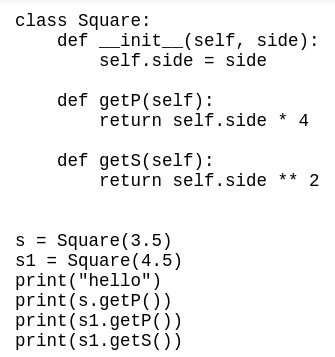
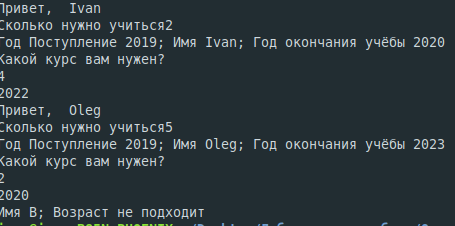
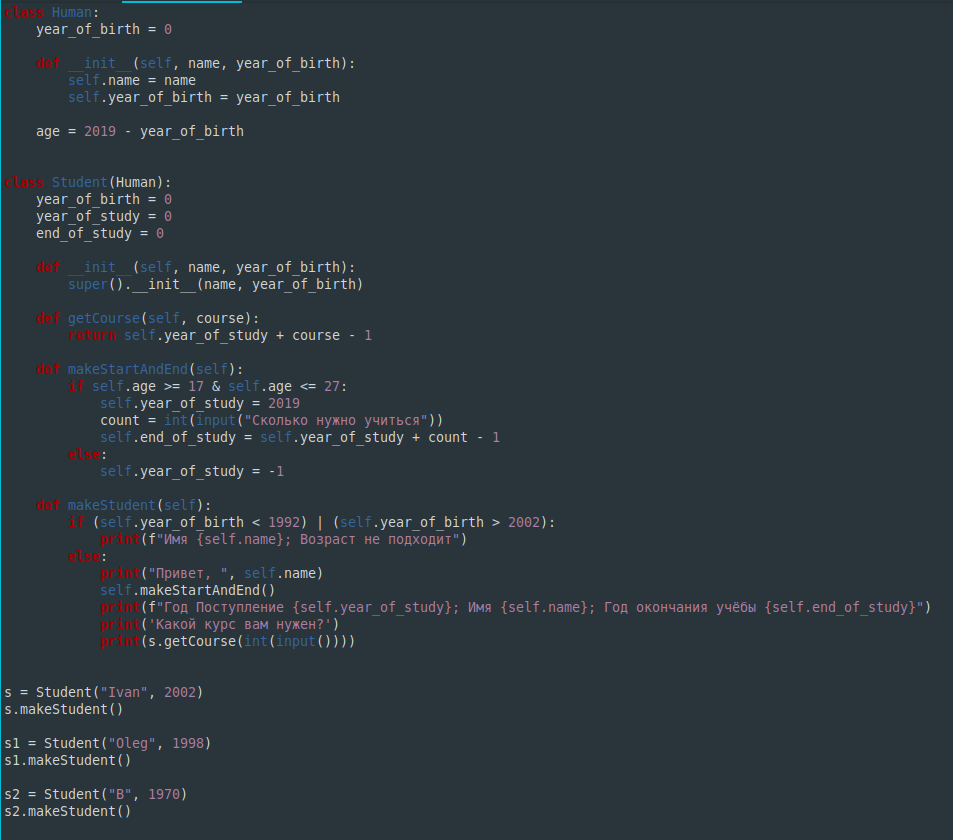
**Задачи на классы**

1. Создайте класс «Квадрат» с полем, принимающим значение стороны квадрата, и методами для получения периметра и площади. Для одного объекта класса выведите периметр, для другого периметр и площадь.

