МИНОБРНАУКИ РОССИИ

Глазовский инженерно-экономический институт (филиал) Федерального государственного бюджетного образовательногоучреждения высшего образования «Ижевский государственный технический университет имени М.Т. Калашникова» (ГИЭИ (филиал) ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

УТВЕРЖДАЮ

Директор _____ М.А.Бабушкин

14 06 2023 г.

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 3DA273AA38C240BA869FA062FFFB332C
Владелец: Золотов Алексей Васильевич
Действителен: с 29.07.2026 до 22.10.2027

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
Объектно-ориентированное программирование

направление подготовки: **09.03.01 «Информатика и вычислительная техника»**

направленность (профиль): **Автоматизированные системы обработки информации и управления**

уровень образования: **бакалавриат**

форма обучения: **очная**

общая трудоемкость дисциплины составляет: **10 зачетных единиц**

Кафедра «Машиностроение и информационные технологии»

Составитель:

Рабочая программа составлена в соответствии с требованиями федерального государственногообразовательного стандарта высшего образования по направлению подготовки 09.03.01 «Информатика и вычислительная техника» и рассмотрена на заседании кафедры.

Рабочая программа составлена в соответствии с требованиями федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.03.01 «Информатика и вычислительная техника» и рассмотрена на заседании кафедры.

Протокол от 22.05.2023 г. № 5

Заведующий кафедрой А.Г. Горбушин

22.05.2023 г.

СОГЛАСОВАНО

Количество часов рабочей программы и формируемые компетенции соответствуют учебному плану по направлению подготовки 09.03.01 «Информатика и вычислительная техника», профиль «Автоматизированные системы обработки информации и управления».

Протокол заседания учебно-методической комиссии от 22 мая 2023 г. № 2

Председатель учебно-методической комиссии ГИЭИ

А.Г. Горбушин

Руководитель образовательной программы А.Г. Горбушин

22.05.2023 г.

Аннотация к дисциплине

Название дисциплины	Объектно-ориентированное программирование
Направление подготовки (специальность)	09.03.01 "Информатика и вычислительная техника"
Направленность (профиль/программа/специализация)	Автоматизированные системы обработки информации и управления
Место дисциплины	Дисциплина относится к части, формируемой участниками образовательных отношений Блока 1 «Дисциплины (модули)» ООП.
Трудоемкость (з.е. / часы)	10 з.е. / 360 часов
Цель изучения дисциплины	Получение теоретических сведений и практических навыков постановки задачи, а также разработки вычислительных и информационных систем на объектно-ориентированном языке программирования (ООП) и их тестирования.
Компетенции, формируемые в результате освоения дисциплины	ПК-1 Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы. ПК-4 Способен разрабатывать тестовые случаи, проводить тестирование и исследование результатов тестирования ПК-5 Способен разрабатывать требования и проектировать программное обеспечение.
Содержание дисциплины (основные разделы и темы)	Тестирование. Очереди, стеки, дженерики. Листы и словари. Делегаты. Элементы функционального программирования. LINQ. Обработка графовых структур. Жадные алгоритмы. Динамическое программирование. Структуры данных. События. Оконные приложения. Многопоточное программирование. Рефлексия типов. Инкапсуляция - теория и практика. Наследование и полиморфизм - теория и практика. Generics. Делегирование. Рефлексия. DDD. Fluent API. Модульность. Управление зависимостями. DI-контейнеры. Функциональный стиль. Управление ресурсами. Работа с файлами.
Форма промежуточной аттестации	Зачет с оценкой (3 семестр) Экзамен (4 семестр), Курсовая работа (4 семестр)

1. Цели и задачи дисциплины:

Целью освоения дисциплины является получение теоретических сведений и практических навыков постановки задачи, а также разработки вычислительных и информационных систем на объектно-ориентированном языке программирования (ООП) и их тестирования.

Задачи дисциплины:

- приобретение знаний о современных объектно-ориентированных языках программирования и умение использовать их как инструмент разработки информационных систем;
- приобрести знания о методах разработки программного обеспечения,
- приобрести знания и умения тестирования разработанных программных систем.

2. Планируемые результаты обучения

В результате освоения дисциплины у студента должны быть сформированы

Знания, приобретаемые в ходе освоения дисциплины

№	Знания
1.	Сложные языковые конструкции C#: обобщённые типы (generics), генераторы последовательностей, LINQ, а также необходимый набор алгоритмов и структур данных
2.	Методы проектирования программной системы: консольные приложения и шаблоны проектирования ПО в C#
3.	Тестирование, виды тестов, техники тестирования

Умения, приобретаемые в ходе освоения дисциплины

№	Умения
1	Проектировать структуру и алгоритмы программной системы
2	Разрабатывать программные системы на языке программирования C#
3	Анализировать тестовые случаи, писать тесты, анализировать отчеты тестирования
4	Выбирать средства реализации требований к программному обеспечению, использовать существующие типовые решения и шаблоны проектирования программного обеспечения, вырабатывать варианты реализации программного обеспечения

Навыки, приобретаемые в ходе освоения дисциплины

№	Навыки
1.	Работы в интегрированной среде разработки
2.	Алгоритмизации и программирования на процедурных и объектно-ориентированных языках программирования
3.	Разработки архитектуры программного обеспечения
4.	Документирования тестов, разработки скриптов для автоматизации тестирования

Компетенции, приобретаемые в ходе освоения дисциплины

Компетенции	Индикаторы	Знания	Умения	Навыки
ПК-1 Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы.	ПК-1.1 Знать: архитектуру, устройство и функционирование вычислительных и информационных систем, программные средства и платформы инфраструктуры информационных технологий организации, современные подходы и стандарты автоматизации организации, современные языки программирования, теорию баз данных, основы современных операционных систем, сетевые протоколы и коммуникационное оборудование ПК-1.2 Уметь: проектировать архитектуру,	1	1-2	1-2

	структуру и алгоритмы функционирования вычислительных и информационных систем, разрабатывать инфраструктуру информационных технологий предприятия, применять современные подходы и стандарты автоматизации организации, проектировать информационное, программное и аппаратное обеспечение, оценивать объемы и сроки выполнения работ ПК-1.3 Владеть: навыками проектирования и реализации вычислительных и информационных систем, навыками создания программ на современных языках программирования, навыками работы с аппаратным и сетевым оборудованием, навыками создания баз данных, навыками проектирования дизайна информационных систем, навыками создания пользовательской документации			
ПК-4 Способен разрабатывать тестовые случаи, проводить тестирование и исследование результатов тестирования	ПК-4.1 Знать: классификацию видов и типов тестирования, техники тестирования, инструменты выполнения тестов, типы дефектов и их классификации, жизненный цикл программного обеспечения и процесса тестирования; ПК-4.2 Уметь: анализировать тестовые случаи, сопоставлять и анализировать информацию, проводить сравнительный анализ, работать с текстовыми редакторами и другими пакетами для создания отчетов по результатам тестирования, пользоваться системами отслеживания ошибок; ПК-4.3 Владеть: навыками документирования тестов, навыками разработки скриптов для автоматизации тестирования, навыками работы в качестве тестировщика в команде с разработчиками, навыками использования специального программного обеспечения для автоматизированного тестирования.	3	3	4
ПК-5 Способен разрабатывать требования и проектировать программное обеспечение.	ПК-5.1 Знать: методологии разработки программного обеспечения и технологии программирования, методы и средства проектирования программного обеспечения, программных интерфейсов и баз данных, языки формирования функциональных спецификаций; ПК-5.2 Уметь: согласовывать требования к программному обеспечению с заинтересованными сторонами, выбирать средства реализации требований к программному обеспечению, использовать существующие типовые решения и шаблоны проектирования программного обеспечения, вырабатывать варианты реализации программного обеспечения, проводить оценку и обоснование рекомендуемых решений; ПК-5.3 Владеть: навыками анализа требований к программному обеспечению, навыками разработки технических спецификаций на программные компоненты и их взаимодействие,	2	4	3

	навыками разработки, изменения и согласования архитектуры программного обеспечения, навыками проектирования структур данных, баз данных, программных интерфейсов.			

