Java Program

1.Bubble sort

**package** interviewQ;

**publicclass** BubbleSort {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**arr[] = **newint**[]{10,23,1,90,34};

**int**len = arr.length;

System.***out***.println("Before sorting an array is");

**for**(**int**i=0;i<len;i++){

System.***out***.print(arr[i]+"\t");

}

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

**for**(**int**i=0;i<len-1;i++){

**for**(**int**k=0;k<len-i-1;k++){

**if**(arr[k]>arr[k+1]){

**int**temp = arr[k];

arr[k] = arr[k+1];

arr[k+1] = temp;

}

}

}

System.***out***.println("After sorting an array is ");

**for**(**int**i=0;i<len;i++){

System.***out***.print(arr[i]+"\t");

}

}

}

2.Merge sort

3.Decimal to binary

**package** interviewQ;

**publicclass** BinaryNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**val =10;

**int**temp=0;

String sum=" ";

**while**(val>0){

temp = val%2;

sum = sum+temp;

val =val/2;

}

//System.out.println(sum);

**for**(**int**i=sum.length()-1;i>0;i--){

System.***out***.print(sum.charAt(i)+"\t");

}

}

}

4.binary to decimal

**package** interviewQ;

**publicclass** DecimalNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

String binValue = "1011011";

**int**mulByTwo = 1;

**int**decVal=0;

System.***out***.println(Integer.*parseInt*(binValue, 2));//built-in method to test our result with this statement

//System.out.println(Integer.toBinaryString(Integer.parseInt(binValue, 2)));

**int**len = binValue.length();

**for**(**int**i=len-1;i>=0;i--){

String charVal = " "+binValue.charAt(i);

**int**temp = Integer.*parseInt*(charVal.trim());

**if**(temp==1 && (len-1)==i){

mulByTwo = mulByTwo\*1;

decVal = decVal+mulByTwo;

}

**elseif**(temp==1){

mulByTwo = mulByTwo\*2;

decVal = decVal+mulByTwo;

}

**elseif**(temp!=1 && (len-1)==i){

mulByTwo = mulByTwo\*1;

}

**else**{

mulByTwo = mulByTwo\*2;

}

}

System.***out***.println(decVal);

}

}

5.Linear search

**package** interviewQ;

**publicclass** DeleteElementsFromArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**a[] = **newint**[]{12,2,9,23,2};

**int**del =9;

**int**count=0;

**int**pos=0;

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

**if**(a[i]==del){

pos = i;

**for**(**int**j=i;j<a.length-1;j++){

a[j]=a[j+1];

}

count++;

**break**;

}

}

**if**(count!=0){

System.***out***.println("Element is found at position "+pos+"and deleted from an array");

}

**else**{

System.***out***.println("Element is not found in array");

}

System.***out***.println("After deletion of an array from positioned");

**for**(**int**i=0;i<a.length-1;i++){

System.***out***.print(a[i]+"\t");

}

}

}

Time complexity is o(n)

6.Binary search

Binary search can be possible on Sorted order only.

Algorithm

We basically ignore half of the elements just after one comparison.

1. Compare x with the middle element.
2. If x matches with middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
4. Else (x is smaller) recur for the left half.

**package** interviewQ;

**class** BinarySearch

{

**int** binarySearch(**int**arr[], **int**l, **int**r, **int**x)

{

**if** (r>=l)

{

**int**mid = l + (r - l)/2;

**if** (arr[mid] == x)

**return**mid;

**if** (arr[mid] >x)

**return** binarySearch(arr, l, mid-1, x);

**return** binarySearch(arr, mid+1, r, x);

}

**return** -1;

}

**publicstaticvoid** main(String args[])

{

BinarySearch ob = **new**BinarySearch();

**int**arr[] = {2,3,4,10,40};

**int**n = arr.length;

**int**x = 40;

**int**result = ob.binarySearch(arr,0,n,x);

**if** (result == -1)

System.***out***.println("Element not present");

**else**

System.***out***.println("Element found at index " +

result);

