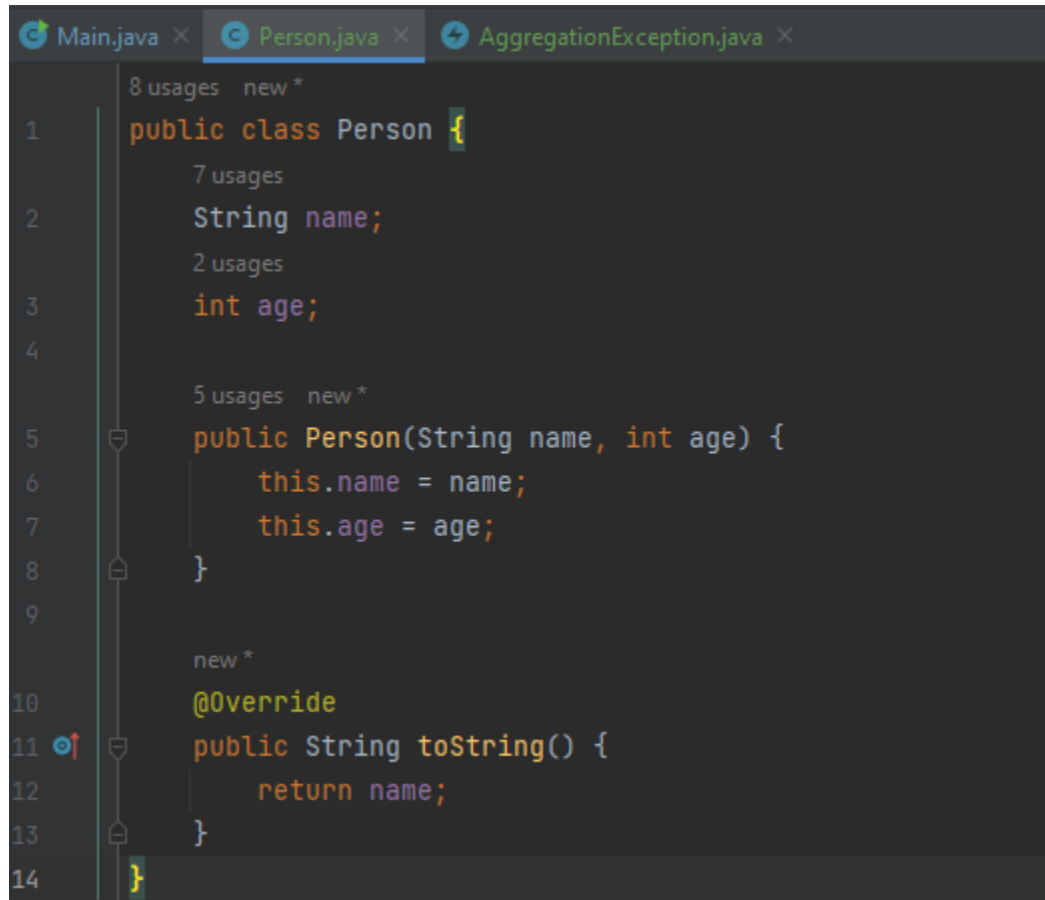


1. Создайте класс `Person`, содержащий информацию о человеке (например, имя и возраст). Затем создайте список объектов этого класса. Если список пуст, выбросите исключение `EmptyListException`, созданное вами. Обработайте это исключение, выводя пользователю сообщение, что список пуст.

