

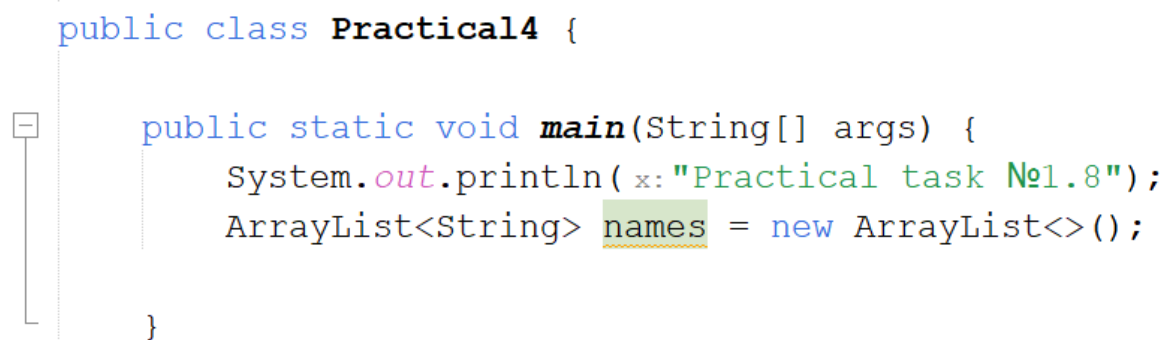
ЗАНЯТИЕ 1.8

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Тема: Создание Java-приложения для работы с коллекциями

Упражнение №1

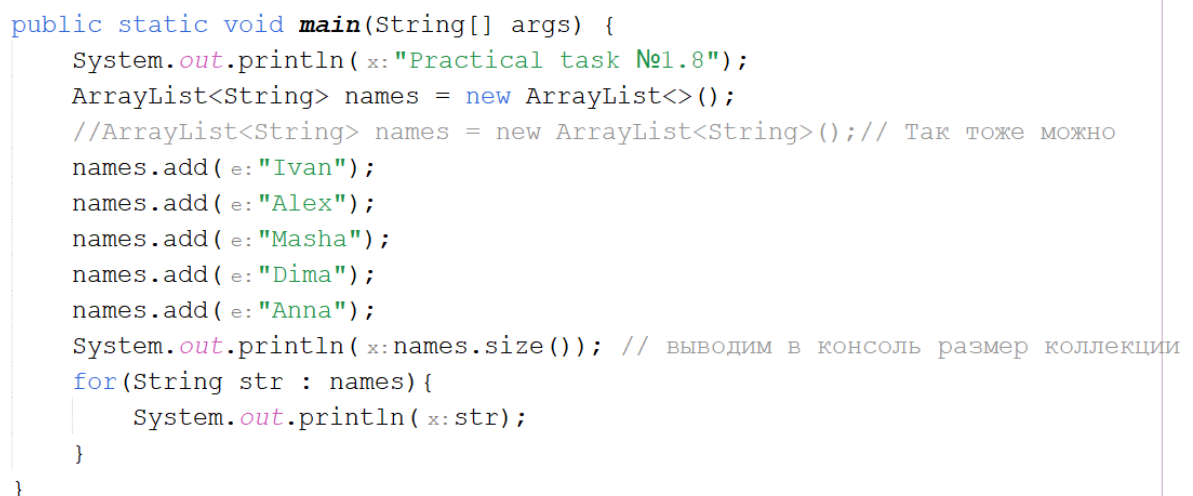
Создайте новое java-приложение. В методе main создадим коллекцию – **ArrayList**, которая будет содержать элементы типа **String** (имена людей).



```
public class Practical4 {  
  
    public static void main(String[] args) {  
        System.out.println( x: "Practical task №1.8");  
        ArrayList<String> names = new ArrayList<>();  
    }  
}
```

