- Arrays.sort(arr, (a,b) -> b-a); // descening

4. Stack // Collection

1. Definition ->Stack st = new Stack<>();
2. insert -> T push(t); // TC: O(1)
3. size -> int size(); // TC: O(1)
4. look up for head element -> T peek(); // TC: O(1)
5. remove head element -> T pop(); // TC: O(1)
6. check for Empty -> boolean isEmpty(); // TC: O(1)

5. Queue // Collection

1. Definition -> Queue queue = new LinkedList<>();
2. insert -> boolean add(t); // TC: O(1)
3. size -> int size(); // TC: O(1)
4. look up for head element -> T peek(); // TC: O(1)
5. remove head element -> T poll(); // TC: O(1)
6. check for Empty -> boolean isEmpty(); // TC: O(1)
7. points to remember :

- queue poll vs stack pop
- queue add vs stack push
- we can define queue via LinkedList, PriorityQueue based on use case

6. String / StringBuilder

1. Definition -> String str = new String();
2. size -> int length();// TC: O(1)
3. convert to char Array -> toCharArray(); // TC: O(n)
4. value for specific index -> charAt(int index); // TC: O(1)
5. substring from string -> substring(a,b] // a : inclusive, b: Exclusive, TC: O(n)
6. transform to Lowercase -> toLowerCase(); // TC: O(n)
7. transform to UpperCase -> toUpperCase(); // TC: O(n)
8. replace all characters in string -> replaceAll(from, to) // TC: O(n)
9. Some useful Character properties

- Character.isLetter();
- Character.isAlphabetic();
- Character.isUpperCase();
- Character.isLowerCase();
- Character.isDigit();

10. Concatenation

- T str1 + str2
- StringBuilder ->
  - new StringBuilder() / new StringBuilder(int)
  - append("adding string") // better way to do
  - toString() // converting back to string

7. HashSet // Collection

1. Definition ->Set set = new HashSet<>();