МИНОБРНАУКИ РОССИИ

Орский гуманитарно-технологический институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования «Оренбургский государственный университет»
(Орский гуманитарно-технологический институт (филиал) ОГУ)

Кафедра программного обеспечения

Методические указания по выполнению и защите лабораторных работ по дисциплине «Б1.Д.В.3 Объектно-ориентированное программирование»

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки

09.03.01 Информатика и вычислительная техника

(код и наименование направления подготовки)

Программное обеспечение средств вычислительной техники и автоматизированных систем

(наименование направленности (профиля) образовательной программы)

Тип образовательной программы

Программа бакалавриата

Квалификация

Бакалавр

Форма обучения

Очная

Год начала реализации программы (набора)

2019

г. Орск 2018

Методические указания предназначены для обучающихся очной формы обучения направления подготовки 09.03.01 Информатика и вычислительная техника профилю Программное обеспечение средств вычислительной техники и автоматизированных систем по дисциплине «Б1.Д.В.3 Объектно-ориентированное программирование»

Составитель _____ В.С. Богданова
_____ О.В. Подсобляева

Методические указания рассмотрены и одобрены на заседании кафедры программного обеспечения, протокол № 1 от «01» сентября 2018 г.

Заведующий кафедрой _____ Е.Е. Сурина

Согласовано:
Председатель методической комиссии по направлению подготовки 09.03.01 Информатика и вычислительная техника

_____ Е.Е.Сурина
«12» сентября 2018 г.

Пояснительная записка

В результате изучения дисциплины «Б1.Д.В.3 Объектно-ориентированное программирование» у обучающихся должны быть сформированы знания, умения и навыки:

– изучить основные принципы объектно-ориентированного программирования; изучить реализацию этих принципов на языках C++ и Java;

– научиться писать программы на языках C++ и Java;

– научиться проектировать и разрабатывать объектно-ориентированные программы.

Одной из наиболее эффективных форм закрепления теоретических знаний и выработки навыков самостоятельной работы являются лабораторные занятия.

Целью проведения лабораторных занятий является:

- закрепление знаний студентов по основам проектной деятельности,

- формирование у студентов навыков использования современных технических средств и технологий для решения проектных и исследовательских задач.

Тематический план

Таблица 1 – Тематический план выполнения лабораторных работ по дисциплине «Б1.Д.В.3 Объектно-ориентированное программирование» для обучающихся направления подготовки 09.03.01 Информатика и вычислительная техника профиль подготовки Программное обеспечение средств вычислительной техники и автоматизированных систем

в 4 семестре

№ занятия	№ раздела	Тема	Кол-во часов
1	1-3	Язык C++. Объекты и классы.	2
2	4	Язык C++. Конструкторы и деструкторы.	2
3	5	Язык C++. Массивы объектов, указатели и ссылки на объекты.	2
4	6	Язык C++. Перегрузка операций.	2
5	7	Язык C++. Наследование.	2
6	8	Язык C++. Виртуальные функции.	2
7	9	Язык C++. Потоки и файлы.	2
8	10	Язык C++. Многофайловые программы.	2
		Итого:	16

в 5 семестре

№ занятия	№ раздела	Тема	Кол-во часов
11	11-14	Язык Java. Среда программирования. Основные конструкции. Объекты и классы.	1
12	15	Язык Java. Наследование.	1
13	16	Язык Java. Интерфейсы и внутренние классы.	1
14	17	Язык Java. Программирование графики.	1
15	18	Язык Java. Обработка событий.	2
16	19	Язык Java. Компоненты пользовательского интерфейса из пакета Swing.	2
17	20	Язык Java. Аплеты.	2

18	21	Язык Java. Исключения и отладка.	2
19	22	Язык Java. Потоки и файлы.	2
20	23	Язык Java. Работа с универсальными типами	2
		Итого:	16

Методические указания по выполнению и оформлению лабораторных работ

Лабораторные работы по дисциплине «Объектно-ориентированное программирование» предполагают решение задач по темам, представленным в тематическом плане.

В практической работе должны быть выполнены все предусмотренные задания. В работе должна просматриваться логическая последовательность и взаимная увязка основных частей работы.

Рекомендуемая структура лабораторных работ:

1) цель практической работы;

2) задание в соответствии с выбранным вариантом;

3) теоретическая часть, включающая краткое изложение теоретических положений по теме практической работы, формулы для решения задания;

4) практическая часть, включающая решение задания по теме практической работы. Дополнительно для наглядности расчетный материал может быть представлен в виде таблиц, графиков;

5) выводы по практической работе;

6) список использованной литературы.

Лабораторные работы могут быть оформлены:

- машинописным текстом на листах формата А4.

Титульный лист оформляется на основе СТО 02069024. 101 – 2014 «РАБОТЫ СТУДЕНЧЕСКИЕ. Общие требования и правила оформления».

Работа защищается устно и принимается к зачету, если нет замечаний по ее выполнению и оформлению. При отсутствии зачтенных лабораторных работ студент не допускается к зачету по дисциплине «Б1.Д.В.3 Объектно-ориентированное программирование».

Лабораторная работа №1
Язык С++. Объекты и классы.

Разработать классы для описанных ниже объектов. Включить в класс методы set (…), get (…), show (…). Определить другие методы.

1.**Student**: Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс. Создать массив объектов. Вывести:

а) список студентов заданного факультета; б) списки студентов для каждого факультета и курса;

в) список студентов, родившихся после заданного года.

2.**Abiturient**: Фамилия, Имя, Отчество, Адрес, Оценки. Создать массив объектов. Вывести:

а) список абитуриентов, имеющих неудовлетворительные оценки; б) список абитуриентов, сумма баллов у которых не меньше заданной;

в) выбрать N абитуриентов, имеющих самую высокую сумму баллов, и список абитуриентов, имеющих полупроходной балл.

3.**Aeroflot**: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Создать массив объектов. Вывести:

а) список рейсов для заданного пункта назначения; б) список рейсов для заданного дня недели;

в) список рейсов для заданного дня недели, время вылета для которых больше заданного.

4.**Book**: Автор, Название, Издательство, Год, Количество страниц. Создать массив объектов. Вывести:

а) список книг заданного автора; б) список книг, выпущенных заданным издательством;

в) список книг, выпущенных после заданного года.

5.**Worker**: Фамилия и инициалы, Должность, Год поступления на работу, Зарплата. Создать массив объектов. Вывести:

а) список работников, стаж работы которых на данном предприятии превышает заданное число лет; б) список работников, зарплата которых больше заданной;

в) список работников, занимающих заданную должность.

6.**Train**: Пункт назначения, Номер поезда, Время отправления, Число общих мест, Купейных, Плацкартных. Создать массив объектов. Вывести:

а) список поездов, следующих до заданного пункта назначения; б) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

в) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

7.**Product**: Наименование, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести:

а) список товаров для заданного наименования; б) список товаров для заданного наименования, цена которых не превышает указанной;

в) список товаров, срок хранения которых больше заданного.

8.**Patient**: Фамилия, Имя, Отчество, Адрес, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести:

а) список пациентов, имеющих данный диагноз; б) список пациентов, номер медицинской карты которых находится

взаданном интервале.

9.**Bus**: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Создать массив объектов. Вывести:

а) список автобусов для заданного номера маршрута; б) список автобусов, которые эксплуатируются больше 10 лет;

в) список автобусов, пробег у которых больше 10 000 км.

10.**Customer**: Фамилия, Имя, Отчество, Адрес, Телефон, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести:

а) список покупателей в алфавитном порядке; б) список покупателей, номер кредитной карточки которых находится в заданном интервале.

11.**File**: Имя файла, Размер, Дата создания, Количество обращений. Создать массив объектов. Вывести:

а) список файлов, упорядоченный в алфавитном порядке; б) список файлов, размер которых превышает заданный;

в) список файлов, число обращений к которым превышает заданное.

12.**Word**: Слово, Номера страниц, на которых слово встречается (от 1 до 10), Число страниц. Создать массив объектов. Вывести:

а) слова, которые встречаются более чем на N страницах; б) слова в алфавитном порядке;

в) для заданного слова номера страниц, на которых оно встречается.

13.**House**: Адрес, Этаж, Количество комнат, Площадь. Создать массив объектов. Вывести:

а) список квартир, имеющих заданное число комнат; б) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в определенном промежутке;

в) список квартир, имеющих площадь, превосходящую заданную.

14.**Phone**: Фамилия, Имя, Отчество, Адрес, Номер, Время внутригородских разговоров, Время междугородних разговоров. Создать массив объектов. Вывести:

а) сведения об абонентах, время внутригородских разговоров которых превышает заданное; б) сведения об абонентах, воспользовавшихся междугородней связью;

в) сведения об абонентах, выведенные в алфавитном порядке.

15.**Person**: Фамилия, Имя, Отчество, Адрес, Пол, Образование, Год рождения. Создать массив объектов. Вывести:

а) список граждан, возраст которых превышает заданный; б) список граждан с высшим образованием; в) список граждан мужского пола.

Лабораторная работа №2
Язык С++. Конструкторы и деструкторы

Разработать перечисленные ниже классы. При разработке каждого класса возможны два варианта решения: а) данные-члены класса представляют собой переменные и массивы фиксированной размерности; б) память для данных-членов класса выделяется динамически.

1.«**Комплексное число**» – **Complex**. Класс должен содержать несколько конструкторов и операции для сложения, вычитания, умножения, деления, присваивания. Создать два вектора размерности *n* из комплексных координат. Передать их в функцию, которая выполняет сложение комплексных векторов.

2.Определить класс «**Дробь**» – **Fraction** в виде пары m,n . Класс

должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Перегрузить операции сложения, вычитания, умножения, деления, присваивания и операции отношения. Создать массив объектов и передать его в функцию, которая изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

3. Разработать класс «**Вектор**» – **Vector** размерности *n*. Определить несколько конструкторов, в том числе конструктор копирования. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Перегрузить операции сложения, вычитания, умножения, инкремента, декремента, индексирования, присваивания для данного класса. Создать массив объектов. Написать функцию, которая для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.

4.Определить класс «**Квадратная матрица**» – **Matrix**. Класс должен содержать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для сложения, вычитания, умножения матриц; вычисления нормы матрицы. Перегрузить операции сложения, вычитания, умножения и присваивания для данного класса. Создать массив объектов класса **Matrix** и передать его в функцию, которая изменяет *i*-ю матрицу путем возведения ее в квадрат. В головной программе вывести результат.

5.Разработать класс **«Многочлен» – Polynom** степени *n*. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для вычисления значения полинома; сложения, вычитания и умножения полиномов. Перегрузить операции сложения, вычитания, умножения, инкремента, декремента, индексирования, присваивания. Создать массив объектов класса. Передать его в функцию, вычисляющую сумму полиномов массива и возвращающую по- лином-результат, который выводится на экран в головной программе.

