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	10.03.01 Информационная безопасность
Профиль	Информационно-аналитические системы финансового мониторинга
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 10.03.01 Информационная безопасность (приказ Минобрнауки России от 17.11.2020 г. № 1427)

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, Курсовая работа	180	96	32	64	57	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-7 Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности;	ИД-1.ОПК-7 Знать: основные языки программирования, современные программные среды, алгоритмы решения задач
	ИД-2.ОПК-7 Уметь: применять языки программирования, современные программные среды для решения прикладных задач

ОПК-7 Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности;	ИД-3.ОПК-7 Владеть навыками программирования и технологии разработки программных средств для решения задач профессиональной деятельности

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	21	2	4		15	
Тема 2.	Основы программирования на Java.	63	16	32		15	
Тема 3.	Объектно-ориентированное программирование на Java.	45	10	20		15	
Тема 4.	Создание графического интерфейса на Java и многопоточное программирование	24	4	8		12	

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.

3 семестр (КР)	Курсовая работа	«Перечень курсовых работ (Приложение 3), Методические рекомендации по выполнению курсовой работы по дисциплине (Приложение 7)	100 баллов

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

7.2 Содержание практических занятий и лабораторных работ

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

7.3. Содержание самостоятельной работы

Тема 2. Основы программирования на Java. Практическое программирование на Java
Тема 3. Объектно-ориентированное программирование на Java. Изучение дополнительных материалов
Тема 4. Создание графического интерфейса на Java и многопоточное программирование 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
Приложение 3

7.4. Электронное портфолио обучающегося
Размещается курсовая работа

7.5. Методические рекомендации по выполнению контрольной работы
не предусмотрено

7.6 Методические рекомендации по выполнению курсовой работы
Приложение 7

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

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”

Приложение 3
к рабочей программе

Темы курсовых работ

1. Программа на JAVA для хранения и управления анкетами пользователей для сервиса знакомств.
2. Программа на JAVA для взаимодействия между начальником и подчинёнными в ВУЗе.
3. Программа на JAVA для взаимодействия между начальником и подчинёнными в школе.
4. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для отдела полиции.
5. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для медицинской клиники.
6. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для ЗАГСа.
7. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для музея.
8. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для агентства недвижимости.
9. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для таможни.
10. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для отделения МЧС.
11. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для департамента ФСБ.
12. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для департамента ГРУ.
13. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для строительной организации.
14. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для фонда помощи бездомным.
15. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для аэропорта.
16. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для ВУЗа.
17. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для школы.
18. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для конструкторского отдела.
19. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для суда по уголовным делам.
20. Программа на JAVA для организации архива документов с проверкой целостности данных файлов для железнодорожного вокзала.

Приложение 7
к рабочей программе

Федеральное государственное бюджетное образовательное учреждение высшего образования
УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНЫ
на заседании кафедры бизнес-информатики

Методические рекомендации по выполнению курсовой работы по дисциплине

Технологии и методы программирования

Цель курсовой работы: получить практические навыки в создании приложений с использованием языка программирования Java.

Порядок выполнения работы

1. Взять у преподавателя задания и уточнить его при необходимости.
2. Изучить методическое указание.
3. Познакомиться с графиком выполнения курсовой работы.
4. В папке для курсовой работы создать отдельный файл для хранения пояснительной записки к курсовой работе. В нем должны помещаться фрагменты, из которых в дальнейшем будет образован полный текст пояснительной записки. Первыми страницами в нем должны быть: титульный лист, лист задания, список литературы. Рекомендуется сразу выставить по тексту названия глав и параграфов в нужном формате. В конце периода из совокупности созданных фрагментов должен сложиться весь отчет – пояснительная записка.
5. Предусмотреть ***постоянное копирование документа на другие носители*** для уменьшения потерь в случае повреждения и удаления папок.
6. Написать программу.
7. Отладить программу.
8. Распечатать пояснительную записку и сдать ее для проверки.
9. Защитить работу перед комиссией.

