Plan for the day:

StringTokenizer

1. Java hands on practice for M1
2. Cheat sheet for Java
3. Online coding challenges
4. Module 1 curriculum
5. Some of our participants are going to present their topics
6. Online coding challenge: few of our team will choose a question online and all of us will take that challenge
7. HashSet, HashMap

HashSet

Is a type of Set,

Is a class that implements Set interface

Set<Integer> marks=new HashSet<>()

Hashing is the technique used in hash set so if you use contains() method, it will say true or false so quickly

Searching in hash set is fast

Whenever you implement HashSet, then you should override 2 methods:

hashCode() returns int

equals() returns boolean

note: HashSet uses HashMap for its internal purpose

You have a hashset of integers, and you want to sort them. How will you do?

Collections.sort(hs); //WRONG

Collections.sort() works only with LIST not SET

ArrayList<Integer> temp=new ArrayList<>(hs);

Collections.sort(temp);

Many things that works only with List:

ListIterator previous() hasPrevious()

Collections.sort

BigDecimal

BigInteger

The modulo operations and conversion from these to and fro Integer, Long should be learnt

Java Cheat Sheet:

<https://www.edureka.co/blog/cheatsheets/java-cheat-sheet/>

<https://introcs.cs.princeton.edu/java/11cheatsheet/>

<https://hackr.io/blog/java-cheat-sheet>

**N-smallest Elements**

Shreya was attending a coding competition and she was progressing good . Suddenly she got stuck in a problem where she needs to find the n-smallest elements in an array . Given an array of integers of size N and n value as inputs , Write a program to help Shreya find the first - n smallest elements from the array but they must be in the same order as they are in given array.  
  
**Input Format :**  
First input is an integer that denotes the size of the array, N.  
Second line consists of series of integers seperated by a space that denotes the array values.  
Third input is an integer that denotes the n value.  
  
**Output Format:**  
Output is a series of integers seperated by a space that denotes the n smallest elements.  
  
**Sample Input 1:**  
5  
4 8 2 5 9  
3  
  
**Sample Output 1:**  
4 2 5  
  
**Sample Input 2:**  
10  
7 3 8 11 55 22 1 6 13 2  
5  
  
**Sample Output 2:**  
7 3 1 6 2

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** App {

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

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

**int** n=sc.nextInt();

**int** arr[]=**new** **int**[n];

**for**(**int** i=0;i<n;i++)

{

arr[i]=sc.nextInt();

}

**int** count=sc.nextInt();

List<Integer> alist=**new** ArrayList<Integer>();

**for**(**int** i=0;i<n;i++)

{

alist.add(arr[i]);

}

Collections.*sort*(alist);

alist=alist.subList(0, count);

// System.out.println(alist);

**int** result[]=**new** **int**[count];

**for**(**int** i=0,j=0;i<n;i++)

{

// if(arr[i] is present in the alist)

**if**(alist.contains(arr[i]))

result[j++]=arr[i];

}

**for**(**int** i=0;i<count;i++)

{

System.***out***.print(result[i]+" ");

}

}

}

<https://nhlink.net/education/120-top-selenium-multiple-choice-questions-and-answers>

above link has 94 mcq on selenium

Challenge:

**32.  Employee Bonus**

A Company wants to give away bonus to its employees. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1.   Read Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.

2.   You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.

3.   If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised salary as value. Assume the age is caculated based on the date 01-09-2014. (Typecast the bonus to integer).

4.   Other Rules:

a. If Salary is less than 5000 store -100.

b. If the age is less than 25 or greater than 60 store -200.

c. a takes more priority than b i.e both if a and b are true then store -100.

5.   You decide to write a function **calculateRevisedSalary** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is “dd-mm-yyyy”

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

20-12-1987

10000

2020

01-01-1985

14400

**Sample Output 1:**

1010

12000

2020

17280

Solution :

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map.Entry;

**import** java.util.TreeMap;

**public** **class** UserMainCode {

**public** TreeMap<Integer, Integer> calculateRevisedSalary(HashMap<Integer, String> hmap1, HashMap<Integer, Integer> hmap2)

{

TreeMap<Integer, Integer> result=**new** TreeMap<>();

**for**(Entry<Integer, String> entry:hmap1.entrySet())

{

Integer id=entry.getKey();

String sdob=entry.getValue();

Integer salary=hmap2.get(id);

// 01-09-2014

Date today=**new** Date();

Date dob=**new** Date();

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

**try** {

today=sdf.parse("01-09-2014");

dob=sdf.parse(sdob);

} **catch** (ParseException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**int** age=0;

//manual method. Get the difference of date in millis (long)

**long** to=today.getTime();

**long** from=dob.getTime();

**long** diff=to-from;

**double** seconds=diff/1000;

**double** mins = seconds/60;

**double** hours=mins/60;

**double** days=hours/24;

**double** years=days/365.25;

//employee age is in years

**double** bonus=0.0;

**if**(years>=25 && years <=30)

{

bonus=(**double**)20/100\*salary;

}

**else** **if**(years>=31 && years<=60)

{

bonus=(**double**)30/100\*salary;

}

result.put(id, (**int**)(salary+bonus));

}

**return** result;

}

}

**import** java.util.HashMap;

**import** java.util.Map.Entry;

**import** java.util.Scanner;

**import** java.util.TreeMap;

**public** **class** Main {

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

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

**int** n=sc.nextInt();

HashMap<Integer, String> hmap1=**new** HashMap<>();

HashMap<Integer, Integer> hmap2=**new** HashMap<>();

**for**(**int** i=0;i<n;i++)

{

Integer id=sc.nextInt();

String dob=sc.nextLine();

**if**(dob.equals(""))

dob=sc.nextLine();

Integer salary=sc.nextInt();

hmap1.put(id, dob);

hmap2.put(id, salary);

}

UserMainCode umc=**new** UserMainCode();

TreeMap<Integer, Integer> result = umc.calculateRevisedSalary(hmap1, hmap2);

**for**(Entry<Integer, Integer> e:result.entrySet())

{

System.***out***.println(e.getKey());

System.***out***.println(e.getValue());

}

}

}

-------------To Find difference between 2 dates using LocalDate-----------

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.time.LocalDate;

**import** java.time.Period;

**import** java.util.Date;

**public** **class** App2 {

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

Date dt1=**new** Date();

Date dt2=**new** Date();

SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MMM-yyyy");

dt1=sdf.parse("10-May-2023");

dt2=sdf.parse("01-Jan-2000");

java.sql.Date sdt1=**new** java.sql.Date(dt1.getTime());

java.sql.Date sdt2=**new** java.sql.Date(dt2.getTime());

LocalDate ldt1 = sdt1.toLocalDate();

LocalDate ldt2 = sdt2.toLocalDate();

Period result = Period.*between*(ldt2, ldt1);

System.***out***.println(result.getYears()+" years, "+result.getMonths()+" months and "+result.getDays()+" days!");

}

}

Task:

Find the methods to convert

String to LocalDate

java.util.Date to LocalDate (via java.sql.Date)

and vice versa