***Dt : 15/11/2022***

***Ex-program : DemoDeque.java***

***package maccess;***

***import java.util.\*;***

***public class DemoDeque {***

***@SuppressWarnings("removal")***

***public static void main(String[] args) {***

***Deque<Integer> ob = new ArrayDeque<Integer>();***

***for(int i=1;i<=5;i++)***

***{***

***ob.add(new ~~Integer~~(i));***

***}//end of loop***

***System.out.println(ob.toString());***

***ob.addFirst(new ~~Integer~~(11));***

***ob.addLast(new ~~Integer~~(19));***

***System.out.println(ob.toString());***

***ob.removeFirst();***

***ob.removeLast();***

***System.out.println(ob.toString());***

***System.out.println("First ele:"+ob.getFirst());***

***System.out.println("Last ele:"+ob.getLast());***

***System.out.println(ob.toString());***

***System.out.println("Poll first:"+ob.pollFirst());***

***System.out.println("Poll last:"+ob.pollLast());***

***System.out.println(ob.toString());***

***System.out.println("peek first:"+ob.peekFirst());***

***System.out.println("peek last:"+ob.peekLast());***

***ob.addFirst(new ~~Integer~~(11));***

***ob.addLast(new ~~Integer~~(11));***

***ob.addFirst(new ~~Integer~~(12));***

***ob.addLast(new ~~Integer~~(12));***

***System.out.println(ob.toString());***

***ob.removeFirstOccurrence(new ~~Integer~~(11));***

***System.out.println(ob.toString());***

***ob.removeLastOccurrence(new ~~Integer~~(12));***

***System.out.println(ob.toString());***

***}***

***}***

***o/p:***

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

***[11, 1, 2, 3, 4, 5, 19]***

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

***First ele:1***

***Last ele:5***

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

***Poll first:1***

***Poll last:5***

***[2, 3, 4]***

***peek first:2***

***peek last:4***

***[12, 11, 2, 3, 4, 11, 12]***

***[12, 2, 3, 4, 11, 12]***

***[12, 2, 3, 4, 11]***

***======================================================================***

***Note:***

***=>LinkedList<E> is the implementation of both List<E> and Deque<E>.***

***=======================================================================***

***faq:***

***define Iterable<E>?***

***=>Iterable<E> is an interface from java.lang package and which is ParentInterface***

***of 'Collection<E>'.***

***=>Iterable<E> will provide the following methods to perform iterations on***

***Collection<E> objects:***

***(i)iterator()***

***(ii)spliterator()***

***(iii)forEach()***

***(i)iterator():***

***=>iterator() method is used to create implementation object for Iterable<E>***

***interface.