6.Определить класс **«Стек» – Stack**. Элементы стека хранятся в массиве. Если массив имеет фиксированную размерность, то предусмотреть контроль выхода за пределы массива. Если память выделяется динамически и ее не хватает, то увеличить размер выделенной памяти. Включение элементов в стек и их извлечение реализовать как

ввиде методов, так и с помощью перегруженных операций. Создать массив объектов класса **Stack**. Передавать объекты в функцию, которая удаляет из стека первый (сверху), третий, пятый и т. д. элементы.

7.Построить классы для описания плоских фигур: круг, квадрат, прямоугольник. Включить методы для изменения объектов, перемещения на плоскости, вращения. Перегрузить операции, реализующие те же действия. Выполнить тестирование класса, создав массив объектов.

8.Определить класс «**Строка**» – **String** длины *n*. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для выполнения конкатенации строк, извлечения символа из заданной позиции, сравнения строк. Перегрузить операции сложения, индексирования, отношения, добавления , присваива-

ния для данного класса. Создать массив объектов и передать его в функцию, которая выполняет сортировку строк.

9.Разработать класс «Множество (целых чисел, символов, строк

ит. д.)» – **Set** мощности *n*. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для определения принадлежности заданного элемента множеству, пересечения, объединения, разности двух множеств. Перегрузить операции сложения, вычитания, умножения (пересечения), индексирования, присваивания. Создать массив объектов и передавать пары объектов в функцию, которая строит множество, состоящее из элементов, входящих только в одно из заданных множеств, т. е. $A\ B \setminus A\ B$, и возвращает его в головную программу.

10.Разработать класс для массива строк. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для поэлементной конкатенации двух массивов, упорядочения строк в лексикографическом порядке, слияния двух массивов с удалением повторяющихся строк, а также для вывода на экран всего массива и заданной строки. Перегрузить операции сложения, умножения, индексирования, присваивания для данного класса. Создать массив объектов и передавать объекты в функцию, которая выполняет слияние объектов и для полученного объекта-результата производит лексикографическое упорядочения строк.

11.Составить описания класса, обеспечивающего представление

матрицы заданного размера *n m* и любого минора в ней. Память для матрицы выделять динамически. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для отображения на экране как матрицы в целом, так и заданного минора, а также для изменения минора; сложения, вычитания, умножения миноров. Перегрузить операции сложения, вычитания, умножения и присваивания для данного класса. Создать массив объектов данного класса и передать его в функцию, которая изменяет для *i*-й матрицы

ееминор путем умножения на константу.

12.Построить класс «**Булев вектор**» – **BoolVector** размерности *n*. Определить несколько конструкторов, в том числе конструктор копирования. Реализовать методы для выполнения поразрядных конъюнкции, дизъюнкции и отрицания векторов, а также подсчета числа единиц и нулей в векторе. Реализовать те же действия над векторами с помощью перегруженных операций. Перегрузить операции отношения и присваивания для данного класса. Создать массив объектов. Передавать объекты в функцию, которая будет их изменять по формуле

13. Реализовать класс «**Троичный вектор**» – **Tvector** размерности *n*. Компоненты вектора принимают значения из множества 0,1, *X* .

Два троичных вектора *tk*=(*t*1*k*,…,t*nk*) и *tl*=(*t*1*l*,…,*tnl*) называются ортогональными, если существует такое *i*, что *tik*, *til* {0,1} и *tik til*. Операцияпересечения не ортогональных векторов выполняется покомпонентно по следующим правилам: 1 1=1 *X*=*X* 1=1, 0 0=0 *X*=*X* 0=0, *X X*=*X*. Реализовать методы для проверки векторов на ортогональность, для пересечения не ортогональных векторов, сравнения векторов, подсчета числа компонент, равных *X*.

Осуществить те же действия над векторами с помощью перегруженных операций. Перегрузить операцию присваивания для данного класса. Выполнить тестирование класса, создав массив объектов.

14. Определить класс «**Булева матрица» – BoolMatrix** размерности $n \times m$. Класс должен содержать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для логического сложения (дизъюнкции), умножения и инверсии матриц. Реализовать методы для подсчета числа единиц в матрице и лексикографического упорядочения строк. Перегрузить операции для логического сложения, умножения и инверсии матриц, а также операцию присваивания. Создать массив объектов класса **BoolMatrix.**

Лабораторная работа №3
Язык С++. Массивы объектов, указатели и ссылки на объекты.

При решении задач необходимо описать класс, который используется для представления элементов динамической структуры данных. Затем разрабатывается класс для работы с используемой динамической структурой данных, которая при тестировании класса может быть построена путем ввода данных: а) с клавиатуры; б) из файла. Возможны два варианта решения:

а) динамическая структура данных постоянно хранится в памяти; б) динамическая структура данных хранится в файле.

1.Создать класс для работы со стеком. Элемент стека – действительное число. Применить класс для вывода возрастающих серий последовательности действительных чисел: а) в обратном порядке; б) в том же порядке (серия – упорядоченная последовательность максимальной длины).

2.Построить класс для работы со стеком. Элемент стека – целое число. Ввести две неубывающие последовательности чисел в два стека. Использовать третий стек для слияния двух последовательностей в одну неубывающую.

3.Создать класс для работы со стеком. Элемент стека – символ. Сформировать два стека, содержащие последовательности символов. Подсчитать общее число элементов в стеках, предусмотреть восстановление их исходного расположения.

4.Создать класс для работы со стеком. Элемент стека – символ. Использовать стек для проверки правильности расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении.

5.Построить класс для работы с односвязным списком. Элемент списка – действительное число. Сформировать список, содержащий неубывающую последовательность чисел, и преобразовать его так, чтобы последовательность была невозрастающей. Для этого необходимо совершить переворот списка, т. е. такую переустановку указателей в списке, при которой элементы его следуют друг за другом в обратном порядке.

6.Построить класс для работы с односвязным списком. Элементы списка – целые числа. Сформировать список, упорядочить элементы списка по возрастанию, используя сортировку: а) методом выбора; б) методом пузырька; в) методом вставки.

7.Построить класс для работы с односвязным списком. Элементы списка – действительные числа. Создать два упорядоченных по невозрастанию списка, слить их в один (также упорядоченный по невозрастанию), построив новый список.

8.Построить класс для работы с односвязным списком. Элементы списка – слова. Создать список, содержащий некоторую последовательность слов. Заменить в списке каждое вхождение заданного слова другим (также заданным).

9.Построить класс для работы с односвязным списком. Создать два списка: List1 и List2. Проверить, содержатся ли элементы списка List1 в списке List2 в указанном списком List1 порядке.

10.Построить класс для работы с односвязным списком. Элементы списка – целые числа. Создать список List1. Построить список List2, содержащий порядковые номера максимальных элементов спи-

ска List1.

11.Построить класс для работы с двусвязным списком. Элементы списка – действительные числа. Создать список List1, содержащий последовательность $x1, x2, \ldots, xn$. Построить список List2, содержа-

щий последовательность $x1, xn, x2, xn\ 1, \ldots, xn, x1$.

12.Создать класс для работы с бинарным деревом, узлы которого содержат целые числа. Построить дерево, затем копию дерева. Подсчитать число листьев в нем (листьями называются узлы, не содержащие поддеревьев).

13.Построить класс для работы с бинарным деревом, узлы которого содержат действительные числа. Создать дерево. Определить высоту дерева (максимальное число узлов, принадлежащих пути от корня дерева до любого из его листьев). Подсчитать число элементов, равных максимальному.

14.Построить класс для работы с бинарным деревом, узлы которого содержат действительные числа. Создать дерево для заданной последовательности чисел. Используя его, упорядочить последовательность по возрастанию, убыванию.

15.Построить класс для работы со списком. Элемент списка содержит информацию о заявке на авиабилет: пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

Программа должна обеспечивать: хранение всех заявок в виде списка, добавление заявок в список, удаление заявок, вывод заявок по заданному номеру рейса и дате вылета, вывод всех заявок.

16.Построить класс для работы с бинарным деревом, узел которого содержит информацию о заявках на авиабилеты (в последовательности, используемой для упорядочения заявок): желаемую дату вылета, номер рейса, фамилию и инициалы пассажира, пункт назначения.

Программа должна обеспечивать: хранение всех заявок в виде бинарного дерева, добавление заявок, удаление заявок, вывод заявок по заданному номеру рейса и дате вылета, вывод всех заявок.

17.Построить класс для работы со списком, который содержит динамическую информацию о наличии автобусов в парке: номер автобуса, фамилию и инициалы водителя, номер маршрута, признак местонахождения автобуса – на маршруте или в парке.

Программа должна обеспечивать: начальное формирование списка, введение номера автобуса при выезде и установление программой значения признака «автобус на маршруте». Аналогичным образом изменяется информация об автобусе при его возвращении в парк. По запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

18.Решить предыдущую задачу, используя не список, а бинарное дерево.

19.Построить класс для работы с бинарным деревом, содержащим англо-русский словарь.

20.Построить класс для работы со списком, содержащим информацию о поездах дальнего следования. Элемент списка содержит следующую информацию о поезде: номер поезда, станция назначения, время отправления.

Составить программу, которая обеспечивает первоначальное формирование списка, производит вывод списка, вводит номер поезда и выводит информацию о нем, вводит название станции назначения и выводит данные о всех поездах, следующих до этой станции.

Лабораторная работа №4

Язык С++. Перегрузка операций.

Для разработки шаблонов классов можно использовать результаты выполнения лабораторных работ № 2 и № 3. При тестировании созданных шаблонов классов необходимо создавать объекты с различными допустимыми значениями параметров шаблона (например, компоненты вектора могут быть целыми, действительными или комплексными числами).

1. Создать шаблон класса для работы со стеком. Применить его для решения задач № 1 – 4 (лаб. работа № 3).

2.Создать шаблон класса для работы с одномерным массивом. Выполнить тестирование путем создания и обработки массивов, содержащих элементы различных типов (например, сортировка элементов массивов различными методами).

3.Создать шаблон класса **Vector** размерности *n* (см. задачу № 3, лаб. работа № 2).

4.Создать шаблон класса «Квадратная матрица» – **Matrix** раз-

мерности *n n* (см. задачу № 4, лаб. работа № 2).

5.Создать шаблон класса **Polynom** степени *n* (см. задачу № 5, лаб. работа № 2) или создать шаблон класса для работы с односвязным списком. Применить его для решения задачи № 5 (лаб. работа № 3).

6.Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 6 (лаб. работа № 3).

7.Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 7 (лаб. работа № 3).

8.Создать шаблон класса для работы с бинарным деревом. Применить его для сортировки действительных чисел и строк, вводимых

склавиатуры или из файла.

9.Создать шаблон класса **Set** (множество) мощности *n* (см. задачу № 9, лаб. работа № 2) или создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 9 (лаб. работа № 3).

10.Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 10 (лаб. работа № 3).

11.Создать шаблон класса, обеспечивающего описание матрицы заданного размера *n m* и любого минора в ней (см. задачу № 11, лаб. работа № 2) или создать шаблон класса для работы с двусвязным списком. Применить шаблон класса для решения задачи № 11 (лаб. работа № 3).

