Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «17» ноября 2022 г. по «07» декабря 2022 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 19-1  Петров Данил Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2022

Содержание

|  |  |
| --- | --- |
| Введение | 3 |
| 1 Решение задач | 4 |
| 1.1 Задача 1 | 4 |
| 1.2 Задача 2 | 6 |
| 1.3 Задача 3 | 9 |
| 1.4 Задача 4 | 15 |
| 1.5 Задача 5 | 17 |
| 1.6 Задача 6 | 18 |
| 1.7 Задача 7 | 24 |
| 1.8 Задача 8 | 26 |
| 1.9 Задача 9 | 30 |
| 1.10 Соединение задач | 36 |
| 1.11 Задача 10 | 38 |
| 2 Разработка АИС «Ресторан» | 40 |
| 2.1 ER-диаграмма, диаграмма классов, диаграмма… |  |
| 2.2 Техническое задание |  |
| 2.3 Разработка структурной и функциональной схемы АИС |  |
| 2.4 Проектирование и реализация интерфейса |  |
| 2.5 Разработка встроенной системы справочного руководства |  |
| 2.6 Проведение тестирования |  |
| 3 Разработка мобильного приложения |  |
|  |  |
| 4 Разработка сайта |  |
|  |  |
| 5 Руководство пользователя |  |
|  |  |
| Заключение |  |
|  |  |
| Список использованных источников |  |
|  |  |
| Приложение |  |

Введение

Задание 1

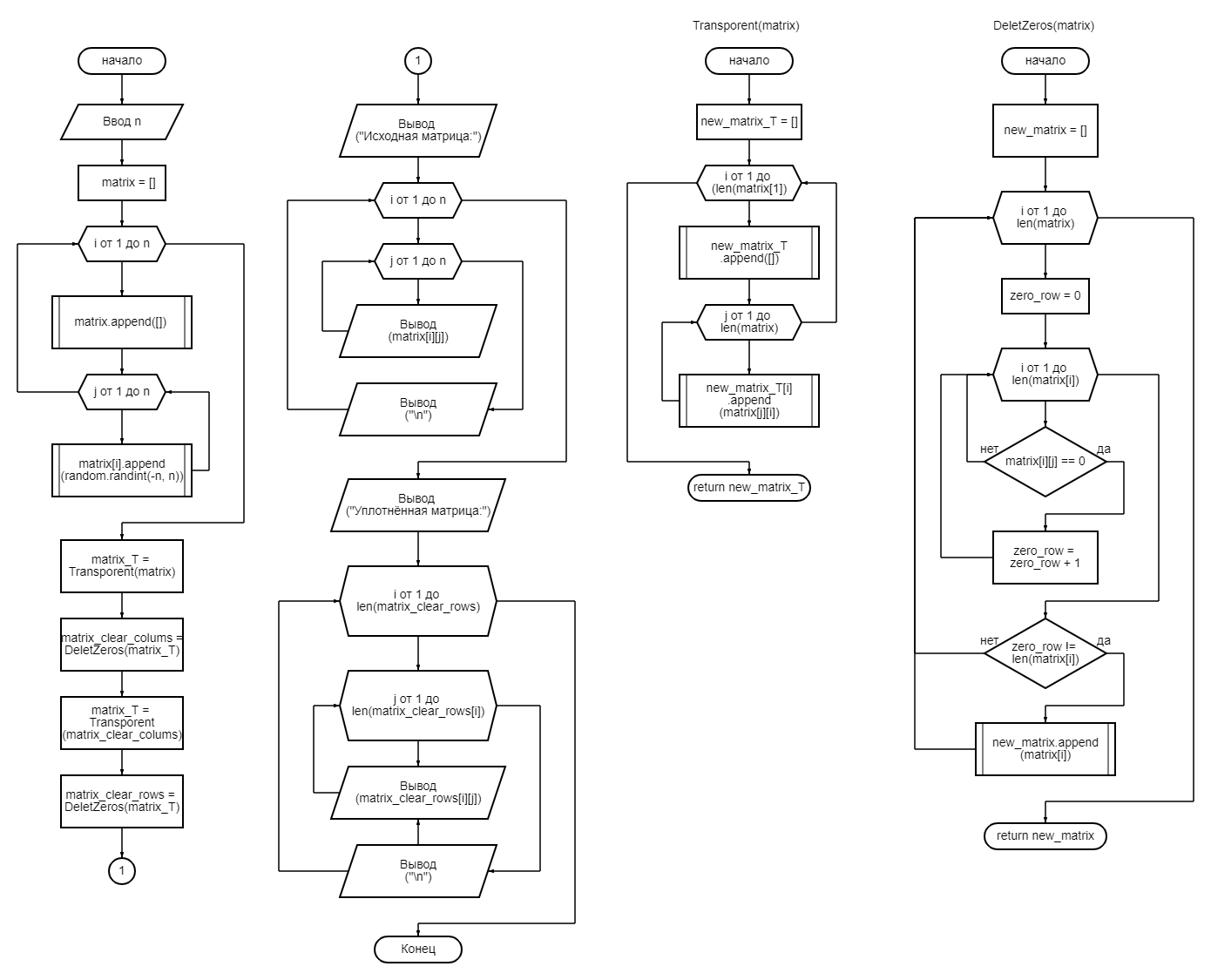
Ввести с консоли n-размерность матрицы a [n] [n]. Задать значения элементов матрицы в интервале значений от -n до n с помощью датчика случайных чисел

11. Уплотнить матрицу, удаляя из нее строки и столбцы, заполненные нулями.

Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| new\_matrix\_T | array | Вспомогательная переменная в функции Transporent для трансформации исходного массива. Транспонирует матрицу. |
| new\_matrix | array | Вспомогательная переменная в функции DeletZeros для трансформации исходного массива. Сохраняет матрицу без нулевых строк. |
| zero\_row | Int | Вспомогательная переменная в функции DeletZeros. Подсчитывает количество нулей в строке матрицы. |
| n | int | Вспомогательная вводимая переменная для задания размерности матрицы. |
| matrix | array | Выводимая переменная хранящая исходную матрицу. |
| matrix\_T | array | Вспомогательная переменная хранящая транспонированную матрицу. |
| matrix\_clear\_colums | array | Выводимая переменная хранящая уплотнённую матрицу. |
| e | Exception | Вспомогательная переменная, хранящая ошибку. Создаётся при неверном выполнении программы. |
| i | Int | Вспомогательная переменная для циклического перебора массива |
| j | int | Вспомогательная переменная для глубокого циклического перебора массива |

Блок схема



Код программы

import random

def Transporent(matrix):

new\_matrix\_T = []

for i in range(len(matrix[1])):

new\_matrix\_T.append([])

for j in range(len(matrix)):

new\_matrix\_T[i].append(matrix[j][i])

return new\_matrix\_T

def DeletZeros(matrix):

new\_matrix = []

for i in range(len(matrix)):

zero\_row = 0

for j in range(len(matrix[i])):

if(matrix[i][j] == 0):

zero\_row = zero\_row + 1

if (zero\_row != len(matrix[i])):

new\_matrix.append(matrix[i])

return new\_matrix

try:

n = int(input("Введите размер матрицы: "))

matrix = []

for i in range(n):

matrix.append([])

for j in range(n):

matrix[i].append(random.randint(-n, n))

matrix\_T = Transporent(matrix)

matrix\_clear\_colums = DeletZeros(matrix\_T)

matrix\_T = Transporent(matrix\_clear\_colums)

matrix\_clear\_rows = DeletZeros(matrix\_T)

print("Исходная матрица:")

for i in range(n):

for j in range(n):

print(matrix[i][j], end=' ')

print("\n")

print("Уплотнённая матрица:")

for i in range(len(matrix\_clear\_rows)):

for j in range(len(matrix\_clear\_rows[i])):

print(matrix\_clear\_rows[i][j], end=' ')

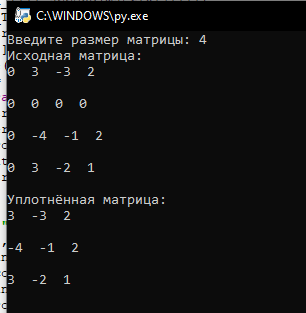
print("\n")

except Exception as e:

print("Некорректная матрица")

input()

Скриншот результата



Ссылка на онлайн компилятор

[main.py - Task 1 - Replit](https://replit.com/@PretrovDS/Task-1#main.py)

Задание 2

3. Определить класс Квадратное уравнение. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив/список/множество объектов и определить наибольшие и наименьшие по значению корни.

