# Введение в ООП. Полиморфизм

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или testmodule.py |
| Вывод | стандартный вывод или output.txt |

Напишите класс **Selector**.  
Экземпляр этого класса при инициализации получает список чисел. Вызов метода get\_odds возвращает нечётные числа из первоначального списка, вызов get\_evens — чётные.

Числа должны идти в том же порядке, в котором они были в изначальном списке.

## Формат ввода

Каждый тест представляет собой код, в котором будет использоваться ваш класс.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| from solution import Selector  values = [11, 12, 13, 14, 15, 16, 22, 44, 66]  selector = Selector(values)  odds = selector.get\_odds()  evens = selector.get\_evens()  print(' '.join(map(str, odds)))  print(' '.join(map(str, evens))) | 11 13 15  12 14 16 22 44 66 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| from solution import Selector  values = [6, 6, 0, 4, 8, 7, 6, 4, 7, 5]  selector = Selector(values)  odds = selector.get\_odds()  evens = selector.get\_evens()  print(' '.join(map(str, odds)))  print(' '.join(map(str, evens))) | 7 7 5  6 6 0 4 8 6 4 |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| from solution import Selector  values = []  selector = Selector(values)  odds = selector.get\_odds()  evens = selector.get\_evens()  print(' '.join(map(str, odds)))  print(' '.join(map(str, evens))) |  |

# Вывод предложений

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или testmodule.py |
| Вывод | стандартный вывод или output.txt |

Напишите два класса: **LeftParagraph** и **RightParagraph** для печати абзаца с выравниванием по левому и правому краю.

При инициализации экземпляры обоих классов должны принимать целое число — ширину поля вывода. В обоих классах нужно реализовать метод add\_word для добавления слова в абзац и метод end, выводящий полученный абзац на печать и начинающий формирование нового.

Гарантируется, что длина любого слова меньше ширины поля вывода.

## Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваши классы.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашими классами, а его вывод сравнивается с правильным решением.

## Пример

| Ввод | Вывод |
| --- | --- |
| from solution import LeftParagraph, RightParagraph  lp = LeftParagraph(8)  lp.add\_word('abc')  lp.add\_word('defg')  lp.add\_word('hi')  lp.add\_word('jklmnopq')  lp.add\_word('r')  lp.add\_word('stuv')  lp.end()  print()  rp = RightParagraph(8)  rp.add\_word('abc')  rp.add\_word('defg')  rp.add\_word('hi')  rp.add\_word('jklmnopq')  rp.add\_word('r')  rp.add\_word('stuv')  rp.end()  print() | abc defg  hi  jklmnopq  r stuv  abc defg  hi  jklmnopq  r stuv |

**Форматы дат**

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или testmodule.py |
| Вывод | стандартный вывод или output.txt |

Форматы записи дат в виде строки в США и Европе отличаются.  
В США принят формат мм.дд.гггг, в Европе — дд.мм.гггг, где дд — день (дополняется нулём слева, если число меньше 10), мм — месяц (так же дополняется нулём слева), гггг — год.  
Например, 10 апреля 2000 года будет записано в американском формате как 04.10.2000, а в европейском — как 10.04.2000. Все годы в задаче — четырёхзначные.

Реализуйте классы **AmericanDate** и **EuropeanDate**. При инициализации они должны принимать год, месяц и число (именно в этом порядке). Так же должны быть реализованы методы set\_year, set\_month, set\_day для изменения одной из компонентов даты, и get\_year, get\_month, get\_day для чтения компонентов даты. Метод format должен возвращать строковое представление (своё для каждого класса).

Гарантируется, что все даты в тестах корректны и существуют в календаре.

**Формат ввода**

Каждый тест представляет собой код, в котором будут использоваться ваши классы.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашими классами, а его вывод сравнивается с правильным решением.

**Пример**

| Ввод | Вывод |
| --- | --- |
| from solution import AmericanDate, EuropeanDate  american = AmericanDate(2000, 4, 10)  european = EuropeanDate(2000, 4, 10)  print(american.format())  print(european.format()) | 04.10.2000  10.04.2000 |

# Статистика

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.py |
| Вывод | стандартный вывод или output.txt |

Реализуйте классы **MinStat**, **MaxStat**, **AverageStat**, которые будут находить минимум, максимум и среднее арифметическое последовательности целых чисел.

Экземпляры классов инициализируются без аргументов. Метод add\_number должен добавлять в статистику число, которое будет учтено при вычислении финального результата методом result. Для экземпляров MinStat и MaxStat result должен возвращать целое число, для AverageStat — число типа float (это число будет сравниваться с правильным ответом до седьмой значащей цифры).