12.Создать шаблон класса для работы с бинарным деревом. Применить шаблон класса для решения задачи № 12 (лаб. работа № 3).

13.Создать шаблон класса для работы с бинарным деревом. Применить шаблон класса для решения задачи № 13 (лаб. работа № 3).

14.Создать шаблон класса для работы с двусвязным списком. Применить шаблон класса для решения задачи № 15 (лаб. работа № 3).

15. Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 16 (лаб. работа № 3).

Лабораторная работа №5
Язык С++. Наследование.

При выполнении данной работы необходимо определить базовый класс и производные от него классы. Предусмотреть передачу аргументов конструкторам базового класса; использование виртуальных и перегруженных функций; обработку исключительных ситуаций.

Вариант А

В следующих заданиях требуется создать базовый класс (как вариант абстрактный базовый класс) и определить общие методы show ( ), get ( ), set ( ) и другие, специфические для данного класса. Создать производные классы, в которые добавить свойства и методы.

Часть методов переопределить. Создать массив объектов базового класса и заполнить объектами производных классов. Объекты производных классов идентифицировать конструктором по имени или идентификационному номеру.

Вызвать метод show ( ) базового класса и просмотреть массив объектов.

Использовать объекты для моделирования реальных ситуаций.

1.Создать базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

2.Создать базовый класс «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость перевозки для указанных городов и расстояний.

3.Создать аналогичный базовый класс «Пассажироперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость передвижения.

4.Изменить задания 1–3, чтобы базовый класс стал абстрактным. Сделать некоторые методы абстрактными.

5.Создать базовый класс «Учащийся» и производные классы «Школьник» и «Студент». Создать массив объектов базового класса и заполнить этот массив объектами. Показать отдельно студентов и школьников.

6.Создать базовый класс «Музыкальный инструмент» и производные классы «Ударный», «Струнный», «Духовой». Создать массив объектов «Оркестр». Выдать состав оркестра, переопределив метод.

7.Определить базовый класс «Множество» и производный класс «Кольцо» (операции сложения и умножения обе коммутативные и ассоциативные, связанные законом дистрибутивности; сложение обладает обратной операцией – вычитанием). Ввести кольца целых чисел, многочленов, систему классов целых чисел, сравнимых по модулю. Кольцо является полем, если в нем определена операция деления, кроме деления на нуль. Рациональные числа, дробно рациональные функции.

8.Создать абстрактный класс «Работник фирмы» и производные классы «Менеджер», «Администратор», «Программист».

9.Создать базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай» и др. С помощью конструктора установить имя каждого животного и его характеристики.

10.Создать базовый класс «Садовое дерево» и производные классы «Яблоня», «Вишня», «Груша» и др. С помощью конструктора автоматически установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

Вариант Б

1.Создать класс Item (единица хранения в библиотеке), содержащий данные-члены: invNumber – инвентарный номер и taken – взято на руки или имеется в наличии, а также методы:

virtual void Show(); //показать информацию о единице хранения bool isAvailable(); // есть ли единица хранения в наличии ?

int GetinvNumber(); //возвращает инвентарный номер void Take(); // операция «взять»

void Return(); // операция «вернуть»

Построить производные классы Book и Magazin. Класс Book со-

держит данные-члены: author, title, publisher, year и методы: Author(); Title(); Publisher(); YearOf Publishing(); Show().

Класс Magazin включает данные-члены: volume; number; year; title

иметоды: Volume(); Title(); Number(); Year(); Show().

2.Создать базовый класс Polygon (многоугольник). Класс должен содержать методы для рисования многоугольника, вычисления периметра, нахождения площади и др. Построить производный класс Triangle (треугольник), содержащий также методы для нахождения точки пересечения медиан, длин медиан, длин биссектрис, координат точек пересечения биссектрис, высот треугольника.

3.Создать абстрактный класс Shape для рисования плоских фигур. Построить производные классы Square (квадрат, который характеризуется координатами левого верхнего угла и длиной стороны), Circle (окружность с заданными координатами центра и радиусом), Ellipse (эллипс с заданными координатами вершин описанного вокруг него прямоугольника), позволяющие рисовать указанные фигуры, а также передвигать их на плоскости.

4.Создать класс CPoint – точка и производные от него классы

CcoloredPoint и CLine. На основе классов CcoloredPoint и CLine со-

здать класс CcoloredLine. Все классы должны иметь методы для установки и получения значений всех координат, а также изменения цвета

иполучения текущего цвета.

5.Описать базовый класс Stroka. Обязательные данные-члены класса: указатель типа char – для хранения строки; значение типа int – длина строки.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметра С-строку (заканчивается нулевым байтом); конструктор копирования; получение длины строки; очистка строки (сделать строку пустой); деструктор.

Описать производный класс «БИТОВАЯ_СТРОКА» (строки данного класса могут содержать только символы ‘0’ и ‘1’). Если в основе инициализирующей строки встретятся любые символы, отличные от допустимых, то БИТОВАЯ_СТРОКА становится пустой. Содержимое строки рассматривается как двоичное представление целого числа со знаковым разрядом. Отрицательные числа хранятся в дополнительном коде.

Обязательные методы: конструктор без параметров; конструктор, принимающий в качестве параметра С-строку; конструктор копирования; деструктор; изменение знака числа (перевод числа в дополнительный код).

Переопределить следующие операции (длина строки результата в случае необходимости расширяется влево знаковым разрядом): присваивание; сложение (+); проверка на равенство (==).

6.Создать производный класс «СТРОКА10» (целое неотрицательное десятичное число) от класса «СТРОКА» (описание приведено выше).

Методы: конструктор без параметров; конструктор, принимающий

вкачестве параметра С-строку; конструктор копирования; деструктор; метод, определяющий, можно ли представить данное число в формате int; метод, определяющий, равно ли число нулю; метод, возвращающий представление числа в виде целого (int); метод, удаляющий незначащие нули.

Переопределить операции: сложение (+); проверка на больше (по значению) (>); проверка на меньше (<); присваивание (=).

7.Создать производный класс «БУЛЕВ ВЕКТОР» (**BoolVector**) от класса **Vector**. Компоненты принимают значения из множества {0,1}.

Методы: конструктор без параметров; конструктор, принимающий

вкачестве параметров указатель на массив целого типа (если элементы массива содержат числа, отличные от 0 и 1, то создается пустой вектор) и размер вектора; конструктор копирования; деструктор; метод, возвращающий число единиц в векторе; метод, возвращающий позицию самой левой единицы в векторе.

Переопределить операции: поразрядная конъюнкция (&); поразрядная дизъюнкция (|); поразрядная инверсия (~); поразрядная операция ИСКЛЮЧАЮЩЕЕ ИЛИ (^); присваивание (=).

8.Создать производный класс «ТРОИЧНЫЙ ВЕКТОР» от класса **Vector**. Компоненты принимают значения из множества {0,1,2}.

Методы: конструктор без параметров; конструктор, принимающий

вкачестве параметров указатель на массив целого типа и размер вектора; конструктор копирования; деструктор; проверка двух векторов на ортогональность (два троичных вектора называются ортогональными, если в них существует пара одноименных компонент, имеющих

водном из векторов значение 0, а в другом – 1); метод, возвращающий число компонент в векторе, принимающих значение 2.

Переопределить операции: присваивание (=); поразрядная конъюнкция (пересечение) двух не ортогональных векторов (&): 0&0=0, 1&1=1, 2&2=2, 0&2=0, 2&0=0, 1&2=1, 2&1=1; индексирование ([]).

9.Создать производный класс «БУЛЕВА МАТРИЦА» от класса «ЦЕЛОЧИСЛЕННАЯ МАТРИЦА».

Методы: конструктор без параметров; конструктор, принимающий

вкачестве параметров целочисленный двумерный массив, содержащий матрицу, и ее размеры *n* и *m*; конструктор копирования; деструктор; метод возвращает число единиц в матрице; метод, возвращающий -каноническую матрицу (в исходной матрице удалены повторяющиеся строки; строки составлены в порядке возрастания неотрицательных чисел, в качестве двоичных кодов которых рассматриваются данные строки).

Переопределить операции: поэлементная конъюнкция двух матриц (&); поэлементная дизъюнкция двух матриц (|); поэлементная операция ИСКЛЮЧАЮЩЕЕ ИЛИ двух матриц (^); произведение двух матриц A=[aij] и B=[bjk], где i=1,*n*, j=1,*m*, k=1,*l* (*). При вычислениях операция целочисленного умножения заменяется конъюнкцией, а сложение – дизъюнкцией; присваивание (=).

10.Создать производный класс «МИНОР» для базового класса «МАТРИЦА» размерности *n m*. Переопределить для производного класса операции и методы (см. лаб. работу № 2, задание № 4).

11.Расширить возможности стандартного класса Time, чтобы можно было выводить время дня: утро, вечер и т. д.

12.Расширить возможности стандартного класса Date, чтобы можно было выводить время года: зима, лето и т. д.

13.Расширить возможности класса Annotation, чтобы можно было выводить время и дату изменения аннотации.

14.Расширить возможности класса Dictionary, чтобы можно было выводить дату последнего изменения в словаре.

15.Расширить возможности класса File, чтобы можно было выводить время и дату создания файла.

16.Расширить возможности класса Stack, чтобы можно было выводить время последнего сеанса работы со стеком.

17.Определить базовый класс для работы с прямоугольными матрицами, предусмотрев ввод-вывод матриц и выполнение следующих операций: сложение матриц; умножение матрицы на скаляр; перестановка строк матрицы по заданному вектору транспозиции; перестановка столбцов матрицы по заданному вектору транспозиции. В производном классе реализовать указанные операции для квадратных матриц, добавив выполнение следующих операций: транспонирование матрицы; умножение матриц.

Лабораторная работа № 6,7,8
Язык С++. Виртуальные функции. Язык С++. Потоки и файлы. Язык С++. Многофайловые программы.

При выполнении приводимых ниже заданий можно использовать классы, разработанные в лабораторных работах № 1–3. Осуществлять контроль состояния потоков. В случае возникновения ошибок потоков генерировать и обрабатывать исключительные ситуации. Для соответствующих классов перегрузить операции вставки в поток и извлечения из потока. При динамическом выделении памяти предусмотреть обработку исключения, возникающего при нехватке памяти.

а) Для класса **Student** (лаб. работа № 1) предусмотреть ввод данных из файла. Полученные при выполнении лаб. работы № 1 списки студентов вывести в файл.

То же задание для классов: б) **Abiturient** (лаб. работа №1); в) **Aeroflot** (лаб. работа № 1); г) **Worker** (лаб. работа № 1); д) **Train** (лаб. работа № 1); е) **Product** (лаб. работа № 1); ж) **Patient** (лаб. работа № 1); з) **Bus** (лаб. работа № 1); и) **Customer** (лаб. работа № 1); к) **File** (лаб. работа № 1); л) **Word** (лаб. работа № 1); м) **House** (лаб. работа № 1); н) **Phone** (лаб. работа № 1); о) **Person** (лаб. работа № 1).

