Mid Exam Cheat Sheet

1. Масиви

- Четене на масив от конзолата:

Чрез "\\s+" може да сплитнем елементите по един или няколко спейса (защитава ни от ексепшън, в случай че имаме повече от 1 спейс между елементите).

```
String[] stringArr = scanner.nextLine().split("\\s+");
```

```
int[] integerArr = Arrays
        .stream(scanner.nextLine().split("\\s+"))
        .mapToInt(Integer::parseInt)
        .toArray();
```

```
double[] doubleArr = Arrays
        .stream(scanner.nextLine().split("\\s+"))
        .mapToDouble(Double::parseDouble)
        .toArray();
```

- Достъпване на елемент на масив:

```
int firstItem = integerArr[0];
```

- Сортиране на масив:

```
Arrays.sort(integerArr);
```

- Принтиране на масив:

- принтиране с for

```
for (int i = 0; i < integerArr.length; i++) {
    System.out.print(integerArr[i] + " ");
}
```

- Принтиране с foreach

```
for (int item : integerArr) {
    System.out.print(item + " ");
}
```

- Принтиране със String.join() – с този метод може да се принтира само String масив

```
System.out.println(String.join(", ", numbersAsStringArr));
```

- Int масив със String.join() – трябва да създаден нов String[], в който да сложим елементите и да ги направим от числа в String

```
int[] integerArr = Arrays.stream(scanner.nextLine().split("\\s+"))
        .mapToInt(Integer::parseInt).toArray();

String[] numbersAsStringArr = new String[integerArr.length];
for (int i = 0; i < integerArr.length; i++) {
    numbersAsStringArr[i] = String.valueOf(integerArr[i]);
}
System.out.println(String.join(", ", numbersAsStringArr));
```

- Принтиране с replaceAll() – няма значение какъв е типът на данните в масива

```
System.out.println(Arrays.toString(integerArr).replaceAll("[\\[\\]]",
""));
```

2. Лист

- Четене на лист от конзолата:

```
List<String> stringList =
Arrays.stream(scanner.nextLine().split("\\s+")).collect(Collectors.toList
());
```

```
List<Integer> integerList = Arrays
        .stream(scanner.nextLine().split("\\s+"))
        .map(Integer::parseInt)
        .collect(Collectors.toList());
```

```
List<Double> doubleList = Arrays
        .stream(scanner.nextLine().split("\\s+"))
        .map(Double::parseDouble)
        .collect(Collectors.toList());
```

- Методи

- `integerList.get(index)` – връща елемента на индекса, подаден като аргумент в скобите
- `integerList.size()` – връща цяло число – брой на елементите в листа
- `integerList.indexOf(element)` – връща цяло число – Индекс, на който се намира елемента
- `integerList.isEmpty()` – връща булева стойност: true – при празен лист, с 0 елемента
- `integerList.contains(element)` – връща булева стойност: true – ако елемента се съдържа в листа
- `integerList.add(element)` – добавя елемента в края на листа
- `integerList.add(index, element)` – добавя елемента на индекс, който сме подали като аргумент
- `integerList.set(1, 333)` – замества елемент на конкретен индекс с новият елемент
- `integerList.remove(index)` – премахва елемент на даден индекс

- `integerList.remove(Integer.valueOf(elment))` – премахва елемент, ако съществува в листа. Ако елемента е примитивен тип данни (double, int), чрез ValueOf() трябва да го превърнем в референтен тип данни, за да знае компилатора, че това е елемент. Ако го оставим int компилатора ще припознае числото като индекс, а не като елемент от листа. Ако елемента не съществува няма да ни даде ексепшън, просто ще игнорира командата.
- `integerList.addAll(numList)` – може да добавим всички елементи от една колекция в друга. Добавят се в края на колекцията.
- `Collections.sort(integerList)` - сортираме елементите в листа
- `Collections.reverse(integerList)` – обръщаме реда на елементите в листа, независимо от това дали са сортирани или не.

- Принтиране на List<>

- принтиране с for

```
for (int i = 0; i < integerList.size(); i++) {
    System.out.print(integerList.get(i) + " ");
}
```

- Принтиране с foreach

```
for (int item : integerList) {
    System.out.print(item + " ");
}
```

- Принтиране със String.join() – с този метод може да се принтира само String колекция

```
System.out.println(String.join(", ", stringList));
```

- Int лист със String.join() – трябва да създаден нов List<String>, в който да сложим елементите и да ги направим от числа в String

```
List<Integer> integerList = Arrays
        .stream(scanner.nextLine().split("\\s+")).map(Integer::parseInt)
        .collect(Collectors.toList());

List<String> stringList = new ArrayList<>(integerList.size());
for (int i = 0; i < integerList.size(); i++) {
    stringList.add(String.valueOf(integerList.get(i)));
}

System.out.println(String.join(", ", stringList));
```

- Принтиране с replaceAll() – няма значение какъв е типът на данните в листа

```
System.out.println(integerList.toString().replaceAll("[\\[\\],]", ""));
```

3. DecimaFormat

Чрез DecimalFormat може да форматирами числови типове данни с конкретен патерн. Може да разгледате примерни патерни тук: https://www.dev2qa.com/java-decimalformat-example/

```
double num = 1.146000;

DecimalFormat decimalFormat = new DecimalFormat("0.###");
String formattedNum = decimalFormat.format(num);

System.out.println(formattedNum); //1.146
```

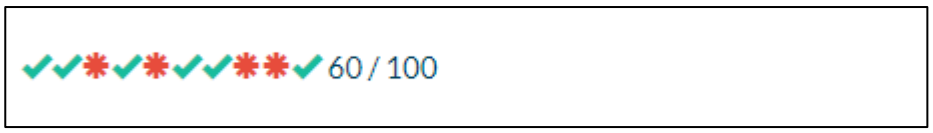
4. Грешки в Judge

- Incorrect answer

Тази грешка се получава, когато някой от изходните тестове (на събмитнатият от нас код) не е верен. В този случай може да отворим "Details", за да се ориентираме по примерните (нулеви) тестове.

- Runtime error

Това е грешка, която получаваме по време на изпълнение на нашия код. Програмата се стартира, но дава грешка при изпълнението на конкретен ред. Такъв пример е когато се опитаме да парснем въведен в конзолата некоректен стринг към цяло число – напр. "3.14" няма как да се парсне към int. Друг случай за такава грешка – ако опитаме да достъпим елемент от колекция (масив / лист) на индекс, който не съществува (извън колекцията – отрицателен или по-голям от размера му).



- Compile time error

Това е компилационна грешка, която виждаме още в IntelliJ Idea. Ако правим всичко коректно (кода ни в IntelliJ работи и връща коректен резултат, копираме и пействаме както трябва), но имаме Compile time error е много вероятно да събмитваме на различен програмен език в Judge, трябва да изберем Java.

- Time limit

Много често е поради претовареност на Judge (особено по време на изпит). Ако обаче след няколко събмита на един и същ код грешката е същата има проблем в кода ни. Това се случва при работа с цикли (най-често while), когато по условие имаме няколко условия за прекъсването му и не сме имплементирали някое от тях. Поради това програмата продължава, цикъла не се прекъсва и конзолата ни чака да въведем вход, т.е. конзолата зависва и JUDGE дава тази грешка.