Создаём класс `Person` с `name` и `age` атрибутами.



```
8 usages new *
1  public class Person {
2      7 usages
3      String name;
4      2 usages
5      int age;
6
7      5 usages new *
8      public Person(String name, int age) {
9          this.name = name;
10         this.age = age;
11     }
12
13     new *
14     @Override
15     public String toString() {
16         return name;
17     }
18 }
```

Заполняем лист и сразу проверяем не пустой ли он

```
try {
    List<Person> people = new ArrayList<>(Arrays.asList(
        new Person( name: "Charlie", age: 40),
        new Person( name: "Alice", age: 30),
        new Person( name: "Bob", age: 20),
        new Person( name: "Max", age: 10),
        new Person( name: null, age: 30)
    ));

    if (people.isEmpty()){
        throw new EmptyListException("Список пуст.");
    }
}
```

Так отлавливаются некоторые исключения

```
catch (EmptyListException | NoMatchesException | SortingException e){
    System.err.println(e.getMessage());
}
```

2. Используйте стрмы для фильтрации списка объектов Person, оставив только тех, кто старше определенного возраста. Если после фильтрации список оказывается пустым, выбросите исключение NoMatchingDataException, созданное вами. Обработайте это исключение, выводя пользователю сообщение, что нет данных, соответствующих условию.

**Фильтруем** используя Stream-ы и проверяем не пустой ли вышел массив.

```
// Фильтрация
int ageLimit = 25;
people = people.stream()
    .filter(p -> p.age > ageLimit)
    .collect(Collectors.toList());

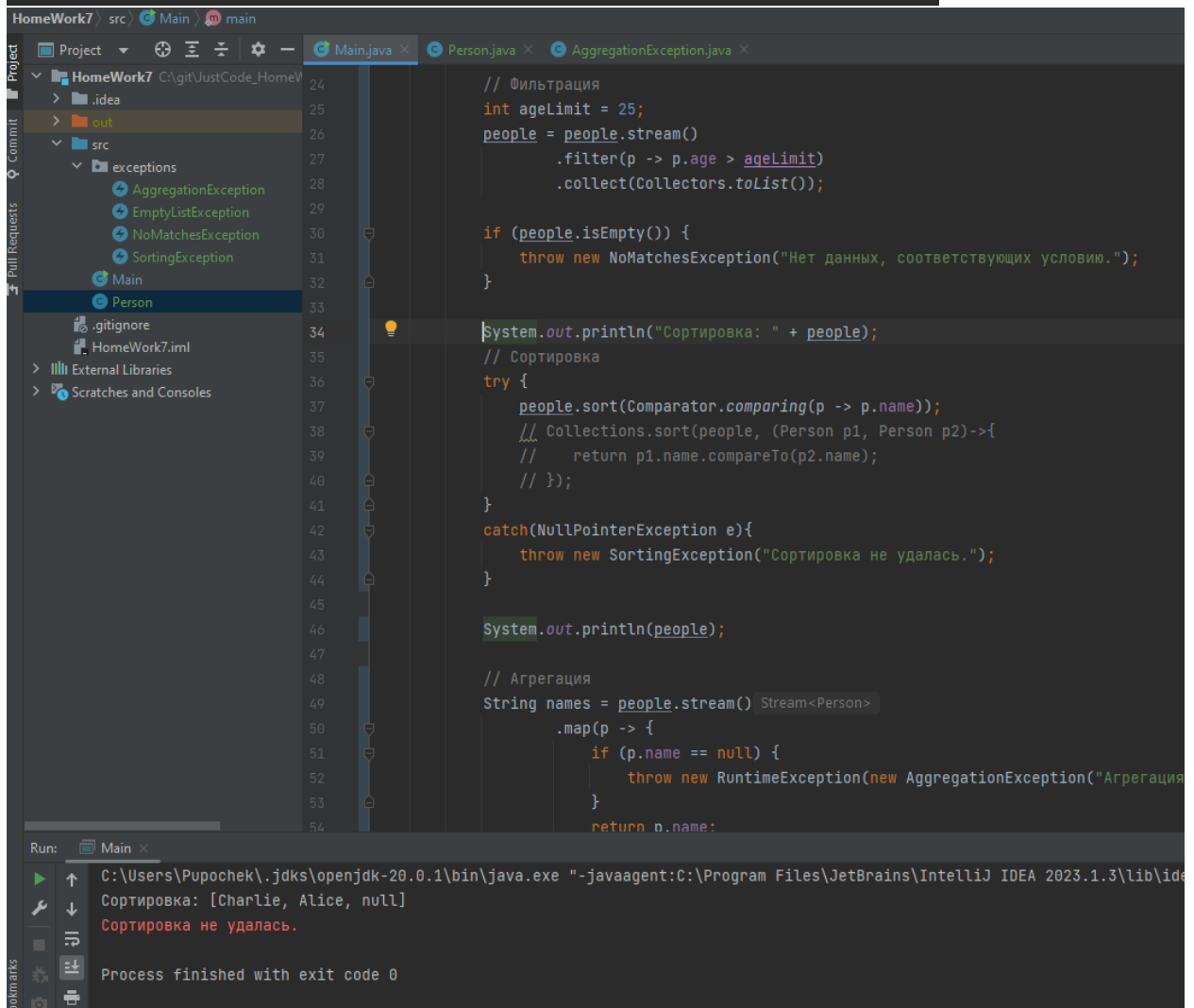
if (people.isEmpty()) {
    throw new NoMatchesException("Нет данных, соответствующих условию.");
}
```