Требования

1. Программный продукт должен быть с комментариями. Названия переменных, методов и пр. должны иметь “говорящие” имена.
2. Изучите [любые] рекомендации по написанию хорошо-стилизованного кода. И пишите программу в соответствии с этими рекомендациями.
3. В программе должны использоваться принципы объектно-ориентированного программирования.
4. Для защиты курсовой работы необходимо подготовить презентацию, в которую следует включить следующие слайды: постановка задачи, использованные средства, обоснование тех или иных решений, ключевые фрагменты кода, снимки экрана с примерами работы программы, заключение.

Рекомендации

В методическом указании имеется большая глава: «Содержательное наполнение разделов пояснительной записки». Эта глава является основной, и она состоит из разделов, наименования которых совпадают с наименованиями параграфов, отраженных в “Структуре и содержании курсовой работы”.

Содержательная часть этих разделов предназначена для того, чтобы помочь в написании текста пояснительной записки. Каждый раздел в общем случае представлен тремя компонентами: а) описанием того, какую информацию и каким образом следует отразить в соответствующем параграфе пояснительной записки; б) пример описания; в) указание литературного источника, где можно получить дополнительную помощь.

Методические указания

Как это следует из цели, поставленной в курсовой работе, основная задача, стоящая перед студентом курса: научиться составлять программы высокого качества. Такие программы должны быть легко модифицируемыми, простыми в обращении. Они должны быть написаны с использованием современных методов программирования, таких как *ООП* (объектно-ориентированные программирование), модульное программирование, процедурное программирование, визуальное программирование, событийное программирование.

Как известно, класс – это определяемый пользователем тип, объединяющий в себе группу данных и функций для работы с этими данными. В определении нового типа всегда лежит идея – отделить (абстрагироваться) несущественные подробности реализации от тех качеств, которые существенны для его правильного использования.

Одно из мощных преимуществ классов, как типов данных, заключается в том, что классам присуща структура, позволяющая моделировать реальные объекты. Любой предмет может быть описан набором своих характеристик, т.е. данных. Работать с моделью реального мира тем проще, чем больше отношения между данными в модели объекта напоминают отношения между характеристиками этого объекта.

Моделирование объектов в программе также называется *абстракцией*. Речь идет об имитации реально существующих объектов, отражающей особенности их взаимодействия в окружающем мире. А концепции виртуальной реальности выводят принцип абстракции на совершенно новый уровень, не связанный с физическими объектами. Абстракция необходима, потому что успешное использование ООП возможно лишь в том случае, если вы сможете выделить содержательные аспекты своей проблемы.

Поэтому основу решения задачи должна составлять разработка иерархии классов. Приступая к построению объектной модели, всегда задавайте себе вопрос: какие свойства и методы должны входить в объект, чтобы он адекватно моделировал ситуацию для решения поставленной задачи?

Структура и содержание курсовой работы

Титульный лист

Аннотация

Задание

Оглавление.

Введение.

1. Основная часть.
   1.1. Постановка задачи
   1.2. Анализ и исследование задачи, построение модели
   1.3. Разработка интерфейса
   1.4. Разработка классов
      1.4.1. Разработка иерархии классов
      1.4.2. Разработка схемы взаимодействия классов
      1.4.3. Разработка и реализация методов классов
2. Тестирование программы
   2.1. Разработка плана тестирования.
   2.2. Оценка результатов проведения тестирования
3. Заключение
4. Список литературы
5. Приложения

Содержательное наполнение разделов пояснительной записки

Аннотация

Содержит перечень используемых ключевых слов, очень краткое[1] содержание работы, число страниц пояснительной записки, число рисунков, таблиц, приложений.

Введение

[1] 2-3 предложения

Введение должно содержать общие сведения по теме курсовой работы. Так, если в основе работы лежат списки, то требуется дать информацию о списках (что это, зачем, особенности и т.д.). Если речь идет о множества, то сведения о множествах и т.д.

