Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Силин Яков Петрович
Должность: Ректор
Дата подписания: 08.06.2026 15:31:33
Уникальный программный ключ:
24f866be2aca16484036a8cbb3c509a9531e6b5f

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Уральский государственный экономический университет»

Одобрена
на заседании кафедры

02.12.2025 г.

протокол № 3

Зав. кафедрой Назаров Д.М.

Утверждена
Советом по учебно-методическим вопросам и качеству образования

16 декабря 2025 г.

протокол № 4

Председатель _____ Карх Д.А.
(подпись)

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование дисциплины	Технологии и методы программирования
Направление подготовки	38.03.05 Бизнес-информатика
Профиль	Цифровой бизнес
Форма обучения	очная
Год набора	2026

Разработана:
Доцент, к.ф.-м.н.
Тюлюкин В.А.

Екатеринбург
2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Рабочая программа дисциплины является частью основной профессиональной образовательной программы высшего образования - программы бакалавриата, разработанной в соответствии с ФГОС ВО

ФГОС ВО	Федеральный государственный образовательный стандарт высшего образования - бакалавриат по направлению подготовки 38.03.05 Бизнес-информатика (приказ Минобрнауки России от 29.07.2020 г. № 838)

1. ЦЕЛЬ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины Технологии и методы программирования является знакомство с методами объектно-ориентированного программирования; изучение основ разработки алгоритмов на основе объектно-ориентированного подхода; формирование умений и навыков программирования экономических задач на основе изучения языка программирования Java.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

Дисциплина относится к обязательной части учебного плана.

3. ОБЪЕМ ДИСЦИПЛИНЫ

Промежуточная аттестация	Часов					З.е.
	Всего за семестр	Контактная работа .(по уч.зан.)			Самостоятельная работа в том числе подготовка контрольных и курсовых	
		Всего	Лекции	Лабораторные		
Семестр 3						
Экзамен	180	64	32	32	89	5

4.ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ ОПОП

В результате освоения ОПОП у выпускника должны быть сформированы компетенции, установленные в соответствии ФГОС ВО.

Шифр и наименование компетенции	Индикаторы достижения компетенций
ОПК-3 Способен управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации;	ИД-1.ОПК-3 Знать: принципы и методы организации управления процессами создания и использования продуктов и услуг в сфере ИКТ

ОПК-3 Способен управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации;	ИД-2.ОПК-3 Уметь: организовывать и управлять процессами создания и использования информационных продуктов и услуг
	ИД-3.ОПК-3 Иметь практический опыт: решения конкретных задач разработки, создания и использования программных продуктов.
ОПК-4 Способен понимать принципы работы информационных технологий; использовать информацию, методы и программные средства ее сбора, обработки и анализа для информационно-аналитической поддержки принятия управленческих решений;	ИД-1.ОПК-4 Знать: основные методы и подходы к поиску, сбору, обработке, анализу и систематизации информации, использованию программных средств и глобальных компьютерных сетей для подготовки обзоров, отчетов и научных публикаций
	ИД-2.ОПК-4 Уметь: использовать информационно-коммуникационные технологии, информационные ресурсы в решении профессиональных задач; самостоятельно проводить анализ информации, делать обоснованные выводы
	ИД-3.ОПК-4 Иметь практический опыт: применения методов системного анализа; инструментов математического моделирования, владения навыками использования программных продуктов для реализации типовых процедур обработки информации, методами анализа данных различного характера

5. ТЕМАТИЧЕСКИЙ ПЛАН

Тема	Часов

	Наименование темы	Всего часов	Контактная работа .(по уч.зан.)			Самост. работа	Контроль самостоятельной работы
			Лекции	Лабораторные	Практические занятия		
Семестр 3		153					
Тема 1.	Технологии и методы программирования (ОПК-3)	26	2	2		22	
Тема 2.	Основы программирования на Java.(ОПК-3, ОПК-4)	54	16	16		22	
Тема 3.	Объектно-ориентированное программирование на Java.(ОПК-3)	46	10	10		26	
Тема 4.	Создание графического интерфейса на Java и многопоточное программирование (ОПК-4)	27	4	4		19	

6. ФОРМЫ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ШКАЛЫ ОЦЕНИВАНИЯ

Раздел/Тема	Вид оценочного средства	Описание оценочного средства	Критерии оценивания
Текущий контроль (Приложение 4)			
Темы 1	Контрольная работа №1 (приложение 4)	Получение практических навыков написания программ на Java.	20 баллов максимум
Темы 2	Контрольная работа №2 (приложение 4)	Получение практических навыков написания программ на Java.	20 баллов максимум
Темы 3	Контрольная работа №3 (приложение 4)	Получение практических навыков проектирования и реализации классов на Java.	10 баллов максимум
Темы 4	Контрольная работа №4 (приложение 4)	Получение практических навыков проектирования и реализации иерархии классов на Java.	10 баллов максимум
Промежуточная аттестация(Приложение 5)			
3 семестр (Эк)	Экзаменационный билет(приложение 5)	Билет содержит 2 теоретических вопроса и 1 практическое задание. По заданию необходимо создать небольшой фрагмент кода, который иллюстрирует работу данного языкового средства	100 баллов. Оценивается правильность выполнения, оптимальность кода, самостоятельность выполнения, уровень понимания.