3. Отсортируйте список объектов Person по имени. Если во время сортировки происходит ошибка (например, из-за null-значений), выбросите исключение SortingException, созданное вами. Обработайте это исключение, выводя пользователю сообщение, что сортировка не удалась.

Здесь я написал 2 метода сортировки, каждый работает правильно. При `p.name=null` выходит исключение `NullPointerException`, которое я обортываю в `SortingException`.

```
// Сортировка
try {
    people.sort(Comparator.comparing(p -> p.name));
    // Collections.sort(people, (Person p1, Person p2)->{
    //     return p1.name.compareTo(p2.name);
    // });
}
catch(NullPointerException e){
    throw new SortingException("Сортировка не удалась.");
}

System.out.println(people);
```



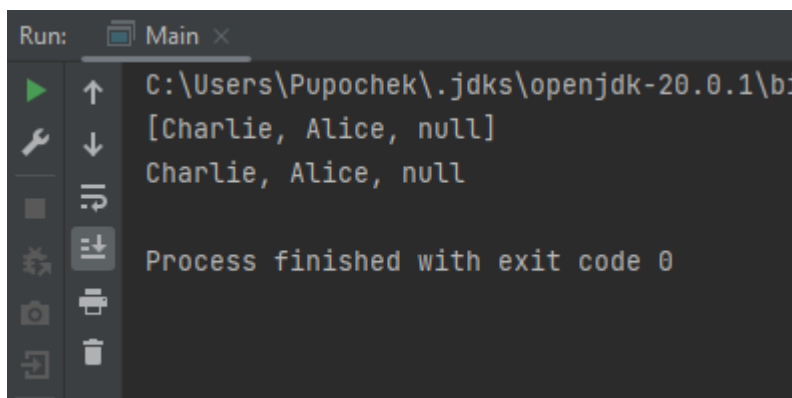
4. Используйте метод collect для агрегации данных, например, для создания строки со всеми именами объектов Person, разделенными запятой. Если во время агрегации происходит ошибка (например, из-за null-значений), выбросите исключение AggregationException, созданное вами. Обработайте это исключение, выводя пользователю сообщение, что агрегация данных не удалась.

Здесь некоторые трудности с обнаружением null значения, для Collectors.joining null значение не проблема и исключение никогда не вызывается.

```
System.out.println(people);

// Агрегация
String names = people.stream() Stream<Person>
    .map(p -> p.name) Stream<String>
    .collect(Collectors.joining( delimiter: ", "));

System.out.println(names);
```



Run: Main ×

C:\Users\Pupochek\.jdk\openjdk-20.0.1\bin\java.exe

[Charlie, Alice, null]