а) При выполнении задания № 1 лаб. работы № 2 (класс **Complex**) предусмотреть формирование массива объектов путем считывания комплексных чисел из файла. Результат также вывести в файл.

То же задание для классов: б) **Fraction** (лаб. работа № 2); в) **Vector** (лаб. работа № 2). Предусмотреть обработку исключения при динамическом выделении памяти;) **Matrix** (лаб. работа № 2); д) **Polynom** (лаб. работа № 2); е) **Stack** (лаб. работа № 2); ж) **Строка** (лаб. работа № 2); з) **Set** (лаб. работа № 2);

и) **«Массив строк»** (зад. № 10 лаб. работы № 2); к) «**Булев вектор»** (лаб. работа № 2); л) «**Троичный вектор»** (лаб. работа № 2); м) «**Булева матрица»** (лаб. работа № 2).

Те же задания, что и в разделах I и II, но для классов, реализующих работу с динамическими структурами данных (см. лаб. работу № 3).

Лабораторная работа №9

Язык Java. Среда программирования. Основные конструкции. Объекты и классы.

Поскольку язык Java изначально является объектно-ориентированным, то любое написанное с его помощью приложение (даже самое простое) будет содержать класс. Для разработки приложений на языке Java требуется компилятор и виртуальная машина, а для удобства разработки можно использовать интегрированную среду разработки (IDE), но это не является обязательным.

Целью данной лабораторной работы будет создание простейшей программы, с написания которой начинается изучение практически любого языка программирования, а именно программы, выводящей на экран фразу «Hello, World!». В данном случае код будет выглядеть следующим образом:

```
public class Hello {

    public static void main(String args[]) {

      System.out.println(«Hello, World!»);

                                          }
}
```

Эта программа, как и любая другая на Java, представляет собой набор классов. В рассматриваемом случае это единственный класс Hello. Для того чтобы программу можно было запустить, в ней должен присутствовать класс, который содержит открытую (public) функцию main, имеющую прототип, как в примере. Эта функция должна быть статической, что нужно для ее вызова без создания экземпляра класса. Для вывода на экран в текстовом режиме (в консоль) используется фунция System.out.println(), которая позволяет выводить различные объекты: строки, символы,

числа и др.

Код необходимо сохранить в файле Hello.java. Применение именно такого имени файла является необходимым в соответствии с требованиями языка. Заметьте, что имя файла должно строго соответствовать имени объявленного в нем класса.

Для компиляции должен быть установленным JDK – (Java Development Kit) и/или одна из известных сред разработки (например, Eclipse).

Компиляция программы производится командой:

$ javac Hello.java

После компиляции будет создан файл Hello.class. А запуск программы производится командой:

$ java Hello

Рассмотрим пример преобразовывания градусы Фаренгейта в градусы Цельсия используя следующую формулу:

FahrenheitToCelsius.java

Источник | История

```
1  import java.util.Scanner;
2
3  public class FahrenheitToCelsius {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);
6
7          System.out.print("Введите градусы в фаренгейтах: ");
8          double fahrenheit = input.nextDouble();
9
10         // Конвертация фаренгейтов в Цельсии
11         double celsius = (5.0 / 9) * (fahrenheit - 32);
12         System.out.println("Фаренгейт " + fahrenheit + " это "
13                 + celsius + " в градусах Цельсия");
14     }
15 }
```

Рис. 1.Листинг программы

Результат работы программы представлен на рисунке 2.

Вывод - FahrenheitToCelsius (run)

```
run:
Введите градусы в фаренгейтах: 100
Фаренгейт 100.0 это 37.77777777777778 в градусах Цельсия
СБОРКА УСПЕШНО ЗАВЕРШЕНА (общее время: 2 секунды)
```

Рис. 2.Результат работы программы

Лабораторная работа №10,11
Язык Java. Наследование.
Язык Java. Интерфейсы и внутренние классы.

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Размер или длина массива — это общее количество элементов в массиве. Размер

массива задаётся при создании массива и не может быть изменён в дальнейшем, т. е. нельзя убрать элементы из массива или добавить их туда, но можно в существующие элементы присвоить новые значения.

Индекс начального элемента — 0, следующего за ним — 1 и т. д. Индекс последнего элемента в массиве — на единицу меньше, чем размер массива.

В Java массивы являются объектами. Это значит, что имя, которое даётся каждому массиву, лишь указывает на адрес какого-то фрагмента данных в памяти. Кроме адреса в этой переменной ничего не хранится. Индекс массива, фактически, указывает на то, насколько надо отступить от начального элемента массива в памяти, чтоб добраться до нужного элемента.

Чтобы создать массив надо объявить для него подходящее имя, а затем с этим именем связать нужный фрагмент памяти, где и будут друг за другом храниться значения элементов массива.

Рассмотрим программу поиска суммы всех положительных элементов массива.

```
1       public static void main(String args[]) {
2           Scanner s = new Scanner(System.in);
3           int numbers = Integer.valueOf(s.nextLine()).intValue();
4           String line = s.nextLine();
5           s.close();
6           String [] all = line.split("\\s");
7           int elements[] = new int [numbers];
8           for(int i = 0; i<numbers; i++)
9               elements[i] = Integer.valueOf(all[i]).intValue();
10
11          int count = 0;
12          for(int i = 0; i<numbers; i++)
13              if(elements[i] > 0)
14                  count++;
15          System.out.println(count);
16      }
```

Задания для самостоятельного выполнения

Вариант 1º. Написать программу, находящую минимальный элемент целочисленного массива *A* размера *N*. С помощью этой программы найти минимальные элементы массивов *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 2. Написать программу, находящую номер максимального элемента вещественного массива *A* размера *N*. С помощью этой программы найти номера максимальных элементов массивов *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 3. Написать программу, находящую номера минимального и максимального элемента вещественного массива *A* размера *N*. Выходные параметры целого типа: *NMin* (номер минимального элемента) и *NMax* (номер максимального элемента). С помощью этой программы найти номера минимальных и максимальных элементов массивов *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 4. Написать программу, меняющую порядок следования элементов вещественного массива *A* размера *N* на противоположный (*инвертирование* массива). Массив *A* является входным и выходным параметром. С помощью этой программы инвертировать массивы *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 5. Написать программу, выполняющую *сглаживание* вещественного массива *A* размера *N* следующим образом: элемент A_K заменяется на среднее арифметическое первых *K* исходных элементов массива *A*. Массив *A* является входным и выходным параметром.

Вариант 6. Написать программу, выполняющую *сглаживание* вещественного массива *A* размера *N* следующим образом: элемент A_1 не изменяется, элемент A_K ($K = 2, \ldots, N$) заменяется на полусумму исходных элементов A_{K-1} и A_K. Массив *A* является

входным и выходным параметром. С помощью этой программы выполнить пятикратное сглаживание данного массива *A* размера *N*, выводя результаты каждого сглаживания.

Вариант 7. Написать программу, выполняющую *сглаживание* вещественного массива *A* размера *N* следующим образом: каждый элемент массива заменяется на его среднее арифметическое с соседними элементами (при вычислении среднего ариф
A размера *N*, выводя результаты каждого сглаживания.

Вариант 8. Написать программу, удаляющую из целочисленного массива *A* размера *N* элементы, равные целому числу *X*. Массив *A* и число *N* являются входными и выходными параметрами. С помощью этой программы удалить числа X_A, X_B, X_C из массивов *A*, *B*, *C* размера N_A, N_B, N_C соответственно и вывести размер и содержимое полученных массивов.

Вариант 9. Написать программу, удаляющую из вещественного массива *A* размера *N* «лишние» элементы так, чтобы оставшиеся элементы оказались упорядоченными по возрастанию: первый элемент не удаляется, второй элемент удаляется, если он меньше первого, третий — если он меньше предыдущего элемента, оставленного в массиве, и т. д. Например, массив 5.5, 2.5, 4.6, 7.2, 5.8, 9.4 должен быть преобразован к виду 5.5, 7.2, 9.4. Массив *A* и число *N* являются входными и выходными параметрами. С помощью этой программы преобразовать массивы *A*, *B*, *C* размера N_A, N_B, N_C соответственно и вывести размер и содержимое полученных массивов.

Вариант 10. Написать программу, дублирующую в целочисленном массиве *A* размера *N* элементы, равные целому числу *X*. Массив *A* и число *N* являются входными и выходными параметрами. С помощью этой программы продублировать числа X_A, X_B, X_C в массивах *A*, *B*, *C* размера N_A, N_B, N_C соответственно и вывести размер и содержимое полученных массивов.

Вариант 11. Написать программу, выполняющую сортировку по возрастанию вещественного массива *A* размера *N*. Массив *A* является входным и выходным параметром. С помощью этой программы отсортировать массивы *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 12. Написать программу, формирующую для вещественного массива *A* размера *N* *индексный массив I* — массив целых чисел того же размера, содержащий номера элементов массива *A* в том порядке, который соответствует
возрастанию элементов массива *A* (сам массив *A* при этом не изменяется). Индексный массив *I* является выходным параметром. С помощью этой программы создать индексные массивы для массивов *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 13. Написать программу, меняющую порядок элементов вещественного массива *A* размера *N* на следующий: наименьший элемент массива располагается на первом месте, наименьший из оставшихся элементов — на последнем, следующий по величине располагается на втором месте, следующий — на предпоследнем и т. д. (в результате график значений элементов будет напоминать *колокол*). Массив *A* является входным и выходным параметром. С помощью этой программы преобразовать массивы *A*, *B*, *C* размера N_A, N_B, N_C соответственно.

Вариант 14. Написать программу, формирующую по вещественному массиву *A* размера N_A два вещественных массива *B* и *C* размера N_B и N_C соответственно; при этом массив *B* содержит все элементы массива *A* с нечетными порядковыми номерами (1, 3, …), а массив *C* — все элементы массива *A* с четными номерами (2, 4, …). Массивы *B* и *C* и числа N_B и N_C являются выходными параметрами. Применить эту программу к данному массиву *A* размера N_A и вывести размер и содержимое полученных массивов *B* и *C*.

Вариант 15. Написать программу, формирующую по целочисленному массиву *A* размера N_A два целочисленных массива *B* и *C* размера N_B и N_C соответственно; при этом массив *B* содержит все четные числа из массива *A*, а массив *C* — все нечетные числа (в том же порядке). Массивы *B* и *C* и числа N_B и N_C являются выходными параметрами.

Применить эту программу к данному массиву A размера N_A и вывести размер и содержимое полученных массивов B и C.

Лабораторная работа №12,13,14,15

Язык Java. Программирование графики.Язык Java. Обработка событий.Язык Java. Компоненты пользовательского интерфейса из пакета Swing.Язык Java. Аплеты.