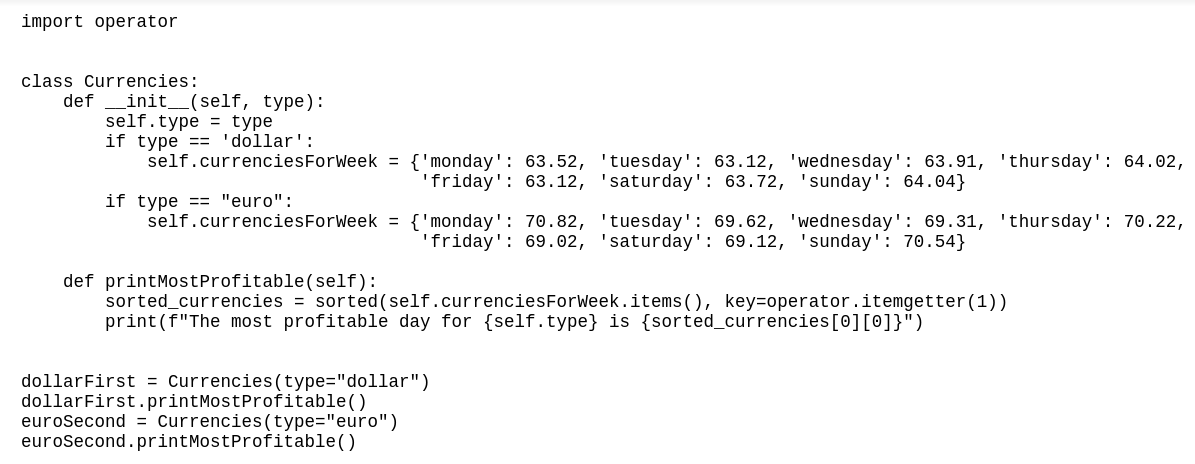




1. Создайте класс «Человек», с полями: имя и год рождения. Определите методы задания имени, и определения возраста (на основании текущего года). Создайте производный класс «Студент», имеющий поле года обучения. Определите методы задания года обучения (в случае, если возраст абитуриента находится в определённых возрастных рамках, например, от 17 до 27, выводит сообщение с годом начала обучения) и увеличения года обучения (до достижения определённого срока, например, 5 лет, затем выводит сообщение с годом окончания обучения).Продемонстрируйте работу с полями и методами производного класса на нескольких объектах.



1. Создайте класс «Курс валют», с полями: тип валюты (например, доллар, евро) и списком значений курса валюты (по отношению к рублю) за неделю. Определите методы задания типа валюты, списка значений курса и определения самого выгодного курса (выводит значение по курсу в рублях и соответствующий день недели). Продемонстрируйте работу с полями и методами класса на нескольких объектах.



**Задача на паттерны**

Используя поведенческий паттерн «Состояние», реализуйте процесс поступления абитуриента в ВУЗ по следующей схеме:

1. Состояние «Начало» – вывод «Приём в ВУЗ открыт», «Документы поданы», «Проходной балл для 1 волны: Значение1, для 2 волны: Значение 2». Переход к состоянию «Анализ 1 волна».
2. Состояние «Анализ 1 волна» – вывод «Результат вступительных испытаний: ЗначениеN», «Конкурс 1 волна». Если «ЗначениеN» больше «Значение1», то вывод «Попадание в 1 волну» и переход к состоянию «Обработка», в противном случае – переход к состоянию «Анализ 2 волна».
3. Состояние «Анализ 2 волна» – вывод «Конкурс 2 волна». Если «ЗначениеN» больше «Значение2», то вывод «Попадание во 2 волну» и переход к состоянию «Обработка», в противном случае – вывод «Конкурс не пройден» и переход к состоянию «Окончание».
4. Состояние «Обработка» – вывод «Приказ о зачислении». Переход к состоянию «Окончание».
5. Состояние «Окончание» – вывод «Приём в ВУЗ закрыт». Состояние None.

Примечание: ЗначениеN задаётся рандомно из диапазона от 60 до 100. Значение1 и Значение2: а) фиксированы (из диапазона от 70 до 85, не рандомно), причём Значение1 строго больше Значение2 не менее, чем на 8 баллов; б) вычисляются как среднее арифметическое значение списков не менее, чем на 100 элементов (из диапазонов соответственно от 78 до 85 и от 70 до 85).

from abc import ABCMeta, abstractmethod

from random import randint

class State(metaclass=ABCMeta):

@abstractmethod

def do\_something(self) -> None:

pass

class Begin(State):

def do\_something(self) -> None:

print(

f"Приём в ВУЗ открыт.\nДокументы поданы.\nПроходной балл для 1 волны: {Admission.n1}, для волны 2: {Admission.n2}")

class Analyze\_First\_Wave(State):

def do\_something(self) -> None:

print(f"Результат вступительных испытаний: {Admission.mark}.\nКонкурс 1 волна.")

