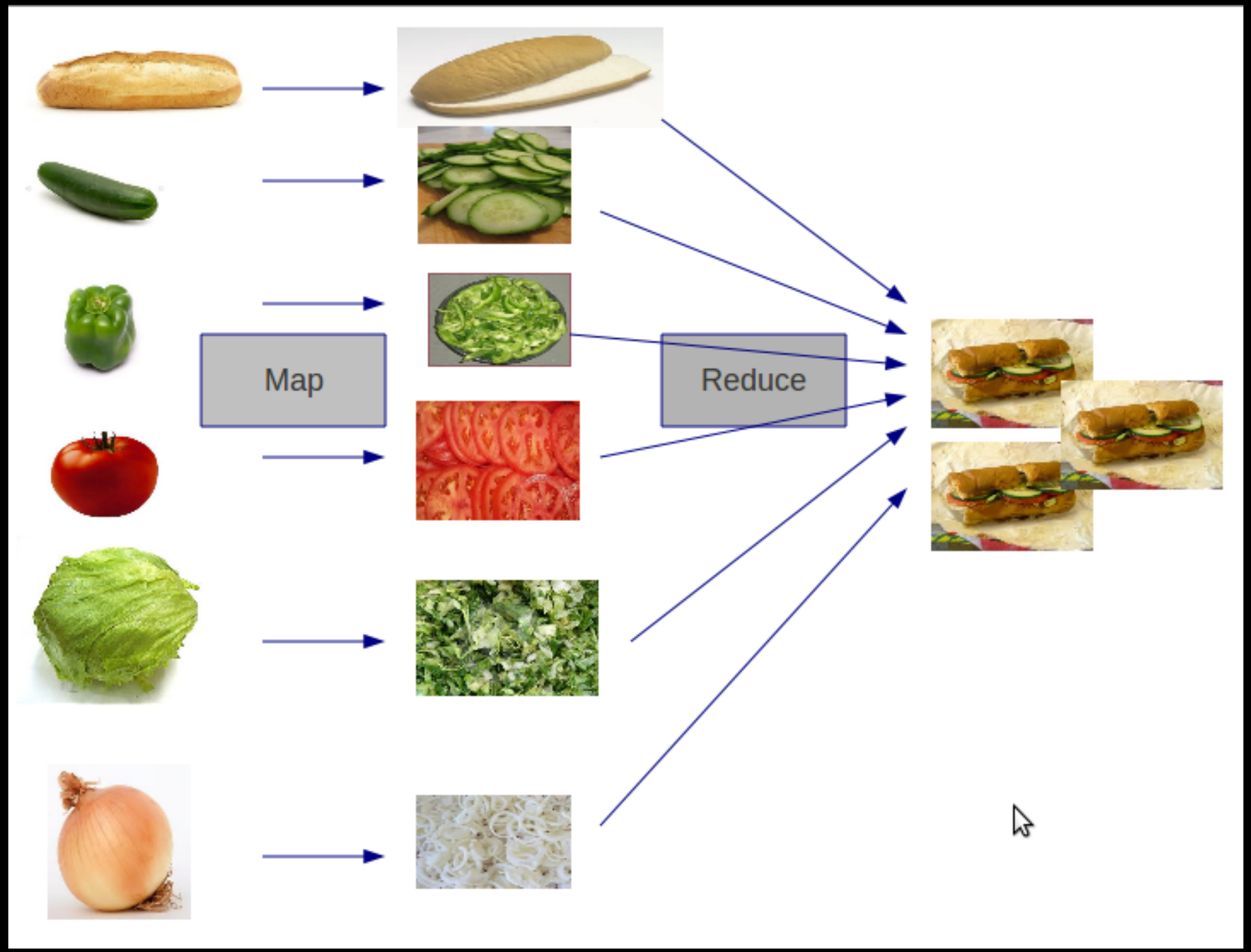
Лямбды. Stream API. Разбор домашних.