Упражнение 1. Создание нового Java приложения

Чтобы начать работу с пакетом Eclipse необходимо выполнить в командной строке d:\eclipse\eclipse.exe. После чего перед вами появится главное окно программы (Рис. 1).

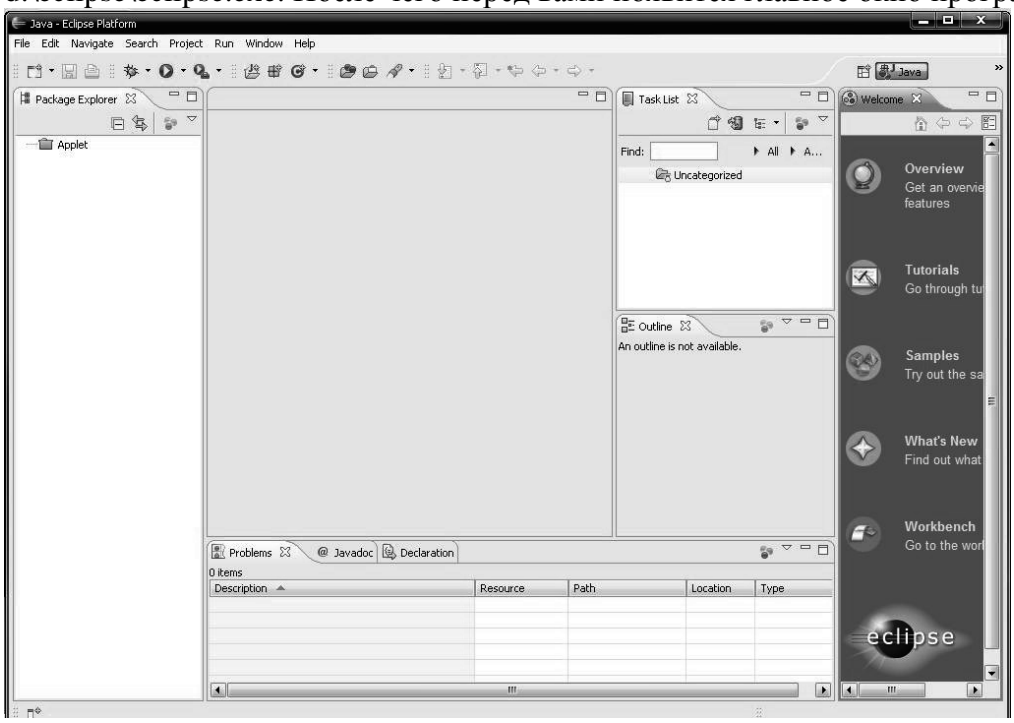


Рис. 1. Интерфейс Eclipse

Меню File->New позволяет запустить конструктор создания проекта, либо компонента.

Чтобы создать новое Java приложение выберите пункт меню File->New->Project... В появившемся списке выберите «Java Project».

Затем нажмите «Next» (Рис. 2).

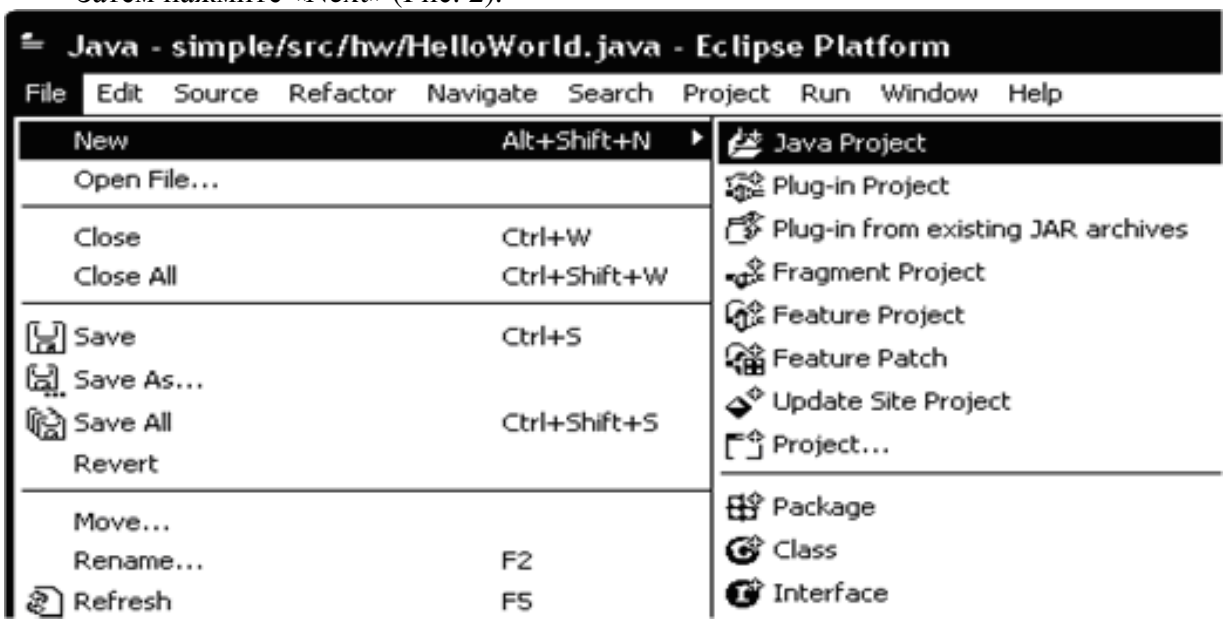


Рис. 2. Меню создания нового проекта

Появится окно конструктора нового проекта. В строке Project name введите название нового проекта (например, Simple) и нажмите «Next» (Рис. 3). Здесь же можно указать место для хранения исходных и скомпилированных файлов, задать любые подпроекты и библиотеки, которые могут понадобиться для работы и компоновки текущего проекта, а также выбрать среду выполнения кода в Java Runtime Environment.

Java Runtime Environment (JRE) — минимальная реализация виртуальной машины Java, необходимая для исполнения Java- приложений без компилятора и других средств разработки. Состоит из виртуальной машины — Java Virtual Machine и библиотеки Java-классов.

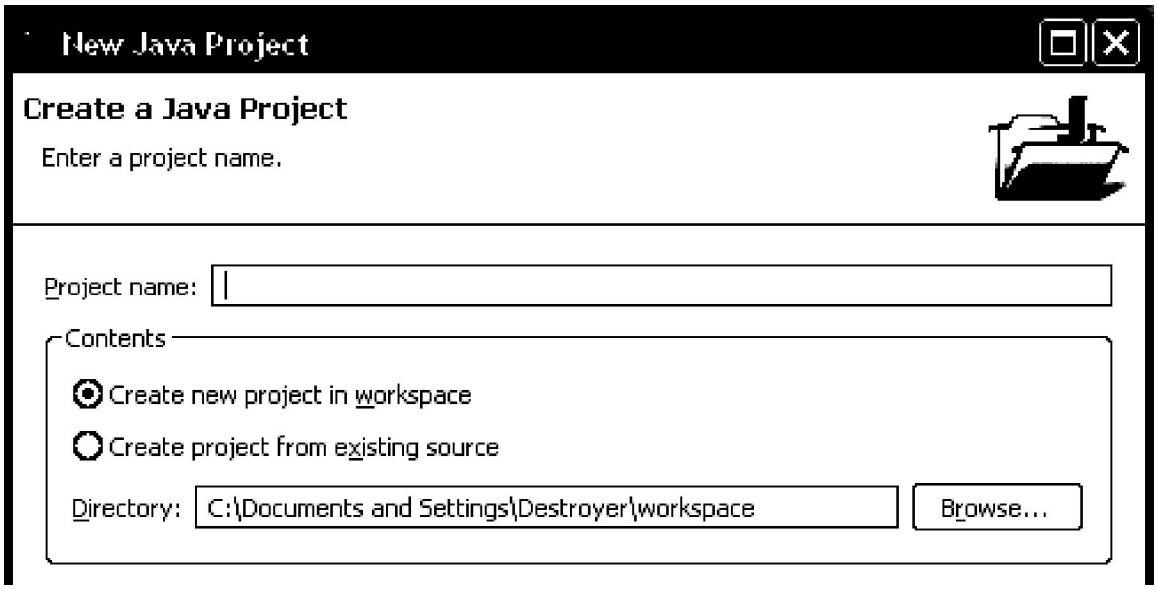


Рис. 3. Окно конструктора проекта

Нажмите «Finish».

Упражнение 2. Создание нового пакета

В созданном проекте щелкните правой кнопкой мыши на папке src. В появившемся меню выберете New -> Package и задайте имя нового пакета. Имена папок и пакетов пишутся малыми буквами (Рис. 4).

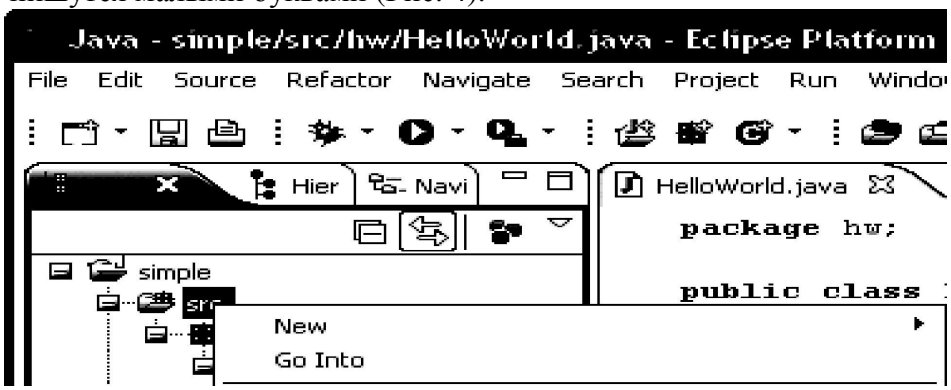


Рис. 4. Меню папки

1.1. Упражнение 3. Создание нового класса

Чтобы создать новый класс в созданном выше пакете выберите пункт меню папки New->Class. Запустится конструктор создания класса (Рис. 5).

В поле Name введите имя класса. Имя класса начинается с заглавной буквы (например, HelloWorld). Затем в строке Which method stubs would you like to create? выбираем шаблон метода для создания класса. Ставим галочку на public static void man (String[] args), иные галочки снимаются. Нажимаем кнопку
«Finish», класс создан.