}

}

7.Find missing number from array

**package** interviewQ;

**publicclass** FindMissingWord {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**a[] = **newint**[]{1,4,5,6,2};

**int**missedWord = *missedWord*(a,a.length);

System.***out***.println(missedWord);

}

**privatestaticint** missedWord(**int**[] a, **int**length) {

// **TODO** Auto-generated method stub

**int**total = (length+1)\*(length+2)/2;

**for**(**int**j=0;j<length;j++){

total-=a[j];

}

**return**total;

}

}

8.Find out the finocci series with recursion

Without recursion

**publicclass** FobonacciSeries {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**first=0,second=1;

System.***out***.print(first+"\t"+second);

//System.out.println("\t");

**for**(**int**i=2;i<=10;i++){

**int**third =first+second;

System.***out***.print(third+"\t");

first = second;

second =third;

}

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

}

}

With recursion

**package** interviewQ;

**class** FibonacciSeries

{

**staticint***n1*=0,*n2*=1,*n3*=0;

**staticvoid**printFibonacci(**int**count){

**if**(count>0){

*n3* = *n1* + *n2*;

*n1* = *n2*;

*n2* = *n3*;

System.***out***.print(" "+*n3*);

*printFibonacci*(count-1);

}

}

**publicstaticvoid** main(String args[]){

**int**count=10;

System.***out***.print(*n1*+" "+*n2*);//printing 0 and 1

*printFibonacci*(count-2);//n-2 because 2 numbers are already printed

}

}

9.Find add all the number present in string

**package** interviewQ;

**publicclass** NumberInString {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

String str = "43String231";

**int**sum=0;

**int**len = str.length();

**for**(**int**i=0;i<len;i++){

**if**(Character.*isDigit*(str.charAt(i))){

sum = sum+Integer.*parseInt*((" "+str.charAt(i)).trim().toString());

}

}

System.***out***.println(sum);

}

}

10.count number of repeated character present in string using own logic and using collection.

Using collection

**package** interviewQ;

**import** java.util.HashMap;

**import** java.util.Map;

**publicclass** RepeatedCharacter {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

String str = "Malyalamboyinnewyorkstreat";

Map<Character,Integer>repeatedChar = **new** HashMap<Character,Integer>();

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

**if**(repeatedChar.containsKey(str.charAt(i))){

repeatedChar.put(str.charAt(i), repeatedChar.get(str.charAt(i))+1);

}

**else**{

repeatedChar.put(str.charAt(i), 1);

}

}

System.***out***.println(repeatedChar);

}

}

Using logic and without any collection or utility method.

package interviewQ;

import java.util.HashMap;

import java.util.Map;

public class RepeatedCharacter1 {

public static void main(String[] args) {

String s = "vickyjaiswalatnewyorkstreat";

int distinct=0;

System.out.println(s.length());

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

for(int j=0;j<s.length();j++){

if(s.charAt(i)==s.charAt(j)){

distinct++;

}

}

System.out.print(s.charAt(i)+"--"+distinct+"\t");

String sRepVal = String.valueOf(s.charAt(i)).toString();

s = s.replaceAll(sRepVal, "");

distinct=0;

}

}

}

11.Swap two number without third variable.

a.by divide

a =10 , b=5

a = a\*b;//50

b = a/b;//5

a = a/b//10

b.by addition

a =10 , b=5

a=a+b;//15

b=a-b;//10

a=a-b;//5

c.by one’s completement

x=10 in binary=1010

y = 5 in binary 0101

x=x^y;

x = x xor y

1010

Xor 0101

\_\_\_\_\_\_\_\_\_

1111

y = x^y

y = 1111 Xor 0101=1010

x = x^y

x = 1111 xor 1010 = 0101

0 for same different for 1

12.Find the HCF or GCD and LCF

13.reverse the string

**package** interviewQ;

**publicclass** ReverseString {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

String name= "Vicky";

**for**(**int**i =name.length()-1;i>=0;i--){

System.***out***.println(name.charAt(i));

}

}

}

14.reverse the number

**package** interviewQ;

**publicclass** ReverseTheNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**number = 12697;

**int**temp=0 ;

**int**rev=0;

**while**(number>0){

temp = number%10;

rev = rev\*10+temp;

number = number/10;