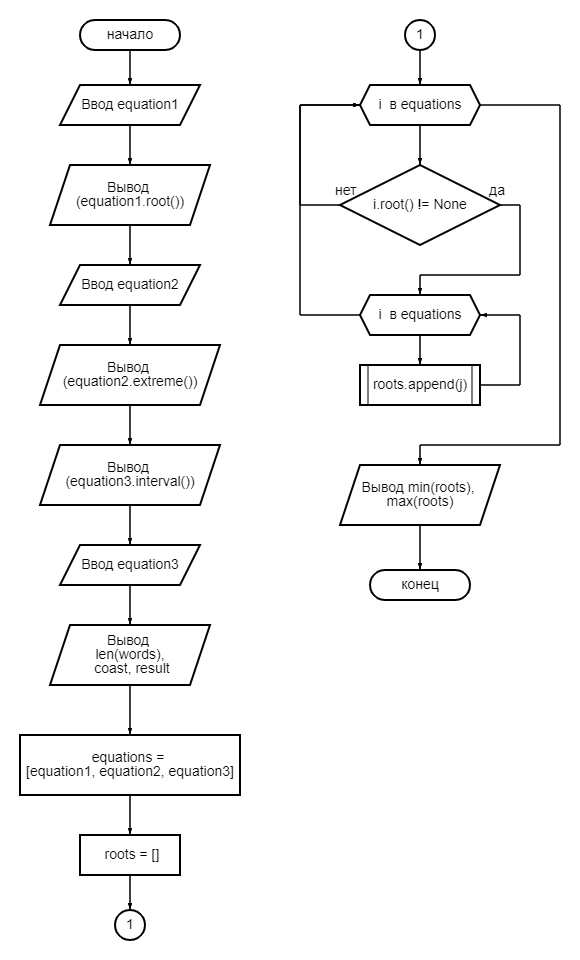
Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| equation1 | Equation | Объект класса Equation. Отвечает за демонстрацию работы программы. |
| equation2 | Equation | Объект класса Equation. Отвечает за демонстрацию работы программы. |
| equation2 | Equation | Объект класса Equation. Отвечает за демонстрацию работы программы. |
| equations | array | Массив класса Equation. Отвечает за демонстрацию работы программы. |
| roots | array | Массив результатов вычисления корней квадратных уравнений из массива equations. |
| i | int | Вспомогательная переменная для циклического перебора массива. |
| j | int | Вспомогательная переменная для глубокого циклического перебора массива. |
| a | int | Переменная класса Equation. Отвечает за хранение коэффициента a . |
| b | int | Переменная класса Equation. Отвечает за хранение коэффициента b. |
| c | int | Переменная класса Equation. Отвечает за хранение коэффициента c. |
| x1 | float | Переменная класса Equation. Отвечает за хранение первого корня квадратного уравнения. |
| x2 | float | Переменная класса Equation. Отвечает за хранение второго корня квадратного уравнения. |
| x | float | Переменная класса Equation. Отвечает за хранение абсциссы |
| y | float | Переменная класса Equation. Отвечает за хранение ординаты |

UML диаграмма класса Equation

|  |
| --- |
| Equation |
| a  b  c |
| \_\_init\_\_(self, a, b, c)  root(self)  extreme(self)  interval(self) |

Блок схема



Код программы

import math

class Equation(object):

def \_\_init\_\_(self, a, b, c):

self.a = a

self.b = b

self.c = c

def root(self):

discr = self.b \*\* 2 - 4 \* self.a \* self.c

if discr > 0:

x1 = (-self.b + math.sqrt(discr)) / (2 \* self.a)

x2 = (-self.b - math.sqrt(discr)) / (2 \* self.a)

return [round(x1, 2), round(x2, 2)]

elif discr == 0:

x = -self.b / (2 \* self.a)

return [round(x, 2)]

else:

return None

def extreme(self):

x = -(self.b / (2 \* self.a))

y = self.a \* x \*\* 2 + self.b \* x + self.c

if self.a >= 0:

return print("Точка минимума[",x,', ',y,"]", sep='')

else:

return print("Точка максимума[",x,', ',y,"]", sep='')

def interval(self):

x = -(self.b / (2 \* self.a))

if self.a >= 0:

return print("Убывает на промежутках[-∞, ",x,']', '\n', "Возрастает на промежутках[",x,', ∞]', sep='')

else:

return print("Возрастает на промежутках[-∞, ",x,']', '\n', "Убывает на промежутках[",x,', ∞]', sep='')

if \_\_name\_\_ == "\_\_main\_\_":

equation1 = Equation(-2,8,3)

print("Корни:",equation1.root())

equation2 = Equation(1,17,18)

equation2.extreme()

equation3 = Equation(1,2,1)

equation3.interval()

equations = [equation1, equation2, equation3]

roots = []

for i in equations:

if (i.root() != None):

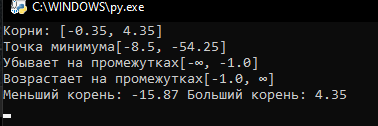
for j in i.root():

roots.append(j)

print("Меньший корень:", min(roots), "Больший корень:", max(roots))

input()

Скриншот результата



Ссылка на онлайн компилятор

[main.py - Task 2 - Replit](https://replit.com/@PretrovDS/Task-2#main.py)

Задание 3

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Наследование применять только в тех заданиях, в которых это логически обосновано. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

15. Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| doors | array | Массив в классе House. Отвечает за хранение объектов класса Door. |
| windows | array | Массив в классе House. Отвечает за хранение объектов класса Window. |
| Door. width | int | Переменная в классе Door. Отвечает за хранение ширины двери. |
| Door. height | int | Переменная в классе Door. Отвечает за хранение высоты двери. |
| Door. material | string | Переменная в классе Door. Отвечает за хранение материала двери. |
| Door. isOpen | boolean | Переменная в классе Door. Отвечает за хранение состояния открытия двери. |
| Window. width | int | Переменная в классе Window. Отвечает за хранение ширины окна. |
| Window. height | int | Переменная в классе Window. Отвечает за хранение высоты окна. |
| Window. glazing | string | Переменная в классе Window. Отвечает за хранение типа стеклопакета окна. |
| Door1 | Door | Объект класса Door. Отвечает за демонстрацию работы программы. |
| Door2 | Door | Объект класса Door. Отвечает за демонстрацию работы программы. |
| Door3 | Door | Объект класса Door. Отвечает за демонстрацию работы программы. |
| Window1 | Window | Объект класса Window. Отвечает за демонстрацию работы программы. |
| Window2 | Window | Объект класса Window. Отвечает за демонстрацию работы программы. |
| Window3 | Window | Объект класса Window. Отвечает за демонстрацию работы программы. |
| House1 | House | Объект класса House. Отвечает за демонстрацию работы программы. |
| House2 | House | Объект класса House. Отвечает за демонстрацию работы программы. |