Если в последовательности отсутствуют числа и статистические величины вычислить невозможно, метод result должен возвращать None.

## Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваши классы.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашими классами, а его вывод сравнивается с правильным решением.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| from solution import MinStat, MaxStat, AverageStat  values = [1, 2, 4, 5]  mins = MinStat()  maxs = MaxStat()  average = AverageStat()  for v in values:  mins.add\_number(v)  maxs.add\_number(v)  average.add\_number(v)  print(mins.result(), maxs.result(), '{:<05.3}'.format(average.result())) | 1 5 3.000 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| from solution import MinStat, MaxStat, AverageStat  mins = MinStat()  maxs = MaxStat()  average = AverageStat()  print(mins.result(), maxs.result(), average.result()) | None None None |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| from solution import MinStat, MaxStat, AverageStat  values = [1, 0, 0]  mins = MinStat()  maxs = MaxStat()  average = AverageStat()  for v in values:  mins.add\_number(v)  maxs.add\_number(v)  average.add\_number(v)  print(mins.result(), maxs.result(), '{:<05.3}'.format(average.result())) | 0 1 0.333 |

# Таблица

Kлассная работа

макс. 3 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.py |
| Вывод | стандартный вывод или output.txt |

Реализуйте класс **Table**, который хранит целые числа в двумерной таблице. При инициализации Table(rows, cols) экземпляру передаются число строк и столбцов в таблице. Строки и столбцы нумеруются с нуля. Ячейки таблицы инициализируются нулями.

table.get\_value(row, col) — прочитать значение из ячейки со строкой row, столбцом col. Если ячейка с индексами row и col не лежит внутри таблицы, нужно вернуть None.

table.set\_value(row, col, value) — записать число в ячейку со строкой row, столбцом col. Гарантируется, что в тестах будет в запись только в ячейки внутри таблицы.

table.n\_rows() — вернуть число строк в таблице.

table.n\_cols() — вернуть число столбцов в таблице.

## Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваш класс.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| from solution import Table  tab = Table(3, 5)  tab.set\_value(0, 1, 10)  tab.set\_value(1, 2, 20)  tab.set\_value(2, 3, 30)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print() | 0 10 0 0 0  0 0 20 0 0  0 0 0 30 0 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| from solution import Table  tab = Table(2, 2)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.set\_value(0, 0, 10)  tab.set\_value(0, 1, 20)  tab.set\_value(1, 0, 30)  tab.set\_value(1, 1, 40)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  for i in range(-1, tab.n\_rows() + 1):  for j in range(-1, tab.n\_cols() + 1):  print(tab.get\_value(i, j), end=' ')  print()  print() | 0 0  0 0  10 20  30 40  None None None None  None 10 20 None  None 30 40 None  None None None None |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| from solution import Table  tab = Table(1, 1)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.set\_value(0, 0, 1000)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  for i in range(-1, tab.n\_rows() + 1):  for j in range(-1, tab.n\_cols() + 1):  print(tab.get\_value(i, j), end=' ')  print()  print() | 0  1000  None None None  None 1000 None  None None None |

# Прямоугольники

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или test.py |
| Вывод | стандартный вывод или output.txt |

Реализуйте класс **Rectangle** для описания прямоугольника, стороны которого параллельны осям координат.

При инициализации экземпляра передаются координаты левой нижней точки прямоугольника x и y, а также его ширина и высота w и h. Таким образом, координаты верхнего правого угла — (x + w) и (y + h).

При вызове метода intersection (например, rect1.intersection(rect2)) должен возвращаться прямоугольник, который возникает как пересечение rect1 и rect2. Если прямоугольники не пересекаются, должен возвращаться объект None.

Также необходимо реализовать метод get для каждого из атрибутов класса, возвращающий заданное значение данного атрибута. Пример: get\_\*() - возвращает значение атрибута \*.

Гарантируется, что во входных данных ширина и высота любого прямоугольника положительны.

Если пересечением прямоугольников является точка или отрезок, то следует считать, что они не пересекаются.

## Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваш класс.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| from solution import Rectangle  rect1 = Rectangle(0, 0, 10, 10)  rect2 = Rectangle(5, 5, 10, 10)  rect3 = rect1.intersection(rect2)  if rect3 is None:  print('No intersection')  else:  print(rect3.get\_x(), rect3.get\_y(), rect3.get\_w(), rect3.get\_h()) | 5 5 5 5 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| from solution import Rectangle  rect1 = Rectangle(0, 0, 10, 10)  rect2 = Rectangle(10, 0, 10, 10)  rect3 = rect1.intersection(rect2)  if rect3 is None:  print('No intersection')  else:  print(rect3.get\_x(), rect3.get\_y(), rect3.get\_w(), rect3.get\_h()) | No intersection |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| from solution import Rectangle  rect1 = Rectangle(3, 5, 2, 1)  rect2 = Rectangle(1, 2, 10, 10)  rect3 = rect1.intersection(rect2)  if rect3 is None:  print('No intersection')  else:  print(rect3.get\_x(), rect3.get\_y(), rect3.get\_w(), rect3.get\_h()) | 3 5 2 1 |

# Таблица с изменяемым размером

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.py |
| Вывод | стандартный вывод или output.txt |

Реализуйте класс **Table**, который хранит целые числа в двумерной таблице. При инициализации Table(rows, cols) экземпляру передаются число строк и столбцов в таблице. Строки и столбцы нумеруются с нуля.

table.get\_value(row, col) — прочитать значение из ячейки в строке row, столбце col. Если ячейка с индексами row и col не лежит внутри таблицы, нужно вернуть None.

table.set\_value(row, col, value) — записать число в ячейку строки row, столбца col. Гарантируется, что в тестах будет в запись только в ячейки внутри таблицы.

table.n\_rows() — вернуть число строк в таблице

table.n\_cols() — вернуть число столбцов в таблице

table.delete\_row(row) — удалить строку с номером row

table.delete\_col(col) — удалить колонку с номером col

table.add\_row(row) — добавить в таблицу новую строку с индексом row.  
Номера строк >= row, должны увеличиться на единицу. Новая строка состоит из нулей.

table.add\_col(col) — добавить в таблицу новую колонку с индексом col.  
Номера колонок >= col, должны увеличиться на единицу. Новая колонка состоит из нулей.

## Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваш класс.  
Файл c решением не обязательно называть solution.py, он будет переименован автоматически.  
Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| from solution import Table  tab = Table(3, 5)  tab.set\_value(0, 1, 10)  tab.set\_value(1, 2, 20)  tab.set\_value(2, 3, 30)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.add\_row(1)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print() | 0 10 0 0 0  0 0 20 0 0  0 0 0 30 0  0 10 0 0 0  0 0 0 0 0  0 0 20 0 0  0 0 0 30 0 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| from solution import Table  tab = Table(2, 2)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.set\_value(0, 0, 10)  tab.set\_value(0, 1, 20)  tab.set\_value(1, 0, 30)  tab.set\_value(1, 1, 40)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  for i in range(-1, tab.n\_rows() + 1):  for j in range(-1, tab.n\_cols() + 1):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.add\_row(0)  tab.add\_col(1)  for i in range(-1, tab.n\_rows() + 1):  for j in range(-1, tab.n\_cols() + 1):  print(tab.get\_value(i, j), end=' ')  print()  print() | 0 0  0 0  10 20  30 40  None None None None  None 10 20 None  None 30 40 None  None None None None  None None None None None  None 0 0 0 None  None 10 0 20 None  None 30 0 40 None  None None None None None |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| from solution import Table  tab = Table(1, 1)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.set\_value(0, 0, 1000)  for i in range(tab.n\_rows()):  for j in range(tab.n\_cols()):  print(tab.get\_value(i, j), end=' ')  print()  print()  for i in range(-1, tab.n\_rows() + 1):  for j in range(-1, tab.n\_cols() + 1):  print(tab.get\_value(i, j), end=' ')  print()  print()  tab.add\_row(0)  tab.add\_row(2)  tab.add\_col(0)  tab.add\_col(2)  tab.set\_value(0, 0, 2000)  tab.set\_value(0, 2, 3000)  tab.set\_value(2, 0, 4000)  tab.set\_value(2, 2, 5000)  for i in range(-1, tab.n\_rows() + 1):  for j in range(-1, tab.n\_cols() + 1):  print(tab.get\_value(i, j), end=' ')  print()  print() | 0  1000  None None None  None 1000 None  None None None  None None None None None  None 2000 0 3000 None  None 0 1000 0 None  None 4000 0 5000 None  None None None None None |

# СМС-рассылка

Kлассная работа

макс. 3 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или testmodule.py |
| Вывод | стандартный вывод или output.txt |

Для создания смс-рассылки напишите классы **Person** и **Company**, а также функцию **send\_sms**.  
Объект класса **Person** при инициализации принимает значения имени, отчества и фамилии человека, а также словарь с номерами его телефонов.  
Класс должен содержать следующие методы:

1. **get\_phone()** – возвращает телефон из словаря по ключу 'private' или None, если такого телефона нет,
2. **get\_name()** – возвращает фамилию, имя и отчество человека через пробел,
3. **get\_work\_phone()** – возвращает телефон из словаря по ключу 'work' или None, если такого телефона нет,
4. **get\_sms\_text()** – возвращает текст «Уважаемый <имя> <отчество>! Примите участие в нашем беспроигрышном конкурсе для физических лиц».