}

System.***out***.println(rev);

}

}

15.count number of digit in given number

**package** interviewQ;

**publicclass** CountNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**num = 123;

**int**sum = 0;

**int**count = 0;

**int**temp = 0;

**while** (num> 0) {

temp = num % 10;

sum = sum + temp;

num = num / 10;

count++;

}

System.***out***.println("sum of the number=" + sum + " "

+ "total number present in number" + " " + count);

}

}

16. How do you find the duplicate number on a given integer array?

**publicclass** DuplicateNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int** [] arry = {10,2,3,1,2,3,17,17,6,7};

**for**(**int**i=0;i<arry.length-1;i++){

**for**(**int**j=i+1;j<arry.length;j++){

**if**(arry[i]==arry[j] &&i!=j){

System.***out***.println("Duplicate Element:="+arry[i]);

}

}

}

}

}

17. How do you find the largest and smallest number in an unsorted integer array?

Ans:-

**package** interviewQ;

**class** Test

{

**staticint***arr*[] = {500,10, 3240, 45, 90, 98};

// Method to find maximum in arr[]

**staticint** largest()

{

**int**i;

// Initialize maximum element

**int**max = *arr*[0];

// Traverse array elements from second and

// compare every element with current max

**for** (i = 1; i<*arr*.length; i++)

**if** (*arr*[i] >max)

max = *arr*[i];

**return**max;

}

**staticint** smallest()

{

**int**i;

// Initialize maximum element

**int**min = *arr*[0];

// Traverse array elements from second and

// compare every element with current max

**for** (i = 1; i<*arr*.length; i++)

**if** (*arr*[i] <min)

min = *arr*[i];

**return**min;

}

// Driver method

**publicstaticvoid** main(String[] args)

{

System.***out***.println("Largest in given array is " + *largest*());

}

}

18. How do you find all pairs of an integer array whose sum is equal to a given number?

**package** interviewQ;

**publicclass** PairOfElementsInArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

FindPairs p = **new**FindPairs(**newint**[]{21,32,12,12,-24,36},24);

FindPairs p1 = **new**FindPairs(**newint**[] {4, 6, 5, -10, 8, 5, 20}, 10);

FindPairs p2 = **new**FindPairs(**newint**[] {4, -5, 9, 11, 25, 13, 12, 8}, 20);

FindPairs p3 = **new**FindPairs(**newint**[] {12, 13, 40, 15, 8, 10, -15}, 25);

FindPairs p4 = **new**FindPairs(**newint**[] {12, 23, 125, 41, -75, 38, 27, 11}, 50);

}

}

19. How do you find duplicate numbers in an array if it contains multiple duplicates?

**publicclass** DuplicateNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int** [] arry = {10,2,3,1,2,3,17,17,6,7};

**for**(**int**i=0;i<arry.length-1;i++){

**for**(**int**j=i+1;j<arry.length;j++){

**if**(arry[i]==arry[j] &&i!=j){

System.***out***.println("Duplicate Element:="+arry[i]);

}

}

}

}

}

**package** interviewQ;

**publicclass** FindPairs {

**public** FindPairs(**int**[] is, **int**i) {

// **TODO** Auto-generated constructor stub

**int**len=is.length;

**for**(**int**l=0;l<len;l++){

**for**(**int**k=l+1;k<len;k++){

**if**(is[l]+is[k]==i){

System.***out***.println(is[l]+"+"+is[k]+"= "+i);

}

}

}

}

}

20. How are duplicates removed from a given array in Java

**publicclass** RemoveDuplicateElementFromArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**arr[] = {10,10,20,20,30,30,50};

**int**len = arr.length;

**int**j=0;

**for**(**int**i=0;i<len-1;i++){

**if**(arr[i]!=arr[i+1]){

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

}

}

arr[j++]=arr[len-1];

**for**(**int**k=0;k<j;k++){

System.***out***.println(arr[k]);

}

}

}

21

|  |  |
| --- | --- |
| Q: | Why can't you use switch with strings? |
| A: | Strings are objects, and switch in Java works only for the primitive types byte, char, short, and int. To compare strings, you have to use nested ifs, which enable more general expression tests, including string comparison. |