ОПИСАНИЕ ШКАЛ ОЦЕНИВАНИЯ

Показатель оценки освоения ОПОП формируется на основе объединения текущего контроля и промежуточной аттестации обучающегося.

Показатель рейтинга по каждой дисциплине выражается в процентах, который показывает уровень подготовки студента.

Текущий контроль.Используется 100-балльная система оценивания. Оценка работы студента в течении семестра осуществляется преподавателем в соответствии с разработанной им системой оценки учебных достижений в процессе обучения по данной дисциплине.

В рабочих программах дисциплин и практик закреплены виды текущего контроля, планируемые результаты контрольных мероприятий и критерии оценки учебный достижений.

В течение семестра преподавателем проводится не менее 3-х контрольных мероприятий, по оценке деятельности студента. Если посещения занятий по дисциплине включены в рейтинг, то данный показатель составляет не более 20% от максимального количества баллов по дисциплине.

Промежуточная аттестация. Используется 5-балльная система оценивания. Оценка работы студента по окончанию дисциплины (части дисциплины) осуществляется преподавателем в соответствии с разработанной им системой оценки достижений студента в процессе обучения по данной дисциплине. Промежуточная аттестация также проводится по окончанию формирования компетенций.

Порядок перевода рейтинга, предусмотренных системой оценивания, по дисциплине, в пятибалльную систему.

Высокий уровень – 100% - 70% - отлично, хорошо.

Средний уровень – 69% - 50% - удовлетворительно.

Показатель оценки	По 5-балльной системе	Характеристика показателя
100% - 85%	отлично	обладают теоретическими знаниями в полном объеме, понимают, самостоятельно умеют применять, исследовать, идентифицировать, анализировать, систематизировать, распределять по категориям, рассчитать показатели, классифицировать, разрабатывать модели, алгоритмизировать, управлять, организовать, планировать процессы исследования, осуществлять оценку результатов на высоком уровне
84% - 70%	хорошо	обладают теоретическими знаниями в полном объеме, понимают, самостоятельно умеют применять, исследовать, идентифицировать, анализировать, систематизировать, распределять по категориям, рассчитать показатели, классифицировать, разрабатывать модели, алгоритмизировать, управлять, организовать, планировать процессы исследования, осуществлять оценку результатов. Могут быть допущены недочеты, исправленные студентом самостоятельно в процессе работы (ответа и т.д.)
69% - 50%	удовлетворительно	обладают общими теоретическими знаниями, умеют применять, исследовать, идентифицировать, анализировать, систематизировать, распределять по категориям, рассчитать показатели, классифицировать, разрабатывать модели, алгоритмизировать, управлять, организовать, планировать процессы исследования, осуществлять оценку результатов на среднем уровне. Допускаются ошибки, которые студент затрудняется исправить самостоятельно.
49 % и менее	неудовлетворительно	обладают не полным объемом общих теоретическими знаниями, не умеют самостоятельно применять, исследовать, идентифицировать, анализировать, систематизировать, распределять по категориям, рассчитать показатели, классифицировать, разрабатывать модели, алгоритмизировать, управлять, организовать, планировать процессы исследования, осуществлять оценку результатов. Не сформированы умения и навыки для решения профессиональных задач
100% - 50%	зачтено	характеристика показателя соответствует «отлично», «хорошо», «удовлетворительно»
49 % и менее	не зачтено	характеристика показателя соответствует «неудовлетворительно»

7. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

7.1. Содержание лекций

Тема 1. Технологии и методы программирования (ОПК-3) Введение в технологии и методы программирования. Основы объектно-ориентированного программирования (ООП). Парадигмы ООП.
Тема 2. Основы программирования на Java.(ОПК-3, ОПК-4) Основы программирования на Java.
Тема 3. Объектно-ориентированное программирование на Java.(ОПК-3) Объектно-ориентированное программирование на Java.
Тема 4. Создание графического интерфейса на Java и многопоточное программирование (ОПК-4) Создание графического интерфейса на Java и многопоточное программирование

7.2 Содержание практических занятий и лабораторных работ

Тема 2. Основы программирования на Java.(ОПК-3, ОПК-4) Разработка Java-программ с помощью IntelliJ IDEA
Тема 3. Объектно-ориентированное программирование на Java.(ОПК-3) Получение практических навыков в ООП
Тема 4. Создание графического интерфейса на Java и многопоточное программирование (ОПК-4) Выполнение Заданиq.