Рисунок 1

Добавим в созданную коллекцию несколько значений – имён. При использовании метода add без параметров элемент добавляется в конец коллекции. И с помощью расширенного цикла for отобразим в консоль значения:



```
public static void main(String[] args) {  
    System.out.println( x: "Practical task №1.8");  
    ArrayList<String> names = new ArrayList<>();  
    //ArrayList<String> names = new ArrayList<String>(); // Так тоже можно  
    names.add( e: "Ivan");  
    names.add( e: "Alex");  
    names.add( e: "Masha");  
    names.add( e: "Dima");  
    names.add( e: "Anna");  
    System.out.println( x: names.size()); // выводим в консоль размер коллекции  
    for(String str : names){  
        System.out.println( x: str);  
    }  
}
```

Рисунок 2

Заменим элемент, хранящийся по нулевому индексу – запишем туда имя «Lena». Также удалим два элемента из коллекции: элемент со значением «Dima» и элемент, хранящийся по индексу 3. Итоговый лист отобразим в консоль. Размер коллекции изменился (стал равен 3), элементы сдвинулись и поменяли свои индексы.

```
public static void main(String[] args) {
    System.out.println("Practical task №1.8");
    ArrayList<String> names = new ArrayList<>();
    //ArrayList<String> names = new ArrayList<String>(); // Так тоже можно
    names.add(e: "Ivan");
    names.add(e: "Alex");
    names.add(e: "Masha");
    names.add(e: "Dima");
    names.add(e: "Anna");
    System.out.println("names.size()"); // выводим в консоль размер коллекции
    for(String str : names){
        System.out.println(str);
    }
    names.set(index: 0, element: "Lena");
    names.remove(o: "Dima");
    names.remove(index: 3);
    System.out.println("names");
}
```

com.mirea.kt.practical4.Practical4 >

Output - Run (Practical4) ×

```
cd C:\Users\User\Documents\NetBeansProjects\Practical4; "JAVA_HOME=C:\\Program Files'
Running NetBeans Compile On Save execution. Phase execution is skipped and output di:
Scanning for projects...

-----< com.mirea.kt:Practical4 >-----
Building Practical4 1.0
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practical4 ---
Practical task №1.8
5
Ivan
Alex
Masha
Dima
Anna
[Lena, Alex, Masha]
```

Рисунок 3

Выполним сортировку с помощью метода **sort** класса **Collections**.

```
3         System.out.println( x:names);
4         Collections.sort( list:names);
5         System.out.println( x:names);
6     }
7
com.mirea.kt.practical4.Practical4 > main > names >
Output - Run (Practical4) x
cd C:\Users\User\Documents\NetBeansProjects\Practical4; "JAVA_H
Running NetBeans Compile On Save execution. Phase execution is
Scanning for projects...

-----< com.mirea.kt:Practical4 >-----
Building Practical4 1.0
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practical4 ---
Practical task №1.8
5
Ivan
Alex
Masha
Dima
Anna
[Lena, Alex, Masha]
[Alex, Lena, Masha]
-----
```

Рисунок 4

Теперь лист отсортирован по значениям!

Добавим в текущий проект класс Person. Определим в нем переменные: имя, фамилия и возраст. Так же реализуем конструктор, сеттеры и геттеры.

```

*/
public class Person {
    private String firstName;
    private String lastName;
    private int age;

    public Person(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }
}

```

Рисунок 5

В главном классе в методе `main` создадим **ArrayList**, который будет содержать элементы типа **Person**. Добавим в созданный лист несколько объектов **Person**. С помощью цикла **for** отобразим в консоль имя и фамилию каждого элемента коллекции.

```

public static void main(String[] args) {
    System.out.println("Practical task №1.8");
    List<Person> persons = new ArrayList<>();
    persons.add(new Person( firstName: "Ivan",   lastName: "Ivanov",   age: 20));
    persons.add(new Person( firstName: "Masha",  lastName: "Rasputina", age: 19));
    persons.add(new Person( firstName: "Alex",   lastName: "Alexsandrov", age: 17));
    persons.add(new Person( firstName: "Jonh",   lastName: "Wick",      age: 40));
    persons.add(new Person( firstName: "Ivan",   lastName: "Petrov",   age: 16));
    for(Person p : persons){
        System.out.println(p.getFirstName() + " " + p.getLastName());
    }
}