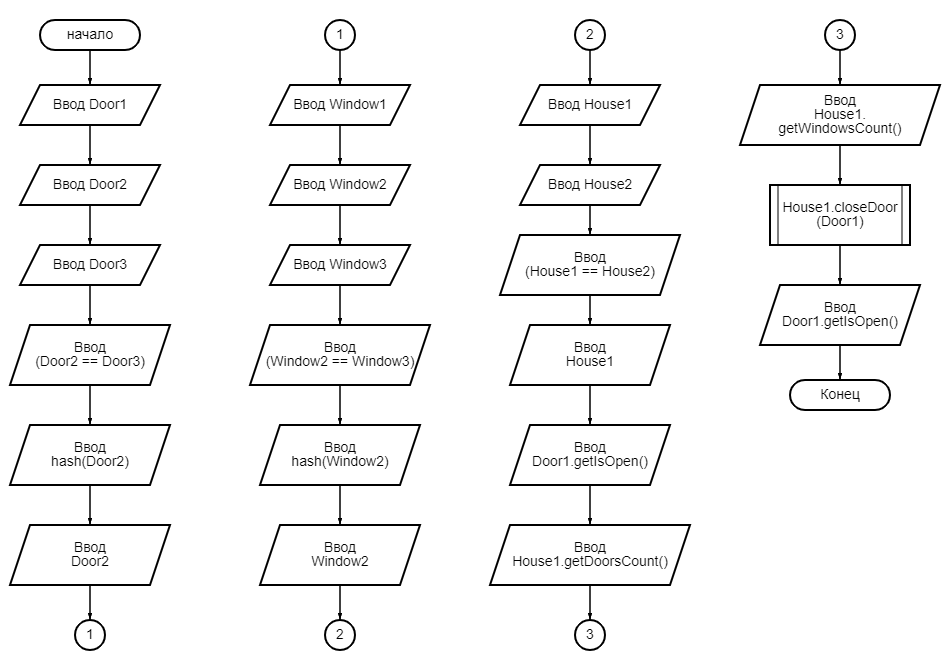
UML диаграмма классов

|  |
| --- |
| Window |
| width  height  glazing |
| \_\_init\_\_(self, width = 0, height = 0, glazing = '')  \_\_eq\_\_(self, other)  \_\_hash\_\_(self)  \_\_str\_\_(self)  getWidth(self)  getHeight(self)  getMaterial(self)  setWidth(self, width)  setHeight(self, height)  setGlazing(self, glazing) |

|  |
| --- |
| Door |
| width  height  material  isOpen |
| \_\_init\_\_(self, width = 0, height = 0, material = '', isOpen = True)extreme(self)  \_\_eq\_\_(self, other)  \_\_hash\_\_(self)  \_\_str\_\_(self)  getWidth(self)  getHeight(self)  getMaterial(self)  getIsOpen(self)  setWidth(self, width)  setHeight(self, height)  setMaterial(self, material)  setIsOpen(self, isOpen) |

|  |
| --- |
| House |
| doors  windows |
| \_\_init\_\_(self, doors = [], windows = [])  \_\_eq\_\_(self, other)  \_\_str\_\_(self)  addDoor(self, door)  addWindow(self,window)  getDoorsCount(self)  getWindowsCount(self)  closeDoor(self, door) |

Блок схема



Код программы

class House:

def \_\_init\_\_(self, doors = [], windows = []):

self.\_doors = doors

self.\_windows = windows

"""Переопределение equals"""

def \_\_eq\_\_(self, other):

return (self.\_doors, self.\_windows) == (other.\_doors, other.\_windows)

"""Переопределение toString"""

def \_\_str\_\_(self):

return "Класс House"

def addDoor(self, door):

self.\_doors.append(door)

def addWindow(self,window):

self.\_windows.append(window)

def getDoorsCount(self):

return len(self.\_doors)

def getWindowsCount(self):

return len(self.\_windows)

def closeDoor(self, door):

print("Дверь закрыта")

door.setIsOpen(False)

class Door:

def \_\_init\_\_(self, width = 0, height = 0, material = '', isOpen = True):

self.\_width = width

self.\_height = height

self.\_material = material

self.\_isOpen = isOpen

"""Переопределение equals"""

def \_\_eq\_\_(self, other):

return (self.\_width, self.\_height,self.\_material, self.\_isOpen) == (other.\_width, other.\_height,other.\_material, other.\_isOpen)

"""Переопределение hashCode"""

def \_\_hash\_\_(self):

return hash((self.\_width, self.\_height,self.\_material, self.\_isOpen))

"""Переопределение toString"""

def \_\_str\_\_(self):

return "Класс Door"

def getWidth(self):

return self.\_width

def getHeight(self):

return self.\_height

def getMaterial(self):

return self.\_material

def getIsOpen(self):

return self.\_isOpen

def setWidth(self, width):

self.\_width = width

def setHeight(self, height):

self.\_height = height

def setMaterial(self, material):

self.\_material = material

def setIsOpen(self, isOpen):

self.\_isOpen = isOpen

class Window:

def \_\_init\_\_(self, width = 0, height = 0, glazing = ''):

self.\_width = width

self.\_height = height

self.\_glazing = glazing

"""Переопределение equals"""

def \_\_eq\_\_(self, other):

return (self.\_width, self.\_height, self.\_glazing) == (other.\_width, other.\_height, other.\_glazing)

"""Переопределение hashCode"""

def \_\_hash\_\_(self):

return hash((self.\_width, self.\_height, self.\_glazing))

"""Переопределение toString"""

def \_\_str\_\_(self):

return "Класс Window"

def getWidth(self):

return self.\_width

def getHeight(self):

return self.\_height

def getMaterial(self):

return self.\_material

def setWidth(self, width):

self.\_width = width

def setHeight(self, height):

self.\_height = height

def setGlazing(self, glazing):

self.\_glazing = glazing

Door1 = Door(75, 200, "Дуб")

Door2 = Door(85, 210, "Фанера")

Door3 = Door(85, 210, "Фанера")

print("Работает ли переопределение equals в классе Door:", Door2 == Door3)

print("Работа hashCode в классе Door:", hash(Door2))

print("Работа toString в классе Door:", end = " ")

print(Door2)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

Window1 = Window(175, 145, "Двойной степлопакет")

Window2 = Window(95, 145, "Одинарный степлопакет")

Window3 = Window(95, 145, "Одинарный степлопакет")

print("Работает ли переопределение equals в классе Window:", Window2 == Window3)

print("Работа hashCode в классе Window:", hash(Window2))

print("Работа toString в классе Window:", end = " ")

print(Window2)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

House1 = House([Door1, Door2],[Window1, Window2])

House2 = House([Door1, Door2],[Window1, Window2])

print("Работает ли переопределение equals в классе House:", House1 == House2)

print("Работа toString в классе House:", end = " ")

print(House1)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print("Открытость первой двери: ",Door1.getIsOpen())

print("Количество дверей: ", House1.getDoorsCount())

