1. 자료구조
2. ArrayDeque

* 추가

push(앞), addFrist(앞), add(뒤), addLast(뒤)

* 값 제거

poll(앞), pop(앞), pollFirst(앞), pollLast(뒤)

* 값을 확인만 하기

peekFirst(앞), peekLast(뒤), getFirst(앞), getLast(뒤)

1. 배열 관련 함수: String.toCharArray() ; list.toArray(); Arrays.sort(array1);
2. 패턴 탐색

|  |
| --- |
| import java.util.regex.\*;  class Test {  public static void main(String[] args) {  String hello = "HelloxxxHelloxxxHello";  Pattern pattern = Pattern.compile("Hello");  Matcher matcher = pattern.matcher(hello);  int count = 0;  while (matcher.find())  count++;  System.out.println(count); // prints 3  }  } |

1. Scanner

next(): 개행 문자, 공백은 무시하고 문자를 입력

nextLine(): 개행 문자 인식하여 한 줄 단위로 입력

1. Sort
   1. Comparable: 기본 정렬기준 구현

|  |
| --- |
| import java.util.\*    class Test {  public static void main(String[] args) {  String[] fruits = new String[]{“Pineapple”, “Apple”, “Orange”, “Banana”};  Arrays.sort(fruits); //오름차순 정렬  }  } |

|  |
| --- |
| import java.util.\*    class Data implements Comaparable<Data> {  int startDate; // 시작일  int endDate; // 종료일    // 생성자  Data(int startDate,int endDate){  this.startDate = startDate;  this.endDate = endDate;  }  public int compareTo(Data data){  return this.endDate – data.endDate; //ascending order  }  } |

* 1. Comparator

|  |
| --- |
| import java.util.Comparator  class Test{  public static Comparator<Data> comp = new Comparator<Data>(){  public int compare( Data o1, Data o2){ return o1.endDate – o2.endDate; }  };  Arrays.sort(fruits, comp);  } |

1. 분할정복

|  |
| --- |
| void mergeSort(int data[], int left, int right)  {  .iIf(left < right)  {  int mid = (left+right)/2;  //분할  mergeSort(data, left, mid);  mergeSort(data, mid+1, right);  //병합  merge(data, left, mid, right);  }  }  void merge(int result[], int left, int mid, int right)  {    } |