Во введении также необходимо отразить:

- актуальность выбранной темы;
- цель (например, приобрести навыки в разработке приложений на языке C# с применением современных технологий программирования);
- задачи, решаемые в проекте;
- используемые парадигмы программирования (например, объектно-ориентированное программирование, функциональное программирование, процедурное программирование и т.п.), фреймворки, технологии, платформы и пр.
- практическую значимость полученных результатов (где можно использовать);
- какого рода ресурсы необходимы для реализации (ПК, программное обеспечение…уточнить какое);
- перспективы совершенствования изготавливаемого программного продукта.

Постановка задачи

Составляющие элементы этого раздела м.б. следующие:

- формулировка задачи;
- выделение объектов, которые фигурируют в задаче;
- определение формы выдачи результатов;
- описание данных (их типов, диапазонов величин, структуры и т.п.).

***Формулировка условия задачи* выдается преподавателем**. В качестве примера для объяснения хода и логической основы некоторых элементов выполнения работы предлагается к рассмотрению следующая постановка:

О товаре, размещенном на складе, имеется информация вида: номер артикула, наименование товара, общее кол-во, выделенное кол-во, не поставленное количество, цена единицы. Разработать приложение (программу), которое выдает информацию о товаре по требованию пользователя и подводит итоги по не поставке товаров.

Выделение объектов, которые фигурируют в задаче. Следует определиться, с какими данными разработчик проекта имеет дело. Если в задаче речь идет о товаре, размещенном на складе, нужно указать, какой товар (перечислить его: стол, стул,…), что связано с поставкой/непоставкой (М.б. документы на заказ и на доставку). Выделить физический объект (множество товаров) и определить его свойства. В том числе,

количество элементов-единиц товара. А т.к. в данном случае физический объект определен как множество, то следует выделить свойства отдельного представителя из этого множества. При этом необходимо выявить самые существенные свойства, необходимые для решения задачи. Выделив наиболее важные факторы, можно пренебречь менее существенными. Параметрами отдельного представителя (единицы товара), например, могут быть: наименование, цена, фирма-изготовитель, поставщик, количество, … .

Определение формы выдачи результатов. Например, представить список поставщиков в виде таблицы, в которой можно было бы увидеть наименование поставляемого им товара … .

Описание данных (их типов, диапазонов величин, структуры и т.п.). Например, наименование товара представляется строкой символов, количество символов не превышает 20. Цена указывается в рублях, может быть представлена вещественным числом, диапазон ценовых колебаний в промежутке от 10000 до 500. ….

Здесь целесообразно подготовить для дальнейшей работы текстовый файл с реальными данными.

Анализ и исследование задачи, построение модели

Составляющими данной главы м.б. следующие элементы:

- выделение математического объекта;
- анализ существующих аналогов;
- анализ технических и программных средств;
- разработка математической модели;
- разработка требований к приложению.

Выделение математического объекта. Сказать, например, что для того, чтобы иметь возможность хранить и обрабатывать данные о физическом объекте, эти данные нужно представить в виде, приспособленном для обработки математическими методами. Для этого нужно перейти от физического объекта к объекту математическому. В нашем случае нужно перейти от физического объекта — множество товаров, где каждый товар имеет свои физические характеристики, к математическому объекту, описывающему это множество. Причем, каждый элемент математического множества должен быть представлен эквивалентными свойствами элемента физического множества (имеет структуру из свойств).

Далее можно сказать о том, что для описания математического множества можно воспользоваться таким математическим объектом, как «массив». В данном случае это будет массив структур.

Анализ существующих аналогов. Сказать, что хранить и обрабатывать массив структур описанных данных можно разными способами, например, средствами базы данных. Привести примеры известных баз, например, Access (особенности…), в Excel (особенности…).

Анализ технических и программных средств. ПК – техническое средство. Объяснить, почему выбрана среда Java, а не Excel или др.).