```

com.mirea.kt.practical4.Practical4 >

put - Run (Practical4) ×

```

cd C:\Users\User\Documents\NetBeansProjects\Practical4; "JAVA_HOME=C:\Program Files\Java\jdk-11.0.2\bin" java -jar Practical4.jar
Running NetBeans Compile On Save execution. Phase execution is skipped and output
Scanning for projects...

-----< com.mirea.kt:Practical4 >-----
Building Practical4 1.0
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practical4 ---
Practical task №1.8
Ivan Ivanov
Masha Rasputina
Alex Alexsandrov
Jonh Wick
Ivan Petrov
-----

```

Рисунок 6

Как реализовать удаление из коллекции «людей», возраст которых не превышает 18 (убрать несовершеннолетних)? Проверьте, будет ли работать такой код?

```
for(Person p : persons){
    if(p.getAge() < 18){
        persons.remove(o:p);
    }
}
```

Рисунок 7

Удаление из коллекции внутри цикла нужно реализовывать с помощью итератора!

```
List<Person> persons = new ArrayList<>();
persons.add(new Person( firstName:"Ivan",  lastName:"Ivanov",  age:20));
persons.add(new Person( firstName:"Masha",  lastName:"Rasputina",  age:19));
persons.add(new Person( firstName:"Alex",  lastName:"Alexsandrov",  age:17));
persons.add(new Person( firstName:"Jonh",  lastName:"Wick",  age:40));
persons.add(new Person( firstName:"Ivan",  lastName:"Petrov",  age:16));
System.out.println("List size: " + persons.size());
Iterator<Person> iter = persons.iterator();
while (iter.hasNext()) {
    Person p = iter.next();
    if(p.getAge() < 18){
        iter.remove();
    }
}
System.out.println("List size: " + persons.size());
```

Рисунок 8

Реализуем сортировку элементов коллекции – по фамилии человека. Для этого нам необходимо реализовать интерфейс **Comparable** в классе **Person** и переопределить метод этого интерфейса – **compareTo**. Этот метод сравнивает текущий объект с объектом, переданным в качестве параметра. Если этот метод возвращает отрицательное число, то текущий объект будет располагаться перед тем, который передается через параметр. Если метод вернет положительное число, то, наоборот, после второго объекта. Если метод возвратит ноль, значит, оба объекта равны.

```

public class Person implements Comparable<Person>{
    private String firstName;
    private String lastName;
    private int age;

    public Person(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    @Override
    public int compareTo(Person o) {
        return lastName.compareTo(o.getLastName());
    }
}

```

Рисунок 9

Теперь можно использовать метод **sort** класса **Collections**.

```

public static void main(String[] args) {
    System.out.println("Practical task №1.8");
    List<Person> persons = new ArrayList<>();
    persons.add(new Person( firstName:"Ivan",   lastName:"Ivanov",   age:20));
    persons.add(new Person( firstName:"Masha",  lastName:"Rasputina", age:19));
    persons.add(new Person( firstName:"Alex",   lastName:"Alexsandrov", age:17));
    persons.add(new Person( firstName:"Jonh",   lastName:"Wick",     age:40));
    persons.add(new Person( firstName:"Ivan",   lastName:"Petrov",   age:16));
    System.out.println("List size: " + persons.size());
    Collections.sort( list:persons);
    for(Person p : persons){
        System.out.println(p.getLastName() + " " + p.getFirstName());
    }
}

```

m.mirea.kt.practical4.Practical4 > main >

put - Run (Practical4) ×

```

cd C:\Users\User\Documents\NetBeansProjects\Practical4; "JAVA_HOME=C:\\Program Fil
Running NetBeans Compile On Save execution. Phase execution is skipped and output
Scanning for projects...

-----< com.mirea.kt:Practical4 >-----
Building Practical4 1.0
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practical4 ---
Practical task №1.8
List size: 5
Alexsandrov Alex
Ivanov Ivan
Petrov Ivan
Rasputina Masha
Wick Jonh

```

Рисунок 10

Упражнение №2

Создайте новый проект или продолжите работать в текущем (старый код в методе `main` можно закомментировать).