7.3. Содержание самостоятельной работы

Тема 2. Основы программирования на Java.(ОПК-3, ОПК-4) Практическое программирование на Java
Тема 3. Объектно-ориентированное программирование на Java.(ОПК-3) Изучение дополнительных материалов
Тема 4. Создание графического интерфейса на Java и многопоточное программирование (ОПК-4) Изучение дополнительных материалов при создании приложение Windows-forms

7.3.1. Примерные вопросы для самостоятельной подготовки к зачету/экзамену
Приложение 1

7.3.2. Практические задания по дисциплине для самостоятельной подготовки к зачету/экзамену
Приложение 2

7.3.3. Перечень курсовых работ
Не предусмотрено

7.4. Электронное портфолио обучающегося
Материалы не размещаются

7.5. Методические рекомендации по выполнению контрольной работы
не предусмотрено

7.6 Методические рекомендации по выполнению курсовой работы
Не предусмотрено

8. ОСОБЕННОСТИ ОРГАНИЗАЦИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ ДЛЯ ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

По заявлению студента

В целях доступности освоения программы для лиц с ограниченными возможностями здоровья при необходимости кафедра обеспечивает следующие условия:

- особый порядок освоения дисциплины, с учетом состояния их здоровья;
- электронные образовательные ресурсы по дисциплине в формах, адаптированных к ограничениям их здоровья;
- изучение дисциплины по индивидуальному учебному плану (вне зависимости от формы обучения);
- электронное обучение и дистанционные образовательные технологии, которые предусматривают возможности приема-передачи информации в доступных для них формах.
- доступ (удаленный доступ), к современным профессиональным базам данных и информационным справочным системам, состав которых определен РПД.

9. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Сайт библиотеки УрГЭУ
http://lib.usue.ru/

Основная литература:

2. Лаврищева Е. М. Программная инженерия и технологии программирования сложных систем [Электронный ресурс]:учебник для вузов. - Москва: Юрайт, 2022. - 432 с – Режим доступа: https://urait.ru/bcode/491029

Дополнительная литература:

2. Лаврищева Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства [Электронный ресурс]:Учебник для вузов. - Москва: Юрайт, 2022. - 280 – Режим доступа: https://urait.ru/bcode/491048

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ВКЛЮЧАЯ ПЕРЕЧЕНЬ ЛИЦЕНЗИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИНФОРМАЦИОННЫХ СПРАВОЧНЫХ СИСТЕМ, ОНЛАЙН КУРСОВ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Перечень лицензионного программного обеспечения:

Microsoft Windows 10 .Договор № 52/223-ПО/2020 от 13.04.2020, Акт № Tr000523459 от 14.10.2020. Срок действия лицензии -Без ограничения срока.

Microsoft Visual Studio Community. Лицензия для образовательных учреждений. Срок действия лицензии - без ограничения срока.

Язык программирования Java.

Intellij IDEA.

Перечень информационных справочных систем, ресурсов информационно-телекоммуникационной сети «Интернет»:

Язык программирования Java
https://metanit.com/java/
Основы программирования на Java
https://intuit.ru/studies/courses/16/16/info

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Реализация учебной дисциплины осуществляется с использованием материально-технической базы УрГЭУ, обеспечивающей проведение всех видов учебных занятий и научно-исследовательской и самостоятельной работы обучающихся:

Специальные помещения представляют собой учебные аудитории для проведения всех видов занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду УрГЭУ.

Все помещения укомплектованы специализированной мебелью и оснащены мультимедийным оборудованием спецоборудованием (информационно-телекоммуникационным, иным компьютерным), доступом к информационно-поисковым, справочно-правовым системам, электронным библиотечным системам, базам данных действующего законодательства, иным информационным ресурсам служащими для представления учебной информации большой аудитории.

Для проведения занятий лекционного типа презентации и другие учебно-наглядные пособия, обеспечивающие тематические иллюстрации.

Приложение 1
к рабочей программе

Примерные вопросы для самостоятельной подготовки к экзамену

1 Язык JAVA. Основные конструкции языка

- История появления языка.
- Место JAVA среди других языков программирования.
- Объектно-ориентированное программирование (ООП).
- JIT-компилятор.
- Версии JAVA.
- Встроенные типы данных.
- Преобразование типов.
- Неявная типизация.
- Операторы *if*, *for*, *while*.
- Сокращённые логические операторы.
- Методы.
- Область видимости.
- Константы и перечисления.
- Массивы.

2 Классы

- Определение класса.
- Инкапсуляция.
- Динамические и статические объекты.
- Создание экземпляра класса.
- Статические поля, методы, классы.
- Методы расширения.
- Атрибуты доступа “private” и “public”.
- Наследование.

3 Методы и параметры

- Переменные ссылочного типа и типы значений.
- Массивы.
- Ступенчатые массивы.
- Неявно типизированные массивы.
- Передача параметров по значению (для элементарных типов) и по ссылке (массивов, классов).
- Конструкторы.
- Ключевое слово *this*.
- Перегрузка (Методов. Конструкторов. Вызов перегруженных конструкторов с помощью ключевого слова *this*).
- Инициализаторы объектов.
- Необязательные аргументы.