if Admission.mark > Admission.n1:

print("Попадание в 1 волну.")

class Analyze\_Second\_Wave(State):

def do\_something(self) -> None:

print(f"Конкурс 2 волна.")

if Admission.mark >= Admission.n2:

print("Попадание во 2 волну")

else:

print("Конкурс не пройден")

class Treatment(State):

def do\_something(self) -> None:

print("Приказ о зачислении.")

class End(State):

def do\_something(self) -> None:

print("Приём в ВУЗ закрыт.")

class Admission:

mark = randint(60, 100)

numbers1 = []

for i in range(100):

numbers1.append(randint(78, 85))

numbers2 = []

for i in range(100):

numbers2.append(randint(70, 85))

n1 = int(input("Введите проходной балл для первой волны"))

n2 = int(input("Введите проходной балл для второй волны"))

def \_\_init\_\_(self, state: State) -> None:

self.\_state = state

def change\_state(self, state: State) -> None:

self.\_state = state

def do\_something(self) -> None:

self.\_state.do\_something()

begin = Begin()

afw = Analyze\_First\_Wave()

asw = Analyze\_Second\_Wave()

treatment = Treatment()

end = End()

# first student

adm = Admission(begin)

adm.do\_something()

adm.change\_state(afw)

adm.do\_something()

if adm.mark >= Admission.n1:

adm.change\_state(treatment)

adm.do\_something()

else:

adm.change\_state(asw)

adm.do\_something()

if Admission.mark >= Admission.n2:

adm.change\_state(treatment)

adm.do\_something()

adm.change\_state(end)

adm.do\_something()

print("")

# second student

adm1 = Admission(begin)

adm1.do\_something()

adm1.change\_state(afw)

adm1.do\_something()

if adm1.mark >= Admission.n1:

adm1.change\_state(treatment)

adm1.do\_something()

else:

adm1.change\_state(asw)

adm1.do\_something()

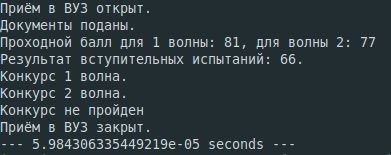
if Admission.mark >= Admission.n2:

adm1.change\_state(treatment)

adm1.do\_something()

adm1.change\_state(end)

adm1.do\_something()



from abc import ABCMeta, abstractmethod

from random import randint

class State(metaclass=ABCMeta):

@abstractmethod

def do\_something(self) -> None:

pass

class Begin(State):

def do\_something(self) -> None:

print(

f"Приём в ВУЗ открыт.\nДокументы поданы.\nПроходной балл для 1 волны: {Admission.n1}, для волны 2: {Admission.n2}")

class Analyze\_First\_Wave(State):

def do\_something(self) -> None:

print(f"Результат вступительных испытаний: {Admission.mark}.\nКонкурс 1 волна.")

if Admission.mark > Admission.n1:

print("Попадание в 1 волну.")

class Analyze\_Second\_Wave(State):

def do\_something(self) -> None:

print(f"Конкурс 2 волна.")

if Admission.mark >= Admission.n2:

print("Попадание во 2 волну")

else:

print("Конкурс не пройден")

class Treatment(State):

def do\_something(self) -> None:

print("Приказ о зачислении.")

class End(State):

def do\_something(self) -> None:

print("Приём в ВУЗ закрыт.")

class Admission:

mark = randint(60, 100)

numbers1 = []

for i in range(100):

numbers1.append(randint(78, 85))

numbers2 = []

for i in range(100):

numbers2.append(randint(70, 85))

n1 = int(sum(numbers1) / len(numbers1))

n2 = int(sum(numbers2) / len(numbers2))

def \_\_init\_\_(self, state: State) -> None:

self.\_state = state

def change\_state(self, state: State) -> None:

self.\_state = state

def do\_something(self) -> None:

self.\_state.do\_something()

begin = Begin()

afw = Analyze\_First\_Wave()

asw = Analyze\_Second\_Wave()

treatment = Treatment()

end = End()