Практика 37 - Айрат Хасьянов

Минута Паттерна

Map, Reduce примеры



Пример 1

```
ArrayList<String> articles = new ArrayList<>(Arrays.asList("one", "two", "three", "four", "five"));

String reduced2 = articles.stream()
    .reduce((acc, item) -> acc + " + " + item)
    .get();
System.out.println(reduced2);
```

```
one + two + three + four + five
```

Пример 2

```
ArrayList<Article> articles = new ArrayList<>(Arrays.asList(
new Article("one"), new Article("two"), new Article("three"), new Article("four"),
new Article("five")));

String reduced = articles.stream().map(art -> art.toString())
    .reduce("", (acc, art) -> acc + " - " + art);
System.out.println(reduced);
```

```
- one - two - three - four - five
```

Пример 3

```
ArrayList<Article> articles = new ArrayList<>(Arrays.asList(
new Article("one"), new Article("two"), new Article("three"),
new Article("four"), new Article("five")));

Article reduced3 = articles.stream().
    reduce(new Article("Collection: "), (acc, art)
       -> acc.setTitle(acc.getTitle() + " & " + art.getTitle()));
System.out.println(reduced3.getTitle());
```

```
Collection:  & one & two & three & four & five
```

```
class Article{
    String _title;
    Article(String title){ _title = title;    }
    public Article setTitle(String title){
        _title = title;
        return this;
    }
    .....
}
```

35.2 Используйте java.util.Map для того, чтобы посчитать количество 1, 2, 3 и так до 12 в выборке из 1000000 случайных чисел.

```
list.stream()
        .filter(number -> number < 13 && number > 0)
        .collect(Collectors.groupingBy(key -> key))
        .forEach((k, v) -> System.out.println("Количество " + k + " равно " + v.size()));
```

```
Map<Integer, Long> result =
list.stream().collect(Collectors.groupingBy(Function.identity(),Collectors.counting()));
```

35.4. Напишите два итератора, которые позволяют обходить вашу приоритетную очередь, в порядке приоритета посадки, так и в алфавитном порядке для распечатки списка пассажиров. Продемонстрируйте обход коллекции, используя разные итераторы.

```
    Comparator<Person> personComparator =
            (o1, o2) -> o1.getId() - o2.getId();
    Comparator<Person> alphabetComparator =
(o1, o2) -> o1.getName().compareTo(o2.getName());
        ……
    Iterator ai =
      new PriorityIterator(a, alphabetComparator);
    Iterator pi =
      new PriorityIterator(a, personComparator);

    while(pi.hasNext()) {
            System.out.println(pi.next());
    }
```

```
class PriorityIterator implements Iterator {
    private PriorityQueue pq;

    public PriorityIterator(List list, Comparator comparator) {
        pq = new PriorityQueue(list.size(), comparator);
        list.stream().forEach(element -> pq.add(element));
    }

    @Override
    public boolean hasNext() {
        return !pq.isEmpty();
    }

    @Override
    public Object next() {
        return pq.poll();
    }}
```

35.6. Используйте метод subList для очистки каждых 5 элементов через 5 элементов, которые остаются нетронутыми.

```
IntStream
        .iterate(0, i -> i + 10)
        .limit(a.size() / 10 + 1)
        .mapToObj(i -> a.subList(Math.min(i, a.size()), Math.min(i + 5, a.size())))
        .forEach(element -> System.out.println(element));
```

```
[100, 101, 102, 103, 104]
[110, 111, 112, 113, 114]
[120, 121, 122, 123, 124]
[130, 131, 132, 133, 134]
[140, 141, 142, 143, 144]
[150, 151, 152, 153, 154]
[160, 161, 162, 163, 164]
[170, 171, 172, 173, 174]
[180, 181, 182, 183, 184]
[190, 191, 192, 193, 194]
[200, 201, 202]
```

```
IntStream.rangeClosed(0, list.size() / 10)
        .forEachOrdered(element -> list
                .subList(5 * element, (5 * element + 5 < list.size()) ? 5 * element + 5 : list.size())
                .clear());
```

