package Algoritmer.exercises;  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class Practice {  
 public void reverse(){  
 int [] arr = {1,2,4,5,6,7,8,9};  
 int n = arr.length;  
 for (int i = n-1; i >=0 ; i--) {  
 System.*out*.println(arr[i]);  
 }  
 }  
 public void insertion(){ */\*\*hvis man vil legge til en verdi i en valgfri rekkefølge kan man lage en variabel int pos og sette den inn i  
 slik : newArr[pos] før man printer ut verdien  
 \*/* int[] arr = {1,2,3,4,5,6};  
 int n = arr.length;  
 int value =32;  
 int[] newArr = new int[n+1];  
 for (int i = 0; i <n ; i++) {  
 newArr[i]=arr[i];  
 }  
 newArr[n]=value;  
 System.*out*.println(Arrays.*toString*(newArr));  
 }  
  
 public void deletion(){  
 int [] arr = {1,2,3,4,5,6,7,8};  
 int n = arr.length;  
 System.*out*.println("Here i your array:");  
 for (int i = 0; i <n ; i++) {  
 System.*out*.println(arr[i]);  
 }  
 System.*out*.println("Delete a value from the array by position:");  
 Scanner input = new Scanner(System.*in*);  
 int pos = input.nextInt();  
 for (int i = pos; i <n-1 ; i++) {  
 arr[i]=arr[i+1];  
 }  
 n=n-1;  
 System.*out*.println("Value from position deleted successfully:\n");  
 for (int i = 0; i < n; i++) {  
 System.*out*.println(arr[i]);  
 }  
 }  
  
 public void searchArr(){  
 int [] arr = {1,2,3,4,5,6,7,8,9,10};  
 int n = arr.length;  
 System.*out*.println("Search for array values:");  
 Scanner input = new Scanner(System.*in*);  
 int s = input.nextInt();  
 for (int i = 0; i <n ; i++) {  
 if (arr[i]==s){  
 System.*out*.println("results found for:"+s+":"+arr[i]);  
 }  
 }  
 }  
  
 public void reverse2(){  
 int [] arr = {1,2,4,5,6,7,8};  
 int n = arr.length;  
 for (int i = n-1; i >=0 ; i--) {  
 System.*out*.println(arr[i]);  
 }  
 }  
  
 public void insertion2() { */\*\*I denne koden kan du legge verdier i en tilfeldig posisjon i arrayet\*/* int [] arr = {1,2,3,4,5,6,7,8};  
 System.*out*.println(Arrays.*toString*(arr));  
 int n = arr.length;  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Enter a value:");  
 int value = input.nextInt();  
 System.*out*.println("Enter a position:");  
 int pos = input.nextInt();  
 for (int i = n-1; i >pos ; i--) {  
 arr[i]=arr[i-1];  
 }  
 arr[pos]=value;  
 System.*out*.println(Arrays.*toString*(arr));  
  
  
 }  
  
 public void deletion2(){  
 int [] arr = {1,2,3,4,5,6,7};  
 System.*out*.println(Arrays.*toString*(arr));  
 int n = arr.length;  
 System.*out*.println("Delete value by entering the position:");  
 Scanner input = new Scanner(System.*in*);  
 int pos = input.nextInt();  
 for (int i = pos; i <n-1 ; i++) {  
 arr[i]=arr[i+1];  
 }  
 for (int i = 0; i <n ; i++) {  
 System.*out*.println(arr[i]);  
 }  
  
 }  
 public void test3(){  
 int[] arr = {1,2,3,4,5,6,7,8};  
 System.*out*.println(Arrays.*toString*(arr));  
 int n = arr.length;  
 System.*out*.println("Enter a value:");  
 Scanner input = new Scanner(System.*in*);  
 int value = input.nextInt();  
 System.*out*.println("Enter a position:");  
 int pos = input.nextInt();  
 for (int i = n-1; i >pos ; i--) {  
 arr[i]=arr[i-1];  
 }  
 arr[pos]=value;  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
}

public void test5() { () */\*\*update array /modify array\*/* int [] arr = {1,2,3,4,5,6,7};  
 System.*out*.println(Arrays.*toString*(arr));  
 int n = arr.length;  
 int [] newArr = new int[n+1];  
 int value=32;  
 int pos=0;  
 for (int i = 0; i <n ; i++) {  
 newArr[i]=arr[i];  
 }  
 newArr[pos]=value;  
 System.*out*.println(Arrays.*toString*(newArr));  
}

public void test5() {  
 int [] arr = {1,2,3,4,5,6,7};  
 int n = arr.length;  
 System.*out*.println(Arrays.*toString*(arr));  
 System.*out*.println("Enter a value:");  
 Scanner input = new Scanner(System.*in*);  
 int value = input.nextInt();  
 System.*out*.println("Enter a position:");  
 int pos= input.nextInt();  
 for (int i = n-1; i >pos ; i--) {  
 arr[i]=arr[i-1];  
 }  
 arr[pos]=value;  
 System.*out*.println(Arrays.*toString*(arr));  
}

