**Java Interview question**

1. Sorted hash Map

package com.map;  
  
  
//sort hash map  
  
  
import java.util.HashMap;  
import java.util.LinkedHashMap;  
import java.util.Map;  
import java.util.stream.Collectors;  
  
class Employee{  
 int id;  
 String name;  
 String city;  
  
 public Employee(int id, String name, String city) {  
 this.id = id;  
 this.name = name;  
 this.city = city;  
 }  
}  
  
public class SortedMap {  
 public static void main(String[] args){  
 System.*out*.println("Saurav Suman");  
 Map<Integer,Employee> mp = new HashMap<>();  
  
 mp.put(1,new Employee(1,"Abc","Mumbai"));  
 mp.put(2,new Employee(2, "pqr","Banglore"));  
 mp.put(3,new Employee(3,"xyz","Delhi"));  
 mp.put(4,new Employee(3,"mnp","pune"));  
  
 //sort hashmap on the basis city in descending order  
  
 HashMap<Integer,Employee> sortedMap = mp.entrySet().stream()  
 .sorted((mp1,mp2)->mp2.getValue().city.compareTo(mp1.getValue().city))  
 .collect(Collectors.*toMap*(  
 Map.Entry::getKey,  
 Map.Entry::getValue,(m1,m2)->m2,LinkedHashMap::new  
  
 ));  
  
 for(Map.Entry<Integer,Employee> entry:sortedMap.entrySet()) {  
 System.*out*.println("key: " + entry.getKey() + " id: " + " ( "+entry.getValue().id+" , city:: "+entry.getValue().city+  
 " , name: " + entry.getValue().name +" ) ");  
 }  
 }  
}

1. JAVA 8 functional interface

present in package: java.util.function;

* Functional interface: A interface contains only one abstract method, and it can be contains private static method, default method.

@FunctionalInterface

public interface Bird{

void canFly(String val);

}

@FunctionalInterface

public interface Bird{

void canFly(String val);

default void getHeight(){

//default method

}

static void canEat(){

}

//don't need to provide implementation

// the who implemented this interface.

//of Object class method.

String toString();//Object class method

}

* 1. Types of Functional Interface
* **Consumer**: takes on parameter but not return any result.
* **Method: void accept(T t)**

@FunctionalInterface

public interface Consumer<T>{

void accept (T t);

}

public class Main{

public static void main(String[] args){

Consumer<Integer> oddOrEven = (Integer val)->{

if(val%10==0){

System.out.println("even");

}else{

System.out.println("odd");

}

}

}

}

* **Supplier**: Not takes any parameter but produces, result.
* **Method: T get()**

@FunctionalInterface

public interface Supplier<T>{

T get();

}

public class Main{

  public static void main(String[] args){

    Supplier<String> quote = ()-> "I am not a quote";

System.out.println(quote.get());

  }

}

* **Function**: It’s accept one parameter and produces result.
* **Method: R apply(T t)**

@FunctionalInterface

public interface Function<T,R>{

**R apply(T t);**

}

public class Main{

public static void main(String[] args){

Function<Integer,String> integerToString = (Integer num)->num.toString();

System.out.println(integerToString.apply(100));

}

}

* Predicate: Accept one parameter and return Boolean result
* **Method: boolean apply(T t)**

@FunctionalInterface

public interface Function<T>{

  boolean apply(T t);

}

public class Main{

  public static void main(String[] args){

    Predicate<Integer> isEven= (val)-> (val %10)==0;

    System.out.println(isEven.test(121));

  }

}

output:

false

1. String strs = "There is a tree tree and leaves leaves are green", find repeated word, using stream API

public class Find\_Repeated\_Words\_In\_Sentence {  
 public static void main(String[] args){  
 String str = "There is a tree tree and leaves leaves are green";  
  
 String[] resultant = Arrays.*stream*(str.split("\\s+")).collect(Collectors.*groupingBy*(Function.*identity*(),Collectors.*counting*()))  
 .entrySet().stream().filter(e->e.getValue()>1).map(e->e.getKey()).toArray(String[]::new);  
  
 System.*out*.println(Arrays.*toString*(resultant));  
 }  
}  
//output [leaves, tree]

1. **what changes comes for HashMap in java 8?**

**Default size [16] of bucket size, always in 2^n, [2^0, 2^1, 2^2, 2^3, 2^5...]**

* In Java 7, HashMap uses linked list for storing object of same hash code, due to high collision on bucket index,
* Time complexity for getting value from HashMap in worst case is **O(n)** due to LinkedList
* In Java 8, there is a concept treefy\_threshold which is 8, after 8 collision, node store in **BALANCED SEARCH TREE OR RED BLACK TREE , time complexity is O(log n)** .
* **getOrDefault(), computeIfAbsent(), computeIfPresent(), putIfAbsent(K, V) and forEach method that’s all comes in JAVA 8**.

1. **final int[] a = {1, 2, 3, 2, 4, 1, 1, 2, 1, 3, 1}; shift all 1 to left**

class Solution {

    public void moveZeroes(int[] a) {

        int i=0;

        while(i<a.length){

            int j=i+1;

            while(a[i]==0 && j<a.length){

                a[i]=a[j];

                a[j]=0;

                j++;

            }

            i++;

        }

    }

}