# first student

adm = Admission(begin)

adm.do\_something()

adm.change\_state(afw)

adm.do\_something()

if adm.mark >= Admission.n1:

adm.change\_state(treatment)

adm.do\_something()

else:

adm.change\_state(asw)

adm.do\_something()

if Admission.mark >= Admission.n2:

adm.change\_state(treatment)

adm.do\_something()

adm.change\_state(end)

adm.do\_something()

print("")

# second student

adm1 = Admission(begin)

adm1.do\_something()

adm1.change\_state(afw)

adm1.do\_something()

if adm1.mark >= Admission.n1:

adm1.change\_state(treatment)

adm1.do\_something()

else:

adm1.change\_state(asw)

adm1.do\_something()

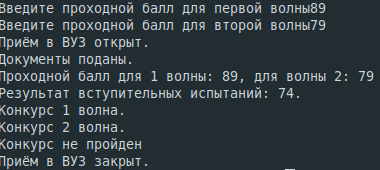
if Admission.mark >= Admission.n2:

adm1.change\_state(treatment)

adm1.do\_something()

adm1.change\_state(end)

adm1.do\_something()

  
Добавление: Паттерн State в Java

package com.company;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Random;

public class Main{

public static void main(String[] args)

{

Admission.makeScores();

for (int i = 0; i < 100; i++) {

Admission.studentAdmission(i);

}

System.out.println("Поступило " + Admission.success + " абитурентов");

}

}

interface State {

String getName(StateContext context);

void doSomething(StateContext context);

}

class StateContext {

private State state = new Begin();

void doSomething()

{

state.doSomething(this);

}

void setState(State state)

{

this.state = state;

}

public State getState()

{

return state;

}

}

class Begin implements State{

private static final String NAME = "begin";

@Override

public String getName(StateContext context) {

return NAME;

}

@Override

public void doSomething(StateContext context)

{

System.out.println("Приём в ВУЗ открыт.");

System.out.println("Документы поданы.");

System.out.println("Проходной балл для 1 волны: " + Admission.n1 + ", для волны 2: " + Admission.n2);

}

}

class AnalyzeFirstWave implements State{

private static final String NAME = "analyze\_first\_wave";

@Override

public String getName(StateContext context) {

return NAME;

}

@Override

public void doSomething(StateContext context)

{

System.out.println("Результат вступительных испытаний: " + Admission.mark + ".");

System.out.println("Конкурс 1 волна.");

if (Admission.mark > Admission.n1) System.out.println("Попадание в 1 волну.");

}

}

class AnalyzeSecondWave implements State{

private static final String NAME = "analyze\_first\_wave";

@Override

public String getName(StateContext context) {

return NAME;

}

@Override

public void doSomething(StateContext context)

{

System.out.println("Конкурс 2 волна.");

if (Admission.mark >= Admission.n2)

System.out.println("Попадание во 2 волну");

else

System.out.println("Конкурс не пройден");

}

}

class Treatment implements State{

private static final String NAME = "treatment";

@Override

public String getName(StateContext context) {

return NAME;

}

@Override

public void doSomething(StateContext context)

{

System.out.println("Приказ о зачислении.");

}

}

class End implements State{

private static final String NAME = "end";

@Override

public String getName(StateContext context) {

return NAME;

}

@Override

public void doSomething(StateContext context)

{

System.out.println("Приём в ВУЗ закрыт.");

}

}

class Admission{

static int success;

static int n1, n2;

static void makeScores()

{

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

try {

System.out.println("Введите проходной балл для первой волны: ");

n1 = Integer.parseInt(reader.readLine());

System.out.println("Введите проходной балл для второй волны: ");

n2 = Integer.parseInt(reader.readLine());

} catch (Exception e) {

e.printStackTrace();

}

}

static int mark;

private void makeMark()

{

Random random = new Random();

mark = 65 + random.nextInt(100 - 65);

}

static void studentAdmission(int x)

{

Admission student = new Admission();

student.makeMark();

System.out.println("Student № " + x);

StateContext context = new StateContext();

context.doSomething();

context.setState(new AnalyzeFirstWave());

context.doSomething();

if (mark >= Admission.n1){

context.setState(new Treatment());

context.doSomething();

success += 1;

}

else{

context.setState(new AnalyzeSecondWave());

context.doSomething();

if (mark >= Admission.n2){

context.setState(new Treatment());

context.doSomething();

success += 1;

}

}

context.setState(new End());

context.doSomething();

System.out.println();

}

}