Inserting a value in the first index:

public void test5() {  
 int[] arr = {1, 2, 3, 4, 5, 6, 7, 8};  
 System.*out*.println(Arrays.*toString*(arr));  
 int n = arr.length;  
 int value =54;  
 int [] newArr = new int[n+1];  
 for (int i = 0; i <n ; i++) {  
 newArr[i]=arr[i];  
 }  
 newArr[n-8]=value;  
 System.*out*.println(Arrays.*toString*(newArr));  
  
}

public void traversal() { */\*\*traversal\*/* int [] arr = new int[4];  
 arr[0]=1;  
 arr[1]=43;  
 arr[2]=54;  
 arr[3]=423;  
 for (int i = 0; i < arr.length ; i++) {  
 System.*out*.println(arr[i]);  
 }  
}

public void test5() { */\*\*deletion\*/* int [] arr = {1,2,3,4,5,6,7,8};  
 int n = arr.length;  
 int pos =-1;  
 for (int i = pos; i <n-1 ; i++) {  
 arr[i]=arr[i+1];  
 }  
 n=n-1;  
 for (int i = 0; i <n ; i++) {  
 System.*out*.println(arr[i]);  
 if (pos <0){  
 System.*out*.println("Invalid input!!");  
 }  
 }

search elements in array + prints its position (riktig)

public void test5() { int [] arr = {1,2,3,4,5,6,7,8,9};  
 int n = arr.length;  
 System.*out*.println(Arrays.*toString*(arr));  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Search for a value:");  
 int s = input.nextInt();  
 for (int i = 0; i <n ; i++) {  
 if (arr[i]==s){  
 System.*out*.println("Results found:"+arr[i] + " in position:"+(i));  
 }  
 }  
 if (!arr.equals(n)){  
 System.*out*.println("Ikke funnet");  
 }  
  
}

Linked list

1. LinkedList<String> al=**new** LinkedList<String>();
2. al.add("Ravi");
3. al.add("Vijay");
4. al.add("Ravi");
5. al.add("Ajay");
7. Iterator<String> itr=al.iterator();
8. **while**(itr.hasNext()){
9. System.out.println(itr.next());
10. }
11. }

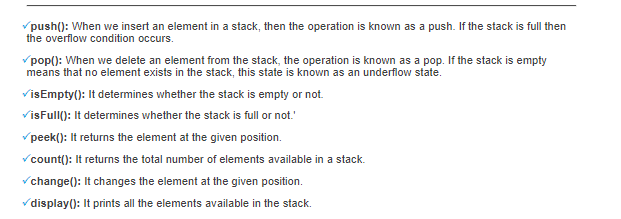
Lecture 3 (Stack, Tree )

Stack:

public static void main(String[] args) {  
 Stack<Integer> STACK = new Stack<>();  
 STACK.push(43);  
 STACK.push(23);  
 STACK.push(23);  
 STACK.push(54);  
 STACK.push(5423);  
 System.*out*.println("Elements:"+ String.*valueOf*(STACK));  
 System.*out*.println("Size:"+STACK.size());  
 System.*out*.println("First popped element:"+String.*valueOf*(STACK.pop()));  
 System.*out*.println("Second popped element"+String.*valueOf*(STACK.pop()));  
  
}  
public void test() {  
 Stack<Integer>stack = new Stack<>();  
 stack.push(534);  
 stack.push(6354);  
 stack.push(2);  
 stack.push(321);  
 stack.push(754);  
  
 System.*out*.println("Elements in stack:"+String.*valueOf*(stack));  
 System.*out*.println("First popped element:"+String.*valueOf*(stack.pop()));  
 System.*out*.println("Second popped element:"+String.*valueOf*(stack.pop()));  
}

push ved bruk av for loop:

public void stack\_push(){  
 Stack<Integer>stack = new Stack<>();  
 for (int i = 0; i < 5; i++) {  
 stack.push(i);  
 }  
 System.*out*.println(String.*valueOf*(stack));  
  