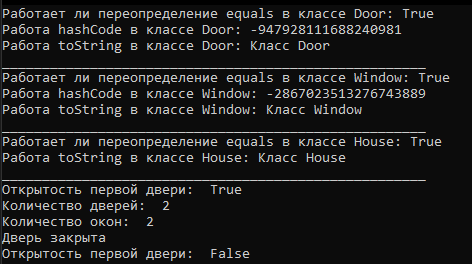
print("Количество окон: ",House1.getWindowsCount())

House1.closeDoor(Door1)

print("Открытость первой двери: ",Door1.getIsOpen())

input()

Скриншот результата



Ссылка на онлайн компилятор

[main.py - Task 3 - Replit](https://replit.com/@PretrovDS/Task-3#main.py)

Задание 4

15. Создать класс Computer с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти

Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| computer1 | Computer | Объект класса Computer. Отвечает за демонстрацию работы программы |
| OS | Specification | Объект класса Specification. Отвечает за демонстрацию работы программы. Хранит характеристики операционной системы. |
| RAM | Specification | Объект класса Specification. Отвечает за демонстрацию работы программы. Хранит характеристики оперативной памяти. |
| processor | Specification | Объект класса Specification. Отвечает за демонстрацию работы программы. Хранит характеристики процессора. |
| id | int | Переменная класса Computer. Отвечает за хранение id компьютера |
| name | String | Переменная класса Specification. Отвечает за хранение названия характеристики. |
| value | String | Переменная класса Specification. Отвечает за хранение значения характеристики. |

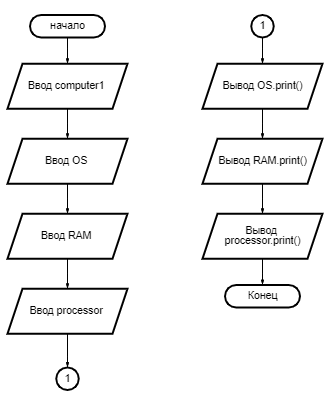
UML диаграмма классов

|  |
| --- |
| Computer |
| id |
| Computer(int id) |

принадлежит

|  |
| --- |
| Specification |
| name  value |
| Specification(String name, String value) |

Блок схема



Код программы

public class Main {

public static void main(String[] args) {

Computer computer1 = new Computer(1 );

Computer.Specification OS = computer1.new Specification("Операционная система", "Linux");

Computer.Specification RAM = computer1.new Specification("Оперативная память", "8Гб");

Computer.Specification processor = computer1.new Specification("Процессор", "i5 8300h");

OS.print();

RAM.print();

processor.print();

}

}

class Computer {

private int id;

public Computer(int id) {this.id = id;}

public class Specification {

String name;

String value;

public Specification(String name, String value) {

this.name = name;

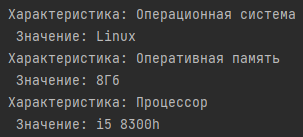
this.value = value;

}

public void print() {

System.out.println("Характеристика: " + name + "\n Значение: " + value); }}}

Скриншот результата



Ссылка на онлайн компилятор

[Main.java - Task 4 - Replit](https://replit.com/@PretrovDS/Task-4#Main.java)

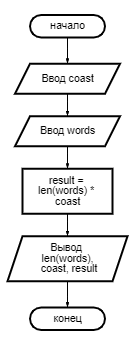
Задание 5

15. Напечатать квитанцию об оплате телеграммы, если стоимость одного слова задана

Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| coast | float | Входная переменная. Отвечает за хранение стоимости одного слова |
| words | array | Входная переменная. Отвечает за хранение массива введенных слов. |
| result | float | Выходная переменная. Отвечает за хранение вычисленной стоимости предложения. |

Блок схема



Код программы

coast = float(input("Введите стоимость слова: "))

words = input("Введите телеграмму: ").split()

result = len(words) \* coast

print("Квитанция об оплате:", '\n',

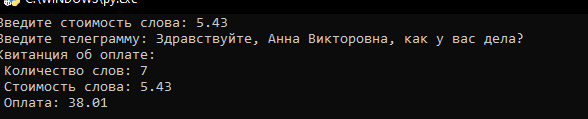
'Количество слов:', len(words), '\n',

"Стоимость слова:", coast, '\n',

"Оплата:", result)

input()

Скриншот результата



Ссылка на онлайн компилятор

[main.py - Task 5 - Replit](https://replit.com/@PretrovDS/Task-5#main.py)

Задание 6

Вариант A

Выполнить задания на основе варианта А задачи 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Вариант B

Выполнить задания из варианта В задачи 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| doors | array | Массив в классе House. Отвечает за хранение объектов класса Door. |
| windows | array | Массив в классе House. Отвечает за хранение объектов класса Window. |
| Door. width | int | Переменная в классе Door. Отвечает за хранение ширины двери. |
| Door. height | int | Переменная в классе Door. Отвечает за хранение высоты двери. |
| Door. material | string | Переменная в классе Door. Отвечает за хранение материала двери. |
| Door. isOpen | boolean | Переменная в классе Door. Отвечает за хранение состояния открытия двери. |
| Window. width | int | Переменная в классе Window. Отвечает за хранение ширины окна. |
| Window. height | int | Переменная в классе Window. Отвечает за хранение высоты окна. |
| Window. glazing | string | Переменная в классе Window. Отвечает за хранение типа стеклопакета окна. |
| Door1 | Door | Объект класса Door. Отвечает за демонстрацию работы программы. |
| Door2 | Door | Объект класса Door. Отвечает за демонстрацию работы программы. |
| Door3 | Door | Объект класса Door. Отвечает за демонстрацию работы программы. |
| Window1 | Window | Объект класса Window. Отвечает за демонстрацию работы программы. |
| Window2 | Window | Объект класса Window. Отвечает за демонстрацию работы программы. |
| Window3 | Window | Объект класса Window. Отвечает за демонстрацию работы программы. |
| House1 | House | Объект класса House. Отвечает за демонстрацию работы программы. |
| House2 | House | Объект класса House. Отвечает за демонстрацию работы программы. |
| Houses | array | Массив объектов House. Отвечает за хранение домов. |
| house\_number | Int | Вводимая переменная. Отвечает за выбор номера дома. |
| i | int | Вспомогательная переменная для циклического перебора массива. |
| e | Exception | Вспомогательная переменная, хранящая ошибку. Создаётся при неверном выполнении программы. |

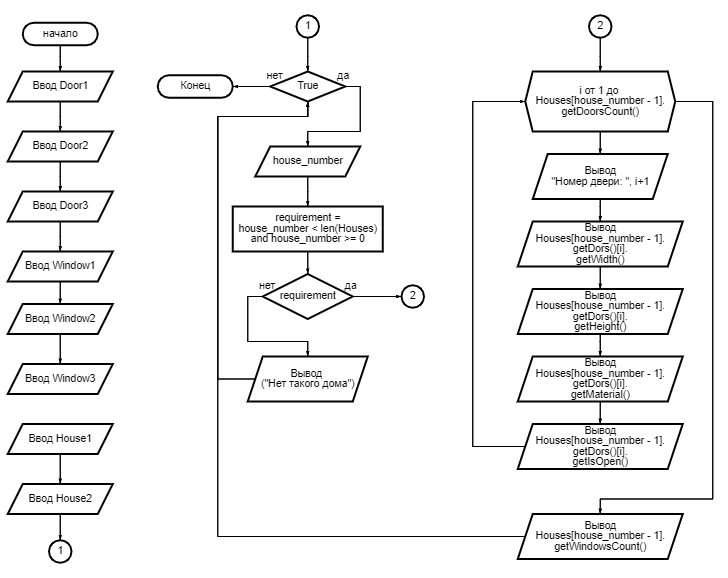
UML диаграмма классов

|  |
| --- |
| Window |
| width  height  glazing |
| \_\_init\_\_(self, width = 0, height = 0, glazing = '')  \_\_eq\_\_(self, other)  \_\_hash\_\_(self)  \_\_str\_\_(self)  getWidth(self)  getHeight(self)  getMaterial(self)  setWidth(self, width)  setHeight(self, height)  setGlazing(self, glazing) |