***

***(ii)spliterator():***

***=>spliterator() method is used to create implementation object for***

***Spliterator<T> interface.***

***(iii)forEach():***

***=>forEach() method is directly used on Collection<E> objects to retrieve***

***elements.***

***======================================================================***

***Limitation of Collection<E>:***

***=>In the process of holding database table data Collection<E> cannot differentiate***

***Primary-Key and NonPrimary-Key-Values.***

***Note:***

***=>Limitation of Collection<E> can be Overcomed using MapMap<K,V>***

***==========================================================================***

***\*imp***

***define Map<K,V>?***

***=>MapMK,V> is an interface from java.util package and which organizes elements***

***in the form of Key-Value pairs.***

***K - Key***

***V - Values***

***=>The following are some important methods of Map<K,V>:***

***public abstract int size();***

***public abstract boolean isEmpty();***

***public abstract boolean containsKey(java.lang.Object);***

***public abstract boolean containsValue(java.lang.Object);***

***public abstract V get(java.lang.Object);***

***public abstract V put(K, V);***

***public abstract V remove(java.lang.Object);***

***public abstract void putAll(java.util.Map<? extends K, ? extends V>);***

***public abstract void clear();***

***public abstract java.util.Set<K> keySet();***

***public abstract java.util.Collection<V> values();***

***public default void forEach***

***(java.util.function.BiConsumer<? super K, ? super V>);***

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

***=>The following are the implementation classes of Map<K,V>:***

***(a)HashMap<K,V>***

***(b)LinkedHashMap<K,V>***

***(c)TreeMap<K,V>***

***(d)Hashtable<K,V>***

***(a)HashMap<K,V>:***

***=>HashMap<K,V> organizes elements without any order and which is NonSynchronized***

***class.***

***(b)LinkedHashMap<K,V>:***

***=>LinkedHashMap<K,V> organizes elements in insertion order and which is***

***NonSynchronized class.***

***(c)TreeMap<K,V>:***

***=>TreeMap<K,V> organizes elements automatically in ascending order based on key***

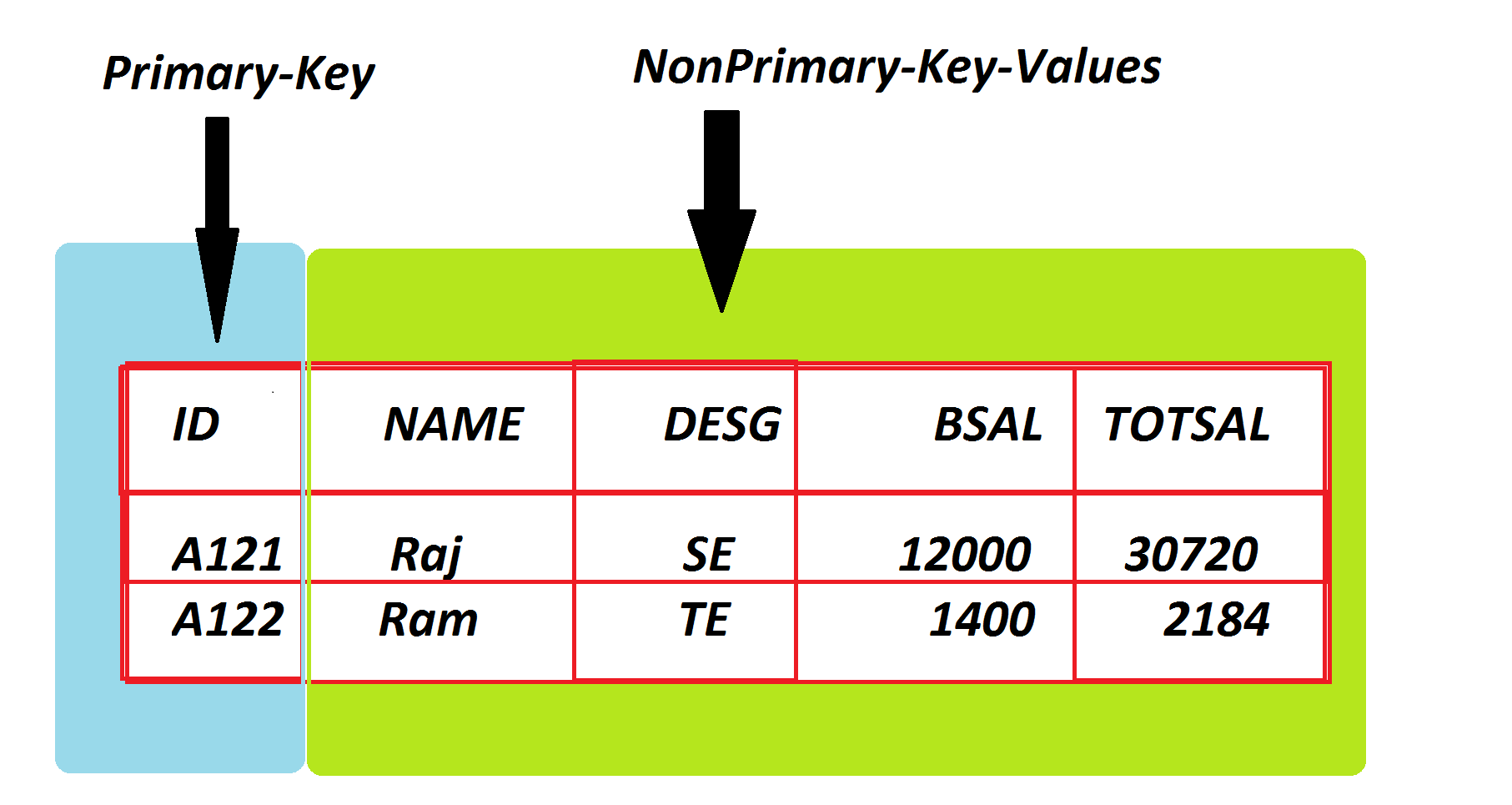
***and which NonSynchronized class.***

***(d)Hashtable<K,V>:***

***=>Hashtable<K,V> organizes elements without any order and which is synchronized***

***class***

***==========================================================================***

******