**public** **void** **traverseInOrder**(Node node) {

**if** (node != null) {

traverseInOrder(node.left);

visit(node.value);

traverseInOrder(node.right);

} }

--------------------------------------------

**String inputLine = sc.nextLine();**

**String[] items = inputLine.split(" ");**

**int[] arr = Arrays.stream(items)**

**.mapToInt(e -> Integer.parseInt(e)).toArray();**

-----------------------------------------

**Scanner sc = new Scanner(System.in);**

**List<Double> numbers = Arrays.stream(sc.nextLine().split(" "))**

**.map(Double::parseDouble).collect(Collectors.toList());**

**List<String> names = new ArrayList<>(Arrays.asList(**

**"Peter", "Michael", "George", "Victor"));**

**Collections.sort(names);**

**System.out.println(String.join(", ", names));**

***// George, John, Michael, Peter, Victor***

**Collections.sort(names);**

**Collections.reverse(names);**

**System.out.println(String.join(", ", names));**

**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());}**

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

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

ArrayList<Integer> nums = new ArrayList<>() {{

add(15); add(25); add(35);

}};

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

.min().getAsInt();

int min = nums.stream()

.min(Integer::compareTo).get();

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

**double avg = nums.stream()**

**.mapToInt(Integer::intValue)**

**.average()**

**.getAsDouble();**

String[] words = Arrays.stream(sc.nextLine().split(" "))

.filter(w -> w.length() % 2 == 0)

.toArray(String[]::new);-------

**String regex =   
"\\b(?<day>\\d{2)**

**Pattern pattern = Pattern.compile(regex)*;***

**Matcher matcher = pattern.matcher(dates);**

**while (matcher.find()) {**

**System.out.println(String.format("Day: %s, Month: %s, Year: %s",**

**matcher.group("day"), matcher.group("month"), matcher.group("year")));**

**}------------------------**

STACK – PUSH(), POP(),PEEK()

QUEUE – add(), offer() REMOVE, POLL()

add()–throws exception if queue is full

offer()–returns false if a queue is full

**-----------------------------------**

Function<T,R>is a function that returns R type

Consumer<T>is a void function

Supplier<T>gets no parameters

Predicate<T>evaluates a condition

BiFunction<T, U, R> accepts two parameters

**public class Library<T> implements Iterable<T> {**

**private final class LibIterator implements Iterator<T> {}**

**}**

**public class Library<Book> implements Iterable<Book> {**

**private Book[] books;**

**public Library(Book... books) {**

**this.books = books;**

**}**

**public Iterator<Book> iterator() {**

**return new LibIterator();}-------------**

**public int compareTo(Book book) {**

**if (this.getTitle().compareTo(book.getTitle()) == 0) {**

**if (this.getYear() >book.getYear()) { return 1;}**

**else if (this.getYear() <book.getYear()) { return -1; }**

**return 0;**

**} else {**

**return this.getTitle().compareTo(book.getTitle());**

**}}**

Questions Iterable vs Iterator