}



public void stack\_push() {  
 Stack<Integer> stack = new Stack<>();  
 int max =5;  
 for (int i = 0; i < 6; i++) {  
 stack.push(i);  
 }

if (max==stack.size()){  
 System.*out*.println("Stack is full");  
 }  
 else {  
 System.*out*.println("There's still space left in the stack");  
 }  
  
 */\*\* System.out.println("Enter a number:");  
 Scanner input = new Scanner(System.in);  
 int n = input.nextInt();  
 stack.push(n); \*/* System.*out*.println("Elements in stack:" + String.*valueOf*(stack));  
 System.*out*.println("is empty:"+stack.isEmpty());/\*returnerer false!\*/  
  
  
}

link list

public void link(){  
 LinkedList<Integer>list = new LinkedList<>();  
 list.add(43);  
 list.add(54);  
 list.add(231);  
 for (int i = 0; i < list.size(); i++) {  
 System.*out*.println(list.get(i));  
 }  
}

public void test7(){  
 Stack<Integer>stack = new Stack<>();  
 int s = stack.size();  
 s=7;  
 int max =s;  
 stack.push(867);  
 stack.push(654);  
 stack.push(54);  
 stack.push(35);  
 stack.push(564);  
 stack.push(543);  
 stack.push(32);  
   
  
 if (stack.size()==max) {  
 System.*out*.println("Stack is full");  
 }else {  
 System.*out*.println("Free spaces left for you to add");  
 }  
 System.*out*.println(String.*valueOf*(stack));  
 System.*out*.println(stack.pop());  
  
}

**Stacks: Riktig**

public void demo2(){  
 Stack <Integer>stack2 = new Stack<>();  
  
 int max =6;  
 stack2.push(6);  
 stack2.push(8);  
 stack2.push(56);  
 stack2.push(1);  
 stack2.push(9);  
 stack2.push(6);  
 stack2.push(7);  
 stack2.push(4);  
 stack2.push(9);  
  
  
 System.*out*.println(String.*valueOf*(stack2));  
  
  
 stackSpace(stack2,max);  
  
  
 System.*out*.println(stack2.peek());  
 stack2.pop();  
 stack2.pop();  
 stack2.pop();  
 stack2.pop();  
 stack2.pop(); */\*\*removes the top element from the stack\*/* System.*out*.println(stack2);  
 stackSpace(stack2, max);  
  
}  
  
public void stackSpace(Stack<Integer>stack2 , int max){  
 if (stack2.size()>=max){  
 System.*out*.println("Stack is full");  
 }else if (stack2.size()<max){  
 System.*out*.println("There's still more space left in the stack , you can add more elements if you wish");  
 }  
}  
  
public static void main(String[] args) {  
 Stacks st = new Stacks();  
 st.demo2();  
}

update Arrays (riktig)

public void test5() {  
 int [] arr = {1,2,3,4,5,6,7,8};  
 int n = arr.length;  
 System.*out*.println(Arrays.*toString*(arr));  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Enter a number:");  
 int value = input.nextInt();  
 System.*out*.println("Enter a position:");  
 int pos = input.nextInt();  
 int [] newArr = new int[n];  
 for (int i = 0; i < n; i++) {  
 newArr[i]=arr[i];  
  
 }  
 newArr[pos]=value;  
 System.*out*.println("Array updated!:");  
 System.*out*.println(Arrays.*toString*(newArr));

add values in array using for loop:

public void test8(){  
 int [] arr = new int[8];  
 int n = arr.length;  
  
 for (int i = 0; i <n ; i++) {  
 arr[i]+=i;  
  
 }  
 System.*out*.println(Arrays.*toString*(arr));  
  
  
}

2D arrays:

public class Main {

public static void main(String[] args) {

int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };

for (int i = 0; i < myNumbers.length; ++i) {

for(int j = 0; j < myNumbers[i].length; ++j) {

System.out.println(myNumbers[i][j]);

}  
 }

}

}

Search in array (søk I array) Riktig!

public void test11(){  
  
 int [] arr = {1,2,3,4,5,6,7};  
 int n = arr.length;  
 int flag=0; /\*kan erstattes med int target\*/  
 System.*out*.println("Enter a number to start searching:");  
 Scanner input = new Scanner(System.*in*);  
 int value = input.nextInt();  
 for (int i = 0; i < n; i++) {  
 if (arr[i]==value){  
 flag=1;  
 System.*out*.println("Results found:"+arr[i] + " in position:"+i);  
 }  
 }  
 if (flag!=1){  
 search(arr,value); /\*kan erstattes med sout(“Not found!!)\*/  
 }  
  