3. Место дисциплины в структуре ООП

Дисциплина относится к части, формируемой участниками образовательных отношений Блока 1 «Дисциплины (модули)» ООП.

Дисциплина изучается на 2 курсе в 3,4 семестрах.

Изучение дисциплины базируется на знаниях, умениях и навыках, полученных при освоении дисциплин (модулей): «Информатика», «Программирование».

Перечень последующих дисциплин (модулей), для которых необходимы знания, умения и навыки, формируемые данной учебной дисциплиной (модулем): «Базы данных», «Интернет-программирование».

4 Структура и содержание дисциплины

4.1 Структура дисциплины

№ п/п	Раздел дисциплины. Форма промежуточной аттестации	Всего часов на раздел	Семестр	Распределение трудоемкости раздела (в часах) по видам учебной работы					Содержание самостоятельной работы
				контактная				СРС	
				лек	пр	лаб	КЧА		
1	2	3	4	5	6	7	8	10	11
1	Тестирование	7	3		2			5	Подготовка к практическим занятиям, тесту
2	Очереди, стеки, дженерики	11	3		2	8		5	Подготовка к практическим занятиям, тесту
3	Листы и словари	7	3		2			5	Подготовка к практическим занятиям, тесту
4	Делегаты	13	3		4			5	Подготовка к практическим занятиям, тесту
5	Элементы функционального програм-мирования	11	3		2			5	Подготовка к практическим занятиям, тесту,
6	LINQ	7	3		2	8		5	Подготовка к практическим занятиям, тесту
7	Обработка графовых структур	14	3		2			8	Подготовка к практическим занятиям, тесту
8	Жадные алгоритмы	7	3		2	8		5	Подготовка к практическим занятиям, тесту
9	Динамическое программирование	18	3		4			10	Подготовка к практическим занятиям, тесту
10	Структуры данных	11	3		2			5	Подготовка к практическим занятиям, тесту
11	События	7	3		2	8		5	Подготовка к практическим занятиям, тесту
12	Оконные приложения	11	3		2			5	Подготовка к практическим занятиям, тесту

13	Многопоточное программирование	7	3		2			5	Подготовка к практическим занятиям, тесту
14	Рефлексия типов	11	3		2			5	Подготовка к практическим занятиям, тесту
	Зачет с оценкой	2					0,4	1,6	Зачет выставляется по совокупности результатов текущего контроля успеваемости
	Итого за 3 семестр	**144**			**32**	**32**	**0,4**	**79,6**	
15	Инкапсуляция - теория и практика	18	4		4	8		10	Подготовка к практическим занятиям, тесту
16	Наследование и полиморфизм - теория и практика	18	4		4			10	Подготовка к практическим занятиям, тесту
17	Generics	7	4		2	8		5	Подготовка к практическим занятиям, тесту
18	Делегирование	11	4		2			5	Подготовка к практическим занятиям, тесту
19	Рефлексия	7	4		2			5	Подготовка к практическим занятиям, тесту
20	DDD	11	4		2			5	Подготовка к практическим занятиям, тесту
21	Fluent API	7	4		2	16		5	Подготовка к практическим занятиям, тесту
22	Модульность	11	4		2			5	Подготовка к практическим занятиям, тесту
23	Управление зависимостями	7	4		2			5	Подготовка к практическим занятиям, тесту
24	DI-контейнеры	11	4		2			5	Подготовка к практическим занятиям, тесту
25	Функциональный стиль	7	4		2			5	Подготовка к практическим занятиям, тесту
26	Управление ресурсами	11	4		2			5	Подготовка к практическим занятиям, тесту
27	Работа с файлами	7	4		2			5	Подготовка к практическим занятиям, тесту
28	Исключения	11	4		2			5	Подготовка к практическим занятиям, тесту
29	Курсовая работа	36					3,0	33,0	Подготовка к защите курсовой работы
	Экзамен	36					0,4	35,6	Экзамен выставляется по совокупности результатов текущего контроля успеваемости или проводится в письменной форме.
	Итого за 4 семестр	**216**			**32**	**32**	**3,4**	**148,6**	
	Итого	**360**			**64**	**64**	**3,8**	**228,2**	

4.2 Содержание разделов курса и формируемых в них компетенций

№ п/п	Раздел дисциплины	Коды компетенции и индикаторов	Знания	Умения	Навыки	Форма контроля
1	Тестирование	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	3	3	4	тест
2	Очереди, стеки, дженерики	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест, выполнение лабораторной работы
3	Листы и словари	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест
4	Делегаты	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	защита лабораторной работы тест
5	Элементы функционального программирования	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест, выполнение лабораторной работы
6	LINQ	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	работа на практических занятиях: текущий контроль решения задач
7	Обработка графовых структур	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест, защита лабораторной работы
8	Жадные алгоритмы	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест
9	Динамическое программирование	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	выполнение лабораторной работы тест
10	Структуры данных	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест, защита лабораторной работы

11	События	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	тест
12	Оконные приложения	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	выполнение лабораторной работы
13	Многопоточное программирование	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	тест
14	Рефлексия типов	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	защита лабораторной работы
15	Инкапсуляция - теория и практика	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	Тест, выполнение лабораторной работы
16	Наследование и полиморфизм - теория и практика	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	Тест, защита лабораторной работы
17	Generics	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	работа на практических занятиях: текущий контроль решения задач
18	Делегирование	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	выполнение лабораторной работы
19	Рефлексия	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	1	1,2	1,2	работа на практических занятиях: текущий контроль решения задач
20	DDD	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	тест, защита лабораторной работы
21	Fluent API	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	3	3	4	тест

22	Модульность	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	выполнение лабораторной работы
23	Управление зависимостями	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	работа на практических занятиях: текущий контроль решения задач
24	DI-контейнеры	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	защита лабораторной работы
25	Функциональный стиль	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	работа на практических занятиях: текущий контроль решения задач
26	Управление ресурсами	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	тест, выполнение лабораторной работы
27	Работа с файлами	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	2	4	2,3	работа на практических занятиях: текущий контроль решения задач
28	Исключения	ПК-1.1; ПК-1.2; ПК-1.3; ПК-4.1; ПК-4.2; ПК-4.3; ПК-5.1; ПК-5.2; ПК-5.3	3	3	4	тест, защита лабораторной работы

4.3 Наименование тем лекций, их содержание и объем в часах

Лекции учебным планом не предусмотрены

4.4 Наименования тем практических занятий, их содержание и объем в часах

4.4.1. Третий семестр

№	№ раздела дисциплины	Название практических работ	Объем в часах
1	1	Тестирование	2
2	2	Очереди, стеки, дженерики	2
3	3	Листы и словари	2
4	4	Делегаты	4
5	5	Элементы функционального программирования	2
6	6	LINQ	2
7	7	Обработка графовых структур	2
8	8	Жадные алгоритмы	2

9	9	Динамическое программирование	4
10	10	Структуры данных	2
11	11	События	2
12	12	Оконные приложения	2
13	13	Многопоточное программирование	2
14	14	Рефлексия типов	2
	Всего		32

4.4.2. Четвертый семестр

№	№ раздела дисциплины	Название практических работ	Объем в часах
1	15	Инкапсуляция - теория и практика	4
2	16	Наследование и полиморфизм - теория и практика	4
3	17	Generics	2
4	18	Делегирование	2
5	19	Рефлексия	2
6	20	DDD	2
7	21	Fluent API	2
8	22	Модульность	2
9	23	Управление зависимостями	2
10	24	DI-контейнеры	2
11	25	Функциональный стиль	2
12	26	Управление ресурсами	2
13	27	Работа с файлами	2
14	28	Исключения	2
	Всего		32

4.5 Наименование тем лабораторных работ, их содержание и объем в часах

4.5.1. Третий семестр

№ п/п	№ раздела дисциплины	Наименование лабораторных работ	Трудоемкость (час)
1	2	Решение задач на стеки и очереди	8
2	6	Решение задач с LINQ	8
3	8	Решение задач с реализацией жадного алгоритма	8
4	11	Решение задач с реализацией событий	8
	Всего		32

4.5.2. Четвертый семестр

№ п/п	№ раздела дисциплины	Наименование лабораторных работ	Трудоемкость (час)
1	15	Решение задач обработки объектов классов	8
2	17	Решение задач разработки оконного приложения	8
3	21	Решение задач с использованием файлов	16
	Всего		32

5 Оценочные материалы для текущего контроля успеваемости и промежуточной аттестации по дисциплине

Для контроля результатов освоения дисциплины проводятся:

– тестирование;
– защиты лабораторных работ;
– защита курсовой работы;
– зачет с оценкой и экзамен.