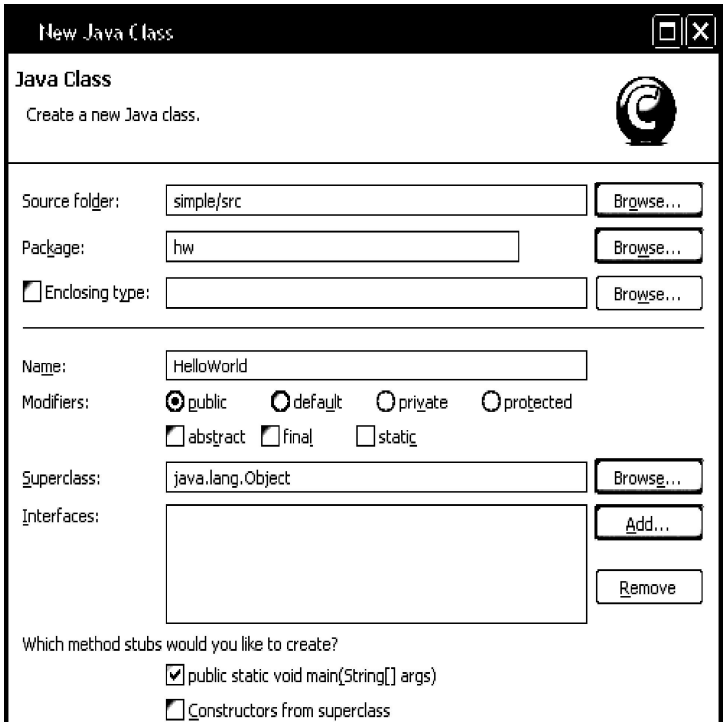


Рис. 5. Окно конструктора создания класса

Упражнение 4. Работа с телом программы

В результате генерации класса сгенерируется тело программы:

```
package hw;
public class HelloWorld {
        /**
           * @param args
           */
          public static void main(String[] args)
          {
                  // TODO Auto-generated method stub
          }
}
```

Для ее выполнения необходимо добавить функцию вывода на экран (или командную строку)

System.*out*.println("Hello World!!!").

Таким образом, программа примет вид (Рис.6).

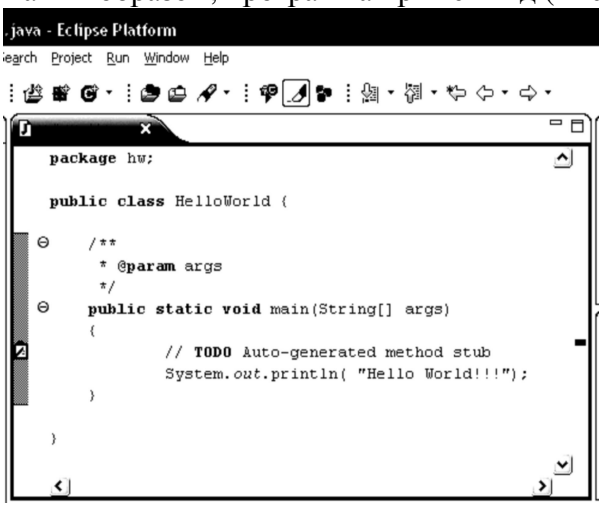


Рис. 6. Рабочее пространство среды Eclipse (поле Edit)

Для выполнения программы выберете меню Run-> Run или нажмите «Ctrl+F11». В консоли Eclipse отобразится выполненная программа.

Упражнение 5. Создание апплета с помощью IDE Eclipse Прежде чем прейти к созданию апплета, необходимо создать новое Java - приложение. Для создания нового проекта воспользуемся описанным ранее способом. Выберите пункт меню File->New->Project... В появившемся списке выберите Java Project. Затем нажмите кнопку «Next». (Рис. 7).

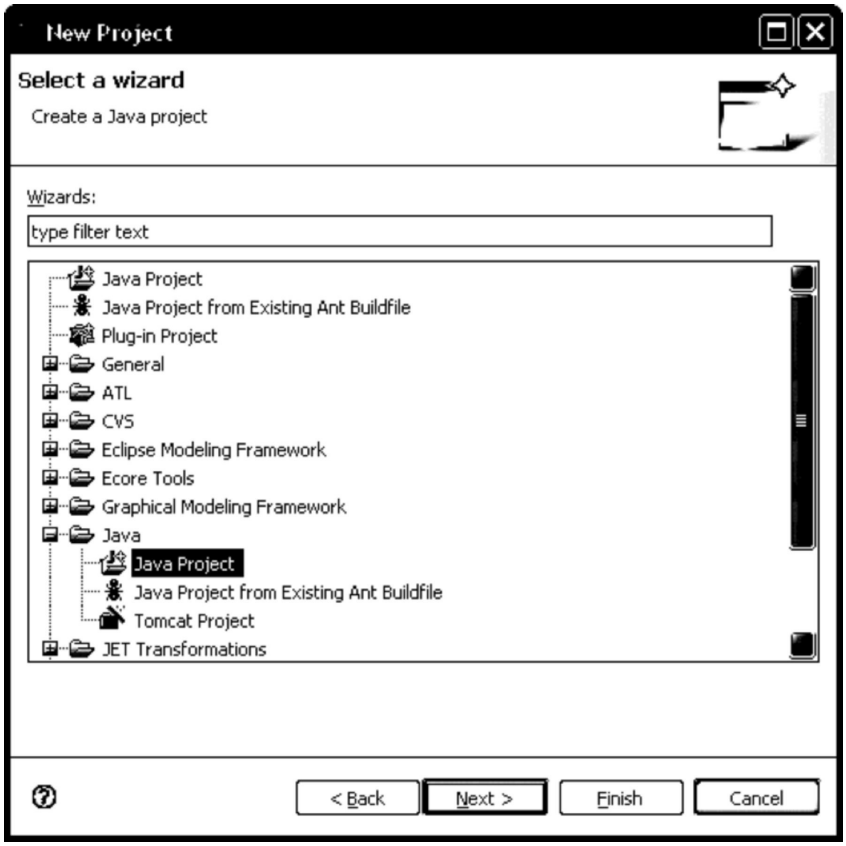


*Рис. 7.*Меню создания нового проекта

В появившемся окне конструктора проекта в строке Project name введите название нового проекта (например, Applet) и нажмите «Next» и затем нажмите «Finish» (Рис. 8).

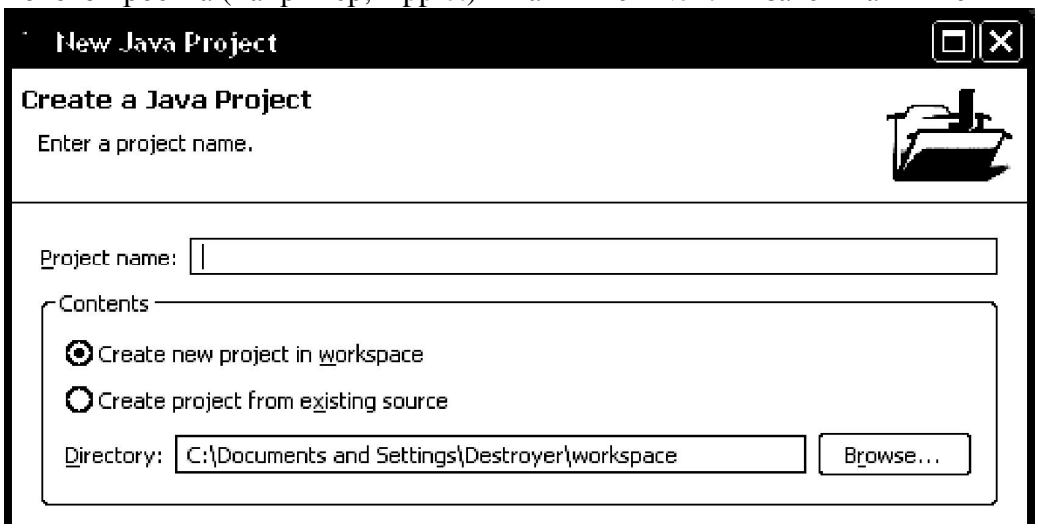


Рис. 8. Окно конструктора проекта

В созданном проекте щелкните правой кнопкой мыши на папке src. В появившемся меню выберете New > Package и задайте имя нового пакета (Рис. 9).

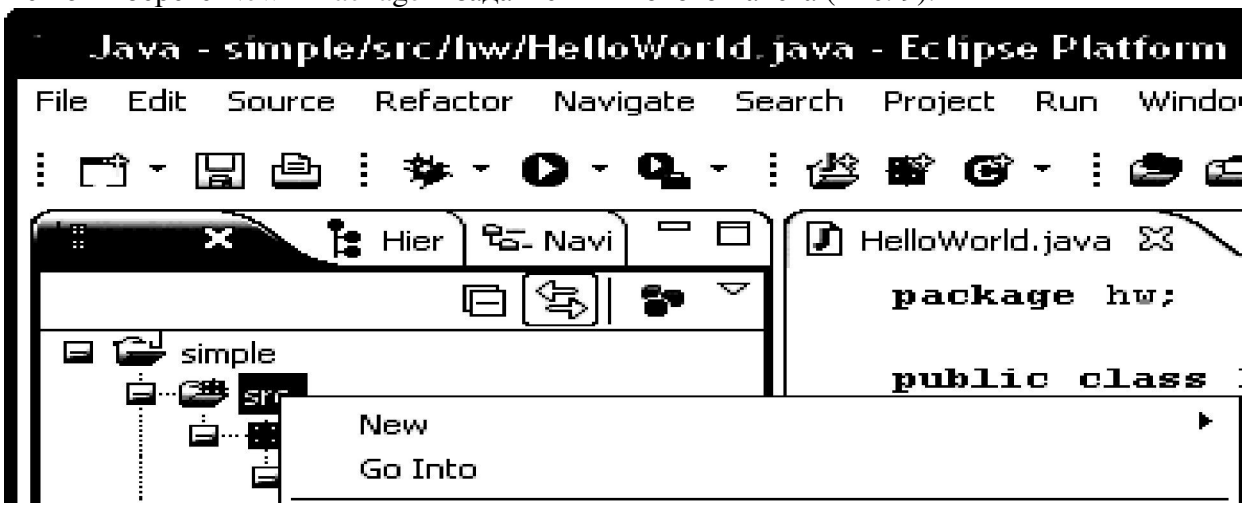


Рис. 9. Всплывающее меню папки

Теперь необходимо создать новый класс апплета. Выберете пункт меню папки New -> Applet (Рис. 10), после чего запустится мастер создания класса. В поле Name введите имя апплета (класса), имя класса начинается с заглавной буквы (например, HelloWorld!). Нажимаем кнопку «Finish», класс создан.