4 Обработка исключительных ситуаций

- Операторы *try – catch*.
- Генерирование собственного исключения.

- Задание характеристик для собственного исключения.
- Создание собственного типа (класса) исключений.
- Использование слова *finally*.
- Несколько блоков *catch*.

5 Наследование

- Наследование и защита доступа.
- Использование свойств для преодоления ограничений защиты.
- Модификатор доступа *protected*.
- Сложно организованное наследование.
- Ссылки на базовый класс и объекты производных классов.
- Использование ссылок на базовый класс в конструкторе.
- Виртуальные методы.
- Абстрактные методы и классы.
- Класс *Object*. Упаковка и распаковка.
- Интерфейсы.

6 Делегаты. События. Лямбда-выражения

- Делегаты.
- Групповая адресация в делегатах.
- Анонимные методы.
- Лямбда-выражения.
- События.

7 Generics

- Обобщённые методы.
- Обобщённые классы.
- Ограничения типов в обобщениях.

8 Динамическая идентификация типов

- Оператор *is*.
- Оператор *as*.
- Оператор *typeof*.

9 Многопоточное программирование

- Потоки и процессы.
- Класс *Thread*.
- Создание и запуск потока.
- Передача потоку аргумента.
- Приоритеты потоков.
- Синхронизация.
- Прерывание потока.

10 Коллекции, перечислители и итераторы

- Необобщенные коллекции.
- Интерфейсы необобщенных коллекций.
- Классы необобщенных коллекций.
- Специальные коллекции.
- Обобщенные коллекции.
- Интерфейсы обобщенных коллекций.
- Классы обобщенных коллекций.

- Параллельные коллекции.
- Доступ к коллекции с помощью перечислителя.
- Итераторы.

11 Строки и форматирование

- Класс *String*.
- Конструкторы класса *String*.
- Операторы класса *String*.
- Форматирование.
- Спецификаторы формата числовых данных.
- Форматирование даты и времени.
- Форматирование промежутков времени.

Приложение 2
к рабочей программе

Практические задания по дисциплине для самостоятельной подготовки к экзамену

ОПК-3, ОПК-4

1 Язык JAVA

1.1 Вопросы для контроля понимания темы:

- Какие ключевые отличия JAVA от других языков?
- Какие ближайшие “родственники” у JAVA и чем он от них отличается?
- JAVA – язык *компилируемого* типа, *интерпретируемого* или какого-то ещё?
- Какие элементы включает парадигма *объектно-ориентированное программирование* (ООП)?
- Что такое *JIT-компилятор*?
- Какие есть версии JAVA?

2 Основные конструкции языка

2.1 Вопросы для контроля понимания темы:

- Какие есть встроенные *типы данных*?
- Что такое *преобразование типов* (явное и неявное)?
- Что такое *неявная типизация*?
- Какие операторы используются для ветвления, циклов?
- Что такое сокращённые логические операторы?
- Что такое *методы*?
- Что такое *область видимости*?
- Что такое *константы*?
- Что такое *перечисления*? Зачем они нужны?
- Какие бывают массивы в JAVA?
- С какого числа начинают нумероваться элементы массива?
- В чём преимущества и недостатки использования ступенчатых массивов по сравнению с обычными многомерными массивами?
- Как создать неявно-типизированный массив?

2.2 Вопрос

При выполнении части кода будет выдавать ошибку. Почему? И как это исправить?

```
int a = 1;
int b = Math.Sin(a);
```

2.3 Вопрос

Какое значение примет переменная *b*?

```
int i = 259;
byte b = (byte) i;
```

2.4 Вопрос

Какое значение примет переменная *i* после первого if и какое – после второго?

```
short d = 12, f = 0, i = 0;
if (d > f | (++i < 10)) Console.WriteLine("i равно {0}", i);
if (d > f || (++i < 10)) Console.WriteLine("i равно {0}", i);
```

2.5 Вопрос

Какое значение примет переменная *absval*?

```
int val = -5;
int absval = val < 0 ? -val : val;
```

2.6 Практическое задание

Что нерационально в следующей программе? Как можно её оптимизировать? Выполните эту оптимизацию:

```
int[] a1 = new int[] { 1, 2, 3, 4, 5, 6 };
int Suma1 = 0;
for (int i = 0; i < a1.Length; i++)
{
        Suma1 += a1[i];
}
ResultTextBox.Text = "Сумма элементов массива a1 = " + Suma1;

int[] b1 = new int[] { 9, 10, 25, 18};
int Sumb1 = 0;
for (int i = 0; i < b1.Length; i++)
{
        Sumb1 += b1[i];
}
ResultTextBox.Text = "Сумма элементов массива b1 = " + Sumb1;
```