Примечание: оценочные материалы (типовые варианты тестов, контрольных работ и др.) приведены в приложении к рабочей программе дисциплины.

Промежуточная аттестация по итогам освоения дисциплины – зачет с оценкой и экзамен

6 Учебно-методическое и информационное обеспечение дисциплины

а) Основная литература

1 Петров, В. Ю. Информатика. Алгоритмизация и программирование. Часть 1 [Электронный ресурс] : учебное пособие / В. Ю. Петров. — Электрон. текстовые данные. — СПб. : Университет ИТМО, 2016. — 93 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/66473.html

2 Тюльпинова, Н. В. Алгоритмизация и программирование [Электронный ресурс] : учебное пособие / Н. В. Тюльпинова. — Электрон. текстовые данные. — Саратов : Вузовское образование, 2019. — 200 c. — 978-5-4487-0470-3. — Режим доступа: http://www.iprbookshop.ru/80539.html

3 Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. А. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html

4 Букунов, С. В. Основы объектно-ориентированного программирования [Электронный ресурс] : учебное пособие / С. В. Букунов, О. В. Букунова. — Электрон. текстовые данные. — СПб. : Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2017. — 196 c. — 978-5-9227-0713-8. — Режим доступа: http://www.iprbookshop.ru/74339.html

б) Дополнительная литература

1 Разумавская, Е. А. Алгоритмизация и программирование [Электронный ресурс] : практическое пособие / Е. А. Разумавская. — Электрон. текстовые данные. — СПб. : Санкт-Петербургский юридический институт (филиал) Академии Генеральной прокуратуры РФ, 2015. — 49 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/65427.html

2 Туральчук, К. А. Параллельное программирование с помощью языка C# [Электронный ресурс] / К. А. Туральчук. — 3-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 189 c. — 978-5-4486-0506-2. — Режим доступа: http://www.iprbookshop.ru/79714.html

3 Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html

в) методические указания:

1. Малина О.В. Методические указания к лабораторным работам для обучающихся по направлению 09.03.01 «Информатика и вычислительная техника», всех форм обучения при изучении дисциплины «Объектно-ориентированное программирование». Ижевск: ИжГТУ, 2019 (Элект. издание) Рег.номер 075/53-ИИВТ

2. Малина О.В. Методические указания к выполнению курсовой работы для обучающихся по

направлению 09.03.01 «Информатика и вычислительная техника», всех форм обучения при изучении дисциплины «Объектно-ориентированное программирование». Ижевск: ИжГТУ, 2019 (Элект. издание) Рег.номер 076/53-ИИВТ

г) **перечень ресурсов информационно-коммуникационной сети Интернет**

1. Электронно-библиотечная система IPRbooks
http://istu.ru/material/elektronno-bibliotechnaya-sistema-iprbooks
2. Электронный каталог научной библиотеки ИжГТУ имени М.Т. Калашникова Web ИРБИС
http://94.181.117.43/cgi-bin/irbis64r_12/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=IBIS&P21DBN=IBIS
3. Национальная электронная библиотека - http://нэб.рф
4. Мировая цифровая библиотека - http://www.wdl.org/ru
5. Международный индекс научного цитирования Web of Science - http://webofscience.com
6. Научная электронная библиотека eLIBRARY.RU – https://elibrary.ru/defaultx.asp
7. Справочно-правовая система КонсультантПлюс http://www.consultant.ru

д) лицензионное и свободно распространяемое программное обеспечение:

1. Microsoft Imagine Premium: Visual Studio
2. Microsoft Office Standard 2007
3. Doctor Web Enterprise Suite

7. Материально-техническое обеспечение дисциплины:

1. Лекционные занятия.

Учебные аудитории для лекционных занятий укомплектованы мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории (наборы демонстрационного оборудования (проектор, экран, компьютер/ноутбук), учебно-наглядные пособия, тематические иллюстрации).

2. Практические занятия.

Учебные аудитории для практических занятий укомплектованы специализированной мебелью и техническими средствами обучения (проектор, экран, компьютер/ноутбук).

3. Лабораторные работы.

Для лабораторных занятий используется аудитория № 209, оснащенная следующим оборудованием: доской, компьютерами с возможностью подключения к сети «Интернет», столами, стульями.

4. Самостоятельная работа.

Помещения для самостоятельной работы оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и доступом к электронной информационно-образовательной среде ИжГТУ имени М.Т. Калашникова:

- научная библиотека ИжГТУ имени М.Т. Калашникова;
- помещение для самостоятельной работы обучающихся

При необходимости рабочая программа дисциплины (модуля) может быть адаптирована для обеспечения образовательного процесса инвалидов и лиц с ограниченными возможностями здоровья, в том числе для обучения с применением дистанционных образовательных технологий. Для этого требуется заявление студента (его законного представителя) и заключение психолого-медико-педагогической комиссии (ПМПК).

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Ижевский государственный технический университет имени М.Т. Калашникова»

Оценочные средства по дисциплине

Объектно-ориентированное программирование
наименование – полностью

направление 09.03.01 «Информатика и вычислительная техника»
код, наименование – полностью

профиль Автоматизированные системы обработки информации и управления
наименование – полностью

уровень образования: бакалавриат

форма обучения: очная
очная/очно-заочная/заочная

общая трудоемкость дисциплины составляет: 10 зачетных единиц

1. Оценочные средства

Оценивание формирования компетенций производится на основе результатов обучения, приведенных в п. 2 рабочей программы и ФОС. Связь разделов компетенций, индикаторов и форм контроля (текущего и промежуточного) указаны в таблице 4.2 рабочей программы дисциплины.

Оценочные средства соотнесены с результатами обучения по дисциплине и индикаторами достижения компетенций, представлены ниже.

№ п/п	Коды компетенции и индикаторов	Результат обучения *(знания, умения и навыки)*	Формы текущего и промежуточного контроля
1	ПК-1.1 ПК-1.2 ПК-1.3	З1: сложные языковые конструкции C#: обобщённые типы (generics), генераторы последовательностей, LINQ, а также необходимый набор алгоритмов и структур данных; У1: проектировать структуру и алгоритмы программной системы; У2: разрабатывать программные системы на языке программирования C#; Н1: навыками работы в интегрированной среде разработки; Н2: навыками алгоритмизации и программирования на процедурных и объектно-ориентированных языках программирования;	Работа на практических занятиях: текущий контроль решения задач Тест защита лабораторных работ Зачет с оценкой Экзамен
2	ПК-4.1 ПК-4.2 ПК-4.3	З3: тестирование, виды тестов, техники тестирования. У3: анализировать тестовые случаи, писать тесты, анализировать отчеты тестирования; Н4: к документирования тестов, разработки скриптов для автоматизации тестирования.	Работа на практических занятиях: текущий контроль решения задач Тест защита лабораторных работ Зачет с оценкой Экзамен
3	ПК-5.1 ПК-5.2 ПК-5.3	З2: методы проектирования программной системы: консольные приложения и шаблоны проектирования ПО в C#; У4: выбирать средства реализации требований к программному обеспечению, использовать существующие типовые решения и шаблоны проектирования программного обеспечения, вырабатывать варианты реализации программного обеспечения; Н3: навыками разработки архитектуры программного обеспечения;	Работа на практических занятиях: текущий контроль решения задач Тест защита лабораторных работ Зачет с оценкой Экзамен

