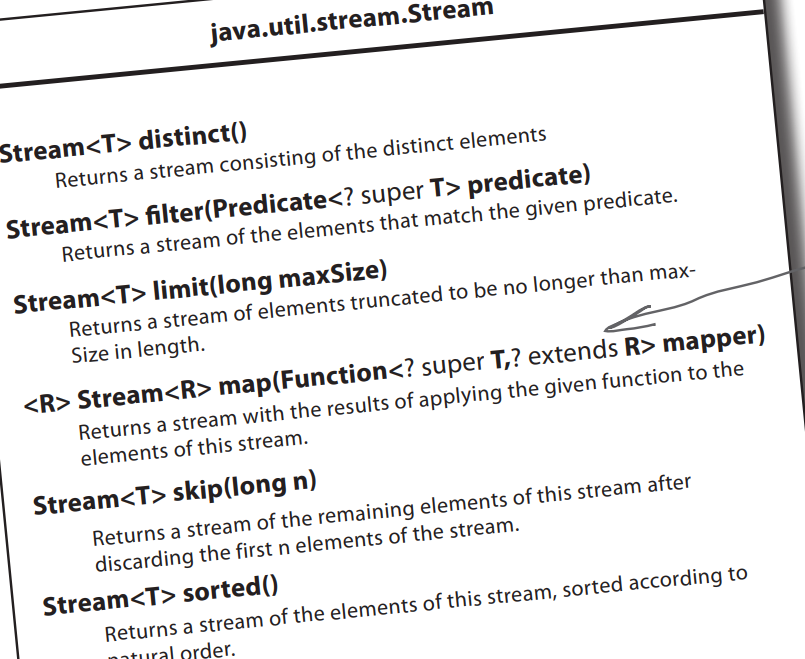
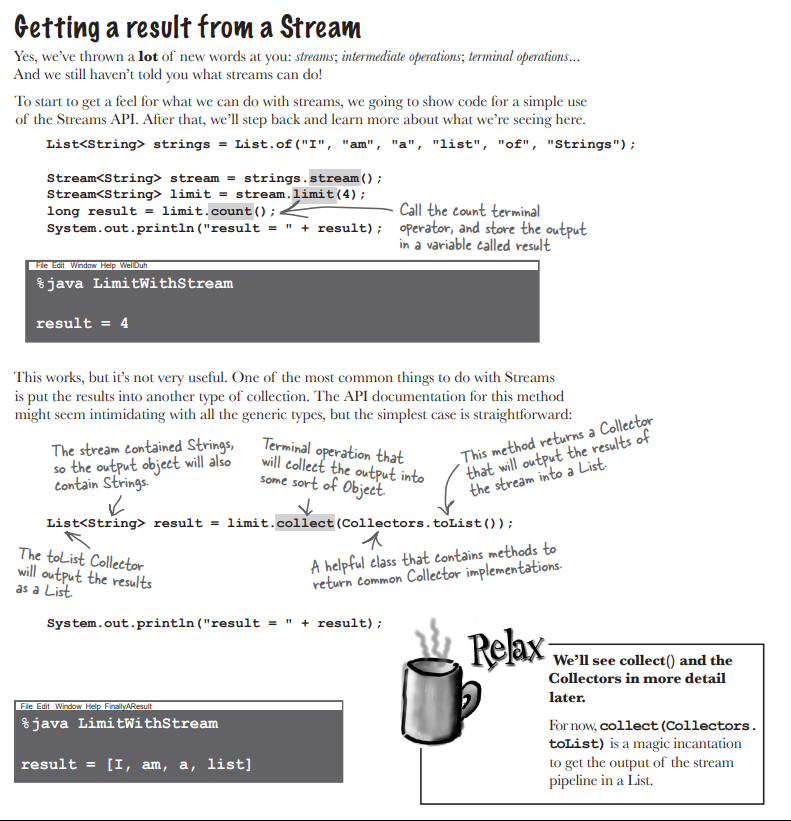
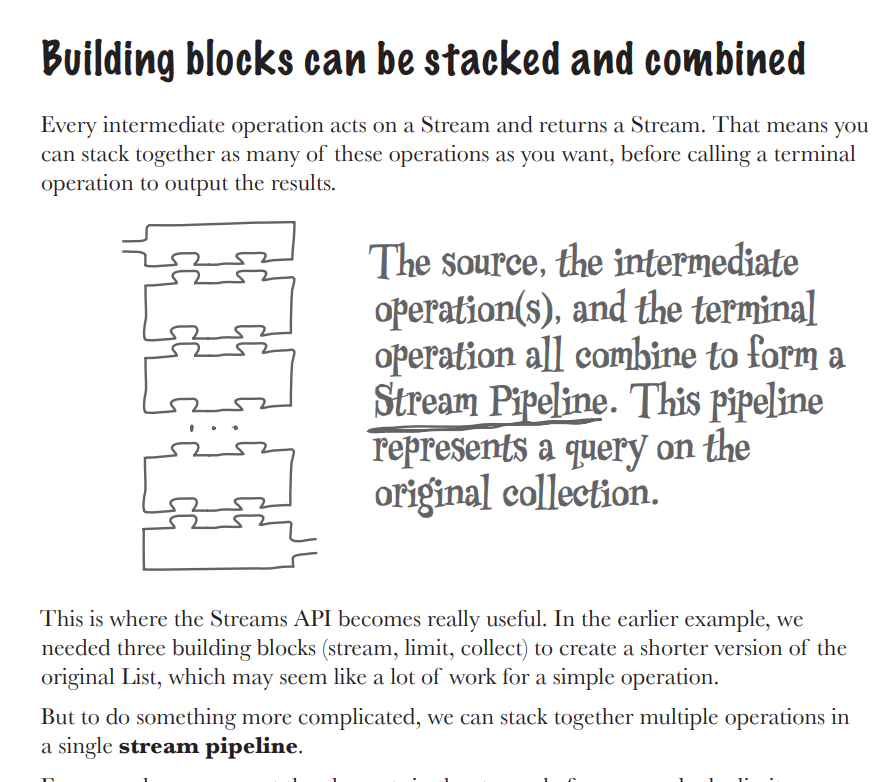
**CHAPTER 12**

