Built-In Methods

- `put(key, value)` method

```
HashMap<String, Integer> airplanes = new HashMap<>();
airplanes.put("Boeing 737", 130);
airplanes.put("Airbus A320", 150);
```

- `remove(key)` method

```
HashMap<String, Integer> airplanes = new HashMap<>();
airplanes.put("Boeing 737", 130);
airplanes.remove("Boeing 737");
```

Built-In Methods (2)

- containsKey(key)

```
HashMap<String, Integer> map = new HashMap<>();
map.put("Airbus A320", 150);
if (map.containsKey("Airbus A320"))
   System.out.println("Airbus A320 key exists");
```

- containsValue(value)

```
HashMap<String, Integer> map = new HashMap<>();
map.put("Airbus A320", 150);
System.out.println(map.containsValue(150)); //true
System.out.println(map.containsValue(100)); //false
```

Iterating Through Map

Software University

- Iterate through objects of type `Map.Entry<K, V>`
- Cannot modify the collection (**read-only**)

```
Map<String, Double> fruits = new LinkedHashMap<>();
fruits.put("banana", 2.20);
fruits.put("kiwi", 4.50);
for (Map.Entry<K, V> entry : fruits.entrySet()) {
System.out.printf("%s -> %.2f%n",
                  entry.getKey(), entry.getValue());
}
```

`entry.getKey()` -> fruit name
`entry.getValue()` -> fruit price

Solution: Count Real Numbers

```
double[] nums = Arrays.stream(sc.nextLine().split(" "))
                .mapToDouble(Double::parseDouble).toArray();
Map<Double, Integer> counts = new TreeMap<>();
for (double num : nums) {
  if (!counts.containsKey(num))
    counts.put(num, 0);
  counts.put(num, counts.get(num) + 1);
}
for (Map.Entry<Double, Integer> entry : counts.entrySet()) {
  DecimalFormat df = new DecimalFormat("#.########");
  System.out.printf("%s -> %d%n", df.format(entry.getKey()), entry.getValue());
}
```

Overwrite the value

Check your solution here: https://judge.softuni.org/Contests/1311/

Solution: Word Synonyms

```
int n = Integer.parseInt(sc.nextLine());
Map<String, ArrayList<String>> words = new LinkedHashMap<>();
for (int i = 0; i < n; i++) {
  String word = sc.nextLine();
  String synonym = sc.nextLine();
  words.putIfAbsent(word, new ArrayList<>());
  words.get(word).add(synonym);
}
//TODO: Print each word and synonyms
```

Adding the key if it does not exist

Check your solution here: https://judge.softuni.org/Contests/1311/

Lambda Expressions

Anonymous Functions

Lambda Functions

- A lambda expression is an anonymous function containing expressions and statements

```
(a -> a > 5)
```

- Lambda expressions
- Use the lambda operator **->**
  - Read as "**goes to**"
- The **left** side specifies the **input** parameters
- The **right** side holds the **expression** or **statement**

Lambda Functions

- Lambda functions are **inline methods** (functions) that take input parameters and return values:

```
x -> x / 2
```

```
static int func(int x) { return x / 2; }
```

```
x -> x != 0
```

```
static boolean func(int x) { return x != 0; }
```

```
() -> 42
```

```
static int func() { return 42; }
```

Stream API

Traversing and Querying Collections

Processing Arrays with Stream API (1)

- `min()` - finds the **smallest** element in a collection:

```
int min = Arrays.stream(new int[]{15, 25, 35}).min().getAsInt();
```

15

```
int min = Arrays.stream(new int[]{15, 25, 35}).min().orElse(2);
```

```
int min = Arrays.stream(new int[]{}).min().orElse(2); // 2
```

- `max()` - finds the **largest** element in a collection:

35

```
int max = Arrays.stream(new int[]{15, 25, 35}).max().getAsInt();
```

Processing Arrays with Stream API (2)

Software University

- `sum()` - finds the **sum** of all elements in a collection:

```
int sum = Arrays.stream(new int[]{15, 25, 35}).sum();
```

75

- `average()` - finds the **average** of all elements:

```
double avg = Arrays.stream(new int[]{15, 25, 35})
                   .average().getAsDouble();
```

25.0

Processing Collections with Stream API (1)


```
ArrayList<Integer> nums = new ArrayList<>() {{
   add(15); add(25); add(35);
}};
```

- min()

```
int min = nums.stream().mapToInt(Integer::intValue)
              .min().getAsInt();
```

15

```
int min = nums.stream()
              .min(Integer::compareTo).get();
```

Processing Collections with Stream API (2)

Software University

- max()

```
int max = nums.stream().mapToInt(Integer::intValue)
               .max().getAsInt();
```

35

```
int max = nums.stream()
               .max(Integer::compareTo).get();
```

- sum()

```
int sum = nums.stream()
               .mapToInt(Integer::intValue).sum();
```

75

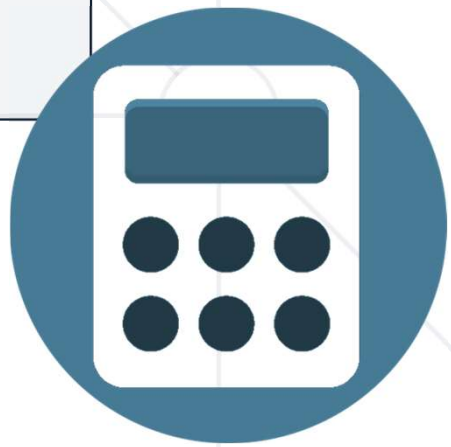
Processing Collections with Stream API (3)

Software University

- average()

```
double avg = nums.stream()
                 .mapToInt(Integer::intValue)
                 .average()
                 .getAsDouble();
```

25.0



Manipulating Collections

- `map()` - manipulates elements in a collection:

```
int[] nums = Arrays.stream(sc.nextLine().split(" "))
           .mapToInt(e -> Integer.parseInt(e))
           .toArray();
```

Parse each element to Integer

```
String[] words = {"abc", "def", "geh", "yyy"};
words = Arrays.stream(words)
              .map(w -> w + "yyy")
              .toArray(String[]::new);
//abcyyy, defyyy, gehyyy, yyyyyy
```

Converting Collections

- Using `toArray()`, `toList()` to convert collections:

```
int[] nums = Arrays.stream(sc.nextLine().split(" "))
                   .mapToInt(e -> Integer.parseInt(e))
                   .toArray();
```

```
List<Integer> nums = Arrays.stream(sc.nextLine()
                           .split(" "))
                           .map(e -> Integer.parseInt(e))
                           .collect(Collectors.toList());
```

Filtering Collections

Software University

- Using `filter()`

```
int[] nums = Arrays.stream(sc.nextLine().split(" "))
                   .mapToInt(e -> Integer.parseInt(e))
                   .filter(n -> n > 0)
                   .toArray();
```

Solution: Word Filter

```
String[] words = Arrays.stream(sc.nextLine().split(" "))
                 .filter(w -> w.length() % 2 == 0)
                 .toArray(String[]::new);

for (String word : words) {
  System.out.println(word);
}
```

Check your solution here: https://judge.softuni.org/Contests/1311/