|  |
| --- |
| Door |
| width  height  material  isOpen |
| \_\_init\_\_(self, width = 0, height = 0, material = '', isOpen = True)extreme(self)  \_\_eq\_\_(self, other)  \_\_hash\_\_(self)  \_\_str\_\_(self)  getWidth(self)  getHeight(self)  getMaterial(self)  getIsOpen(self)  setWidth(self, width)  setHeight(self, height)  setMaterial(self, material)  setIsOpen(self, isOpen) |

|  |
| --- |
| House |
| doors  windows |
| \_\_init\_\_(self, doors = [], windows = [])  \_\_eq\_\_(self, other)  \_\_str\_\_(self)  addDoor(self, door)  addWindow(self,window)  getDoorsCount(self)  getWindowsCount(self)  closeDoor(self, door) |

Блок схема



Код программы

class House:

def \_\_init\_\_(self, doors = [], windows = []):

self.\_doors = doors

self.\_windows = windows

"""Переопределение equals"""

def \_\_eq\_\_(self, other):

return (self.\_doors, self.\_windows) == (other.\_doors, other.\_windows)

"""Переопределение toString"""

def \_\_str\_\_(self):

return "Класс House"

def addDoor(self, door):

self.\_doors.append(door)

def addWindow(self,window):

self.\_windows.append(window)

def getDoorsCount(self):

return len(self.\_doors)

def getWindowsCount(self):

return len(self.\_windows)

def closeDoor(self, door):

print("Дверь закрыта")

door.setIsOpen(False)

def getDors(self):

return self.\_doors

def getWindows(self):

return self.\_windows

class Door:

def \_\_init\_\_(self, width = 0, height = 0, material = '', isOpen = True):

self.\_width = width

self.\_height = height

self.\_material = material

self.\_isOpen = isOpen

"""Переопределение equals"""

def \_\_eq\_\_(self, other):

return (self.\_width, self.\_height,self.\_material, self.\_isOpen) == (other.\_width, other.\_height,other.\_material, other.\_isOpen)

"""Переопределение hashCode"""

def \_\_hash\_\_(self):

return hash((self.\_width, self.\_height,self.\_material, self.\_isOpen))

"""Переопределение toString"""

def \_\_str\_\_(self):

return ("Класс Door" )

def getWidth(self):

return self.\_width

def getHeight(self):

return self.\_height

def getMaterial(self):

return self.\_material

def getIsOpen(self):

return self.\_isOpen

def setWidth(self, width):

self.\_width = width

def setHeight(self, height):

self.\_height = height

def setMaterial(self, material):

self.\_material = material

def setIsOpen(self, isOpen):

self.\_isOpen = isOpen

class Window:

def \_\_init\_\_(self, width = 0, height = 0, glazing = ''):

self.\_width = width

self.\_height = height

self.\_glazing = glazing

"""Переопределение equals"""

def \_\_eq\_\_(self, other):

return (self.\_width, self.\_height, self.\_glazing) == (other.\_width, other.\_height, other.\_glazing)

"""Переопределение hashCode"""

def \_\_hash\_\_(self):

return hash((self.\_width, self.\_height, self.\_glazing))

"""Переопределение toString"""

def \_\_str\_\_(self):

return "Класс Window"

def getWidth(self):

return self.\_width

def getHeight(self):

return self.\_height

def getMaterial(self):

return self.\_material

def setWidth(self, width):

self.\_width = width

def setHeight(self, height):

self.\_height = height

def setGlazing(self, glazing):

self.\_glazing = glazing

Door1 = Door(75, 200, "Дуб")

Door2 = Door(85, 210, "Фанера")

Door3 = Door(65, 210, "Осина")

Window1 = Window(175, 145, "Двойной степлопакет")

Window2 = Window(95, 145, "Одинарный степлопакет")

Window3 = Window(115, 165, "Двойной степлопакет")

House1 = House([Door1, Door2],[Window1, Window2])

House2 = House([Door1, Door2, Door3],[Window1, Window2, Window3])

Houses = [House1, House2]

while True:

try:

house\_number = int(input("Введите номер: "))

if (house\_number > len(Houses) or house\_number <= 0):

print("Да нет такого дома!")

continue

print("Дверей в этом доме: ", Houses[house\_number - 1].getDoorsCount())

for i in range(Houses[house\_number - 1].getDoorsCount()):

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print("Номер двери: ", i+1)

print("Ширина: ", Houses[house\_number - 1].getDors()[i].getWidth())

print("Высота: ", Houses[house\_number - 1].getDors()[i].getHeight())

print("Материал: ", Houses[house\_number - 1].getDors()[i].getMaterial())

print("Открытость: ", Houses[house\_number - 1].getDors()[i].getIsOpen())

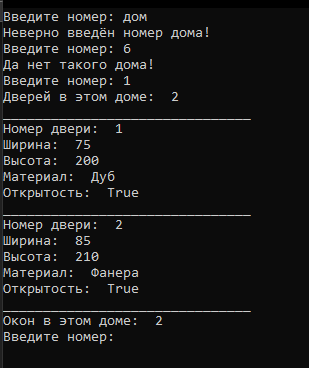
print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print("Окон в этом доме: ", Houses[house\_number - 1].getWindowsCount())

except Exception as e:

print("Неверно введён номер дома!")

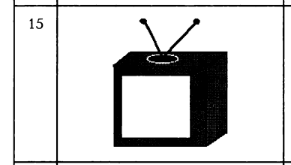
Скриншот результата



Ссылка на онлайн компилятор

[main.py - Task 6 - Replit](https://replit.com/@PretrovDS/Task-6#main.py)

Задание 7



Код программы

import javax.swing.\*;

import java.awt.\*;

import java.awt.geom.Ellipse2D;

public class Main extends JFrame {

Main(String s) {

super(s);

setLayout(null);

setSize(600, 750);

setVisible(true);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

public void paint(Graphics g) {

g.setColor(Color.BLACK);

Graphics2D gr = (Graphics2D)g;

g.fillRect(50, 300, 400, 400);

g.setColor(Color.WHITE);

g.fillRect(100, 350, 300, 300);

int px1[]={50,450,550,200,50};

int py1[]={300,300,240,240, 300};

g.setColor(Color.GRAY);

g.fillPolygon(px1,py1,4);

int px2[]={450,550,550,450,450};

int py2[]={300,240,600,700, 300};

g.setColor(Color.DARK\_GRAY);

g.fillPolygon(px2,py2,4);

BasicStroke pen;

pen=new BasicStroke(5,BasicStroke.CAP\_ROUND,BasicStroke.JOIN\_ROUND);

((Graphics2D) g).setStroke(pen);

g.setColor(Color.WHITE);

g.drawOval(250, 250, 100, 40);

g.setColor(Color.BLACK);

g.drawLine(300, 270, 270,150);

g.fillOval(260, 140, 20, 20);

g.drawLine(300, 270, 330,150);

g.fillOval(320, 140, 20, 20);