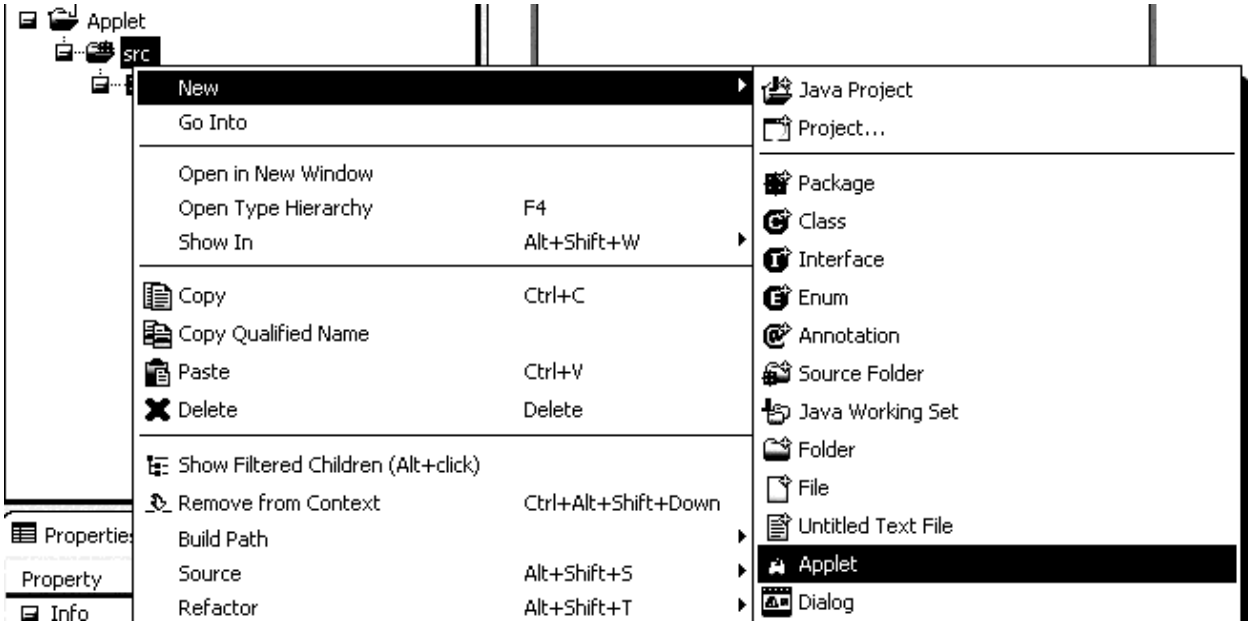


Рис. 10. Меню создания класса

В результате выполненных действий в рабочей области отобразится предварительный шаблон апплета (визуальная форма), с правой стороны появится палитра элементов (Рис. 11).

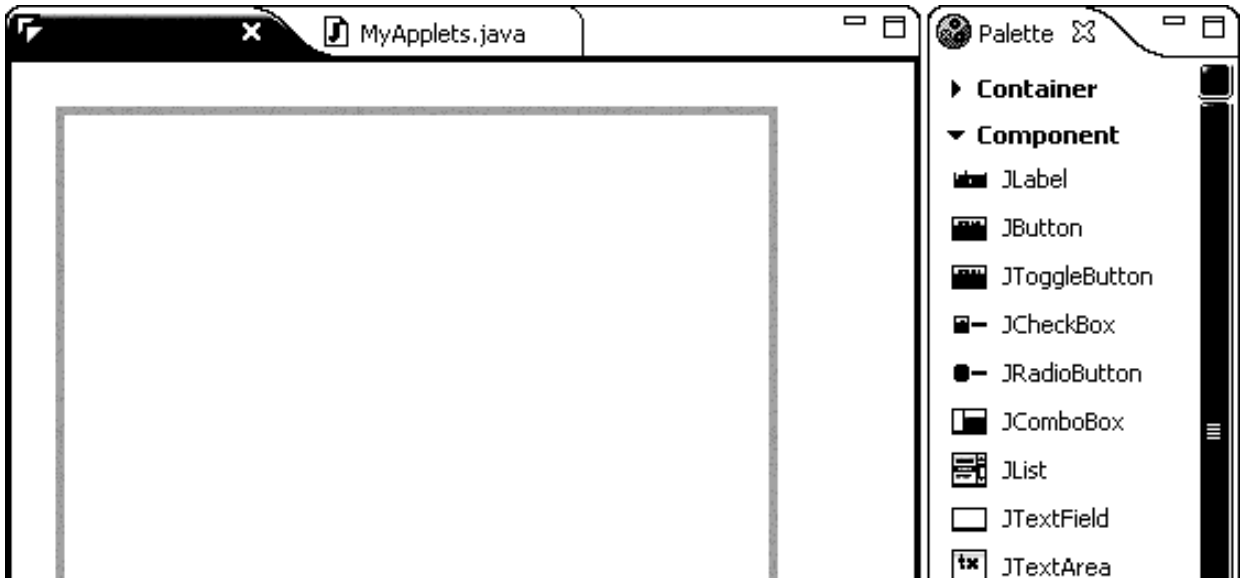


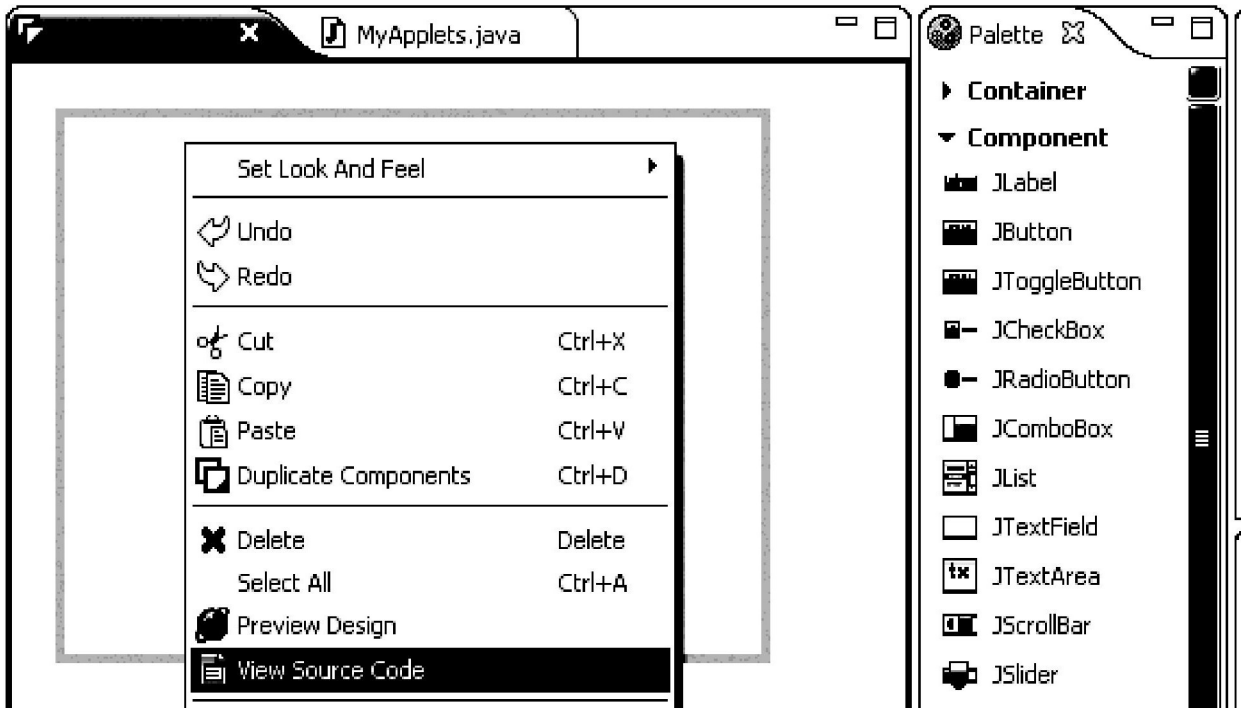
Рис. 11. Рабочая область

1.2. Упражнение 6. Работа с исходным кодом апплета

Для получения исходного кода апплета необходимо в рабочей области правой кнопкой мыши вызвать всплывающее меню и выбрать пункт «View Source Code» (Рис.12).

Рис. 12. Просмотр исходного кода паалета Появится

исходный код созданного апплета:




```
import java.awt.EventQueue; import
javax.swing.JApplet;
//VS4E -- DO NOT REMOVE THIS LINE!
public class MyAplets extends JApplet {
```

```
        private static final long serialVersionUID    =
1L;
        public void init() { try {
                        EventQueue.invokeAndWait(new
Runnable() {
                                @Override
                                public void run() { initComponents();
                                }
                        });
                } catch (Exception ex) { ex.printStackTrace();
                }
        }
        private void initComponents() { setSize(320, 240);
        } }
```

Определенный нами класс MyApplets с помощью ключевого слова extends наследуется от класса JApplet. При этом методам класса HelloApplet становятся доступными все методы и данные класса, за исключением определенных как private. Класс JApplet определен в библиотеке классов javax.swing.JApplet, которую мы подключили оператором import.

В палитре элементов выберите компонент JLabel и перенесите в форму апплета (Рис. 13).

Рис. 13. Перенос компонента JLabel в форму апплета

В левом нижнем окне Properties отобразятся свойства данного компонента (Рис. 14).

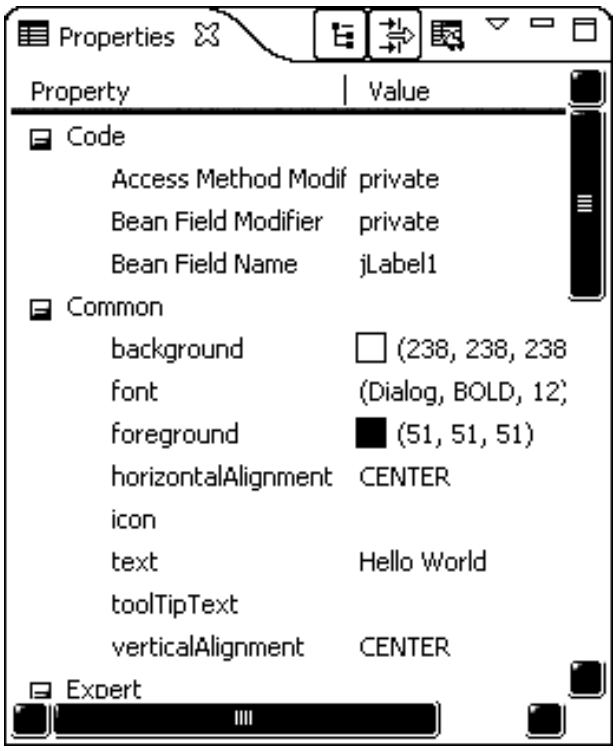


Рис. 14. Окно свойств компонента JLabel

Во вкладке Common выберите поля horizontalAligment и verticalAligment и в сплывающем меню установите CENTER, в поле text введите Hello World.

Изменившийся текст апплета выглядит следующим образом:

```
import java.awt.BorderLayout; import
java.awt.EventQueue; import javax.swing.JApplet; import
javax.swing.JLabel;
import javax.swing.SwingConstants;

//VS4E -- DO NOT REMOVE THIS LINE!
public class MyAplets extends JApplet {

        private static final long serialVersionUID    =
;
private JLabel jLabel1; public void init() {
        try {
                EventQueue.invokeAndWait(new Runnable() {
                        @Override
                        public void run() { initComponents();
                        }
                });
        } catch (Exception ex) { ex.printStackTrace();
        }
}

private void initComponents() { add(getJLabel1(), BorderLayout.CENTER); setSize(320,
        240);
}

private JLabel getJLabel1() { if (jLabel1 == null) {
                jLabel1 = new JLabel();
```

```
        jLabel1.setHorizontalAlignment(SwingConstants
.CENTER);
                        jLabel1.setText("Hello World");
                }
                return jLabel1;
        }

        public MyAplets() { initComponents();
        }
}
```

Для выполнения программы выберете меню Run-> Run или нажмите «Ctrl+F11».