Разработка математической модели. Под математической моделью понимают систему математических соотношений – формул, уравнений, неравенств и т.д., отражающих существенные свойства объекта. Метод математического моделирования сводит исследование поведения объекта или его свойств к математическим задачам. Следует выписать формулы, например с использованием знаков суммирования ……

При этом можно пользоваться и такой схемой действий:

1. выделить предположения, на которых будет основываться математическая модель. Например, предположить, что информация о товаре будет сосредоточена в структурах данных (или содержатся в файле, или в таблице, или…);
2. определить, что считать исходными данными и результатами;
3. записать математические соотношения, связывающие результаты с исходными данными. Какие методы обработки (с описанием подхода к решению, например, «…. решение сводится к задаче накопления суммы элементов…формула….») данных потребуются для решения задачи?

Выведенные зависимости и формулы в общем случае могут быть представлены уравнениями, системами уравнений – линейных, интегральных, матричных, дифференциальных и др.. На этом этапе для нахождения решения также строится неформальный алгоритм (производная от длины и решение уравнения: производная равна 0, ….).

При построении математических моделей далеко не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких случаях используются математические методы, позволяющие дать ответы той или иной степени точности.

Разработка требований к приложению. Например, сказать о том, что пользователь должен иметь возможность для задания исходных данных вручную, возможность для сохранения данных в файле, возможность формировать файл данных с целью повторного

использования; извлекать данные из файла данных; выбирать вид отображаемой информации (например, общее количество товара, суммарная стоимость всего товара, номенклатура товара и т.д в соответствии с требованиями); отображать нужную информацию в соответствии с выбранным видом. …., а также обеспечение соединений этих данных в соответствии с пользовательской логикой (все исходя из поставленной задачи и личных мотивов).

Разработка классов

Следует сказать о том, что для решения задачи на машине следует от математического объекта перейти к программному объекту. Таким программным объектом будет являться класс, определенный пользователем. Класс по своей сути представляет запись математической модели на языке программирования.

Разработку классов рекомендуется проводить в следующем порядке:

- разработка структуры данных, помещаемых в поля класса;
- разработка модели поведения математического объекта;
- разработка схемы иерархии классов;
- разработка интерфейсов классов.

Разработка структуры данных, помещаемых в поля класса. Здесь следует отметить, что математический объект «массив структур» ` определяется двумя характеристиками: именем этого массива и его размерностью — *количеством элементов*. Все остальные свойства следуют однозначно из этих двух его характеристик.

А т.к. структура элемента/представителя массива, имеет набор характеристик, представленных разными типами, имеет смысл объединить характеристики, описывающие отдельный товар, в единое целое.

Далее следует расположить интерфейс структуры, у которой должны быть осмысленные имена, записанные на английском языке. Поля структуры следует снабдить комментариями.

Следует отметить, что элементы массива хранятся в массиве, память для которого должна будет выделяться автоматически в момент создания объекта/экземпляра типа класса. Именно благодаря такому решению можно будет создавать объекты – массивы с любым количеством элементов.

Разработка модели поведения математического объекта. Под моделью поведения, а эта терминология принята в технологии ООП, принято понимать совокупность

методов, необходимых для решения, определенного в классе, набора задач. Класс должен предоставлять пользователям следующие возможности: решать задачи, заявленные в условии: (перечислить); задавать исходные данные (что и как); генерировать значения по какому-либо правилу (как); формировать файл с целью повторного использования (…),; извлекать данные (из файла, …) данных и т.д. Сформулированные выше возможности будут определять поведение объектов класса и должны быть заложены в методы класса.

Естественно, что кроме заявленных выше методов класс-вектор, как и любой другой класс, должен иметь несколько конструкторов и деструктор.

Разработка схемы иерархии классов. При проектировании классов, как и при создании любой программы, разработчик должен изначально исходить из предположения, что предложенная задача со временем может претерпевать изменения, связанные с возрастанием и изменением запросов пользователя. Поэтому необходимо предусмотреть механизм, который бы позволял мобильно изменять интерфейсы классов, не затрагивая их реализацию. С этой целью имеет смысл сразу проектировать семейства классов.