}

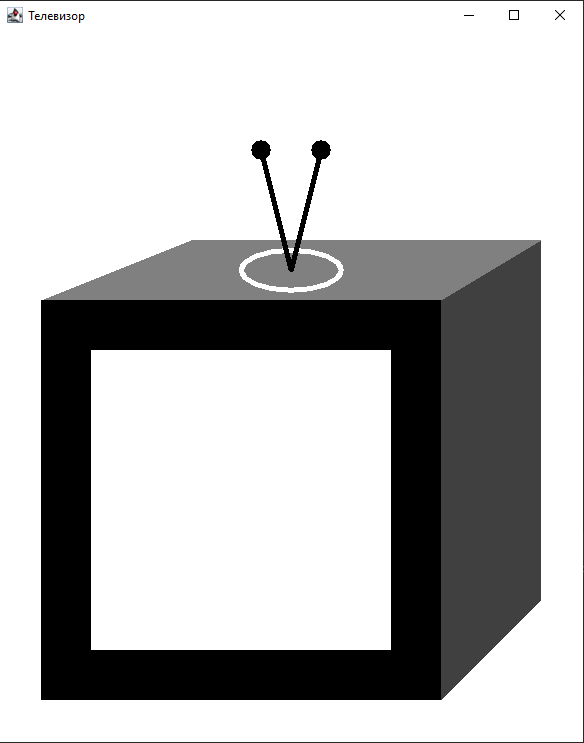
public static void main(String[] args) {

new Main("Телевизор");

}

}

Скриншот результата



Ссылка на онлайн компилятор

[Main.java - Task 7 - Replit](https://replit.com/@PretrovDS/Task-7#Main.java)

Задание 8

15. Составить описание класса многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть при этом методы вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта многочлена, печать (вывод на экран) описания многочлена.

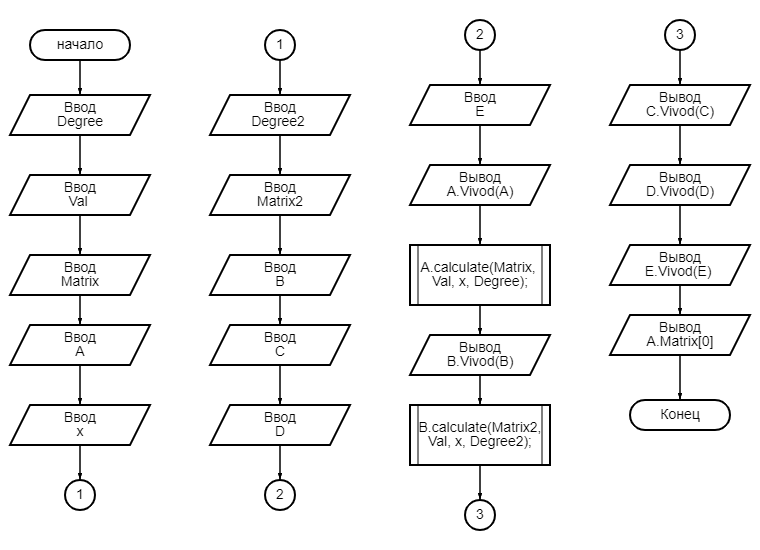
Таблица переменных

| Название | Тип | Назначение |
| --- | --- | --- |
| Degree | int | Переменная хранящая степень многочлена. |
| Val | double | Переменная хранящая значение первого многочлена. |
| Matrix | double[] | Переменная хранящая коэффициенты первого многочлена. |
| A | MultiMember | Объект класса MultiMember. Отвечает за демонстрацию работы программы. |
| x | double | Переменная хранящая значение неизвестной. |
| Degree2 | int | Переменная хранящая степень второго многочлена. |
| Matrix2 | double[] | Переменная хранящая коэффициенты второго многочлена. |
| B | MultiMember | Объект класса MultiMember. Отвечает за демонстрацию работы программы. |
| C | MultiMember.MultiMemberSum | Переменная хранящая объект результатов вычисления сложения многочлена A и B. |
| D | MultiMember.MultiMemberSub | Переменная хранящая объект результатов вычисления вычитания многочлена A и B. |
| E | MultiMember.MultiMemberMult | Переменная хранящая объект результатов вычисления умножения многочлена A и B. |
| MultiMember.Matrix | double[] | Переменная класса MultiMember, хранящая массив коэффициентов многочлена. |
| MultiMember.Degree | int | Переменная класса MultiMember, хранящая степень многочлена. |
| MultiMember.D1 | int | Вспомогательная переменная класса MultiMember. Содержит степень многочлена. |
| MultiMember.M1 | double[] | Вспомогательная переменная класса MultiMember. Содержит массив коэффициентов многочлена. |
| MultiMember. C | MultiMember | Вспомогательный объект класса MultiMember. Содержит результат вычисления многочленов. |
| MultiMember. value | double | Вспомогательная переменная класса MultiMember. Содержит результат решения многочлена. |

UML диаграмма классов

|  |
| --- |
| MultiMember |
| Matrix  Degree |
| MultiMember(double[] matrix, int degree)  MultiMemberSum (MultiMember A, MultiMember B)  MultiMemberSub (MultiMember A, MultiMember B)  MultiMemberMult(MultiMember A, MultiMember B)  Vivod(MultiMember A)  calculate(double[] matrix, double value, double x, int Degree) |

Блок схема



Код программы

public class Main {

public static void main(String[] args) {

int Degree = 2;

double Val = 0;

double[] Matrix={1,2,3};

MultiMember A = new MultiMember(Matrix, Degree);

double x = 1;

int Degree2 =2;

double[] Matrix2 = {4,5,6};

MultiMember B = new MultiMember(Matrix2, Degree2);

MultiMember C = MultiMember.MultiMemberSum(A, B);

MultiMember D = MultiMember.MultiMemberSub(A, B);

MultiMember E = MultiMember.MultiMemberMult(A, B);

System.out.print("A=");

System.out.println(A.Vivod(A));

A.calculate(Matrix, Val, x, Degree);

System.out.print("B=");

System.out.println(B.Vivod(B));

B.calculate(Matrix2, Val, x, Degree2);

System.out.print("A+B=");

System.out.println(C.Vivod(C) + "\n");

System.out.print("A-B=");

System.out.println(D.Vivod(D)+"\n");

System.out.print("A\*B=");

System.out.println(E.Vivod(E) + "\n");

System.out.print("Получение коэффициента по индексу 0 для многочлена А: "+ A.Matrix[0]);

}

}

class MultiMember

{

double[] Matrix;

int Degree;

public MultiMember(double[] matrix, int degree)

{

Matrix = matrix;

Degree = degree;

}

public static MultiMember MultiMemberSum (MultiMember A, MultiMember B)

{

int D1 = A.Degree ;

double[] M1=new double[D1+1] ;

MultiMember C = new MultiMember(M1,D1);

for (int i = 0; i < A.Degree+1; i++)

{

C.Matrix[i]=A.Matrix[i]+B.Matrix[i] ;

}

return C;

}

public static MultiMember MultiMemberSub (MultiMember A, MultiMember B)

{

int D1 = A.Degree;

double[] M1 = new double[D1 + 1];

MultiMember C = new MultiMember(M1, D1);

for (int i = 0; i < A.Degree + 1; i++)

{

C.Matrix[i] = A.Matrix[i] - B.Matrix[i];

}

return C;

}

public static MultiMember MultiMemberMult(MultiMember A, MultiMember B)

{

int D1 = A.Degree;

double[] M1 = new double[D1 + 1];

MultiMember C = new MultiMember(M1, D1);

for (int i = 0; i < A.Degree + 1; i++)

{

C.Matrix[i] = A.Matrix[i] \* B.Matrix[i];

}

return C;

