МИНОБРНАУКИ РОССИИ
Воткинский филиал
Федерального государственного бюджетного образовательного
учреждения высшего образования
«Ижевский государственный технический университет имени М.Т. Калашникова»
(ВФ ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

УТВЕРЖДАЮ
Директор
_____/Давыдов И.А.
__03 июня___ 2020_ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Объектно-ориентированное программирование

направление 09.03.01 «Информатика и вычислительная техника»

профиль «Автоматизированные системы обработки информации и управления»

уровень образования: бакалавриат

форма обучения: очная

общая трудоемкость дисциплины составляет: ___10___ зачетных единиц(ы)

Кафедра Естественные науки и информационные технологии

Составитель _____

Рабочая программа составлена в соответствии с требованиями федерального государственного образовательного стандарта высшего образования и рассмотрена на заседании кафедры

Протокол от ___03__июня__ 2020 г. №___4_

Заведующий кафедрой _____К.Б. Сентяков

__03__июня____ 2020_ г.

СОГЛАСОВАНО

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

Председатель учебно-методической комиссии по направлению 09.03.01 «Информатика и вычислительная техника», профиль «Автоматизированные системы обработки информации и управления»

_____К.Б. Сентяков

__03__июня____ 2020_ г.

Руководитель образовательной программы _____ К.Б. Сентяков

__03__июня____ 2020__ 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	Блока 1 Дисциплины (модули) Часть, формируемая участниками образовательных отношений
Трудоемкость (з.е. / часы)	10 з.е./ 360 часов
Цель изучения дисциплины	**Целью** преподавания 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
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 - Исключения
Форма промежуточной аттестации	КР зачет с оценкой, экзамен

Целью преподавания 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.

В результате изучения дисциплины студент должен

знать:

- сложные языковые конструкции C#: обобщённые типы (generics), генераторы последовательностей, LINQ, а также необходимый набор алгоритмов и структур данных;
- методы проектирования программной системы: консольные приложения и шаблоны проектирования ПО в C#;
- тестирование, виды тестов, техники тестирования.

уметь:

- проектировать структуру и алгоритмы программной системы;
- разрабатывать программные системы на языке программирования C#;
- анализировать тестовые случаи, писать тесты, анализировать отчеты тестирования;
- выбирать средства реализации требований к программному обеспечению, использовать существующие типовые решения и шаблоны проектирования программного обеспечения, вырабатывать варианты реализации программного обеспечения;

владеть:

- навыками работы в интегрированной среде разработки;
- навыками алгоритмизации и программирования на процедурных и объектно-ориентированных языках программирования;
- навыками разработки архитектуры программного обеспечения;
- к документирования тестов, разработки скриптов для автоматизации тестирования.

2 Место дисциплины в структуре ООП

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

Для изучения дисциплины студент должен:

знать:

- основные понятия дисциплин: Информатика, Программирование;

уметь:

- использовать полученные ранее знания в практических задачах;

владеть:

- навыками работы с персональным компьютером на высоком пользовательском уровне;
- основами работы с научно-технической литературой.

Изучение дисциплины базируется на знаниях, полученных при изучении дисциплин: Информатика, Программирование.

3 Требования к результатам освоения дисциплины

3.1 Знания, приобретаемые в ходе изучения дисциплины

№ п/п	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

3.2 Умения, приобретаемые в ходе изучения дисциплины

№ п/п	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

3.3 Навыки, приобретаемые в ходе изучения дисциплины

№ п/п Н	**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

3.4 Компетенции, приобретаемые в ходе изучения дисциплины

Компетенции		**Знания (№№ из 3.1)**	**Умения (№№ из 3.2)**	**Навыки (№№ из 3.3)**
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, структуру и алгоритмы функционирования вычислительных и информационных систем, разрабатывать инфраструктуру	1	1-2	1-2