2.7 Вопрос

При выполнении части кода будет выдавать ошибку. Почему? Исправьте ошибку, чтобы не нарушить задумку автора.

```
int[] C1 = new int[] { 2, 5, 7, 11 };
int SumCSquares = 0;
for (int i = 0; i < C1.Length; i++)
{
        int CurSquare = C1[i] * C1[i];
        SumCSquares += C1[i];
}
ResultTextBox.Text = "Сумма квадратов элементов массива C1 = " + SumCSquares;
ResultTextBox.Text += "Квадрат последнего из просуммированных чисел = " + CurSquare;
```

2.8 Практическое задание

Отыщите ошибку в программе:

```
using System;
class ScopeDemo {
static void Main() {
  int x;
  x = 10;
  if(x == 10)
  {    int y = 20;
       Console.WriteLine("x и y: " + x + " " + y);
       x = y * 2;
  }
   y = 100;
   Console.WriteLine("x равно " + x);
  }
```

```
}
```

2.9 Практическое задание

Отыщите ошибку в программе:

```
using System;
class NestVar
{
 static void Main()
 {
  int count;
  for (count = 0; count < 10; count = count+1)
  {
   Console.WriteLine("Это подсчет: " + count);
   int count;
   for(count = 0; count < 2; count++)
   Console.WriteLine("В этой программе есть ошибка!");
  }
 }
}
```

2.10 Практическое задание

В приведённом ниже коде сразу две ошибки. Объясните, почему этот код ошибочный и исправьте его:

```
class Product
{
        public string Name;
        public double Price;
        public DateTime DateOfManufacture;
        public static double Discount;

        public void PrintInfo()
        {
                Console.WriteLine("Товар: " + Name);
                Console.WriteLine("Дата изготовления товара: " + DateOfManufacture);
                        Console.WriteLine("Цена товара: " + Price + " руб.");
                        Console.WriteLine("Цена товара со скидкой: " + Price*(100 -
Discount)/100 + " руб.");
                Console.WriteLine("\n");
        }
}
static void Main(string[] args)
{
        Product Bag = new Product();
        Bag.Discount = 15;
        Product.Name = "Товар";
```

2.11 Практическое задание

Улучшите читаемость кода с помощью *перечисления* вместо цифр

```
private string TimeOfWorking(int DayOfWeek)
{
        switch (DayOfWeek)
        {
                case 7:
```

```
                return "В этот день мы не работаем";
            case 6:
                return "Сегодня мы работаем с 10 до 19";
            default:
                return "Сегодня мы работаем с 9 до 21";
        }
}
```

2.12 Практическое задание

Создайте статический класс с методом void Print (string stroka, int color), который выводит на экран строку заданным цветом. Используя перечисление, создайте набор цветов, доступных пользователю. Ввод строки и выбор цвета предоставьте пользователю.

2.13 Практическое задание

Создать массив размерностью *N* элементов, заполнить его произвольными целыми значениями.

Вывести наибольшее значение массива, наименьшее значение массива, общую сумму элементов, среднее арифметическое всех элементов, вывести все нечетные значения.

2.14 Практическое задание

Создать массив с именем Train, содержащую следующие поля: название пункта назначения, номер поезда, время отправления.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа Train (записи должны быть упорядочены по номерам поездов);
- вывод на экран информации о поезде, номер которого введен с клавиатуры (если таких поездов нет, вывести соответствующее сообщение).

3 Классы

3.1 Вопросы для контроля понимания темы:

- Дайте определение *класса*.
- Какие *члены класса* могут быть?
- Чем *класс* отличается от *экземпляра* класса?
- Что такое *инкапсуляция*?
- Зачем нужен атрибут *static*?
- Что такое *методы расширения*?
- Зачем нужны и как влияют атрибуты доступа “*private*” и “*public*”?
- Что такое *наследование*? Зачем нужно *наследование*? Как атрибут доступа “*private*” проявляется при *наследовании*?
- Может ли в JAVA производный класс наследовать от нескольких родительских классов?

3.2 Практическое задание

Придумайте удачное название классу:

```
class *******
{
        public string Name;             // Название
        public int NumOfFloors; // Количество этажей
        public int Area;                // Общая площадь
        public int Occupants;   // Количество жильцов
        public DateTime DateOfConstruction;         // Дата постройки
}
```

3.3 Практическое задание

Для класса из предыдущего задания напишите конструктор, а также метод, который в виде string возвращает информацию о данном объекте

3.4 Практическое задание

Создать класс, который описывает вклад в банке.
В классе должны быть поля, которые характеризуют вклад (максимальный срок, возможность автоматической пролонгации, тип начисления процентов и т.п.).
В классе должен быть конструктор.
В классе должен быть метод, который в виде строки (string) возвращает информацию о вкладе.
В классе должен быть метод, который возвращает (рассчитывает) возвращаемую сумму вклада при заданном начальном размере вклада и сроке вклада

3.5 Практическое задание

Создать класс Invoice. В теле класса создать три поля int account, string customer, string provider, которые должны быть проинициализированы один раз (при создании экземпляра данного класса) без возможности их дальнейшего изменения.
В теле класса создать два закрытых поля string article, int quantity
Создать метод расчета стоимости заказа с НДС и без НДС.
Написать программу, которая выводит на экран сумму оплаты заказанного товара с НДС или без НДС.