**Lambdas and Streams: What, Not How (EXERCISE)`**

****



**EXERCISE**

**1.MIXED MESSAGES**

1. for (int i = 1; i < nums.size(); i++)

output += nums.get(i) + " ";

**Ans - 2 3 4 5**

2. for (int i = 0; i <= nums.size(); i++)

output += nums.get(i) + " ";

**Ans –**

**[1, 2, 3, 4, 5]**

**[1, 2, 3, 4, 5]**

**[1, 2, 3, 4, 5]**

**[1, 2, 3, 4, 5]**

**[1, 2, 3, 4, 5]**

3. for (int i = 0; i <= nums.length; i++)

output += nums.get(i) + " ";

**Ans - Compiler error**

4. for (Integer num : nums)

output += nums;

**Ans- Exception thrown**

**2.WHO DOES WHAT**

filter ->Only allows elements that match the given criteria to remain in the stream

skip -> This is the number of elements at the start of the Stream that will not be processed

limit -> Sets the maximum number of elements that can be output from this Stream

distinct ->Use this to make sure duplicates are removed

sorted -> States the result of the stream should be ordered in some way

map -> Changes the current element in the stream into something else

dropWhile -> Will only process elements while the given criteria is true

takeWhile -> While a given criteria is true, will not process elements

**3.CODE MAGNETS**

import java.util.\*;

import java.util.stream.\*;

public class CoffeeOrder {

public static void main(String[] args) {

List<String> coffees = List.of(

"Cappuccino", "Americano", "Espresso",

"Cortado", "Mocha", "Cappuccino",

"Flat White", "Latte"

);

List<String> coffeesEndingInO = coffees.stream()

.filter(s -> s.endsWith("o"))

.sorted()

.distinct()

.collect(Collectors.toList());

System.out.println(coffeesEndingInO);

}

}

**4.BE THE COMPILER**

Runnable r = () -> System.out.println("Hi!");

Consumer<String> c = s -> System.out.println(s);

Function<String, Integer> f = s -> s.length();

Supplier<String> s = () -> "Some string";

**5.SHARPEN YOUR PENCIL**

BiPredicate,Function and Action Listener has only one abstract classes.

**6.FIVE MINUTES MYSTERY**

Alex should have sorted the coffee first and then mapped with the name.This is because by sorting first,the weakest coffee would have been kept first and such that during mapping the initial coffee would have been displayed.But Alex did a vice versa and so she received Americano as first

**7.POOL PUZZLE**

public class StreamPuzzle {

public static void main(String[] args) {

SongSearch songSearch = new SongSearch();

songSearch.printTopFiveSongs();

songSearch.search("The Beatles");

songSearch.search("The Beach Boys");

}

}

class SongSearch {

private final List<Song> songs = new JukeboxData.Songs().getSongs();

void printTopFiveSongs() {

List<String> topFive = songs.stream()

.sorted(Comparator.comparingInt(Song::getTimesPlayed))

.map(song -> song.getTitle())

.limit(5)

.collect(Collectors.toList());

System.out.println(topFive);

}

void search(String artist) {

Optional<Song> result = songs.stream()

.filter(song -> song.getArtist().equals(artist))

.findFirst();

if (result.isPresent()) {

System.out.println(result.get().getTitle());

} else {

System.out.println("No songs found by: " + artist);

}

}

}