	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	2	4	3

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

4 Структура и содержание дисциплины

4.1 Разделы дисциплин и виды занятий

№ п/п	**Раздел дисциплины**	**Семестр**	**Неделя семестра**	**Виды контактной работы, включая самостоятельную работу студентов и трудоемкость (в часах)**				**Формы текущего контроля успеваемости *(по неделям семестра)***
				лек	***прак***	***лаб***	***СРС***	**Форма промежуточной аттестации *(по семестрам)***
1	Тестирование	3	1		2		5	тест
2	Очереди, стеки, дженерики	3	2		2	4	5	тест, выполнение лабораторной работы
3	Листы и словари	3	3		2		5	тест
4	Делегаты	3	4		2	4	5	защита лабораторной работы
		3	5		2		3	тест
5	Элементы функционального программирования	3	6		2	4	5	тест, выполнение лабораторной работы
6	LINQ	3	7		2		5	работа на практических занятиях: текущий контроль решения задач
7	Обработка графовых структур	3	8		2	4	5	тест, защита лабораторной работы
8	Жадные алгоритмы	3	9		2		5	тест

9	Динамическое программирование	3	10		2	4	5	выполнение лабораторной работы
		3	11		2		5	тест
10	Структуры данных	3	12		2	4	5	тест, защита лабораторной работы
11	События	3	13		2		5	тест
12	Оконные приложения	3	14		2	4	5	выполнение лабораторной работы
13	Многопоточное программирование	3	15		2		5	тест
14	Рефлексия типов	3	16		2	4	5	защита лабораторной работы
							2	Дифференцированный зачет
	Итого за 3 семестр				**32**	**32**	**80**	
	В том числе контроль самостоятельной работы				2			
15	Инкапсуляция - теория и практика	4	1		2		5	тест
		4	2		2	4	5	выполнение лабораторной работы
16	Наследование и полиморфизм - теория и практика	4	3		2		5	тест
		4	4		2	4	5	защита лабораторной работы
17	Generics	4	5		2		5	работа на практических занятиях: текущий контроль решения задач
18	Делегирование	4	6		2	4	5	выполнение лабораторной работы
19	Рефлексия	4	7		2		5	работа на практических занятиях: текущий контроль решения задач
20	DDD	4	8		2	4	5	тест, защита лабораторной работы
21	Fluent API	4	9		2		5	тест
22	Модульность	4	10		2	4	5	выполнение лабораторной работы
23	Управление зависимостями	4	11		2		5	работа на практических занятиях: текущий контроль решения задач
24	DI-контейнеры	4	12		2	4	5	защита лабораторной работы
25	Функциональный стиль	4	13		2		5	работа на практических занятиях: текущий контроль решения задач
26	Управление ресурсами	4	14		2	4	5	тест, выполнение лабораторной работы

27	Работа с файлами	4	15		2		5	работа на практических занятиях: текущий контроль решения задач
28	Исключения	4	16		2	4	5	тест, защита лабораторной работы
	Курсовая работа						36	защита курсовой работы
							36	Экзамен
	Итого за 3 семестр				**32**	**32**	**152**	
	В том числе контроль самостоятельной работы				2			

4.2 Содержание разделов курса

№ п/п	Раздел дисциплины	Знания (номер из 3.1)	Умения (номер из 3.2)	Навыки (номер из 3.3)
1	**2**	**3**	**4**	**5**
1	**1.** Модульные тесты **2.** Покрытие тестами **3.** Функциональное тестирование **4.** Внедрение тестов	3	3	4
2	**1.** Стеки и очереди **2.** Очередь на связных списках **3.** Дженерик-классы	1	1,2	1,2
3	**1.** Листы и индексация **2.** Перегрузка операторов **3.** Хеширующие функции	1	1,2	1,2
4	**1.** Делегаты для динамических методов **2.** Дженерик-делегаты **3.** Анонимные делегаты **4.** Лямбда-выражения	1	1,2	1,2
5	**1.** О функциональном программировании **2.** Делегаты для диагностики кода	1	1,2	1,2
6	**1.** Фильтрация и преобразование **2.** Работа с кортежами **3.** Функции агрегирования **4.** Группировка	1	1,2	1,2
7	**1.** Простейшая реализация графа **2.** Неориентированные графы и целостность данных **3.** Обходы графа	1	1,2	1,2
8	**1.** Комбинаторные задачи **2.** Задача о планировании времени **3.** Жадный алгоритм для задачи разбиения **4.** Жадный алгоритм для задачи коммивояжера	1	1,2	1,2
9	**1.** Динамическое программирование на задаче	1	1,2	1,2