В методе `main` создадим отображение (ассоциативный массив) – **HashMap**, которое будет хранить пары «ключ-значения», где в качестве значений будут переменные типа `Person`, а в качестве ключей будут использоваться целые числа (номер паспорта). Заполним ассоциативный массив несколькими значениями с помощью метода `put`:

```
public class Practical4 {  
    public static void main(String[] args) {  
        System.out.println("Practical task №1.8");  
        HashMap<Integer, Person> personsMap = new HashMap<>();  
        personsMap.put(key: 123456, new Person(firstName: "Ivan", lastName: "Ivanov", age: 20));  
        personsMap.put(key: 223344, new Person(firstName: "Masha", lastName: "Rasputina", age: 19));  
        personsMap.put(key: 334455, new Person(firstName: "Jonh", lastName: "Wick", age: 40));  
    }  
}
```

Рисунок 11

Integer – класс-обертка для типа «`int`». Для коллекций и отображений необходимо использовать только ссылочные типы данных. Так же классы-обертки предоставляют дополнительные методы для конвертации и преобразования данных.

Для обращения к элементам `HashMap` можно использовать значение «ключа». Получим возраст для одного из людей:

```
public class Practical4 {  
    public static void main(String[] args) {  
        System.out.println("Practical task №1.8");  
        HashMap<Integer, Person> personsMap = new HashMap<>();  
        personsMap.put(key: 123456, new Person(firstName: "Ivan", lastName: "Ivanov", age: 20));  
        personsMap.put(key: 223344, new Person(firstName: "Masha", lastName: "Rasputina", age: 19));  
        personsMap.put(key: 334455, new Person(firstName: "Jonh", lastName: "Wick", age: 40));  
  
        Person personMasha = personsMap.get(key: 223344);  
        System.out.println("Masha age: " + personMasha.getAge());  
    }  
}
```

Рисунок 12

Выполним итерацию по всему `HashMap` с помощью улучшенного цикла `for` и отобразим в консоль имя, фамилию человека, а также номер его паспорта. Для этого необходимо получить множество элементов `Entry` с помощью метода `entrySet`. Этот метод позволяет работать по набору записей в `HashMap`.

```

        for (Entry<Integer, Person> entr : personsMap.entrySet()) {
            int passportId = entr.getKey();
            String fName = entr.getValue().getFirstName();
            String lName = entr.getValue().getLastName();
            System.out.println(fName + " " + lName + ", passportId: " + passportId);
        }
    }
}

com.mirea.kt.practical4.Practical4 > main > for (Entrv<Integer, Person> entr: personsMap.entrvSet()) > entr >
Output - Run (Practical4) X

cd C:\Users\User\Documents\NetBeansProjects\Practical4; "JAVA_HOME=C:\\Program Files\\Ja
Running NetBeans Compile On Save execution. Phase execution is skipped and output direct
Scanning for projects...

-----< com.mirea.kt:Practical4 >-----
Building Practical4 1.0
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practical4 ---
Practical task №1.8
Ivan Ivanov, passportId: 123456
Jonh Wick, passportId: 334455
Masha Rasputina, passportId: 223344
-----
BUILD SUCCESS