Объект класса **Company** при инициализации принимает название компании, тип компании, словарь с её телефонами, а также неограниченное количество работников компании (объектов класса **Person**).  
Класс должен содержать следующие методы:

1. **get\_phone()** – возвращает телефон из словаря по ключу 'contact', если его нет, то телефон первого работника, у которого есть телефон по ключу 'work', или None, если таких работников не найдётся,
2. **get\_name()** – возвращает название компании,
3. **get\_sms\_text()** – возвращает текст «Для компании <название компании> есть супер предложение! Примите участие в нашем беспроигрышном конкурсе для <тип компании>».

Функция **send\_sms** должна принимать неограниченное количество объектов класса **Person** или **Company** и в случае, если найден номер для отправки (с помощью метода get\_phone()), выводить сообщение «Отправлено СМС на номер <номер> с текстом: <Текст СМС>», иначе – текст «Не удалось отправить сообщение абоненту: <ФИО человека или название компании>».

### Пример 1

| Ввод | Вывод |
| --- | --- |
| from solution import Person, Company, send\_sms  person1 = Person("Ivan", "Ivanovich", "Ivanov", {"private": 123, "work": 456})  person2 = Person("Ivan", "Petrovich", "Petrov", {"private": 789})  person3 = Person("Ivan", "Petrovich", "Sidorov", {"work": 789})  person4 = Person("John", "Unknown", "Doe", {})  company1 = Company("Bell", "ООО", {"contact": 111}, person3, person4)  company2 = Company("Cell", "АО", {"non\_contact": 222}, person2, person3)  company3 = Company("Dell", "Ltd", {"non\_contact": 333}, person2, person4)  send\_sms(person1, person2, person3, person4, company1, company2, company3) | Отправлено СМС на номер 123 с текстом: Уважаемый Ivan Ivanovich! Примите участие в нашем беспроигрышном конкурсе для физических лиц  Отправлено СМС на номер 789 с текстом: Уважаемый Ivan Petrovich! Примите участие в нашем беспроигрышном конкурсе для физических лиц  Не удалось отправить сообщение абоненту: Sidorov Ivan Petrovich  Не удалось отправить сообщение абоненту: Doe John Unknown  Отправлено СМС на номер 111 с текстом: Для компании Bell есть супер предложение! Примите участие в нашем беспроигрышном конкурсе для ООО  Отправлено СМС на номер 789 с текстом: Для компании Cell есть супер предложение! Примите участие в нашем беспроигрышном конкурсе для АО  Не удалось отправить сообщение абоненту: Dell |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| from solution import Person, Company, send\_sms  person1 = Person("Степан", "Петрович", "Джобсов", {"private": 555})  person2 = Person("Боря", "Иванович", "Гейтсов", {"private": 777, "work": 888})  person3 = Person("Семен", "Робертович", "Возняцкий", {"work": 789})  person4 = Person("Леонид", "Арсенович", "Торвальдсон", {})  company1 = Company("Яблочный комбинат", "ООО", {"contact": 111}, person1, person3)  company2 = Company("ПластОкно", "АО", {"non\_contact": 222}, person2)  company3 = Company("Пингвинья ферма", "Ltd", {"non\_contact": 333}, person4)  send\_sms(person1, person2, person3, person4, company1, company2, company3) | Отправлено СМС на номер 555 с текстом: Уважаемый Степан Петрович! Примите участие в нашем беспроигрышном конкурсе для физических лиц  Отправлено СМС на номер 777 с текстом: Уважаемый Боря Иванович! Примите участие в нашем беспроигрышном конкурсе для физических лиц  Не удалось отправить сообщение абоненту: Возняцкий Семен Робертович  Не удалось отправить сообщение абоненту: Торвальдсон Леонид Арсенович  Отправлено СМС на номер 111 с текстом: Для компании Яблочный комбинат есть супер предложение! Примите участие в нашем беспроигрышном конкурсе для ООО  Отправлено СМС на номер 888 с текстом: Для компании ПластОкно есть супер предложение! Примите участие в нашем беспроигрышном конкурсе для АО  Не удалось отправить сообщение абоненту: Пингвинья ферма |

## Примечания

Задача дополнительно проверяется преподавателем.

[Открыть редактор](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2506/tasks/17703/solutions)