	планирования времени **2.** Динамическое программирование в комбинаторике **3.** Динамическое программирование для задачи разбиения			
10	**1.** Очередь с приоритетами **2.** Бинарная куча	1	1,2	1,2
11	**1.** Программирование GUI **2.** Событийная модель **3.** Событийная модель с делегатами **4.** Мультикаст-делегаты **5.** Целостность событийной модели **6.** События	1	1,2	1,2
12	**1.** Windows Forms и WPF **2.** Расположение контролов на форме **3.** Рисование **4.** Повороты и переносы рисунка **5.** Таймеры и анимация **6.** Антипаттерн интеллектуального интерфейса	2	4	2,3
13	**1.** Треды, домены и процессы **2.** Класс Thread **3.** Общие ресурсы и lock **4.** Блокирование потока GUI **5.** Асинхронные операции в GUI	2	4	2,3
14	**1.** Рефлексия. Класс Type **2.** Создание объекта с помощью рефлексии **3.** Рефлексия для свойств, методов и полей **4.** Биндинг и атрибуты	2	4	2,3
15	1.Пререквизиты 2. public, private, static 3. Модификатор internal 4. Конструкторы 5. Порядок инициализации 6. Перегруженные методы и параметры по умолчанию 7. Свойства 8. Индексаторы 9. Структуры	1	1,2	1,2
16	**1.** Наследование **2.** Интерфейсы **3.** Полиморфизм и абстрактные базовые классы	1	1,2	1,2
17	**1.** Generic-классы **2.** Generic-методы **3.** Ковариация и контравариация	1	1,2	1,2
18	**1.** Простое делегирование **2.** Сложное делегирование **3.** Делегирование без делегатов **4.** Обратная совместимость **5.** Декораторы	1	1,2	1,2
19	**1.** Профилирование рефлексии **2.** Рефакторинг рефлексии	1	1,2	1,2

	3. Оптимизация рефлексии			
20	**1.** Слоистая архитектура **2.** Моделирование предметной области	2	4	2,3
21	**1.** Fluent API **2.** FluentAssertions **3.** Реализация Fluent-Интерфейса	3	3	4
22	**1.** Критерии чистого кода **2.** SRP **3.** Модульность **4.** SRP и командная работа	2	4	2,3
23	**1.** Процедурный подход **2.** Принципы OCP и DIP **3.** DIP и расширяемость **4.** DIP и тестируемость	2	4	2,3
24	**1.** Service Locator **2.** DI Container **3.** Коллекции **4.** Циклические зависимости **5.** Время жизни **6.** Контексты	2	4	2,3
25	**1.** Устранение интерфейсов **2.** ФП и DI-контейнеры **3.** Чистые функции **4.** Рефакторинг сумматора **5.** Зависимости между сборками	2	4	2,3
26	**1.** Потоки **2.** Исключения **3.** Управляемая память **4.** Финализаторы	2	4	2,3
27	**1.** Текстовые и бинарные потоки **2.** Архитектура потоков **3.** MemoryStream и NetworkStream	2	4	2,3
28	**1.** Exception и его поля **2.** Перевыброс исключения	3	3	4

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

4.3.1. 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

4.3.2. 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

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

4.4.1. Третий семестр

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

4.4.2. Четвертый семестр

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

5 Содержание самостоятельной работы студентов. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины

5.1 Содержание самостоятельной работы

5.1.1. Третий семестр

№ п/п	№ раздела дисциплины	Наименование тем	Трудоемкость (час)
1	1	Тестирование	5
2	2	Очереди, стеки, дженерики	5
3	3	Листы и словари	5
4	4	Делегаты	8
5	5	Элементы функционального программирования	5
6	6	LINQ	5
7	7	Обработка графовых структур	5
8	8	Жадные алгоритмы	5
9	9	Динамическое программирование	10
10	10	Структуры данных	5
11	11	События	5
12	12	Оконные приложения	5
13	13	Многопоточное программирование	5
14	14	Рефлексия типов	5
15	1-14	Зачет с оценкой	2
	ВСЕГО		80

5.1.2. Четвертый семестр

№ п/п	№ раздела дисциплины	Наименование тем	Трудоемкость (час)
1	15	Инкапсуляция - теория и практика	10
2	16	Наследование и полиморфизм - теория и практика	10
3	17	Generics	5
4	18	Делегирование	5
5	19	Рефлексия	5
6	20	DDD	5
7	21	Fluent API	5
8	22	Модульность	5
9	23	Управление зависимостями	5
10	24	DI-контейнеры	5
11	25	Функциональный стиль	5
12	26	Управление ресурсами	5
13	27	Работа с файлами	5
14	28	Исключения	5
15	1-28	Курсовая работа	36
16	1-28	Экзамен	36
	ВСЕГО		152

5.2. Оценочные средства, используемые для текущего контроля успеваемости и промежуточной аттестации обучающихся по итогам освоения дисциплины, их виды и формы, требования к ним и шкалы оценивания приведены в приложении к рабочей программе дисциплины «Фонд оценочных средств по дисциплине Объектно-ориентированное программирование», которое оформляется в виде отдельного документа.

6 Учебно-методическое и информационное обеспечение дисциплины

а) Основная литература

Номер	Наименование книги	Год издания
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	2015
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	2019
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	2016

б) Дополнительная литература

Номер	Наименование книги	Год издания
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	2016
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	2019
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	2017
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	2017

в) 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

г) программное обеспечение

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

д) методические указания:

1. Николаев, Е. И. Объектно-ориентированное программирование. Часть 1 [Электронный ресурс] : лабораторный практикум / Е. И. Николаев. — Электрон. текстовые данные. — Ставрополь : Северо-Кавказский федеральный университет, 2015. — 183 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/62966.html
2. Николаев, Е. И. Объектно-ориентированное программирование. Часть 2 [Электронный ресурс] : лабораторный практикум / Е. И. Николаев. — Электрон. текстовые данные. — Ставрополь : Северо-Кавказский федеральный университет, 2015. — 156 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/63218.html
3. Новиков, П. В. Объектно-ориентированное программирование [Электронный ресурс] : учебно-методическое пособие к лабораторным работам / П. В. Новиков. — Электрон. текстовые данные. — Саратов : Вузовское образование, 2017. — 124 c. — 978-5-4487-0011-8. — Режим доступа: http://www.iprbookshop.ru/64650.html

7. Материально-техническое обеспечение дисциплины

1. Специальные помещения - учебные аудитории для проведения: занятий семинарского типа, групповых и индивидуальных консультаций, оборудованные доской, столами, стульями.
2. Специальные помещения - учебные аудитории для проведения лабораторных занятий, оборудованные доской, столами лабораторными, стульями, лабораторным оборудованием различной степени сложности.
3. Специальные помещения - учебные аудитории для проведения курсового проектирования/выполнения курсовой работы и выпускной квалификационной работы, оборудованные доской, компьютерами с возможностью подключения к сети «Интернет», столами, стульями.
4. Специальные помещения - учебные аудитории для проведения текущего контроля успеваемости и промежуточной аттестации обучающихся, оборудованные доской, столами, стульями.
5. Специальные помещения - учебные аудитории для организации и проведения самостоятельной работы студентов, оборудованные доской, компьютерами с возможностью подключения к сети «Интернет», столами, стульями.