Типовые задания для оценивания формирования компетенций

Наименование: зачет с оценкой
***Представление в ФОС*:** перечень вопросов
Перечень вопросов для проведения зачета:
1) Тестирование
2) Очереди, стеки, дженерики
3) Листы и словари

4) Делегаты
5) Элементы функционального программирования
6) LINQ
7) Обработка графовых структур
8) Жадные алгоритмы
9) Динамическое программирование
10) Структуры данных
11) События
12) Оконные приложения
13) Многопоточное программирование
14) Рефлексия типов

Критерии оценки:
Приведены в разделе 2

Наименование: экзамен
***Представление в ФОС*:** перечень вопросов
Перечень вопросов для проведения экзамена:
1) Тестирование
2) Очереди, стеки, дженерики
3) Листы и словари
4) Делегаты
5) Элементы функционального программирования
6) LINQ
7) Обработка графовых структур
8) Жадные алгоритмы
9) Динамическое программирование
10) Структуры данных
11) События
12) Оконные приложения
13) Многопоточное программирование
14) Рефлексия типов
15) Инкапсуляция - теория и практика
16) Наследование и полиморфизм - теория и практика
17) Generics
18) Делегирование
19) Рефлексия
20) DDD
21) Fluent API
22) Модульность
23) Управление зависимостями
24) DI-контейнеры
25) Функциональный стиль
26) Управление ресурсами
27) Работа с файлами
28) Исключения

Критерии оценки:
Приведены в разделе 2

Наименование: тест
***Представление в ФОС*:** набор вариантов заданий

Варианты заданий:

по разделу «Тестирование»

1. Чем автоматизированные тесты лучше ручного тестирования?

Автотесты выполняются быстрее, поэтому их можно запускать часто, но тратить время программиста только на непрошедшие тесты

- [ ] Для больших программ полное ручное тестирование предполагает огромное количество ручной работы, поэтому проводится крайне редко
- [ ] Работа автоматических тестов не зависит от концентрации, усидчивости и других личных качеств тестировщика

2. Почему тестирование важно?

- [ ] Некоторые ошибки могут привести к значительному ущербу, а мероприятия по их поиску и исправлению выгоднее исправления последствий
- [ ] Ошибки совершают даже очень опытные разработчики
- [ ] Тестирование ускоряет процесс разработки на начальном этапе
- [ ] Тестирование позволяет гарантировать, что в создаваемом ПО не будет ни одной ошибки
- [ ] При разработке веб-сервисов (например, заказа билетов) ошибки могут мгновенно приводить к финансовым убыткам разработчиков сервиса

Варианты заданий:

по разделу «Очереди, стеки»

Тест

Изучите следующий код:

```
Queue<int> queue = new Queue<int>();
queue.Enqueue(0);
queue.Enqueue(1);
queue.Dequeue();
queue.Enqueue(2);
queue.Dequeue();
```

1. Какой элемент остался в стеке после выполнения этого кода? 1 балл

- ○ 0
- ○ 1
- ○ 2

2. Какой элемент остался в очереди после выполнения этого кода? 1 балл

- ○ 0
- ○ 1
- ○ 2

3. Какую из проблем решает использование односвязного списка вместо List<int> при реализации очереди? 1 балл

- ○ Enqueue работает за O(n)
- ○ Dequeue работает за O(n)
- ○ Никакой проблемы нет, просто другой способ реализации

4. Если в очереди, реализованной на связных списках, tail == head, что это может означать? 1 балл

- [ ] Очередь пуста
- [ ] Очередь содержит один элемент
- [ ] Очередь содержит более одного элемента
- [ ] Произошло нарушение целостности очереди

Варианты заданий:
по разделу «Листы, словари»

Тест
Изучите следующий код:

```
public class List<T> : IEnumerable<T>
{
    T[] collection = new T[100];
    int count = 0;

    public void Add(T value)
    {
        if (count == collection.Length)
        {
            var enlargedCollection = new T[count * 2];
            Array.Copy(collection, enlargedCollection, collection.Length);
            collection = enlargedCollection;
        }
        collection[count++] = value;
    }

    public T this[int index]
    {
        get { return collection[index]; }
        set { collection[index] = value; }
    }

    public bool Contains(T value)
    {
        for (int i = 0; i < count; i++)
            if (collection[i].Equals(value)) return true;
        return false;
    }
}
```

1. Что вы можете сказать об этом коде? 1 балл

- [ ] Сложность операции Add всегда \Theta(1)Θ(1)
- [ ] Сложность операции доступа к элементу по индексу \Theta(1)Θ(1)
- [ ] Сложность операции Contains \Theta(n)Θ(*n*)

2. Оператор == является синонимом вызова метода Equals 1 балл

- Верно
- Неверно

3. В C# возможно переопределить оператор сложения (+) двух чисел типа int 1 балл

- Верно
- Неверно

4. В C# возможно переопределить оператор сложения (+) числа с объектом собственного нового класса 1 балл

- Верно
- Неверно

5. В C# можно сделать так, чтобы выражение a == b || a != b равнялось false 1 балл

- Верно
- Неверно

6. В C# можно сделать так, чтобы выражение a == b | a != b равнялось строке "Не-не-не, Дэвид Блэйн" 1 балл

- Верно
- Неверно

7. Что из этого является корректным определением оператора в классе A? 1 балл

- public static A operator ==(A a, A b)
- public static bool operator ==(A a, int b)
- static bool operator ==(int a, bool b)
- public bool operator ==(A a, A b)
- bool operator ==(A a, A b)
- public static operator ==(int a, A b)

8. Когда стоит перегружать операторы? Выберите верные утверждения 1 балл

- Всегда, когда можно — операторы делают код более компактным.
- Когда вы можете придумать интересную и забавную семантику для оператора. Например, ^ для преобразования строки в верхний регистр.
- Не стоит перегружать оператор, если это сделает код более загадочным.
- Не стоит перегружать оператор, если это может подтолкнуть читателя к неверной интерпретации кода.

Варианты заданий:
по разделу «Делегаты»

1. Какой тип соответствует функции, которая принимает два параметра типов int и string и возвращает double? 1 балл

- delegate<int, string>
- delegate
- Action
- Func

- Action<int, string, double>
- Action<int, string>
- Func<int, string, double>

Вы увидели такой код:

```
int result = (f[0](0))[0];
```

2. Какой тип может быть у f? *1 балл*

- Func<int>
- Func<int>[]
- Func<int[]>[]
- Func<int[], int>[]
- Func<int, List<int>>[]
- Action<Func<int, List<int>>>
- Action<Func<int, List<int>>>[]

Изучите код ниже:

```
var dictionary = new Dictionary<char, Action>();
for(char c='A'; c < 'Z'; c++)
   dictionary.Add(c, () => Console.Write(c));

dictionary['X']();
```

3. Что выводит этот код? *1 балл*

- символ 'A'
- символ 'X'
- символ 'Y'
- символ 'Z'
- ошибку

Варианты заданий:

по разделу «Элементы функционального программирования»

Тест

```
private static Func<int, int, int> Apply1(Func<int, int, int, int> func, int arg)
{
        return (x, y) => func(x, arg, y);
}

private static Func<int, int> Apply2(Func<int, int, int> func, int arg)
{
        return x => func(arg, x);
}

public static void Main()
{
        Func<int, int, int, int>f = (x, y, z) => x * y + z;
```

```
var x0 = f(1, 2, 3);
var x1 = Apply1(f, 100)(1, 11);

var g = Apply2(Apply1(f, 10), 5);
var x2 = g(3);
```

}*1. Чему равен x0? 1 балл*

2. Чему равен x1? 1 балл

3. Чему равен x2? 1 балл

Изучите следующий код

```
IEnumerable<int> GetSequence()
{
 for (var i = 0; i < 2; ++i)
  {
    Console.WriteLine("s");
    yield return i;
  }
}
foreach (var element in GetSequence())
   Console.WriteLine("f");
```

1. Что напечатает на консоль цикл выше? 1 балл

- s f s f
- s s f f
- f s f s

```
foreach (var element in GetSequence().ToList())
   Console.WriteLine("f");
```