}

public void search(int [] arr , int value){  
 for (int i = 0; i < arr.length; i++) {  
 if (arr[i] !=value) {  
 System.*out*.println("Results not found!");  
 break;  
  
 }  
 }

**Bubble Sort:**

package Algoritmer.exercises;// Java program for implementation of Bubble Sort  
  
class BubbleSort {  
 void bubbleSort(int arr[])  
 {  
 int n = arr.length;  
 for (int i = 0; i < n - 1; i++)  
 for (int j = 0; j < n - i - 1; j++)  
 if (arr[j] > arr[j + 1]) {  
 // swap arr[j+1] and arr[j] viktig del  
 int temp = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = temp;  
 }  
 }  
  
 /\* Prints the array \*/  
 void printArray(int arr[])  
 {  
 int n = arr.length;  
 for (int i = 0; i < n; ++i)  
 System.*out*.print(arr[i] + " ");  
 System.*out*.println();  
 }  
  
 // Driver method to test above  
 public static void main(String args[])  
 {  
 BubbleSort ob = new BubbleSort();  
 int arr[] = { 5, 1, 4, 2, 8 };  
 System.*out*.println("Original Unsorted array");  
 ob.printArray(arr);  
 ob.bubbleSort(arr);  
 System.*out*.println("Sorted array");  
 ob.printArray(arr);  
 }  
}

**MergeSort:**

public class MergeSort {  
 // Merges two subarrays of arr[].  
 // First subarray is arr[l..m]  
 // Second subarray is arr[m+1..r]  
 void merge(int arr[], int l, int m, int r)  
 {  
 // Find sizes of two subarrays to be merged  
 int n1 = m - l + 1;  
 int n2 = r - m;  
  
 /\* Create temp arrays \*/  
 int L[] = new int[n1];  
 int R[] = new int[n2];  
  
 /\*Copy data to temp arrays\*/  
 for (int i = 0; i < n1; ++i)  
 L[i] = arr[l + i];  
 for (int j = 0; j < n2; ++j)  
 R[j] = arr[m + 1 + j];  
  
 /\* Merge the temp arrays \*/  
  
 // Initial indexes of first and second subarrays  
 int i = 0, j = 0;  
  
 // Initial index of merged subarray array  
 int k = l;  
 while (i < n1 && j < n2) {  
 if (L[i] <= R[j]) {  
 arr[k] = L[i];  
 i++;  
 }  
 else {  
 arr[k] = R[j];  
 j++;  
 }  
 k++;  
 }  
  
 /\* Copy remaining elements of L[] if any \*/  
 while (i < n1) {  
 arr[k] = L[i];  
 i++;  
 k++;  
 }  
  
 /\* Copy remaining elements of R[] if any \*/  
 while (j < n2) {  
 arr[k] = R[j];  
 j++;  
 k++;  
 }  
 }  
  
 // Main function that sorts arr[l..r] using  
 // merge()  
 void sort(int arr[], int l, int r)  
 {  
 if (l < r) {  
 // Find the middle point  
 int m = l + (r - l) / 2;  
  
 // Sort first and second halves  
 sort(arr, l, m);  
 sort(arr, m + 1, r);  
  
 // Merge the sorted halves  
 merge(arr, l, m, r);  
 }  
 }  
  
 /\* A utility function to print array of size n \*/  
 static void printArray(int arr[])  
 {  
 int n = arr.length;  
 for (int i = 0; i < n; ++i)  
 System.*out*.print(arr[i] + " ");  
 System.*out*.println();  
 }  
  
 // Driver code  
 public static void main(String args[])  
 {  
 int arr[] = { 12, 11, 13, 5, 6, 7 };  
  
 System.*out*.println("Given Array");  
 *printArray*(arr);  
  
 MergeSort ob = new MergeSort();  
 ob.sort(arr, 0, arr.length - 1);  
  