Лист согласования рабочей программы дисциплины «Объектно-ориентированное программирование» на учебный год

Рабочая программа дисциплины «Объектно-ориентированное программирование» по направлению подготовки 09.03.01 «Информатика и вычислительная техника»
по профилю «Автоматизированные системы обработки информации и управления»

согласована на ведение учебного процесса в учебном году:

Учебный год	***«Согласовано»:*** *заведующий кафедрой,* *ответственной за РПД* *(подпись и дата)*
2020 – 2021	7.09.20
2021 – 2022	7.09.21
2022 – 2023	
2023 – 2024	

Приложение к рабочей программе дисциплины

МИНОБРНАУКИ РОССИИ
Воткинский филиал
Федерального государственного бюджетного образовательного
учреждения высшего образования
«Ижевский государственный технический университет имени М.Т. Калашникова»
(ВФ ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Оценочные средства
по дисциплине

Объектно-ориентированное программирование

направление 09.03.01 «Информатика и вычислительная техника»

профиль «Автоматизированные системы обработки информации и управления»

уровень образования: бакалавриат

форма обучения: очная

общая трудоемкость дисциплины составляет: 10 зачетных единиц(ы)

Паспорт

фонда оценочных средств

по дисциплине «Объектно-ориентированное программирование»

(наименование дисциплины)

№ п/п	Раздел Дисциплины	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
1	Тестирование	ПК-1, ПК-4, ПК-5	тест
2	Очереди, стеки, дженерики		тест
3	Листы и словари		тест
4	Делегаты		защита лабораторных работ (отчет по лабораторной работе)
			тест
5	Элементы функционального программирования		тест
6	LINQ		работа на практических занятиях: текущий контроль решения задач
7	Обработка графовых структур		тест, защита лабораторных работ (отчет по лабораторной работе)
8	Жадные алгоритмы		тест
9	Динамическое программирование		
			тест
10	Структуры данных		тест, защита лабораторных работ (отчет по лабораторной работе)
11	События		тест
12	Оконные приложения		
13	Многопоточное программирование		тест
14	Рефлексия типов		защита лабораторных работ (отчет по лабораторной работе)
15	Инкапсуляция - теория и практика		тест
			выполнение лабораторной работы
16	Наследование и полиморфизм - теория и практика		тест
			защита лабораторных работ (отчет по лабораторной работе)
17	Generics		работа на практических занятиях: текущий контроль решения задач
18	Делегирование		выполнение лабораторной работы
19	Рефлексия		работа на практических занятиях: текущий контроль решения задач

20	DDD		тест, защита лабораторных работ (отчет по лабораторной работе)
21	Fluent API		тест
22	Модульность		выполнение лабораторной работы
23	Управление зависимостями		работа на практических занятиях: текущий контроль решения задач
24	DI-контейнеры		защита лабораторных работ (отчет по лабораторной работе)
25	Функциональный стиль		работа на практических занятиях: текущий контроль решения задач
26	Управление ресурсами		тест, выполнение лабораторной работы
27	Работа с файлами		работа на практических занятиях: текущий контроль решения задач
28	Исключения		тест, защита лабораторных работ (отчет по лабораторной работе)
	Все разделы курса		защита курсовой работы, экзамен

Описания элементов ФОС

1. 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) МногопоточноепрограммированиеРефлексия типов
14) Инкапсуляция - теория и практика
15) Наследование и полиморфизм - теория и практика
16) Generics
17) Делегирование
18) Рефлексия
19) DDD
20) Fluent API
21) Модульность
22) Управление зависимостями
23) DI-контейнеры
24) Функциональный стиль
25) Управление ресурсами

26) Работа с файлами
27) Исключения

Критерии оценки:
Приведены в разделе 2

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 ответа *1 балл*

- Верно
- Неверно

3. Для всех комбинаторных задач существуют алгоритмы эффективнее перебора вариантов ответа *1 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

3 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.

4 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.

***5 Наименование*:** защита курсовых работ

Представление в ФОС: задания и требования к выполнению представлены в методических указаниях к курсовой работе по дисциплине «Объектно-ориентированное программирование».