22. How is an integer array sorted in place using the quicksort algorithm?

23. How do you reverse an array in place in Java?

**package** interviewQ;

**publicclass** ReversenArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**a[] = {1,3,4,3,6,7,8,10};

**for**(**int**i =a.length-1;i>=0;i--){

System.***out***.print("\t"+a[i]);

}

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

}

}

24. How are duplicates removed from an array with librarylibrary?

**package** interviewQ;

**import**java.util.Arrays;

**import** java.util.HashSet;

**import**java.util.List;

**import** java.util.Set;

**publicclass** RemoveDuplicateElementFromArrayUsingCollection {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

Integer array[] = {10,20,10,30,50,60,30};

SetsetArray = **new**HashSet();

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

setArray.add(array[i]);

}

System.***out***.println(setArray);

}

}

25. How do you find the second largest and smallest number in an unsorted integer array?

**package** interviewQ;

**class** GFG {

**publicstaticvoid** main(String[] args) {

**int**arr[] = { -1,14,18,-1,7,20,22 };

**int**largest = arr[0];

**int**secondLargest=0;

System.***out***.println("The given array is:" );

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

System.***out***.print(arr[i]+"\t");

}

**for** (**int**i = 1; i<arr.length; i++) {

**if** (arr[i] >largest) {

secondLargest = largest;

largest = arr[i];

} /\*else if (arr[i] > secondLargest) {

secondLargest = arr[i];

}\*/

}

System.***out***.println("\nSecond largest number is:" + secondLargest);

}

}

27.)what is alternative of InstanceOf?

Ans:-Polymorphism

e.g with InstanceOf

**publicfinalclass** BadInstanceOf {

**publicstaticvoid** doSomething(Animal animal){

**if** (animal **instanceof** Fish){

Fish fish = (Fish)animal;

fish.swim();

}

**elseif** (animal **instanceof** Spider){

Spider spider = (Spider)animal;

spider.crawl();

}

}

// PRIVATE

**privatestaticclass** Animal {}

**privatestaticfinalclass** Fish **extends** Animal {

**void** swim(){}

}

**privatestaticfinalclass** Spider **extends** Animal {

**void** crawl(){}

}

}

e.g. with Polymorphism

The mistake is corrected by using an overridable method: 

/\*\*

\* Using polymorphism to call different methods.

\* Does not use instanceof.

\*/

**publicfinalclass** BadInstanceOfFixed {

**publicstaticvoid** main(String... aArgs){

log("Starting...");

Animal animal = **new**Animal();

doSomething(animal);

//repoint the same 'animal' reference to other objects:

animal = **new** Fish();

doSomething(animal);

animal = **new** Spider();

doSomething(animal);

log("Done.");

}

/\*\*

\* This method implementation doesn't care at all

\* about Fish/Spider. It just knows that it has

\* been passed an Animal. Different versions of

\* 'move' are called, specific to each Animal.

\*/

**publicstaticvoid** doSomething(Animal aAnimal){

aAnimal.move();

}

// PRIVATE

**privatestaticclass** Animal {

**void** move(){

log("Move like an animal...");

}

}

**privatestaticfinalclass** Fish **extends** Animal {

**@Overridevoid**move(){

log("Move like a fish...");

}

}

**privatestaticfinalclass** Spider **extends** Animal {

**@Overridevoid**move(){

log("Move like a spider...");

}

}

**privatestaticvoid** log(String aMessage){

System.out.println(aMessage);

}

}

3rd approach is getClass in reflection

4th approach is visitor pattern in java 1.8

Object msg = //...

whenTypeOf(msg).

is(Date.class). then(date -> println(date.getTime())).

is(String.class). then(str -> println(str.length())).

is(Number.class). then(num -> println(num.intValue())).

orElse(obj -> println("Unknown " + obj));

28.)why we use static in java?

Suppose we have a class employee

First thing is the name of the employee

So the value of the string will vary for each employee but when it comes to company name it will be same for all those employees.

1.When we have any value common to all objects in that case we make it static.

Whenever static variable is declared in java it belongs to class not the object.