 System.*out*.println("\nSorted array");  
 *printArray*(arr);  
 }  
}

**Merge Sort (My code):**

public class MergeSort {  
 public void merge(int [] arr, int l , int m , int r){  
 int n1= m-l+1;  
 int n2 = r-m;  
  
 int L[] = new int[n1];  
 int R[] = new int[n2];  
  
 for (int i = 0; i < n1; i++) {  
 L[i]=arr[l+i];  
 }  
 for (int j = 0; j<n2 ; j++) {  
 R[j]=arr[m+1+j];  
 }  
  
 int i =0 , j =0;  
  
 int k = l;  
  
 while (i<n1 && j <n2){  
 if (L[i] <=R[j]){  
 arr[k]=L[i];  
 i++;  
 }else{  
 arr[k]=R[j];  
 j++;  
 }  
 k++;  
 }  
 while (i<n1){  
 arr[k]=L[i];  
 i++;  
 k++;  
 }  
 while (j<n2){  
 arr[k]=R[j];  
 j++;  
 k++;  
 }  
 }  
 void sort(int arr [] , int l , int r){  
  
 if (l<r){  
 int m = l +(r-l)/2;  
 sort(arr,l,m);  
 sort(arr, m+1 , r);  
 merge(arr,l,m,r);  
 }  
 }  
 static void printArray( int arr[]){  
 int n = arr.length;  
  
 for (int i = 0; i <n ; i++) {  
 System.*out*.println(arr[i]);  
 }  
 }  
  
 public static void main(String[] args) {  
 int arr [] = {1,12,11,5,13,9,17};  
 *printArray*(arr);  
  
 MergeSort ob = new MergeSort();  
 ob.sort(arr,0,arr.length-1);  
  
 System.*out*.println("Sorted array:");  
 *printArray*(arr);  
 }  
}

**QuickSort:**

public class QuickSort {  
 static void swap(int [] arr , int i , int j){  
 int temp = arr[i];  
 arr[i]=arr[j];  
 arr[j]=temp;  
 }  
  
  
 static int partition(int [] arr , int low , int high){  
 int pivot = arr[high];  
  
 int i = (low-1);  
  
 for (int j = low; j <=high-1 ; j++) { /\*\*husk j <= high-1\*/  
  
 if (arr[j] < pivot){  
 i++;  
 *swap*(arr,i,j);  
 }  
  
 }  
 *swap*(arr,i+1,high);  
 return (i+1);  
 }  
  
 static void quickSort(int [] arr , int low , int high){  
 if (low < high){  
 int pi = *partition*(arr,low,high);  
  
 *quickSort*(arr,low,pi-1);  
 *quickSort*(arr,pi+1,high);  
 }  
 }  
  
  
 static void printArray(int[] arr , int size){  
 for (int i = 0; i <size ; i++) {  
 System.*out*.println(arr[i] +" ");  
  
 System.*out*.println();  
 }  
 }  
  
 public static void main(String[] args) {  
 int [] arr = {10,7,8,9,1,5};  
 int n = arr.length;  
 System.*out*.println("Unsorted array:");  
 *printArray*(arr,n);  
 *quickSort*(arr,0,n-1);  
 System.*out*.println("Sorted array:");  
 *printArray*(arr,n);  
 }  
  
}

**Searching in hashtables java(chatgpt example)**

// Create a hash table  
 Hashtable<String, Integer> ht = new Hashtable<>();  
  
// Add some values to the hash table  
 ht.put("apple", 1);  
 ht.put("banana", 2);  
 ht.put("orange", 3);  
  
// Search for a value in the hash table  
 int value = ht.get("banana");  
  
// Print the value  
 System.*out*.println(value);

**Binary Search (code):**

package Algoritmer.exercises;  
  
import java.util.Arrays;  
  
public class BinarySearch {  
  
 public void search(){  
 int [] arr ={1,2,3,4,5,6,7,8,9,10};  
 int n = arr.length;  
 System.*out*.println("Input array:");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 int key =1;  
 int low =0;  
 int high = n-1;  
 int mid = (low+high)/2;  
 while (low<=high){  
 if (key == arr[mid]){  
 System.*out*.println("Key is found at index:"+mid);  
 break;  
 }else if (arr[mid] < key){  
 low=mid+1;  
 }else {  
 high=mid-1;  
 }  
 mid = (low+high)/2;  
  
 if (high > low){  
 System.*out*.println("Element not found!!");  
  
 }  
 }  
  
  
 }  
  
 public static void main(String[] args) {  
 BinarySearch bs = new BinarySearch();  
 bs.search();  
 }  
}

**Finding the largest value/number in array:**

public void largestNumberInArray(){  
 int [] arr = {1,2,3,4,32,6,787,8,9,11};  
 int n = arr.length;  
 int max =0;  
 for (int i = 0; i < n; i++) {  
 if (arr[i]>max){  
 max=arr[i];  
 }  
 }  
 System.*out*.println("Max value in the array:"+max);  
}