3.6 Практическое задание

Создайте класс Vehicle.
В теле класса создайте поля: координаты и параметры средств передвижения (цена, скорость, год выпуска).
Создайте 3 производных класса Plane, Car и Ship.
Для класса Plane должна быть определена высота и количество пассажиров.
Для класса Ship – количество пассажиров и порт приписки.
Написать программу, которая выводит на экран информацию о каждом средстве передвижения.

3.7 Практическое задание

Где-то в программе ниже компилятор будет недоволен. Не копируя код в Visual Studio, догадайтесь где и почему.

```
public void MyMeth()
{
 char a, b;
 if(a==b)
 {
  Console.WriteLine("равно");
  return;
 }
 else
 {
  Console.WriteLine("не равно");
  return;
 }
 Console.WriteLine("Программа выполнена!");
}
```

3.8 Практическое задание

Создайте класс DocumentWorker.
В теле класса создайте три метода OpenDocument(), EditDocument(), SaveDocument().

В тело каждого из методов добавьте вывод на экран соответствующих строк: "Документ открыт", "Редактирование документа доступно в версии Про", "Сохранение документа доступно в версии Про".
Создайте производный класс ProDocumentWorker.
Переопределите соответствующие методы. При переопределении методов выводите следующие строки: "Документ отредактирован", "Документ сохранен в старом формате, сохранение в остальных форматах доступно в версии Эксперт".
Создайте производный класс ExpertDocumentWorker от базового класса ProDocumentWorker. Переопределите соответствующий метод. При вызове данного метода необходимо выводить на экран "Документ сохранен в новом формате".
В теле метода Main() реализуйте возможность приема от пользователя номера ключа доступа pro и exp. Если пользователь не вводит ключ, он может пользоваться только бесплатной версией (создается экземпляр базового класса), если пользователь ввел номера ключа доступа pro и exp, то должен создаться экземпляр соответствующей версии класса, приведенный к базовому – DocumentWorker.

4 Методы и параметры

4.1 Вопросы для контроля понимания темы:

- Чем отличаются переменные *ссылочного типа* и *типы значений*? Какие типы данных в JAVA соответствуют *ссылочным типам*, а какие – *типам значений*? На что это влияет?
- Зачем нужен модификатор параметра *Ref*?
- Зачем нужен модификатор параметра *Out*?
- Как создать метод, в котором часть параметров будет необязательна?
- Что такое *Конструктор*?
- Что обычно означает ключевое слово *this*?
- Что такое *перегрузка* методов?
- Какое преимущество для программиста даёт *перегрузка методов*? Чем это удобнее по сравнению с языками программирования, где нет перегрузки?
- Может ли один перегруженный конструктор вызвать другой?

4.2 Вопрос

Какое значение примет переменная ttt после вызова метода AddTwo()?

```
        int ttt = 5;
        AddTwo(ttt);
}

void AddTwo(int Number)
{
        Number = Number + 2;
}
```

4.3 Практическое задание

Как надо модифицировать программу из предыдущего примера, чтобы после возвращения из метода AddTwo к переменной ttt прибавлялось 2?

4.4 Практическое задание

Метод GetMin() находит минимальное значение функции на отрезке [*a*, *b*]. Для этого перебираются все значения функции с шагом *h*. Функция передаётся ему в виде делегата –

ссылки на метод, который вычисляет эту функцию. (Для тестирования можете задать, к примеру, $a = 3$, $b = 7$, $h = 0{,}0001$).
Перепишите программу так, чтобы после работы метода GetMin() можно было узнать не только само минимальное значение функции, но и то значение аргумента, при котором оно достигается. Предложите минимум два способа, как это сделать.

```
class Program
{
delegate double Func(double x);
static double Sin(double x)
{ return Math.Sin(x); }
static double Cos(double x)
{ return Math.Cos(x); }

static double GetMin(Func f, double a, double b, double h)
{
        double min = double.MaxValue;
        for (double x = a; x <= b; x += h)
        {
                double y = f(x);
                if (y < min) min = y;
        }
        return min;
}
static void Main(string[] args)
{
        Func f1 = Sin;
        Func f2 = Cos;
        Console.WriteLine("Введите начало отрезка: ");
        double a = double.Parse(Console.ReadLine());
        Console.WriteLine("Введите начало конец: ");
        double b = double.Parse(Console.ReadLine());
        Console.WriteLine("Введите шаг: ");
        double h = double.Parse(Console.ReadLine());
        double ymin = GetMin(Sin, a, b, h);
        Console.WriteLine("Минимум функции на отрезке [{0};{1}] равен {2}", a, b, ymin);
}
}
```

