, управление версиями.
8. Кросс-компиляция. Процессоры и микроконтроллеры семейства ARM.
9. Написание тестов: Модульное тестирование.
10. Разработка через тестирование.
11. Циклы разработки через тестирование.
12. Рефакторинг.
13. Интеграционное тестирование.
14. Тестирование: Fake-, mock-объекты.
15. Контейнеризация приложений.

16. Средства обеспечивающие эмуляцию и виртуализацию.
17. Оркестровка контейнеризированных приложений: концепция.
18. Оркестровка контейнеризированных приложений: архитектура и компоненты, подсистема управления.
19. Оркестровка контейнеризированных приложений: компоненты узлов, масштабируемость.
20. Кластеризация контейнерных приложений.
21. Оркестровка контейнеризированных приложений: разработка и развертывание.

7 ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ В ПРОЦЕССЕ ОБУЧЕНИЯ

Информационные технологии – обучение в электронной образовательной среде с целью расширения доступа к образовательным ресурсам, увеличения контактного взаимодействия с преподавателем, построения индивидуальных траекторий подготовки, объективного контроля и мониторинга знаний студентов.

В образовательном процессе по дисциплине используются следующие информационные технологии, являющиеся компонентами Электронной информационно-образовательной среды БГПУ:

- Официальный сайт БГПУ;
- Корпоративная сеть и корпоративная электронная почта БГПУ;
- Система электронного обучения ФГБОУ ВО «БГПУ»;
- Система тестирования на основе единого портала «Интернет-тестирования в сфере образования www.i-exam.ru»;
- Система «Антиплагиат.ВУЗ»;
- Электронные библиотечные системы;
- текстовый процессор Microsoft Office Word;
- офисное приложение Microsoft Office Excel;
- офисное приложение Microsoft Office Power Point;
- средства разработки Java: JDK, JRE, NetBeans, Git, Maven, Gradle, Docker.

8 ОСОБЕННОСТИ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ ИНВАЛИДАМИ И ЛИЦАМИ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

При обучении лиц с ограниченными возможностями здоровья применяются адаптивные образовательные технологии в соответствии с условиями, изложенными в раздел «Особенности организации образовательного процесса по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья» основной образовательной программы (использование специальных учебных пособий и дидактических материалов, специальных технических средств обучения коллективного и индивидуального пользования, предоставление услуг ассистента (помощника), оказывающего обучающимся необходимую техническую помощь и т.п.) с учётом индивидуальных особенностей обучающихся.

9 СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ

9.1 Литература

1. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для вузов / С. В. Зыков. – Москва : Издательство Юрайт, 2022. – 155 с. – (Высшее образование). – ISBN 978-5-534-00850-0. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/490423 (дата обращения: 10.10.2022).
2. Стелтинг, С. Применение шаблонов Java тм : [Справоч. руководство разработчика по архитектуре шаблонов для платформы Java] / С. Стелтинг, О. Маассен. – М. ; СПб. ; Киев : Вильямс, 2002. – 563 с. (6 экз.)
3. Лаврищева, Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства : учебник для вузов / Е. М. Лаврищева. – 2-е изд., испр. – Москва : Издательство Юрайт, 2022. – 280 с. – (Высшее образование). – ISBN 978-5-534-01056-5. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/491048 (дата обращения: 10.10.2022).
4. Тузовский, А. Ф. Проектирование и разработка web-приложений : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2022. – 218 с. – (Высшее образование). – ISBN 978-5-534-00515-8. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/490128 (дата обращения: 10.10.2022).
5. Хорев, Павел Борисович. Объектно-ориентированное программирование : учеб. пособие для студ. вузов / П. Б. Хорев. – 3-е изд., испр. – М. : Академия, 2011. – 446 с. (10 экз.)

9.2 Базы данных и информационно-справочные системы

1. Информационная система «Единое окно доступа к образовательным ресурсам» – Режим доступа: http://www.window.edu.ru
2. Сайт Федеральной службы по интеллектуальной собственности, патентам и товарным знакам (Роспатента). – Режим доступа: http://www.fips.ru/rospatent/index.htm
3. Федеральный портал «Российское образование» – Режим доступа: http://www.edu.ru
4. Федеральный центр информационно-образовательных ресурсов – Режим доступа: http://fcior.edu.ru

9.3 Электронно-библиотечные ресурсы

1. ЭБС «Юрайт». – Режим доступа: https://urait.ru
2. Полпред (обзор СМИ). – Режим доступа: https://polpred.com/news

10 МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ БАЗА

Для проведения занятий лекционного и семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации используются аудитории, оснащённые учебной мебелью, аудиторной доской, компьютером(рами) с установленным лицензионным специализированным программным обеспечением, коммутатором для выхода в электронно-библиотечную систему и электронную информационно-образовательную среду БГПУ, мультимедийными проекторами, экспозиционными экранами, учебно-наглядными пособиями мультимедийные презентации).

Самостоятельная работа студентов организуется в аудиториях оснащенных компьютерной техникой с выходом в электронную информационно-образовательную среду вуза,

в специализированных лабораториях по дисциплине, а также в залах доступа в локальную сеть БГПУ, в лаборатории психолого-педагогических исследований и др.

Лицензионное программное обеспечение: операционные системы семейства Windows, Linux; офисные программы Microsoft office, Libreoffice, OpenOffice; Java Development Kit, NetBeans, Git, Maven, Gradle, Docker.

Разработчик: Антонов А.А., к.ф.-м.н., доцент кафедры информатики и МПИ.

11 ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ

Утверждение изменений и дополнений в РПД для реализации в 2025/2026 уч. г.

РПД обсуждена и одобрена для реализации в 2025/2026 уч. г. без изменений на заседании кафедры информатики и методики преподавания информатики (протокол №6 от 26.05.2025 г.).