Charlie, Alice, null

Process finished with exit code 0

Поэтому я написал проверку в map. Но, из map не могут выходить checked exceptions, поэтому пришлось обернуть исключение в RuntimeException. Так делать на практике, уверен, не стоит.

```
names = people.stream() Stream<Person>
    .map(p -> {
        if (p.name == null) {
            throw new RuntimeException(new AggregationException("Name is null"));
        }
        return p.name;
    }) Stream<String>
    .collect(Collectors.joining( delimiter: ", "));

System.out.println(names);

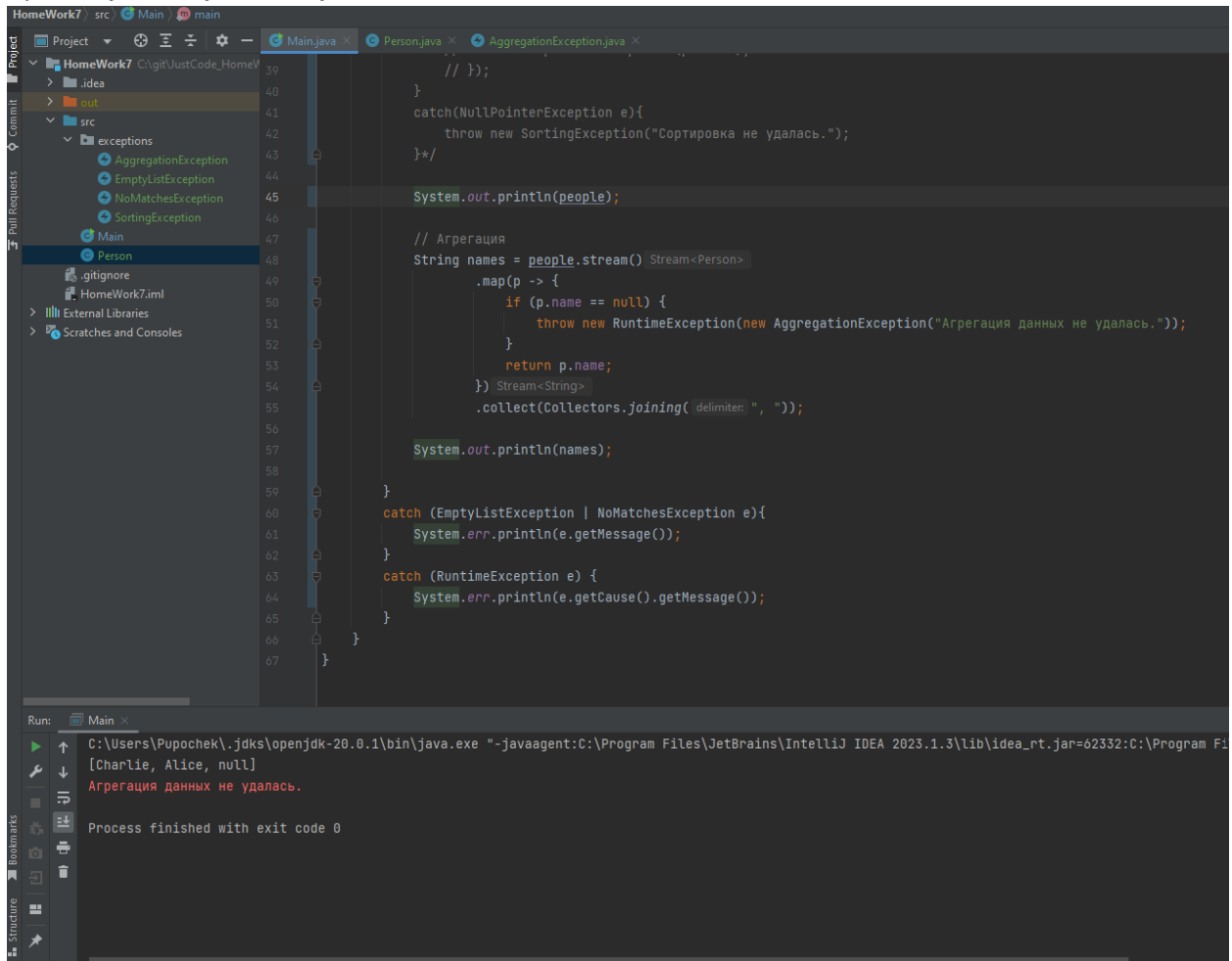
} catch (Exception e) {
    System.err.println(e.getCause().getMessage());
}
```

```
// Агрегация
String names = people.stream() Stream<Person>
    .map(p -> {
        if (p.name == null) {
            throw new RuntimeException(new AggregationException("Name is null"));
        }
        return p.name;
    }) Stream<String>
    .collect(Collectors.joining( delimiter: ", "));

System.out.println(names);
```

```
catch (RuntimeException e) {
    System.err.println(e.getCause().getMessage());
}
```