2. Что напечатает на консоль цикл выше? 1 балл

- s f s f
- s s f f
- f s f s

```
var element = GetSequence().FirstOrDefault();

Console.WriteLine("f");
```

3. Что напечатает на консоль код выше? 1 балл

- s f
- s s f
- f

Изучите следующий код:

```
foreach (var element in GetSequence().ToList().Take(1))
   Console.WriteLine("f");
```

4. Что напечатает на консоль цикл выше? 1 балл

- s f
- s s f

- s s f f
- s f s f
- s

Варианты заданий:
по разделу «Обработка графовых структур»

Тест

В неориентированном графе нет вершин с нулевой степенью, зато есть вот такие ребра:
(0, 1)
(0, 2)
(0, 3)
(1, 2)
(1, 4)

1. Чему равна степень вершины 2? 1 балл

2. Какая вершина инцидентна вершине 4? 1 балл

3. Отметьте все верные факты про данный граф 1 балл

- Он является деревом
- В нем есть цикл
- Он связен
- В нем есть путь из вершины 0 в вершину 4
- В нем есть ровно одна вершина со степенью 1
- Вершина 0 является корнем
- Он является лесом

Варианты заданий:
по разделу «Жадные алгоритмы»

Тест

1. Комбинаторные задачи — это такие задачи, в которых ответ составлен из объектов той же природы, что и вход задачи. 1 балл

- Верно
- Неверно

2. Любую комбинаторную задачу можно решить полным перебором возможных вариантов ответа1 балл

- Верно
- Неверно

3. Для всех комбинаторных задач существуют алгоритмы эффективнее перебора вариантов ответа1 балл

- Верно
- Неверно

Варианты заданий:
по разделу «Динамическое программирование»

Тест

В онлайн игре World of Students есть возможность зарабатывать игровое серебро за участие в контрольных мероприятиях.
Сегодня запланировано 4 мероприятия:

1. С 9-00 до 11-00 семинар истории (50 серебра)
2. С 12-00 до 15-00 тест по русскому языку (200 серебра)
3. С 14-00 до 16-00 большая контрольная по основам программирования (90 серебра)
4. С 10-00 до 13-00 экзамен по математике (190 серебра)

Вы хотите заработать максимальное количество серебра, но, естественно, в каждый момент времени можете участвовать лишь в одном.

1. Как можно решать эту задачу сегодня и подобные задачи в последующие дни? 1 балл

- ☐ Перебрать все подмножества мероприятий и выбрать наилучшую комбинацию.
- ☐ С помощью динамического программирования
- ☐ Жадным алгоритмом, выбирая мероприятия, в порядке убывания их стоимости
- ☐ Жадным алгоритмом, выбирая мероприятия, в порядке убывания их времени окончания

2. Какое оптимальное расписание на сегодня? 1 балл

- ○ история, русский
- ○ история, математика
- ○ история, русский, основы программирования
- ○ русский и основы программирования
- ○ основы программирования и математика
- ○ русский, основы программирования и математика
- ○ русский, история, основы программирования и математика

Варианты заданий:
по разделу «Структуры данных»

Тест

Есть бинарное дерево, для каждого узла которого выполняются два условия:

1. Значение узла *меньше* значения в левом сыне, если он есть.
2. Значение узла *не меньше* значения в правом сыне, если он есть.

1. Что вы можете сказать о таком бинарном дереве? 1 балл

- ○ Это бинарное дерево поиска
- ○ Это куча
- ○ Это может быть не тем и не другим
- ○ Это и то и другое

2. Отметьте свойства реализации бинарного дерева поиска из лекций (то есть без алгоритма балансировки) 1 балл

- ☐ Максимальный элемент всегда находится в листе

- Сложность поиска в дереве логарифмически зависит от высоты дерева
- При добавлении N чисел по порядку начиная с меньших к большим, высота дерева окажется порядка N
- Если в левом сыне корня значение меньше, чем в корне, то и в правом поддереве корня может быть элемент меньший значения в корне
- Вставка элемента в бинарное дерево поиска имеет сложность \Theta(h)Θ(*h*), где h*h* — высота дерева

Есть бинарное дерево, для каждого узла которого выполняются два условия:

1. Значение узла *меньше* значения в левом сыне, если он есть.
2. Значение узла *меньше* значения в правом сыне, если он есть.

3. Что вы можете сказать о таком бинарном дереве? 1 балл

- Это бинарное дерево поиска
- Это куча
- Это может быть не тем и не другим
- Это и то и другое

```
var heap = new[] {-1, 1, 2, 3, 6, 5, 4};
```

4. Этот массив приоритетов представляет корректную кучу, в которой элементы кучи хранятся начиная с первой ячейки массива. 1 балл

- Верно
- Неверно

Варианты заданий:
по разделу «События»

Тест

1. Отметьте верные утверждения 1 балл

- Мультикаст-делегаты позволяют комбинировать несколько делегатов
- Мультикаст-делегаты можно вызывать так же, как и обычные делегаты
- Событие — это полный аналог свойства типа мультикаст-делегат
- Мультикаст-делегаты позволяют комбинировать делегаты различных типов
- Мультикаст-делегаты позволяют комбинировать даже делегаты, возвращающие значения

2. Отметьте верные утверждения 1 балл

- Событие — обертка над делегатом, которая помогает обеспечивать целостность
- Событие — один из предопределенных типов
- Событие класса нельзя вызвать из метода другого класса

Варианты заданий:
по разделу «Многопоточное программирование»

Тест

1. В одном процессе может быть несколько потоков 1 балл

- Верно

- Неверно

2. В одном потоке может быть несколько процессов *1 балл*

- Верно
- Неверно

3. Все потоки одного домена имеют общую память (видят одни и те же значения статических полей) *1 балл*

- Верно
- Неверно

4. В одном процессе может быть несколько доменов *1 балл*

- Верно
- Неверно

5. Несколько процессов одного потока могут иметь разделяемую память *1 балл*

- Верно
- Неверно

6. Процесс и поток это одно и то же *1 балл*

- Верно
- Неверно

Тест

1. Зачем нужна синхронизация потоков? *1 балл*

- Чтобы определить очередность выполнения разных потоков
- Чтобы повысить скорость работы многопоточных программ
- Чтобы обеспечить корректное использование разделяемого между потоками ресурса
- Чтобы передавать информацию из одного потока в другой

2. `lock` используется для синхронизации доступа к разделяемому ресурсу между потоками *1 балл*

- Верно
- Неверно

3. Потокобезопасные методы объекта можно вызывать из разных потоков без дополнительной синхронизации *1 балл*

- Верно
- Неверно

4. Потоки выполняются по очереди один за другим. Это можно использовать, для доказательства корректности программы *1 балл*

- Верно
- Неверно

5. Одним из классических способов взаимодействия потоков является потокобезопасная очередь *1 балл*

- Верно
- Неверно

6. Секция кода заключенная внутрь операции `lock(obj)` не начнет выполняться потоком до тех пор, пока 1 балл

- ... все остальные потоки не окажутся вне этой секции
- ... все остальные потоки не окажутся вне секций, заключенных в `lock(obj)`, с тем же `obj`
- ... есть хоть один поток, выполнение которого уже находится в этой секции

7. Какие операции следует считать потокобезопасными? 1 балл

- По умолчанию считать все операции потокобезопасными, а если возникают ошибки — добавлять синхронизацию
- Сверяться с документацией
- По умолчанию считать, что все операции не являются потокобезопасными

Варианты заданий:
по разделу «Инкапсуляция - теория и практика»

Тест

```
class SomeClass {
  public static int s;
  private int p;
  public int d;
}
```

1. Отметьте все корректные обращения к полям объявленного выше класса SomeClass 1 балл

- SomeClass.s = 42
- SomeClass.d = 42
- new SomeClass().s = 42
- new SomeClass().d = 42
- new SomeClass().p = 42

```
namespace MyNamespace
{
   internal class ClassA { }

   public class ClassB
   {
      public ClassA Method1() { return null; }
      private ClassB Method2() { return null; }
   }
```