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

6. Наименование: зачет
***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

2 Критерии оценки:

Уровень освоения компетенции							
№	**Компе-тенции**	**Дескрипторы**	**Вид, форма оценочного мероприятия**	**Компетенция освоена**			
				отлично	**хорошо**	**удовлетворительно**	**неудовлетворительно**
1		У1 Составлять алгоритмы решения задачи, оформлять блок-схемы алгоритмов У2 Выбирать необходимые типы данных для решения задачи У3 Реализовывать программы, реализующие линейные, разветвляющиеся и циклические вычислительные процессы У4 Обрабатывать структуры данных, файлы. У5 Создавать программы на процедурном языке У6 Создавать программы с использованием идеологии ООП Н3 Алгоритмизации и программирования на процедурных и объектно-ориентированных языках	Работа на практических занятиях: текущий контроль решения задач	Правильно выполнены все задания, даны ответы на все поставленные вопросы. Продемонстрирован высокий уровень владения материалом, как в теории, так и на практике.	Правильно выполнена бо льшая часть заданий. Присутствуют незначительные ошибки. Продемонстрирован хороший уровень владения материалом.	Задания выполнены более чем наполовину. Присутствуют серьёзные ошибки. Продемонстрирован удовлетворительный уровень владения материалом. Проявлены низкие способности применять знания и умения к выполнению конкретных заданий.	Задания выполнены менее чем наполовину. Продемонстрирован неудовлетворительный уровень владения материалом. Проявлены недостаточные способности применять знания и умения на практике
		Дескрипторы	**Вид, форма оценочного мероприятия**	**Компетенция освоена**			
				отлично	**хорошо**	**удовлетворительно**	**неудовлетворительно**
2	ПК-1, ПК-4, ПК-5	З1 Основные этапы решения задачи. Понятия алгоритма, блок-схемы алгоритма, основные виды вычислительных процессов З2 Алфавит и лексику языков программирования, типы данных, правила записи выражений и операций З3 Операторы ввода-вывода, операторы ветвления, операторы цикла З4 Простые типы данных: множество значений, способ хранения, операции, особенности обработки З5 Структурированные ссылочные типы данных: множество значений, способ хранения, операции, особенности обработки З6 Основы процедурного программирования (процедуры, функции, библиотеки) З7 Основы объектно-ориентированного программирования (классы, наследование, инкапсуляция, полиморфизм) З8 Интегрированные среды программирования З9 Принципы рефакторинга, отладки, тестирования программы	Экзамен	Обучающийся продемонстрировал высокий уровень знаний основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по специальности, справился с выполнением заданий, предусмотренных программой дисциплины.	Обучающийся продемонстрировал хороший уровень знаний основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по специальности, справился с выполнением заданий частично (2/3 от возможного максимального результата)	Обучающийся продемонстрировал низкий уровень знаний основного учебно-программного материала, справился с выполнением заданий частично, (1/3 от возможного максимального результата).	Обучающийся обнаружил значительные пробелы в знаниях основного учебно-программного материала, допустил принципиальные ошибки в выполнении заданий и не способен продолжить обучение без дополнительных занятий по соответствующей дисциплине
		Дескрипторы	**Вид, форма оценочного мероприятия**	**Компетенция освоена**			
				отлично	**хорошо**	**удовлетворительно**	**неудовлетворительно**