```

Рисунок 13

Как реализовать сортировку по значению ключей в HashMap? Используем **TreeMap**! Создадим новый объект класса TreeMap и в конструктор передадим ранее созданный HashMap. Теперь все ключи будут отсортированы автоматически. Это можно проверить таким же циклом for:

```

28 public static void main(String[] args) {
29     System.out.println("Practical task №1.8");
30     HashMap<Integer, Person> personsMap = new HashMap<>();
31     personsMap.put(key: 123456, new Person(firstName: "Ivan", lastName: "Ivanov", age: 20));
32     personsMap.put(key: 223344, new Person(firstName: "Masha", lastName: "Rasputina", age: 19));
33     personsMap.put(key: 334455, new Person(firstName: "Jonh", lastName: "Wick", age: 40));
34     TreeMap<Integer, Person> sortedPersonsMap = new TreeMap<>(map: personsMap);
35     for (Entry<Integer, Person> entr : sortedPersonsMap.entrySet()) {
36         int passportId = entr.getKey();
37         String fName = entr.getValue().getFirstName();
38         String lName = entr.getValue().getLastName();
39         System.out.println(fName + " " + lName + ", passportId: " + passportId);
40     }
}

com.mirea.kt.practical4.Practical4 > main >
Output - Run (Practical4) X

cd C:\Users\User\Documents\NetBeansProjects\Practical4; "JAVA_HOME=C:\\Program Files\\Java\\jdk1.
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of
Scanning for projects...

-----< com.mirea.kt:Practical4 >-----
Building Practical4 1.0
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Practical4 ---
Practical task №1.8
Ivan Ivanov, passportId: 123456
Masha Rasputina, passportId: 223344
Jonh Wick, passportId: 334455
-----

```

Рисунок 14

Индивидуальное задание

Задание: разработать консольную Java-программу для работы с интерфейсами Collections и Map в соответствии с вариантом.

Требования:

- 1) На сдачу практического задания отводится **7 дней** (или до следующего практического занятия).
- 2) Вариант определяется согласно порядковому номеру студента в журнале.
- 3) В случае дистанционного выполнения практического задания код программы необходимо выложить на Github и предоставить ссылку на него.
- 4) Итоговая программа должна компилироваться без ошибок, полностью выполнять требуемый функционал и при старте выводить в консоль номер варианта и ФИО студента.
- 5) Названия переменных и методов должны отражать суть и нести смысловую нагрузку.
- 6) Необходимо придерживаться стилистике по написанию Java-кода.
- 7) Обязательно использование реализаций интерфейсов Collection и/или Map (в зависимости от типа решаемых задач выбрать наиболее подходящую – при необходимости **быть готовым обосновать**).
- 8) Необходимо предусмотреть защиту от ввода «аномальных» (ошибочных) значений. Например, если один из параметров характеризует год, то необходимо предусмотреть соответствующую обработку вводимого с клавиатуры значения и не допустить подобного: 5555 или 2k23.

Дополнительная информация:

- 1) Перед выполнением задания рекомендуется ознакомиться с Лекцией №1.7.
- 2) Для работы со Collections и Map рекомендуется ознакомиться со справочным материалом из официальной документации Java SE:
<https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html>
<https://docs.oracle.com/javase/8/docs/api/java/util/Map.html>
- 3) Примерный вид консоли после запуска программы и работы пользователя:

```

--- exec-maven-plugin:3.0.0:exec (default-cli) @ PracticalTask1 ---
Practical task №4. Variant 1. Student Ivanov A.A. Group ABC 12345
Enter command number: 1 - show list of students, 2 - insert item, 3 - remove item, 4 - exit
5
You entered an invalid command number!
Enter command number: 1 - show list of students, 2 - insert item, 3 - remove item, 4 - exit
1
Ivan Ivanov, 20, true
Ekaterina Petrova, 20, true
Alex Jordan, 21, false
Enter command number: 1 - show list of students, 2 - insert item, 3 - remove item, 4 - exit
4
-----

```

Вариант 1. Разработать программу для работы со списком **АВТОМОБИЛЕЙ (Car)**. Каждый автомобиль содержит данные: марка машины (строка), год выпуска (число) и регистрационный номер (строка). Заполнить список начальными значениями (минимум 3) необходимо в коде программы. Программа должна предоставлять пользователю три функции:

- Добавление нового автомобиля в список. При этом если пользователь пытается добавить автомобиль с регистрационным номером, который уже существует, необходимо выводить соответствующее сообщение.
- Удаление автомобиля из списка по его регистрационному номеру.
- Удаление всех автомобилей из списка.