Проектируя семейство классов, необходимо распределить, какие методы будут реализованы в базовом классе и какие из них будут абстрактными; сколько необходимо потомков и каким набором методов и свойств они должны обладать.

В рамках поставленной задачи целесообразно выделить семейство из трех классов. В базовый класс поместить методы, общие для перечисленных задач, в класс – наследник первой очереди поместить методы решения этих перечисленных задач. А в третий класс – наследник от наследника поместить методы, связанные с вводом и отображением данных. В таком представлении первые два класса будут мобильными, то есть они практически без исправлений могут компилироваться под любой системой программирования для Java и для различных операционных систем. Третий класс будет содержать методы с учетом работы в конкретной системе программирования. То есть он будет дополнять предыдущие классы методами, которые привязаны к типовым элементам, таким как классы и компоненты.

Разработка интерфейсов классов. В этом разделе следует на листингах представить интерфейсы классов, сопроводив их пояснениями, можно в следующей последовательности:

- характеристика класса. (Например, «…среди методов класса два конструктора и один деструктор и методы, предназначенные ….. Поля класса … перечислены в секции protected. Эта метка открывает эти поля для доступа к ним со стороны будущих наследников. Такое решение позволяет выявить ошибки типа нарушения прав доступа к данным еще на этапе компиляции программы. Методы класса перечислены в секции

public, т.к. они должны быть доступны из основной программы. Один конструктор будет использоваться для работы с объектами классов в случае ручного ввода данных, а второй – в случае ввода данных из файла…»);

- листинг с интерфейсом;
- спецификации методов класса.

На данном этапе разработки проектировать поведение методов класса следует по принципу «черного ящика»: объект реагирует на входные параметры, результаты являются выходными параметрами, а фактическая реализация остается неизвестной. Причем, входные и выходные параметры являются либо полями класса, либо параметрами метода.

Следует иметь в виду, что все имена: класса, полей, методов должны быть осмысленными, начинаться с заглавной буквы, записаны на английском языке (не следует использовать русские слова с английскими буквами).

Формализация расчетов

В данном разделе следует поместить описания методов классов по схеме:

1. Заголовок, в котором слова о назначении метода.
2. Прототип.
3. Словесное описание алгоритма в общем виде (суть действий).
4. Детальное описание алгоритма.

Разработка структурной схемы интерфейса

Разработчик должен предоставить ответы на следующие вопросы:

- Какие требования выставляются к интерфейсу?
- Почему именно такой интерфейс?
- Есть ли другие варианты?

На этом этапе разрабатывается структурная схема интерфейса программы и детали управления – логика (текстовое пояснение) решения задачи в программе. Разработка структурной схемы позволяет выверить все детали проекта, определить взаимоотношения (текстовое пояснение) между отдельными частями программы. Разработка структурной схемы определяет (текстовое пояснение) содержание программных сообщений. На основании этой схемы в дальнейшем строится *схема движения информационных* потоков ~~и~~ т.д.

Можно начать фразой: «Существует четыре основных (все остальные – производные) критерия оценки качества любого интерфейса, а именно:

- скорость работы пользователей;
- количество человеческих ошибок;
- скорость обучения;
- субъективное удовлетворение.

Скорость выполнения работы является важным критерием эффективности интерфейса. Длительность выполнения работы пользователем состоит из длительности восприятия исходной информации, длительности интеллектуальной работы (пользователь думает, что он должен сделать), длительности физических действий пользователя и длительности реакции системы. Как правило, длительность реакции системы является наименее значимым фактором.

Согласно Дональду Норману, взаимодействие пользователя с системой (не только компьютерной) состоит из семи шагов:

- формирование цели действий;
- определение общей направленности действий;
- определение конкретных действий;
- выполнение действий;
- восприятие нового состояния системы;
- интерпретация состояния системы;
- оценка результата.