}

public int Vivod(MultiMember A)

{

for (int i=0;i<=Degree ;i++)

{

System.out.print(A.Matrix[i] + "X" + "^" + (Degree - i) + "+");

}

return 0;

}

public double calculate(double[] matrix, double value, double x, int Degree)

{

for( int i = Degree; i >0; i--)

{

for (int j = i; j < Degree; j++) x \*= x;

value += matrix[i-1] \* x;

}

value+=matrix[Degree];

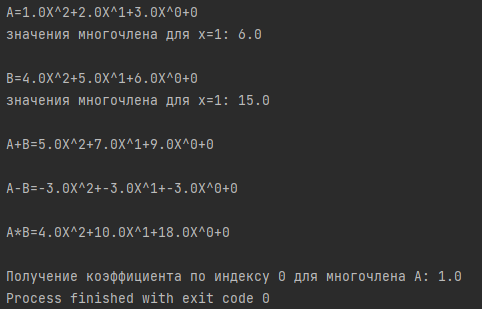
System.out.println("значения многочлена для x=1: "+value+"\n");

return value;

}

}

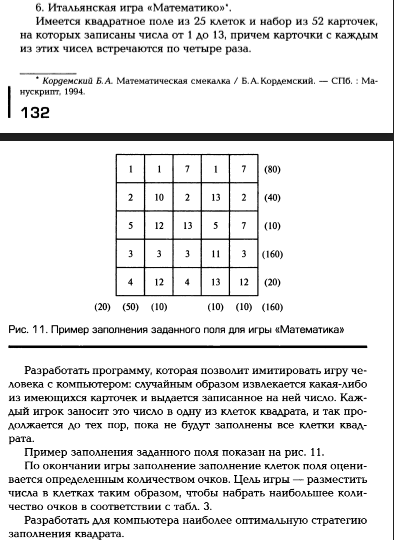
Скриншот результат



Ссылка на онлайн компилятор

[Main.java - Task 8 - Replit](https://replit.com/@PretrovDS/Task-8#Main.java)

Задание 9



Код программы

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import javax.swing.\*;

public class Main {

final static int btns\_count = 14;

final static int btn\_width = 60;

final static int btn\_height = 50;

static List<JButton>player\_buttons = new LinkedList<>();

static List<JButton> bot\_buttons = new LinkedList<>();

static List<Integer> bot\_buttons\_active = new LinkedList<>();

static Map<Integer, Integer> integers = new HashMap<>();

static Map<Integer, Integer> bot\_integers = new HashMap<>();

static int steps = 0;

static JLabel player\_score\_label = new JLabel("Очки: 0");

static JLabel bot\_score\_label = new JLabel("Очки: 0");

static int score\_sum = 0;

static int score\_sum\_bot = 0;

public static void main(String[] args) {

JFrame f = new JFrame();

JPanel panel = new JPanel();

JLabel player = new JLabel("Игрок");

player.setBounds(0, 10, 200, 50);

JLabel bot = new JLabel("Компьютер");

bot.setBounds(400, 10, 200, 50);

ActionListener actionListener = new TestActionListener();

panel.add(player);

panel.add(bot);

player\_score\_label.setBounds(0, 300, 200, 10);

bot\_score\_label.setBounds(400, 300, 200, 10);

panel.add(player\_score\_label);

panel.add(bot\_score\_label);

player\_score\_label.setVisible(false);

bot\_score\_label.setVisible(false);

for (int i = 0; i < 14; i++){

integers.put(i, 0);

bot\_integers.put(i, 0);

}

int value;

while (bot\_buttons\_active.size() < 25){

value = (int) (Math.random() \* 25);

if (!bot\_buttons\_active.contains(value)){

bot\_buttons\_active.add(value);

}

}

int x = 0;

int y = 0;

for (int i = 0; i < 25; i++){

if (i % 5 == 0){

x = 0;

y++;

}

JButton button1 = new JButton("0");

button1.setActionCommand(i + "");

button1.setBounds(btn\_width \* x,btn\_height \* y,btn\_width,btn\_height);

panel.add(button1, i);

button1.addActionListener(actionListener);

player\_buttons.add(button1);

x++;

}

x = 0;

y = 0;

for (int i = 0; i < 25; i++){

if (i % 5 == 0){

x = 0;

y++;

}

JButton button1 = new JButton("0");

button1.setActionCommand(i + "");

button1.setBounds(btn\_width \* x + 400,btn\_height \* y,btn\_width,btn\_height);

panel.add(button1, i);

button1.setEnabled(false);

bot\_buttons.add(button1);

x++;

}

panel.setLayout(null);

panel.setLocation(0,0);

f.setSize(720, 360);

f.add(panel);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setVisible(true);

}

public static class TestActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

Object source = e.getSource();

((JButton) source).setText("" + rnd(btns\_count, false));

((JButton) source).setEnabled(false);

bot\_buttons.get(bot\_buttons\_active.get(0)).setText("" + rnd(btns\_count, true));

bot\_buttons\_active.remove(0);

if(steps == 24){

int[] result = result(player\_buttons,bot\_buttons);

player\_score\_label.setVisible(true);

player\_score\_label.setText("Очки: " + result[0]);

bot\_score\_label.setVisible(true);

bot\_score\_label.setText("Очки: " + result[1]);

}

else{

steps++;

}

}

}

public static int[] result( List<JButton>player\_buttons, List<JButton>bot\_buttons)

{

int[] result = new int[2];

result[0] = 0;

result[1] = 0;

int x1 = 0;

int y1 = 0;

int[][] values = new int[5][5];

for (int i = 0; i < 24; i++){

if (i % 5 == 0){

x1 = 0;

y1++;

}

values[y1-1][x1] = Integer.parseInt(player\_buttons.get(i).getText());

x1++;

}

int score\_row\_sum = 0;

int score\_row = 0;

for (int k = 0; k < 14; k++){

for (int i = 0; i < 5; i++){

for (int j = 0; j < 5; j++){

if (values[i][j] == k && values[i][j]!=-1){

score\_row++;

}

}

if (score\_row > 1){

score\_row\_sum = score\_row\_sum + score\_row;

}

score\_row = 0;

}

if (score\_row\_sum == 2){

score\_sum = score\_sum + 10;

}

if (score\_row\_sum >= 3){

score\_sum = score\_sum + 40;

}

score\_row\_sum = 0;

}

result[0] = score\_sum;

x1 = 0;

y1 = 0;

values = new int[5][5];

for (int i = 0; i < 24; i++){

if (i % 5 == 0){

x1 = 0;

y1++;

}

values[y1-1][x1] = Integer.parseInt(bot\_buttons.get(i).getText());

x1++;

}

score\_row\_sum = 0;

score\_row = 0;

for (int k = 0; k < 14; k++){

for (int i = 0; i < 5; i++){

for (int j = 0; j < 5; j++){

if (values[i][j] == k && values[i][j]!=-1){

score\_row++;

}

}

if (score\_row > 1){

score\_row\_sum = score\_row\_sum + score\_row;

}

score\_row = 0;

}

if (score\_row\_sum == 2){

score\_sum\_bot = score\_sum\_bot + 10;

}

if (score\_row\_sum >= 3){

score\_sum\_bot = score\_sum\_bot + 40;

}

score\_row\_sum = 0;

}

result[1] = score\_sum\_bot;

return result;

}

public static int rnd(final double max, boolean isBot)

{

int value = (int) (Math.random() \* max);

if (!isBot){

while (integers.get(value) > 3){

value = (int) (Math.random() \* max);

}

integers.put(value, integers.get(value) + 1);

}

else {

while (bot\_integers.get(value) > 3){

value = (int) (Math.random() \* max);

}

bot\_integers.put(value, bot\_integers.get(value) + 1);

}

return value;}}

Скриншот результат



Ссылка на онлайн компилятор

[Main.java - Task 9 - Replit](https://replit.com/@PretrovDS/Task-9#Main.java)

Выбор задач реализовать через меню! Одно приложение должно содержать решение 9-ти задач.