Для запуска апплета с помощью браузера исходный текст HTML страницы должен содержать следующие строки:

```
<applet code="MyApplets" width=320 height=240>
</applet>
```

Тег <applet> используется для запуска апплета как из HTML- документа, так и из программы appletviewer. Программа appletviewer выполняет каждый найденный ей тег <applet> в отдельном окне, в то время как браузеры позволяют разместить на одной странице несколько апплетов. Синтаксис тэга <APPLET> в настоящее время таков:

```
<APPLET
CODE = appletFile OBJECT =
appletSerialFile WIDTH = pixels
HEIGHT = pixels [ARCHIVE =
jarFiles]
[CODEBASE = codebaseURL] [ALT =
alternateText]
[NAME = appletInstanceName] [ALIGN =
alignment] [VSPACE = pixels]
[HSPACE = pixels]
[< PARAM NAME = AttributeNamel VALUE = AttributeValuel >] [< PARAM NAME =
AttributeName2 VALUE = AttributeValue2 >] [HTML-текст, отображаемый при отсутствии
поддержки Java]
</APPLET>
```

В HTML странице содержится следующий код:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1- transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<title>Fierst Applet</title>
</head>

<body>

<applet code= MyApplets.class width=320
height=240> </applet>

</body>
```

</html>

аборатоpная работа №16,17
Язык Java. Исключения и отладка. Язык Java. Потоки и файлы.

Для работы с потоком ввода необходимо создать объект класса Scanner, при создании указав, с каким потоком ввода он будет связан.

```
import java.util.Scanner; // импортируем класс
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // создаём объект класса Scanner
        int i = 2;
        System.out.print("Введите целое число: ");
        if(sc.hasNextInt()) { // возвращает истинну если с потока ввода
можно считать целое число
          i = sc.nextInt(); // считывает целое число с потока ввода и сохраняем
в переменную
          System.out.println(i*2);
        } else {
          System.out.println("Вы ввели не целое число");
        }
```

Метод hasNextDouble(), применённый объекту класса Scanner, проверяет, можно ли считать с потока ввода вещественное число типа double, а метод nextDouble() — считывает его. Если попытаться считать значение без предварительной проверки, то во время исполнения программы можно получить ошибку (отладчик заранее такую ошибку не обнаружит). Например, попробуйте в представленной далее программе ввести какое-то вещественное число:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

Имеется также метод nextLine(), позволяющий считывать целую последовательность символов, т.е. строку, а, значит, полученное через этот метод значение нужно сохранять в объекте класса String. В следующем примере создаётся два таких объекта, потом в них

поочерёдно записывается ввод пользователя, а далее на экран выводится одна строка, полученная объединением введённых последовательностей символов.

```
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      String s1, s2;

      s1 = sc.nextLine();

      s2 = sc.nextLine();

      System.out.println(s1 + s2);

   }
```

Существует и метод hasNext(), проверяющий остались ли в потоке ввода какие-то символы.

В классе String существует масса полезных методов, которые можно применять к строкам (перед именем метода будем указывать тип того значения, которое он возвращает):

1. int length() — возвращает длину строки (количество символов в ней);
2. boolean isEmpty() — проверяет, пустая ли строка;
3. String replace(a, b) — возвращает строку, где символ a (литерал или переменная типа char) заменён на символ b;
   String toLowerCase() — возвращает строку, где все символы исходной строки преобразованы к строчным;
4. String toUpperCase() — возвращает строку, где все символы исходной строки преобразованы к прописным;
5. boolean equals(s) — возвращает истинну, если строка к которой применён метод, совпадает со строкой s указанной в аргументе метода (с помощью оператора == строки сравнивать нельзя, как и любые другие объекты);
6. int indexOf(ch) — возвращает индекс символа ch в строке (индекс это порядковый номер символа, но нумероваться символы начинают с нуля). Если символ совсем не будет найден, то возвратит -1. Если символ встречается в строке несколько раз, то вовзвратит индекс его первого вхождения.
7. int lastIndexOf(ch) — аналогичен предыдущему методу, но возвращает индекс последнего вхождения, если смивол встретился в строке несколько раз.
8. int indexOf(ch,n) — возвращает индекс символа ch в строке, но начинает проверку с индекса n (индекс это порядковый номер символа, но нумероваться символы начинают с нуля).
9. char charAt(n) — возвращает код символа, находящегося в строке под индексом n (индекс это порядковый номер символа, но нумероваться символы начинают с нуля).

```
public class Main {

   public static void main(String[] args) { String

      s1 = "firefox";

      System.out.println(s1.toUpperCase()); // выведет «FIREFOX» String
```

```
System.out.println(i); // выведет
7 i = s1.indexOf('f');
System.out.println(i); // выведет
0 i = s1.indexOf('r');
System.out.println(i); // выведет
2 i = s1.lastIndexOf('f');
System.out.println(i); // выведет
4 i = s1.indexOf('t');
System.out.println(i); // выведет -
1 i = s1.indexOf('r',3);
System.out.println(i); // выведет -
1
```

Пример программы, которая выведет на экран индексы всех пробелов в строке, введенное пользователем с клавиатуры:

```
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();

        for(int i=0; i < s.length(); i++) {
            if(s.charAt(i) == ' ') {

                System.out.println(i);
            }
        }
    }
```

Задания для самостоятельного выполнения

Вариант 1. Написать программу, проверяющую, является ли строка *S* допустимым *идентификатором*, то есть непустой строкой, которая содержит только латинские буквы, цифры и символ подчеркивания «_» и не начинается с цифры. Если *S* является допустимым идентификатором, то функция возвращает 0. Если *S* является пустой строкой, то возвращается –1, если *S* начинается с цифры, то возвращается –2. Если *S* содержит недопустимые символы, то возвращается номер первого недопустимого символа.

Вариант 2. Написать программу, возвращающую строку длины *N*, заполненную повторяющимися копиями строки-*шаблона S* (последняя копия строки-шаблона может входить в результирующую строку частично).

Вариант 3. Написать программу,, преобразующую все строчные русские буквы строки *S* в прописные (остальные символы строки *S* не изменяются). Строка *S* является входным и выходным параметром.

Вариант 4. Написать программу, преобразующую все прописные русские буквы строки *S* в строчные (остальные символы строки *S* не изменяются). Строка S является входным и выходным параметром.

Вариант 5. Написать программу, удаляющую в строке S начальные символы, совпадающие с символом C. Строка S является входным и выходным параметром. Дан символ C и пять строк.

Вариант 6. Написать программу, удаляющую в строке S конечные символы, совпадающие с символом C. Строка S является входным и выходным параметром. Дан символ C и пять строк.

Вариант 7. Написать программу, возвращающую инвертированную подстроку строки S, содержащую в обратном порядке N символов строки S, начиная с ее K-го символа. Если K превосходит длину строки S, то возвращается пустая строка; если длина строки меньше K + N, то инвертируются все символы строки, начиная с ее K-го символа.

Вариант 8. Написать программу, возвращающую номер позиции, начиная с которой в строке S содержится первое вхождение строки S_0, причем анализируются только N символов строки S, начиная с ее K-го символа (таким образом, PosSub обеспечивает поиск в подстроке). Если K превосходит длину строки S, то возвращается 0, если длина строки меньше K + N, то анализируются все символы строки, начиная с ее K-го символа. Если в требуемой подстроке строки S вхождения S_0 отсутствуют, то функция возвращает 0.

Вариант 9. Написать программу, возвращающую номер позиции, начиная с которой в строке S содержится последнее вхождение подстроки S_0. Считать, что перекрывающихся вхождений подстрок S_0 строка S не содержит.

Вариант 10. Написать программу, возвращающую номер позиции, начиная с которой в строке *S* содержится *K*-е вхождение подстроки S_0 ($K > 0$). Если количество вхождений S_0 в строке *S* меньше *K*, то функция возвращает 0. Считать, что перекрывающихся вхождений подстрок S_0 строка *S* не содержит.

Вариант 11. Написать программу, возвращающую K-е слово строки S (словом считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки).

Вариант 12. Написать программу, которая формирует по данной строке S массив W слов, входящих в S (массив W и его размер N являются выходными параметрами). Словом считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки; предполагается, что строка S содержит не более 10 слов.

Вариант 13. Написать программу, выполняющую сжатие строки S по следующему правилу: каждая подстрока строки S, состоящая из более чем четырех одинаковых символов C, заменяется текстом вида «C{K}», где K — количество символов C (предполагается, что строка S не содержит фигурных скобок «{» и «}»). Например, для строки S = «bbbccccce» программа вернет строку «bbbc{5}e».

Вариант 14. Написать программу, возвращающую строковое представление целого неотрицательного числа N в 16-ричной системе счисления. Результирующая строка состоит из символов «0»–«9», «A»–«F» и не содержит ведущих нулей (за исключением представления числа 0). Используя эту программу, получить 16-ричные представления пяти данных чисел.

Вариант 15. Написать программу, определяющую целое неотрицательное число по его строковому представлению *S* в 16-ричной системе счисления. Параметр *S* имеет строковый тип, состоит из символов «0»–«9», «A»–«F» и не содержит ведущих нулей (за исключением значения «0»). Используя эту программу, вывести пять чисел, для которых даны их 16-ричные представления.

Рекомендуемая литература

Основная литература

1. Лафоре, Р. Объектно-ориентированное программирование в C++ [Текст] / Р. Лафоре.- 4-е изд.. - Санкт-Петербург : Питер, 2014. - 928 с. : ил. - (Классика Komputer Science) - ISBN 978-5-496-00353-7. (20)

2. Васильев, А. Н. Java. Объектно-ориентированное программирование [Текст] : для магистров и бакалавров. Базовый курс по объектно-ориентированному программированию / А. Н. Васильев. - Санкт-Петербург : Питер, 2014. - 400 с. - (Учебное пособие) - ISBN 978-5-496-00044-4. (15)

Дополнительная литература

1. Павловская, Т.А. C++. Объектно-ориентированное программирование: практикум / Павловская, Т.А. . - СПб. : Питер, 2006. - 265с. : ил. (5)

2. Кирнос, В.Н. Информатика II. Основы алгоритмизации и программирования на языке C++ : учебно-методическое пособие / В.Н. Кирнос ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2013. - 160 с. : ил.,табл., схем. - ISBN 978-5-4332-0068-5 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=208651.

Периодические издания

1. Журнал «Вестник компьютерных и информационных технологий »
2. Журнал «Информационные технологии и вычислительные системы»
3. Журнал «Стандарты и качество»
4. Журнал «Прикладная информатика»