Из этого списка становится видно, что процесс размышления занимает почти все время, в течение которого пользователь работает с компьютером, во всяком случае, шесть из семи этапов полностью заняты умственной деятельностью. Соответственно, повышение скорости этих размышлений приводит к существенному улучшению скорости работы.

К сожалению, существенно повысить скорость собственно мышления пользователей невозможно. Тем не менее, уменьшить влияние факторов, усложняющих процесс мышления, вполне возможно.

Пользователь должен знать:

- что он хочет получить на выходе (решение задачи);
- как минимум одну последовательность действий, приводящую к успешному результату;
- где ему найти все объекты, участвующие в процедуре решения;
- как определять пригодность объектов для их использования;
- как управляться с объектами.

Список, как видим, довольно внушительный. И если с первым пунктом проблем обычно не возникает, то остальные требуют определенных усилий. А помочь разобраться в том должен интерфейс, с встроенной системой подсказок действий. Следовательно, должен быть продуман механизм *управления программой через элементы интерфейса*.

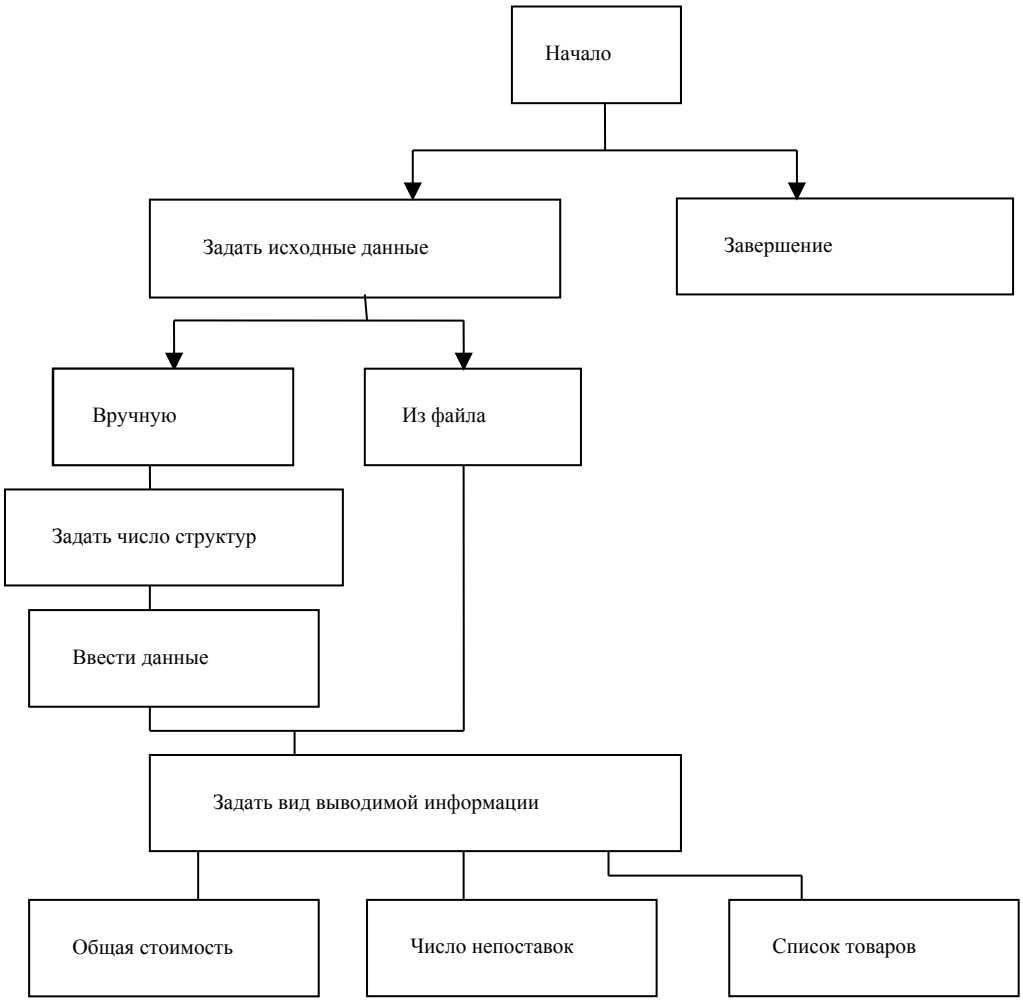

Рис.3. Структурная схема интерфейса

Структурную схему желательно согласовать с преподавателем!

В описательной части следует отметить, что в соответствии с проведенным выше анализом задачи разрабатываемая программа должна содержать: последовательные задания исходных данных (каких) для эксперимента, определяющих условия решения; выбор вида решения (расписать) и вида результатов (охарактеризовать в соответствии с ранее определенными функциями). В результате может быть сформирована структурная схема функционирования интерфейса подобная тому, как это представлено на рис.3.

Пояснить, что структурная схема интерфейса представляет собой графическую интерпретацию конструкции диалога, задающей требуемую последовательность обменов данными между пользователем и системой. Пояснить, что в каждом состоянии (каком) диалога разрабатываемая система ожидает ввода сообщения — реакции (уточнить) от пользователя и в зависимости от введенной информации переходит в другое состояние