2. Перечислите все способы, которыми можно избавиться от ошибок компиляции в этом коде1 балл

- Перенести этот код в другую сборку
- Сменить модификатор доступа Method1 с public на private
- Сменить модификатор доступа ClassB с public на internal
- Сменить модификатор доступа ClassA с internal на public
- Сменить модификатор доступа Method2 с private на public
- Сменить модификатор доступа Method1 с public на internal

Варианты заданий:
по разделу «Наследование и полиморфизм - теория и практика»

Тест

Изучите следующий код:

```
class ClassA { }

class ClassB : ClassA { }

class ClassC : ClassA { }

class Program
{
   public static void Main()
   {
      ClassA a = new ClassA();
      ClassB b = new ClassB();
      ClassC c = new ClassC();
      ClassA d = (ClassA) b;
   }
}
```

1. Конверсия переменной b к ClassA — это... 1 балл

- Upcast
- Downcast
- Ни то, ни другое

2. Конверсия переменной c к ClassA... 1 балл

- Пройдет без ошибок
- Вызовет ошибку компиляции
- Вызовет ошибку выполнения

3. Конверсия переменной a к ClassB - это 1 балл

- Upcast
- Downcast
- Ни то, ни другое

4. Конверсия переменной d к ClassC... 1 балл

- Пройдет без ошибок
- Вызовет ошибку компиляции
- Вызовет ошибку выполнения

5. К каким типам можно без ошибок привести переменную d? 1 балл

- [ ] ClassA
- [ ] ClassB
- [ ] ClassC

6. Конверсия переменной c к ClassB... 1 балл

- Upcast
- Downcast

- Ни то, ни другое

Тест

```
interface A { }
interface B : A { }

class X : A {}
class Y : X, B {}
class Z : X {}
```

1. Какие из этих следующих операторов не выбросят исключение? *1 балл*

- (A) X
- X as B
- Y as B
- (B) Z
- (A)

2. Какие из этих утверждений верны для абстрактных базовых классов, но не верны для интерфейсов *1 балл*

- Могут содержать методы с реализованной логикой
- Могут содержать свойства
- Могут содержать поля
- Могут содержать приватные члены
- Могут содержать реализацию методов базового класса или интерфейса

Варианты заданий:
по разделу «DDD»

Тест

1. Как вы думаете, зачем может быть полезно разделение кода на слои? *1 балл*

- Проще навигация по коду — новичку проще понять, где искать тот или иной класс
- Без разделения на слои, код нельзя будет протестировать
- Код, для которого критична производительность, можно вынести на более низкий уровень, и он будет работать быстрее
- Появляется возможность повторного использования слоя предметной области в разных приложениях

2. В каком слое должен оказаться класс, описывающий контрол Windows Forms, для отображения информации о пациенте? *1 балл*

- UI
- Application
- Domain
- infrastructure

3. В каком слое должен оказаться вспомогательный метод расширения класса FlightLevel, определяющего каким курсом движется воздушное судно, преимущественно северным или южным? 1 балл

- UI
- Application
- Domain
- infrastructure

4. Отметьте верные утверждения про разделение кода на слои согласно DDD 1 балл

- Классы, описывающие предметную область должны быть в слое приложения
- Слой предметной области может зависеть только от слоя инфраструктуры
- Каждый слой может зависеть только от одного следующего слоя
- Каждый слой может зависеть от любого другого слоя.

5. Как разделение на слои может выглядеть в коде на C# 1 балл

- Выделять принадлежность каждого члена к тому или иному слою комментарием
- Помещать члены класса, относящиеся к слою X в отдельный регион с именем X
- Создать по отдельной сборке на каждый слой
- Создать по отдельному пространству имен на каждый слой

Варианты заданий:

по разделу «Fluent API»

Тест

```
void M1()
{
   var settings = new Settings();
   settings.Delimiter = '\t';
   var reader = new XmlReader(settings);
   ReadDocument(reader);
}
```

1. Метод M1 является примером использования Fluent API 1 балл

- Верно
- Неверно

```
int[] M2(){
   return ImmutableList<int>.Empty
      .Add(42).Concat(32, 45, 28)
      .Where(n => n % 2 == 0).ToArray();
}
```

2. Метод M2 является примером использования Fluent API 1 балл

- Верно
- Неверно

3. Отметьте все отличительные особенности, характерные для Fluent API 1 балл

- Fluent API удобен при изучении за счет подсказок среды разработки
- Код использующий Fluent API читается почти как текст на естественном языке

- Fluent API активно используют методы с out и ref параметрами
- Fluent API активно использует технику method chaining
- Linq является примером Fluent API

Варианты заданий:
по разделу «Управление ресурсами»

Тест
1. Когда будет вызван финализатор для объекта? 1 балл

- Сразу же, как только управление покинет метод, где этот объект был создан
- Сразу же, как только на объект перестанут указывать ссылки
- Когда-нибудь после того, как на объект перестанут указывать ссылки
- После того, как программа выйдет из метода Main

2. Что из этого относится к ресурсам в неуправляемой памяти? 1 балл

- Созданный внутри C# массив чисел
- Экземпляр StreamReader
- Область системной памяти, выделенная в результате вызова API-функции
- Экземпляр класса, импортированного из библиотеки на C/C++

Варианты заданий:
по разделу «Исключения»

Тест

1. Отметьте все идейно корректные способы, которыми можно перевыбросить исключение внутри блока catch(Exception e) 1 балл

- throw;
- throw e;
- throw new MyException("My message");
- throw new MyException("My message", e);

Критерии оценки:
Приведены в разделе 2

Наименование: защита лабораторных работ
***Представление в ФОС*:** задания и требования к выполнению представлены в методических указаниях по дисциплине
Варианты заданий: задания и требования к выполнению представлены в методических указаниях по дисциплине

Критерии оценки:
Приведены в разделе 2.

Наименование: курсовая работа

***Представление в ФОС*:** задания и требования к выполнению представлены в методических указаниях по дисциплине

Варианты заданий:

Задание на курсовую работу по «Объективно-ориентированному программированию»

на тему:	***"Сортировка массива заданных объектов заданным методом по заданному принципу"***
Объект:	дуга окружности (радиус, угол).
Метод:	Сортировка выбором
Принцип:	по возрастанию длины дуги.
Операция сравнения:	CompareTO.

	Ход выполнения работы
1.	Изучение метода сортировки
2.	Блок-схема алгоритма метода сотрировки
3.	Разработка программы сортировки массива целых чисел заданным методом на языке программирования "Паскаль".
	Размерность массива задается числом в разделе CONST. Ввод исходного массива с помощью генератора случайных чисел реализуется в головной программе.
	Вывод отсортированного массива - функция;
	(передача данных - список параметров).
	Сортировка - функция;
	(передача данных - общая область).
4.	Разработка программы сотрировки массива заданных объектов на языке программирования C#.
	Для решения задачи создать необходимые классы, предусмотреть конструкторы классов и свойства полей.
5.	Разработка необходимого набора тестов, подтверждающих корректность работы на различных массивах исходных данных
6.	Оценка сложности предлагаемого решения.

	Структура отчета	
1.	Отчет в бумажном варианте	
	а)	титульный лист (с подписью исполнителя);
	б)	задание на курсовую работу (первая страница);
	в)	описание метода сортировки (математика);
	г)	блок-схема алгоритма сортировки;
	д)	программа на языке программирования "Паскаль" с комментариями по использованию типов и назначению переменных;
	е)	примеры работы программы сортировки массива целых чисел;
	ж)	программа сортировки объектов заданного класса на языке программирования C# с комментариями по использованию типов и назначению переменных;
	з)	примеры работы программы сортировки массива заданных объектов;

	и)	обоснование необходимости предлагаемых тестов;
	к)	программа с тестами;
	л)	результаты "тестирования";
	м)	оценка сложности алгоритма.
2.	Отчет по курсовой работе (с указанным выше содержанием) в формате PDF, высланный на электронную почту ***asoiu-program-2017-2018@mail.ru***	
3.	Программные продукты, записанные на диск иди флэш-карту.	