3		З4 Простые типы данных: множество значений, способ хранения, операции, особенности обработки З5 Структурированные ссылочные типы данных: множество значений, способ хранения, операции, особенности обработки З6 Основы процедурного программирования (процедуры, функции, библиотеки) З7 Основы объектно-ориентированного программирования (классы, наследование, инкапсуляция, полиморфизм)	Тесты	Правильно выполнены все задания. Продемонстрирован высокий уровень владения материалом. Проявлены высокие способности применять знания и умения к выполнению конкретных заданий.	Правильно выполнена большая часть заданий. Присутствуют незначительные ошибки. Продемонстрирован хороший уровень владения материалом. Проявлены средние способности применять знания и умения к выполнению конкретных заданий	Задания выполнены более чем наполовину. Присутствуют серьёзные ошибки. Продемонстрирован удовлетворительный уровень владения материалом. Проявлены низкие Способности применять знания и умения к выполнению конкретных заданий.	Задания выполнены менее чем наполовину. Продемонстрирован неудовлетворительный уровень владения материалом. Проявлены недостаточные способности применять знания и умения к выполнению
		Дескрипторы	**Вид, форма оценочного мероприятия**	**Компетенция освоена**			
				отлично	**хорошо**	**удовлетворительно**	**неудовлетворительно**
4		У5 Создавать программы на процедурном языке У6 Создавать программы с использованием идеологии ООП У7 Работать в среде программирования, реализовывать рефакторинг, отладку, тестирование программы Н1 Работы в интегрированной среде разработки Н2 Настройки среды разработки Н3 Алгоритмизации и программирования на процедурных и объектно-ориентированных языках Н4 Владения техникой рефакторинга, раздельной компиляции, отладчиком среды программирования, тестирования.	Защита курсовых работ	Задание на курсовую работу выполнено в полном объёме на высоком уровне.	Задание на курсовую работу выполнено в полном объёме на хорошем уровне.	Задание на курсовую работу выполнено в полном объёме на удовлетворительном уровне.	Задание на курсовую работу выполнено частично с грубыми ошибками.
		Дескрипторы	**Вид, форма оценочного мероприятия**	**Компетенция освоена**			
				зачёт			**незачёт**
5		У1 Составлять алгоритмы решения задачи, оформлять блок-схемы алгоритмов У2 Выбирать необходимые типы данных для решения задачи У3 Реализовывать программы, реализующие линейные, разветвляющиеся и циклические вычислительные процессы У4 Обрабатывать структуры данных, файлы. У5 Создавать программы на процедурном языке У6 Создавать программы с использованием идеологии ООП У7 Работать в среде программирования, реализовывать рефакторинг, отладку, тестирование программы Н1 Работы в интегрированной среде разработки Н2 Настройки среды разработки	Защита лабораторных работ	Задание на лабораторную работу выполнено, даны ответы на все поставленные вопросы, оформлен и защищён отчёт.			Задание на лабораторную работу не выполнено, отсутствует или не защищён отчёт.

		Н3 Алгоритмизации и программирования на процедурных и объектно-ориентированных языках Н4 Владения техникой рефакторинга, раздельной компиляции, отладчиком среды программирования, тестирования.					
		Дескрипторы	**Вид, форма оценочного мероприятия**	**Компетенция освоена**			
				отлично	**хорошо**	**удовлетворительно**	**неудовлетворительно**
6		З1 Основные этапы решения задачи. Понятия алгоритма, блок-схемы алгоритма, основные виды вычислительных процессов З2 Алфавит и лексику языков программирования, типы данных, правила записи выражений и операций З3 Операторы ввода-вывода, операторы ветвления, операторы цикла З4 Простые типы данных: множество значений, способ хранения, операции, особенности обработки З5 Структурированные ссылочные типы данных: множество значений, способ хранения, операции, особенности обработки З6 Основы процедурного программирования (процедуры, функции, библиотеки)	Зачет с оценкой	Обучающийся продемонстрировал высокий уровень знаний основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по специальности, справился с выполнением заданий, предусмотренных программой дисциплины.	Обучающийся продемонстрировал хороший уровень знаний основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по специальности, справился с выполнением заданий частично (2/3 от возможного максимального результата)	Обучающийся продемонстрировал низкий уровень знаний основного учебно-программного материала, справился с выполнением заданий частично, (1/3 от возможного максимального результата).	Обучающийся обнаружил значительные пробелы в знаниях основного учебно-программного материала, допустил принципиальные ошибки в выполнении заданий и не способен продолжить обучение без дополнительных занятий по соответствующей дисциплине