В результате работы программы после каждой операции (добавления или удаления) необходимо выводить на экран текущее содержимое списка автомобилей, например:

Volvo s90, 2015, x001xx

Lada Vesta, 2020, x002xx

Kia Rio, 2012, x003xx

Вариант 2. Разработать программу для работы со списком **ТЕЛЕФОННЫХ АППАРАТОВ (Telephone)**. Каждый телефонный аппарат содержит следующие данные: модель (строка), серийный номер (строка), цвет (строка), тип телефона — мобильный или нет (логический тип). Заполнить список начальными значениями (минимум 3) необходимо в коде программы. Программа должна предоставлять пользователю три функции:

- Добавление телефонного аппарата в список. При этом если пользователь пытается добавить телефонный аппарат с серийным номером, который уже существует, необходимо выводить соответствующее сообщение.
- Удаление телефонного аппарата из списка по его серийному номеру.
- Удаление всех телефонных аппаратов из списка

В результате работы программы после каждой операции (добавления или удаления) необходимо выводить на экран текущее содержимое списка телефонов, например:

Panasonic, X35235ZMD, white, false

Samsung S10, XYZ123456789, black, true

Iphone X, ASIOS77777QWERTY, black, true

Вариант 3. Разработать программу для работы со списком **ВРАЧЕЙ (Doctor)**. Для каждого **ВРАЧА** содержится следующая информация: ФИО (строка), специальность (строка), порядковый номер в списке сотрудников (положительное число), количество рабочих смен в **месяц** (положительное число), отметка о прохождении аттестации (логический тип). Заполнить список начальными значениями (минимум 3) необходимо в коде программы. Программа должна предоставлять пользователю две функции:

- Добавление данных нового врача в список. При этом если пользователь пытается добавить врача с уже существующим порядковым номером, необходимо выводить соответствующее сообщение.
- Изменение флага о прохождении аттестации для выбранного врача.

В результате работы программы после каждой операции необходимо выводить на экран текущее содержимое списка врачей **С СОРТИРОВКОЙ ПО ФИО**, например:

Антонов Антон Антонович, терапевт, 345, 15, true

Иванов Иван Иванович, хирург, 5, 20, true

Юрьев Юрий Аристархович, офтальмолог, 65, 10, false

Вариант 4. Разработать программу для работы со списком **ПРОПУСКОВ (Passport)**. Для каждого **ПРОПУСКА** содержится следующая информация: номер пропуска (число), ФИО (строка), должность (строка), отметка о прохождении допуске на закрытую территорию (логический тип). Заполнить список начальными значениями (минимум 5) необходимо в коде программы. Программа должна предоставлять пользователю функцию:

- Добавление пропусков в список. При этом, пользователю разрешается добавлять пропуск с уже существующим номером, если **ФИО отличаются** (в этом случае старый пропуск удаляется).

В результате работы программы после **каждой операции** добавления необходимо выводить на экран текущее содержимое списка пропусков сразу в двух вариантах: **С СОРТИРОВКОЙ ПО ФИО И ПО НОМЕРУ ПРОПУСКА**, например:

Антонов Андрей Антонович, 445566, сантехник, false

Иванов Иван Иванович, 112134, директор, true

Яковлева Юлия Аристарховна, 306306, секретарь, false

Иванов Иван Иванович, 112134, директор, true

Яковлева Юлия Аристарховна, 306306, секретарь, false

Антонов Андрей Антонович, 445566, сантехник, false

Вариант 5. Разработать программу для работы со **СЛОВАРЕМ (Dictionary) англо-русского переводчика**. Словарь содержит **уникальные** слова на английском и один (или несколько) вариантов перевода на русском. Заполнить список начальными значениями (минимум 5) необходимо в коде программы. Программа должна предоставлять пользователю две функции:

- Добавление новых слов и их значений в словарь. При этом если пользователь добавляет перевод для слова, которое уже есть в словаре, то оно дописывается к нему одним из вариантов.
- Удаление записи из словаря по введенному пользователем слову.

В результате работы программы после каждой операции необходимо выводить на экран текущее содержимое словаря **С СОРТИРОВКОЙ**, например:

- *Abdication – отречение; отказ; сложение полномочий*
- *Aperitif – аперитив*
- *Mead – мёд; луг*
- *Zizz – жужжание*