Критерии оценки:
Приведены в разделе 2

Наименование: работа на практических занятиях: текущий контроль решения задач**.**
***Представление в ФОС*:** сборник задач:

LINQ

1) LINQ удобно использовать для чтения из файла и разбора простых текстовых форматов. Особенно удобно сочетать методы LINQ с методами класса File: File.ReadLines(filename), File.WriteLines(filename, lines).

На выходе метода ReadAllLines получается массив строк, поэтому часто возникает задача обработки именно массива строк.
Пусть у вас есть файл, в котором каждая строка либо пустая, либо содержит одно целое число. Напишите метод, который вернет массив всех чисел, присутствующих в исходном списке строк.
Ваш код будет проверяться с помощью такого метода Main:

```
public static void Main()
   foreach (var num in ParseNumbers(new[] {"-0", "+0000"}))
      Console.WriteLine(num);
   foreach (var num in ParseNumbers(new List<string> {"1", "", "-03", "0"}))
      Console.WriteLine(num);
}
```

Реализуйте метод ParseNumbers в одно LINQ-выражение.

2) Теперь у вас есть список строк, в каждой из которой написаны две координаты точки (X, Y), разделенные пробелом.

Реализуйте метод ParsePoints в одно LINQ-выражение.

```
public static void Main()
{
   // Функция тестирования ParsePoints

   foreach (var point in ParsePoints(new[] { "1 -2", "-3 4", "0 2" }))
      Console.WriteLine(point.X + " " + point.Y);
   foreach (var point in ParsePoints(new List<string> { "+01 -0042" }))
      Console.WriteLine(point.X + " " + point.Y);
}

public class Point
{
   public Point(int x, int y)
   {
      X = x;
      Y = y;
   }
   public int X, Y;
```

```
}
```

Реализуйте метод ParsePoints в одно LINQ-выражение.

3) Вам дан список всех классов в школе. Нужно получить список всех учащихся всех классов.

Учебный класс определен так:

```
public class Classroom
{
   public List<string> Students = new List<string>();
}
```

Без использования LINQ решение могло бы выглядеть так:

```
var allStudents = new List<string>();
foreach (var classroom in classes)
{
   foreach (var student in classroom.Students)
   {
      allStudents.Add(student);
   }
}
return allStudents.ToArray();
```

Напишите решение этой задачи с помощью LINQ в одно выражение.

```
public static void Main()
{
   Classroom[] classes =
   {
      new Classroom {Students = {"Pavel", "Ivan", "Petr"},},
      new Classroom {Students = {"Anna", "Ilya", "Vladimir"},},
      new Classroom {Students = {"Bulat", "Alex", "Galina"},}
   };
   var allStudents = GetAllStudents(classes);
   Array.Sort(allStudents);
   Console.WriteLine(string.Join(" ", allStudents));
}
```

Generics

В анализе данных бывают очень полезны таблицы. Например, строки могут соответствовать датам, столбцы — департаментам, а в ячейках может хранится выручка департамента за контракты на соответствующую дату.

Сделайте такую таблицу, которая бы:

1. Индексировалась величинами типов, указанных при создании таблицы.
2. Имела бы две возможности для индексирования:
    1. С автоматическим созданием нужных строк и столбцов «на лету» при обращении к таблице по соответствующим индексам;
    2. Которая бы требовала создания столбцов и строк заранее и выбрасывала исключение при доступе к несуществующим столбцам или строкам.

Скачайте проект Generics.Tables и изучите тесты, которые должна проходить ваша таблица, чтобы понять детали задания.

Дополнительно подумайте над тем, как не хранить лишней информации в таблице: если в данную дату в данном департаменте не было заключено контрактов, то значение будет 0, и хранить сотни нулей, очевидно, не нужно.

Рефлексия

Для нагрузочного тестирования вашей программы вам нужно уметь создавать большое количество экземпляров классов, при этом они должны быть существенно различны. Вы решили использовать для этой цели генератор случайных чисел, и решили использовать атрибуты для того, чтобы указать, из какого распределения брать значения для тех или иных свойств в объектах.
Понятно, что решение с прикреплением распределения к свойствам "намертво" недостаточно гибкое, поэтому вы заранее озаботились тем, чтобы можно было это распределение менять настройками генератора объектов.
Для простоты, будем рассматривать только значения типа double и два распределения: нормальное и экспоненциальное.
Вы можете сделать оптимизированное решение (с Expression.Bind), если хотите, но и менее оптимальное решение тоже подойдет.
Проект Reflection.Randomness

Файлы

Необходимость писать собственные стримы бывает не так уж и часто. Однако, такие ситуации бывают.
Например, допустим, что вы разрабатываете компьютерную игру с множеством мелких файлов. Очевидно, что хотелось бы эти файлы убрать в один. Допустим, что вы по какой-то причине не хотите использовать zip-сжатие (что было бы самым адекватным подходом к этой ситуации), и вместо этого хотите изобрести свой формат.
Ваша задача — по известному формату написать стрим, который читает секцию файла. Ваш стрим будет получать другой, базовый стрим, который содержит данные, и ваша задача — найти нужную секцию и прочитать. Эта задача осмысленна, поскольку, например, Bitmap.FromStream принимает именно Stream, и вы можете подставить туда ваш стрим для того, чтобы все работало.
Дополнительное ограничение: из базового стрима нужно читать порциями ровно по 1024 байт. Число произвольное, но это ограничение нужно: плохо слишком часто обращаться к стриму (сам факт чтения несет дополнительные расходы, в некоторых случаях не зависящие от количества прочитанных байт), и плохо читать все сразу, поскольку стрим может быть очень большой и не поместиться в памяти. Найдите стандартный способ обеспечить это условие.
Скачайте проект Streams.Resources. Детали формата можно посмотреть в конструкторе TestStream.

Файлы

1. Вводится информация об итогах зимней сессии на 1 курсе. Сведения о каждом студенте (всего их 25) заданы в виде следующего текста: «фамилия», «имя», «отчество», «год рождения», «номер группы», «оценка 1», «оценка 2», «оценка 3», причем первая оценка - за экзамен по высшей математике, вторая - по физике, третья - по программированию), «форма обучения (бюджетная, договорная)» Ввод каждого значения завершается нажатием <ENTER>. **Массив записей не использовать!!!** В группе определить средний балл после зимней сессии, абсолютную успеваемость. Распечатать ФИО студентов по возрастанию среднего балла.

2. Вводится информация об итогах зимней сессии на 1 курсе. Сведения о каждом студенте (всего их 25) заданы в виде следующего текста: «фамилия», «имя», «отчество», «год рождения», «номер группы», «оценка 1», «оценка 2», «оценка 3», причем первая оценка - за экзамен по высшей математике, вторая - по физике, третья - по программированию), «форма обучения (бюджетная, договорная)» Ввод каждого значения завершается нажатием <ENTER>. **Массив записей не использовать!!!** В группе студентов определить средний балл каждого. Распечатать список по убыванию среднего балла. Вывести ФИО студентов, у которых больше одной тройки.

При проведении диагностики освоения компетенций и оценки минимального уровня знаний могут быть использованы тестовые материалы:

1. Наследование – это способность языка …

1. Скрывать излишние детали реализации от пользователя объекта.
2. Трактовать связанные объекты в сходной манере.
3. Позволять строить новые определения классов на основе определений существующих классов.
4. Определять стандартную реализацию, которая может быть изменена.

2. Полиморфизм – это способность языка …

1. Скрывать излишние детали реализации от пользователя объекта.
2. Позволять переопределять стандартную реализацию.
3. Позволять строить новые определения классов на основе определений существующих классов.
4. Которая не предусматривает стандартную реализацию.

3. Способность прятать детали реализации от пользователей этих объектов – это:

1. Полиморфизм.
2. Абстракция.
3. Наследование.
4. Перегрузка.
5. Инкапсуляция.
6. Специализация.
7. Хеширование.