36.2. Преобразовать каждый текст в последовательность слов;

```
Files.lines(file.toPath(), StandardCharsets.UTF_8)
        .forEach(line -> Arrays.asList(line.split("[^A-zA-я]+")))
        .forEach(word -> {
            if (word.length() > 0)
                words.add(word.toLowerCase());
        }));

Map<String, Integer> words = new HashMap<>();
        wordSequence.stream()
                .filter(element -> Pattern.compile("[а-яА-Я]+").matcher(element).matches())
                .map(String::toLowerCase)
                .forEach(element -> {
                    if (words.containsKey(element))
                        words.put(element, words.get(element) + 1);
                    else
                        words.put(element, 1);
                });
```

36.3. Преобразовать коллекцию текстов в коллекцию слов с частотой употребления каждого слова;

```
words.stream()
        .collect(Collectors.groupingBy(key -> key))
        .forEach((k, v) -> statistics.put(k, v.size()));
```

```
Map<String, Long> wordsMap = words.stream()
        .collect(Collectors.groupingBy(Function.identity(),Collectors.counting()));
```

36.4. Отфильтровать ненужные слова не на кириллице;

```
statistics.forEach((k, v) -> {
        if (Pattern.compile("[А-я]+").matcher(k).matches())
                statisticsKir.put(k, v);
        });
```

```
statistics.filter((k, v) -> Pattern.compile("[А-я]+").matcher(k).matches()));
```

36.5. Найти наиболее и наименее употребительное слово;

```
statisticsKir = statisticsKir.entrySet()
        .stream()
        .sorted(Map.Entry.comparingByValue())
    .collect(Collectors.toMap(
            Map.Entry::getKey,
            Map.Entry::getValue,
            (e1, e2) -> e1,
            LinkedHashMap::new
    ));
```

Примеры кода на паттерны, угадываем :)

Что за паттерн?

```
public class Cook {
    private BreakfestComposer breakfastComposer;
……..

    public Breakfast getBreakfast() {
        return breakfastComposer.getBreakfast();
    }

    public void composeBreakfast() {
        this.breakfastComposer.makeSandvich();
        this.breakfastBuilder.makeBeverage();
        this.breakfastBuilder.makeDessert();
    }
}
```

Builder

Что за паттерн?

```
public class Department implements Employee{

    private ArrayList<Employee> employees = new ArrayList<>();

    @Override
    public void getInformation() {
        for (Employee employee : employees)
            employee.getInformation();
    }

    public void add(Employee employee) {
        employees.add(employee);
    }

    public void remove(Employee employee) {
        employees.remove(employee);
    }
}
```

Composite

Что за паттерн?

```
public class test {
    public static void main(String[] args) {
        Originator originator = new Originator();
        Caretaker caretaker = new Caretaker();
        originator.setState(5);
        System.out.println(originator.getState());
        caretaker.setMemento(originator.saveState());
        originator.setState(4);
        System.out.println(originator.getState());
        originator.restoreState(caretaker.getMemento());
        System.out.println(originator.getState());
    }
}
```

Memento

Что за паттерн?

```
public class SmartDrawer implements IDrawer{
    Drawer drawer;

    public void draw() {
        if (drawer == null) drawer = new Drawer();
        // checking...
        drawer.draw();
    }
}
```

Proxy

Что за паттерн?

```
public class Context {
    private Satus status;

    public Context(Status status) {
        this.status = status;
    }

    public void setStatus(Status status) {
        this.state = state;
    }

    public void action() {
        status.action(this);
    }
}
```

State

Домашнее задание

1. Обзор калькулятора с точки зрения SOLID - взять у любого.
2. Преобразовать текст в последовательность слов, отсекая все лишнее. Получить слово, состоящее из самых часто встречающихся букв в соответствующих позициях. Например, для текста Абба, Баба, Лаб, Ассоль резульат будет «Ааба».