4.5 Вопрос

Какие значения будут содержаться в каждом из массивов после завершения данного фрагмента кода?

```
int[] nums1 = new int[] {1, 2, 3, 4, 5};
int[] nums2 = new int[] { 6, 7, 8, 9, 10, 11, 12 };

nums1[1] = 22;
nums2[1] = 33;
nums2 = nums1;
nums2[1] = 44;
```

4.6 Вопрос

В конструкторе приведённого ниже примера допущена некоторая ошибка. Что это за ошибка и как её исправить?

```
class doc
{
    public string Name;
    public int Size;
    public string Author;

    public doc(string Name, int Size, string Author)
    {
        Name = Name;
        Size = Size;
        Author = Author;
    }
```

4.7 Вопрос

В примере ниже какой из двух методов SubstructTax будет вызван? Как компилятор догадается, какой из них надо вызвать? Что будет, если убрать модификатор ref?

```
        double MyAnnualIncome = 200000;
        SubstructTax(ref MyAnnualIncome, 26);
}

void SubstructTax(ref double Val)
{
        Val = Val * 0.87;
}
void SubstructTax(ref double Val, double Tax)
{
        Val = Val * (1 - Tax / 100);
}
```

4.8 Практическое задание

Сократите код за счёт вызова одного перегруженного конструктора другим:

```
class FoodProduct
{ public string Name;                    // Название
  public string RegionofManufacturer;    // Область, в которой произведено
  public DateTime DateOfManufacture;     // Дата производства
  public TimeSpan ExpireRange;           // Срок годности
  public double Price;
  public FoodProduct()                   // Конструктор
  { RegionofManufacturer = "Свердловская область";
    DateOfManufacture = DateTime.Today;
    ExpireRange = new TimeSpan(7, 0, 0, 0);
  }
  public FoodProduct(string Region)      // Конструктор с одним параметром
  { RegionofManufacturer = Region;
    DateOfManufacture = DateTime.Today;
    ExpireRange = new TimeSpan(7, 0, 0, 0);
  }
```

```
 public FoodProduct(string Region, int Days)          // Конструктор с двумя параметрами
 { RegionofManufacturer = Region;
   DateOfManufacture = DateTime.Today;
   ExpireRange = new TimeSpan(Days, 0, 0, 0);
 }
}
```

4.9 Вопрос

Для чего предназначен метод GetSomeVal? Зачем там используется ключевое слово *params*?

```
class SomeClass
{
  public int GetSomeVal(params int[] nums)
  {
    int m;
    if(nums.Length == 0)
    {
      Console.WriteLine("Ошибка: нет аргументов.");
      return 0;
    }
    m = nums[0];
    for(int i=1; i < nums.Length; i++)
    if(nums[i] < m) m = nums[i];
    return m;
  }
}
```

4.10 Вопрос

Какое значение discount будет использоваться в методе PrintInfo при расчёте цены со скидкой?

```
static void Main(string[] args)
{
  PrintInfo("Молоко Ирбитское", 47, 20);
}
public static void PrintInfo(string Name, double Price, double discount = 0)
{
    Console.WriteLine("Цена товара со скидкой " + discount + "% составляет " + Price * (1 - discount / 100) + " руб." + "\n");
}
```

5 Обработка исключительных ситуаций

5.1 Вопросы для контроля понимания темы:

- Что такое *обработка исключительных ситуаций* и зачем это нужно?
- Какие основные операторы для *обработки исключительных ситуаций* и зачем нужен каждый из них?
- Как самому в программе сгенерировать исключение и зачем это может быть нужно?

5.2 Практическое задание

Используя предложенную заготовку кода, создайте собственные классы исключений для проверки силы и здоровья. Дополните *свойства* HP и Power проверками:

- Если задаётся HP меньше нуля, генерируется соответствующее исключение.
- Если задаётся Power больше 200, генерируется соответствующее исключение.

- Обеспечьте перехват сгенерированных вами исключений.

```
class Player
{
        public string Name;    //
        private int hp;     // Здоровье
        private int power;       // сила персанажа
        public int HP
        {
                get { return hp; }
                set { hp = value; }
        }

        public int Power
        {
                get { return power; }
                set { power = value; }
        }
}
```

5.3 Практическое задание

Описать структуру с именем Worker, содержащую следующие поля:
• фамилия и инициалы работника;
• название занимаемой должности;
• год поступления на работу.
Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из пяти элементов типа Worker (записи должны быть упорядочены по алфавиту);
• если значение года введено не в соответствующем формате, ваша программа должна перехватывать исключение и предложить оператору ввести данные ещё раз.
• вывод на экран фамилии работника, стаж работы которого превышает введенное значение.

5.4 Практическое задание

Создайте класс Calculator.
В теле класса создайте четыре метода для арифметических действий: (Add – сложение, Sub – вычитание, Mul – умножение, Div – деление).
Метод деления должен делать проверку деления на ноль, если проверка не проходит, сгенерировать собственное исключение.
Пользователь вводит значения, над которыми хочет произвести операцию и выбирает саму операцию. При возникновении ошибок он должен получать информацию об ошибке, но программа не должна “вылетать”.