Закомментировал часть с сортировкой, так как она выбросила бы исключение первой, а нам нужно протестировать агрегацию.



```
39 // });
40 }
41 catch(NullPointerException e){
42     throw new SortingException("Сортировка не удалась.");
43 }*/
44
45 System.out.println(people);
46
47 // Агрегация
48 String names = people.stream() Stream<Person>
49     .map(p -> {
50         if (p.name == null) {
51             throw new RuntimeException(new AggregationException("Агрегация данных не удалась."));
52         }
53         return p.name;
54     }) Stream<String>
55     .collect(Collectors.joining( delimiter: " ", ));
56
57 System.out.println(names);
58
59 }
60 catch (EmptyListException | NoMatchesException e){
61     System.err.println(e.getMessage());
62 }
63 catch (RuntimeException e) {
64     System.err.println(e.getCause().getMessage());
65 }
66 }
67 }
```

Run: Main x

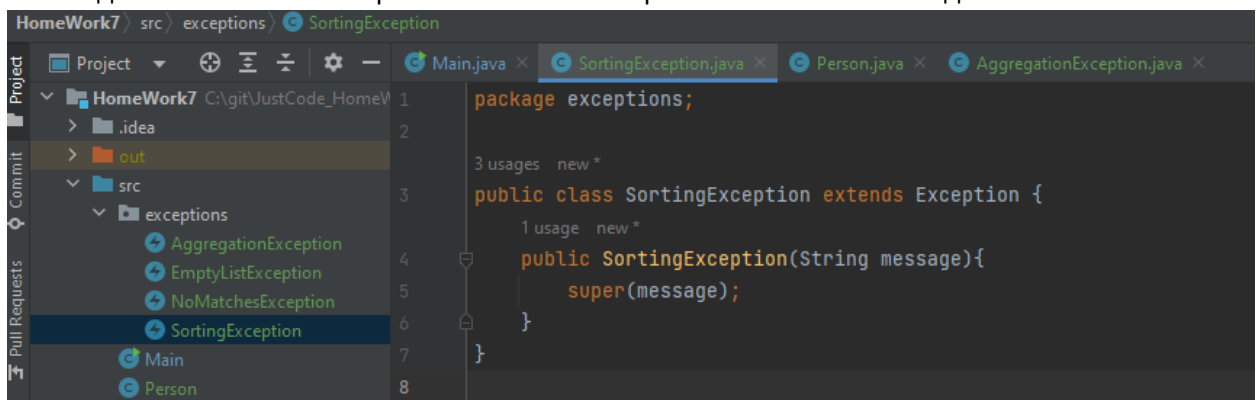
C:\Users\Pupochek\.jdk\openjdk-20.0.1\bin\java.exe --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1.3\lib\idea\_rt.jar=62332:C:\Program F...

[Charlie, Alice, null]

Агрегация данных не удалась.

Process finished with exit code 0

Все созданные **исключения** хранятся в пакете exceptions и имеют такой вид:



```
1 package exceptions;
2
3 3 usages new *
4 public class SortingException extends Exception {
5     1 usage new *
6     public SortingException(String message){
7         super(message);
8     }
9 }
```