Код программы

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.net.URL;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class Main extends JFrame {

private JTextField textField;

public Main() {

super("Соединение заданий");

createGUI();

}

public void createGUI() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new FlowLayout());

ActionListener actionListener = new TestActionListener();

JButton button1 = new JButton("Задание 1");

button1.setActionCommand("https://replit.com/@PretrovDS/Task-1#main.py");

panel.add(button1);

button1.addActionListener(actionListener);

JButton button2 = new JButton("Задание 2");

button2.setActionCommand("https://replit.com/@PretrovDS/Task-2#main.py");

panel.add(button2);

button2.addActionListener(actionListener);

JButton button3 = new JButton("Задание 3");

button3.setActionCommand("https://replit.com/@PretrovDS/Task-3#main.py");

panel.add(button3);

button3.addActionListener(actionListener);

JButton button4 = new JButton("Задание 4");

button4.setActionCommand("https://replit.com/@PretrovDS/Task-4#Main.java");

panel.add(button4);

button4.addActionListener(actionListener);

JButton button5 = new JButton("Задание 5");

button5.setActionCommand("https://replit.com/@PretrovDS/Task-5#main.py");

panel.add(button5);

button5.addActionListener(actionListener);

JButton button6 = new JButton("Задание 6");

button6.setActionCommand("https://replit.com/@PretrovDS/Task-6#main.py");

panel.add(button6);

button6.addActionListener(actionListener);

JButton button7 = new JButton("Задание 7");

button7.setActionCommand("https://replit.com/@PretrovDS/Task-7#Main.java");

panel.add(button7);

button7.addActionListener(actionListener);

JButton button8 = new JButton("Задание 8");

button8.setActionCommand("https://replit.com/@PretrovDS/Task-8#Main.java");

panel.add(button8);

button8.addActionListener(actionListener);

JButton button9 = new JButton("Задание 9");

button9.setActionCommand("https://replit.com/@PretrovDS/Task-9#Main.java");

panel.add(button9);

button9.addActionListener(actionListener);

getContentPane().add(panel);

setPreferredSize(new Dimension(320, 200));

}

public class TestActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

try {

Desktop.getDesktop().browse(new URL(e.getActionCommand()).toURI());

} catch (Exception ex) {

System.out.println(ex.toString());

}

}

}

public static void main(String[] args) {

javax.swing.SwingUtilities.invokeLater(new Runnable() {

public void run() {

JFrame.setDefaultLookAndFeelDecorated(true);

Main frame = new Main();

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

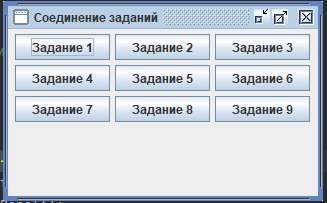
}

});

}

}

Скриншот результата

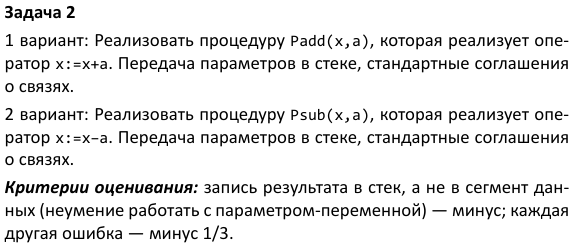


Ссылка на онлайн компилятор

\*не работает в онлайн компиляторе (онлайн компилятор не поддерживает BROWSE action)

[Main.java - Tasks connection - Replit](https://replit.com/@PretrovDS/Tasks-connection#Main.java)

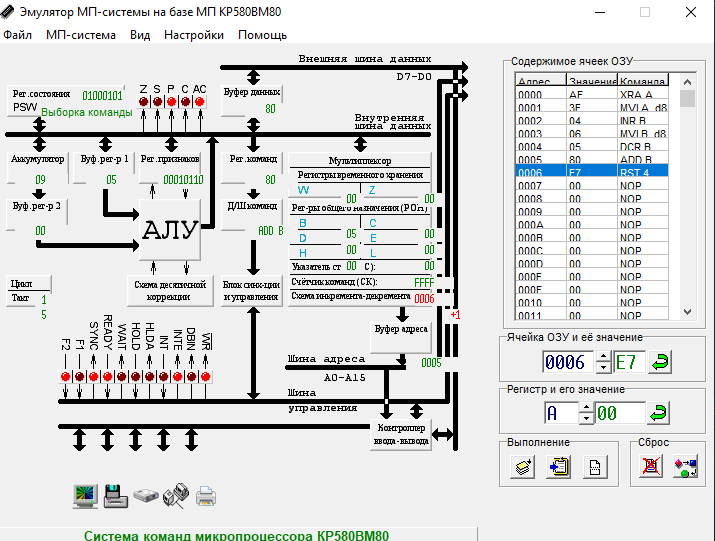
Задача 10



Структура программы

| Код команды | Мнемокод | Комментарий |
| --- | --- | --- |
| AF | XRA A | Очистить аккумулятор |
| 3E | MVI A | Записать в аккумулятор |
| 04 |  | Число x |
| 06 | MVI B | Записать в регистр B |
| 05 |  | Число a |
| 80 | ADD B | Сложить x и a |
| E7 | RST 7 | Прервать выполнение программы |

Скриншот результата

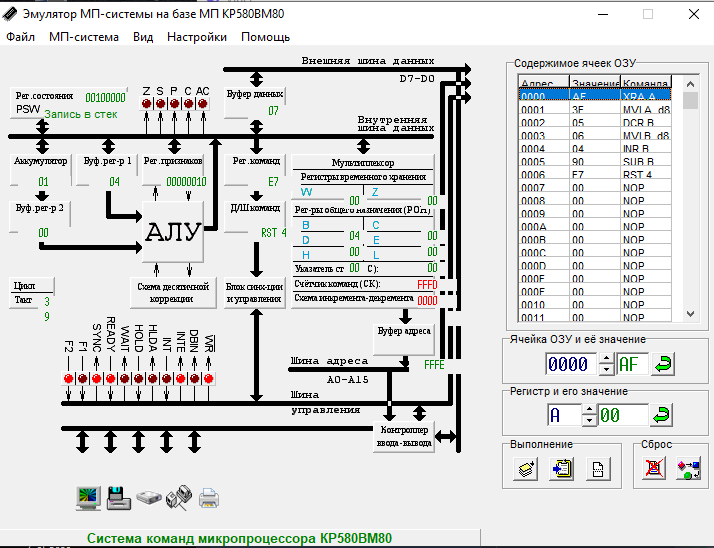


После выполнения программы в аккумуляторе появляется число 9

Структура программы

| Код команды | Мнемокод | Комментарий |
| --- | --- | --- |
| AF | XRA A | Очистить аккумулятор |
| 3E | MVI A | Записать в аккумулятор |
| 04 |  | Число x |
| 06 | MVI B | Записать в регистр B |
| 05 |  | Число a |
| 90 | SUB B | Вычесть из x и a |
| E7 | RST 7 | Прервать выполнение программы |

Скриншот результата



После выполнения программы в аккумуляторе появляется число 1