6 Наследование. Виртуальные методы, абстрактные методы и классы. Интерфейсы.

6.1 Вопросы для контроля понимания темы:

- Существует ограничение, что при использовании модификатора *private* соответствующее поле становится недоступным не только извне класса, но и в производных классах? Какие есть варианты преодолеть это ограничение?
- Чем модификатор доступа *protected* отличается от *private*?
- Может ли переменная базового класса ссылаться на экземпляр производного класса и наоборот?

- Что такое *виртуальные методы*. Что такое переопределение виртуального метода? Зачем это всё может быть надо?
- Что такое *абстрактные методы* и *классы*. Что означает *реализация* абстрактного метода? В каких ситуациях лучше использовать *виртуальные*, а в каких – *абстрактные* методы?
- Какие члены могут быть абстрактными?
- Может ли абстрактный метод быть объявлен в неабстрактном классе?
- Чем абстрактный класс отличается от конкретного (обычного)?
- Может ли абстрактный класс иметь модификатор static?
- Какой "самый родительский (базовый)" класс в JAVA? Что такое *упаковка* и *распаковка*?
- Что такое *интерфейсы*? Что такое *реализация* интерфейса? Как может пригодиться использование *интерфейсов* при программировании?
- Чем абстрактный класс отличается от интерфейса?

6.2 Вопрос

Какая ошибка в данной программе?

```
abstract class Robot
{
        public string ID;               // Идентификатор
        public int CellSize;    // Ёмкость аккумулятора
        public double x, y;             // Координаты робота
        public abstract int GetTimeOfWork();
}

class BattleDroid : Robot
{
        public double Power;        // Мощность
        double Step;                            // Размер шага
        public void DoStep()
        {
                x += Step;
                y += Step;
        }
}
```

6.3 Вопрос

Компилятор будет недоволен. Почему? Предложите варианты, как исправить ошибку.

```
class Program
  {
    static void Main(string[] args)
    {
      IRoar someRoar = new Cat();
      someRoar.Roar();
      someRoar.Attack();
    }
  }

  interface IRoar
  {
    void Roar();
  }
```

```
class Cat : IRoar
{
    public void Roar()
    {
        Console.WriteLine("Мяу!");
    }
    public void Attack()
    {
        Console.WriteLine("На вас набросилась кошка!");
    }
}
```

7 Динамическая идентификация типов

7.1 Вопросы для контроля понимания темы:

- Зачем нужны операторы *is*, *as* и *typeof*?

8 Многопоточное программирование

8.1 Вопросы для контроля понимания темы:

- Что такое *поток*? Чем поток отличается от *процесса*?
- Сколько потоков может быть запущено в программе на JAVA?
- Зачем в программе может понадобиться запускать отдельные потоки?
- Каким способом можно передать в поток параметры?
- Что такое *синхронизация* при многопоточном программировании?
- Когда поток завершает свою работу? Как прервать поток до его обычного (самостоятельного) завершения?
- Для чего разработана библиотека *TPL* в JAVA 4.0?
- Что такое *асинхронное программирование*? Какие ключевые слова в JAVA для этого используются?

8.2 Практическое задание

С помощью создания отдельного потока реализуйте вывод на экран таймера обратного отсчёта. Основной поток должен обрабатывать нажатия кнопок запуска таймера, остановки, задания времени таймера, паузы. Поток, который выводит на экран текущее время таймера, может отслеживать действия пользователя через опрос *public* полей в *public* классе, либо получать уведомления через *события*.

8.3 Практическое задание

С использованием *параллельного foreach* создайте метод для вычисления среднего значения элементов двумерного массива.

9 Коллекции, перечислители и итераторы

9.1 Вопросы для контроля понимания темы:

- Что такое *коллекции*?
- Какие бывают коллекции и какие задачи они помогают решать?
- Чем отличаются *необобщенные коллекции* от *обобщённых*?
- Зачем нужны коллекции с распараллеливанием?

9.2 Практическое задание

С помощью коллекции реализуйте хранение информации о сегодняшнем меню в ресторане: наименование блюда и цена.

9.3 Практическое задание

С помощью коллекции реализуйте в программе электронную очередь в банк.

10 Строки и форматирование

10.1 Вопросы для контроля понимания темы:

- Можно ли в JAVA менять отдельные символы в строке прямо в том месте памяти, где строка размещается (как в обычном массиве)?
- Что такое *форматирование* строк? Как можно форматировать числовые данные, даты и время, промежутки времени?

10.2 Практическое задание

Реализуйте разделение строки на токены (отдельные слова). Строка для разделения: “Уральский государственный экономический университет – федеральное государственное бюджетное образовательное учреждение высшего образования, расположенное в городе Екатеринбурге, которое готовит экономистов различного профиля, технологов, юристов и специалистов в области государственного и муниципального управления”