4. Виртуальными могут быть:

1. Абстрактные методы.
2. Статические методы.
3. Приватные методы.
4. Публичные методы.

5. В языке C# структура – это:

1. Ссылочный тип.
2. Тип-значение.
3. Концептуально и физически полный эквивалент класса.

6. В языке C# выполнится ли блок finally, если исключения не было?

1. Зависит от режима работы.
2. Да.
3. Зависит от кода.
4. Нет.

7. В языке C# может ли класс наследоваться сразу от нескольких классов?

1. Да.
2. Да, но так делать не рекомендуется.
3. Нет.

8. В языке C# может ли класс реализовывать сразу несколько интерфейсов?

1. Да
2. Нет.
3. Классы вообще не могут реализовывать интерфейсы.

9. В языке C# какой способ выброса исключения является корректным:

1. throw Exception();
2. throw new Exception();
3. call new Exception();
4. Exception();
5. new Exception();
6. call Exception();

10. К типам-значениям языка C# относятся:

1. Строки.
2. Все арифметические типы, кроме типа double.
3. Все арифметические типы.
4. Массивы.

Ключи к тесту

Вопрос	1.	2.	3.	4.	5	6	7	8	9	10
Ответ	3	2	5	1, 3, 4	2	2	3	1	2	3

2.Критерии и шкалы оценивания

Для контрольных мероприятий (текущего контроля) устанавливается минимальное и максимальное количество баллов в соответствии с таблицей. Контрольное мероприятие считается пройденным успешно при условии набора количества баллов не ниже минимального.

Результат обучения по дисциплине считается достигнутым при успешном прохождении обучающимся всех контрольных мероприятий, относящихся к данному результату обучения.

3 семестр:

Разделы дисциплины	***Форма контроля***	***Количество баллов***	
		min	***max***
6	Контрольная работа № 1	10	20
2	Лабораторная работа № 1	10	20
6	Лабораторная работа № 2	10	20
8	Лабораторная работа № 3	10	20
11	Лабораторная работа № 4	10	20
	Итого:	**50**	**100**

4 семестр:

Разделы дисциплины	***Форма контроля***	***Количество баллов***	
		min	***max***
17	Контрольная работа № 1	5	10
19	Контрольная работа № 2	5	10
23	Контрольная работа № 3	5	10
25	Контрольная работа № 4	5	10
27	Контрольная работа № 5	5	10
15	Лабораторная работа № 1	5	10
17	Лабораторная работа № 2	10	20
21	Лабораторная работа № 3	10	20
	Итого:	**50**	**100**

При оценивании результатов обучения по дисциплине в ходе текущего контроля успеваемости используются следующие критерии. Минимальное количество баллов выставляется обучающемуся при выполнении всех показателей, допускаются несущественные неточности в изложении и оформлении материала.

Наименование, назначение	***Показатели выставления минимального количества баллов***
Лабораторная работа	Лабораторная работа выполнена в полном объеме; Представлен отчет, содержащий необходимые этапы, выводы, оформленный в соответствии с установленными требованиями; Продемонстрирован удовлетворительный уровень владения материалом при защите лабораторной работы, даны правильные ответы не менее чем на 50% заданных вопросов.
Контрольная работа	Продемонстрирован удовлетворительный уровень владения материалом. Правильно решено не менее 50% заданий
Тест	Правильно решено не менее 50% тестовых заданий

Выполнение и защита курсовой работы оценивается согласно шкале, приведенной ниже. На защите курсовой работы обучающемуся задаются 4 вопроса по теме курсовой работы; оцениваются формальные и содержательные критерии.

Результаты защиты курсовой работы оцениваются максимально 100 баллами.

Критерии оценивания курсовой работы

№	***Показатель***	***Максимальное количество баллов***
I.	**Выполнение курсовой работы**	**5**
1.	Соблюдение графика выполнения	2
2.	Самостоятельность и инициативность при выполнении	3
II.	**Оформление курсовой работы**	**10**
5.	Грамотность изложения текста, безошибочность	3
6.	Владение информационными технологиями при оформлении	4
4.	Качество графического материала	3
III.	**Содержание курсовой работы**	**15**
8.	Полнота раскрытия темы	10
9.	Качество введения и заключения	3
10.	Степень самостоятельности в изложении текста (оригинальность)	2
IV.	**Защита курсовой работы**	**70**
11	Понимание цели	5
12	Владение терминологией по тематике	5
13	Понимание логической взаимосвязи разделов	5
14	Владение применяемыми методиками расчета	5
15	Степень освоения рекомендуемой литературы	5
16	Умение делать выводы по результатам выполнения	5
17	Степень владения материалами, изложенными в работе (проекте), качество ответов на вопросы по теме	40
	Всего	**100**

Промежуточная аттестация в 3 семестре по дисциплине проводится в форме зачета.

Итоговая оценка по дисциплине может быть выставлена на основе результатов текущего контроля с использованием следующей шкалы:

Оценка	***Набрано баллов***
«неудовлетворительно»	менее 39
«удовлетворительно»	39-50
«хорошо»	51-84
«отлично»	85-100

Если сумма набранных баллов менее 39 – обучающийся не допускается до промежуточной аттестации.

Если сумма баллов составляет от 39 до 84 баллов, обучающийся допускается до зачета.

По сумме набранных баллов студенту может быть выставлена оценка за промежуточную аттестацию, согласно приведенной шкале. Обучающийся имеет право сдать экзамен в письменной форме для изменения балла

Билет к зачету включает 2 теоретических вопроса.

Время на подготовку: 30 минут.

Промежуточная аттестация в 4 семестре по дисциплине проводится в форме экзамена.

Оценка	***Набрано баллов***
«отлично»	73-90
«хорошо»	64-72
«удовлетворительно»	55-63
«неудовлетворительно»	45-55

Если сумма набранных баллов менее 54 – обучающийся не допускается до промежуточной аттестации.

Если сумма баллов более 55, обучающийся допускается до экзамена, при условии что выполнены и защищены лабораторные работы.

По сумме набранных баллов студенту может быть выставлена оценка за промежуточную аттестацию, согласно приведенной шкале. Обучающийся имеет право сдать экзамен в письменной форме для изменения балла.

Промежуточная аттестация проводится в письменной форме.

Билет к экзамену включает 2 теоретических вопроса.

Время на подготовку: 40 минут.

При оценивании результатов обучения по дисциплине в ходе промежуточной аттестации используются следующие критерии и шкала оценки:

Оценка	***Критерии оценки***
«отлично»	Обучающийся показал всестороннее, систематическое и глубокое знание учебного материала, предусмотренного программой, умение уверенно применять на их практике при решении задач (выполнении заданий), способность полно, правильно и аргументировано отвечать на вопросы и делать необходимые выводы. Свободно использует основную литературу и знаком с дополнительной литературой, рекомендованной программой
«хорошо»	Обучающийся показал полное знание теоретического материала, владение основной литературой, рекомендованной в программе, умение самостоятельно решать задачи (выполнять задания), способность аргументировано отвечать на вопросы и делать необходимые выводы, допускает единичные ошибки, исправляемые после замечания преподавателя. Способен к самостоятельному пополнению и обновлению знаний в ходе дальнейшей учебной работы и профессиональной деятельности
«удовлетворительно»	Обучающийся демонстрирует неполное или фрагментарное знание основного учебного материала, допускает существенные ошибки в его изложении, испытывает затруднения и допускает ошибки при выполнении заданий (решении задач), выполняет задание при подсказке преподавателя, затрудняется в формулировке выводов. Владеет знанием основных разделов, необходимых для дальнейшего обучения, знаком с основной и дополнительной литературой, рекомендованной программой
«неудовлетворительно»	Обучающийся при ответе демонстрирует существенные пробелы в знаниях основного учебного материала, допускает грубые ошибки в формулировании основных понятий и при решении типовых задач (при выполнении типовых заданий), не способен ответить на наводящие вопросы преподавателя. Оценка ставится обучающимся, которые не могут продолжить обучение или приступить к профессиональной

	деятельности по окончании образовательного учреждения без дополнительных занятий по рассматриваемой дисциплине