(какое). Пояснить, что при завершении диалога осуществляется соответствующая обработка данных (какая) и выдается определенная информация (какая) на экран.

Описание интерфейса приложения

Этот раздел может начинаться словами: “Для решения поставленной задачи в соответствии с разработанной схемой было создано приложение, интерфейс которого включает в себя N^2 форм. Далее следует представить рисунки с изображением форм + описание деталей соответствующей формы (со ссылками на структурную схему) + комментарии к деталям формы”.

Описание структуры приложения и схема связности модулей

Можно начать словами: “Функционирование программ приложения формируется на базе следующих модулей:

Project – главный модуль, процедурами которого являются следующие модули: Unit1, MyUnit, ...

Unit1 – модулт, соответствующий главной форме Form1, с помощью которой осуществляется презентация данного программного продукта, а также вызов модуля: MyUnit, ……..

MyUnit – модуль, содержащий классы для решения задачи (реализующий класс (имя класса) и его подкласс (наследник) (имя)).

Между модульной структурой и структурами данных существует связь, которая может быть представлена в виде следующей схемы (рис.5).

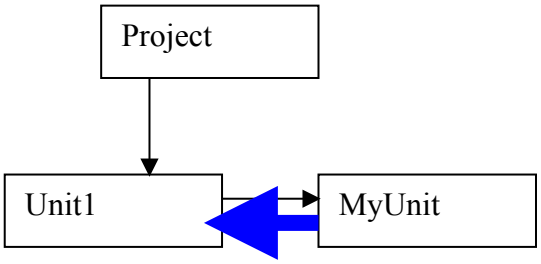

Рис.5. Схема связности модулей

Схема движения информационных потоков (связать с экземпляром класса)

Функционирование любой приложения (программы) можно рассматривать, как обработку некоторого входного потока данных. Данные поступают от входа, преобразуются по правилам решения задачи, и в преобразованном виде передаются к выходу внешним пользователям.

---

[2] N – равно числу созданных форм, обеспечивающих функционирование приложения

Задача анализа решения состоит в установлении правильности обработки данных.

Отсюда, задача определения способа хранения, порядка и правил организации данных, является очень важным элементом при разработке приложения. Четкое структурирование данных, выполняемое в процессе разработки, способствует уменьшению сложности системы и снижает вероятность ошибок из-за их неправильного использования.

Существуют различные способы организации и хранения информационных объектов: файлы, списки, строки, массивы, таблицы, структуры данных, классы, представляющие собой организованные наборы записей данных с методами их обработки и т.д.

Ряд данных, используемых многими модулями и группами системы, определяются как глобальные (файлы, структуры, классы, …). Это переменные, характеризуемые наиболее широким использованием и соответствующие высшему иерархическому уровню среди данных. Все эти данные в интерфейсе приложения отображаются с помощью тех или других визуальных компонент в определенной последовательности и в соответствии с выполненным преобразованием.

Пример описания: "*В интерфейсе приложения были определены следующие информационные объекты:*

1. *Файл, содержащий исходные данные, который имеет следующую структуру: ...(описание структуры файла).*
2. *Компонент Edit1, позволяющий создать однострочное текстовое поле, предназначенное для задания числа строк в матрице.*
3. *Компонент Edit2, позволяющий создать однострочное текстовое поле, предназначенное для задания числа столбцов в матрице.*
4. *Компонент StringGrid1, позволяющий задать элементы матрицы.*
5. *Компонент Edit3, позволяющий создать однострочное текстовое поле, предназначенное для отображения на форме результата*
6. *...*

Привести схему движения информационных потоков

Тестирование программы

Тестирование созданного программного продукта следует начинать с разработки плана тестирования. При этом следует помнить, что весь процесс тестирования можно разделить на три этапа:

- Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы.

- Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий — это граничные объемы данных, когда массивы состоят из слишком малого или слишком большого числа элементов.
- Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области изменений.

Известно, что все программы разрабатываются в расчете на обработку какого-то ограниченного набора данных. Поэтому важно получить ответ на следующие вопросы:

- Что произойдет, если в программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки придется иметь дело как раз с такими данными?
- Как будет вести себя программа, работающая с массивами, если количество их элементов превысит величину, указанную в объявлении массива?
- Что произойдет, если числа будут слишком малыми или слишком большими?

Наихудшая ситуация складывается тогда, когда *программа воспринимает неверные данные как правильные и выдает неверный, но правдоподобный результат*. Программа должна сама отвергать любые данные, которые она не в состоянии обрабатывать правильно.

Следует попытаться представить возможные типы ошибок, которые пользователь способен допустить при работе с Вашей программой и которые могут иметь неприятные для нее последствия. Нужно не забывать, что способ мышления пользователя отличается от способа мышления программиста. Если помнить об этом, то нужно предусмотреть обработку ошибок типа: отсутствие нужного файла, неправильные форматы данных и т.д. Список действий, которые могут привести к неправильному функционированию программы, довольно длинный и зависит от того, что делает приложение в данный момент времени.

Основной смысл этого этапа состоит в проверке того, насколько программный продукт в том виде, в котором он получился, соответствует требованиям, установленным в процессе согласования спецификации. Каждая функция или метод класса должны соответствовать требованиям, определенным для них на этапе спецификации.

Разработка плана тестирования - состоит из следующих шагов:

1)определение последовательности действий, которые позволяют проверить как работу отдельных методов, так и их совокупности;

2)подготовка входных данных, для которых известны результаты тестирования;

3)определение места расположения тестовых данных.

Разработка алгоритма процедуры тестирования состоит в разработке процедуры в соответствии с составленным выше планом тестирования. Эту процедуру можно представить блок-схемой с соответствующими комментариями.

Оценка результатов тестирования – содержит сравнение выходных данных работы отдельных процедур с контрольными значениями, которые получены в результате выполнения процедуры тестирования.

Заключение

Подводятся общие итоги проделанной работы, дается их оценка, делаются общие выводы.

Например, начало может быть следующим:

“В процессе разработки программного приложения (программной системы) для решения конкретной задачи было проделано следующее:

1. изучена литература по теме проекта;
2. разработан алгоритм решения задачи по обработке …;
3. разработан интерфейс приложения;
4. разработана схема движения информационных потоков;
5. разработаны классы и модули приложения…(сколько, особенности);
6. ¼

Продумать и изложить перспективы[3] в усовершенствовании разрабатываемого приложения.

[3] Что хотелось бы реализовать, но не было сделано