In runtime only one string variable of company will be created which will also help in saving the memory.

You will also notice that the main method is also static …..so in order to execute any java program you need to have atleast one static main method or else you will not be able to execute your program.

2.If a method doesn't modify state of object, or not using any instance variables.

3.

Real example of static given below.

Factory design pattern

Ans:-

29.)which one is god to use String or StringBuilder?

Since String is immutable in java, whenever we do String manipulation like concat, substring etc, it generates a new String and discard the older String for garbage collection.

These are heavy operations and generate a lot of garbage in heap. So Java has provided StringBuffer and StringBuilder class that should be used for String manipulation.

StringBuffer and StringBuilder are mutable objects in java and provide append(), insert(), delete() and substring() methods for String manipulation.

30.)write a program to create a utilities method for the union and intersection

**package** interviewQ;

**import**java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Set;

**publicclass** IntersectionAndUnionOfList {

**publicstaticvoid** main(String... args) **throws** Exception {

List<String>list1 = **new**ArrayList<String>(Arrays.*asList*("A", "B", "C"));

List<String>list2 = **new**ArrayList<String>(Arrays.*asList*("B", "C", "D", "E", "F"));

System.***out***.println(**new** IntersectionAndUnionOfList().intersection(list1, list2));

System.***out***.println(**new** IntersectionAndUnionOfList().union(list1, list2));

}

**public**<T> List<T> union(List<T>list1, List<T>list2) {

Set<T>set = **new** HashSet<T>();

set.addAll(list1);

set.addAll(list2);

**returnnew**ArrayList<T>(set);

}

**public**<T> List<T> intersection(List<T>list1, List<T>list2) {

List<T>list = **new**ArrayList<T>();

**for** (T t : list1) {

**if**(list2.contains(t)) {

list.add(t);

}

}

**return**list;

}

}

//written by me

**package** interviewQ;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Set;

**publicclass** IntersectionAndUnion {

**publicstaticvoid** main(String[] args) {

ArrayList<Integer>listA = **new** ArrayList<Integer>(Arrays.*asList*(2,3,4,5,6));

ArrayList<Integer>listB = **new** ArrayList<Integer>(Arrays.*asList*(2,3,4,5,6,7,1));

System.***out***.println(*union*(listA,listB));

System.***out***.println(*intersection*(listA,listB));

}

**privatestatic** List<Integer> intersection(ArrayList<Integer>listA,

ArrayList<Integer>listB) {

// **TODO** Auto-generated method stub

//Set<Integer>

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

**for**(Integer val :listA){

**if**(listB.contains(val)){

list.add(val);

}

}

**return**list;

}

**privatestatic** Set<Integer> union(ArrayList<Integer>listA,

ArrayList<Integer>listB) {

// **TODO** Auto-generated method stub

Set<Integer>union = **new** HashSet<Integer>();

union.addAll(listA);

union.addAll(listB);

**return**union;

}

}

31.)how to combined two List object in java?

Using addAll()

32.)How to delete the particular element from array?use linear search to delete an element and display all elements after deletion.

**package** interviewQ;

**publicclass** DeleteElementsFromArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**a[] = **newint**[]{12,2,9,23,2};

**int**del =9;

**int**count=0;

**int**pos=0;

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

**if**(a[i]==del){

pos = i;

**for**(**int**j=i;j<a.length-1;j++){

a[j]=a[j+1];

}

count++;

**break**;

}

}

**if**(count!=0){

System.***out***.println("Element is found at position "+pos+"and deleted from an array");

}

**else**{

System.***out***.println("Element is not found in array");

}

System.***out***.println("After deletion of an array from positioned");

**for**(**int**i=0;i<a.length-1;i++){

System.***out***.print(a[i]+"\t");

}

}

}

**package** interviewQ;

**import**java.util.Arrays;

**import** java.util.HashSet;

**import**java.util.List;

**import** java.util.Set;

**publicclass** RemoveDuplicateElementFromArrayUsingCollection {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

Integer array[] = {10,20,10,30,50,60,30};

SetsetArray = **new**HashSet();

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

setArray.add(array[i]);

}

System.***out***.println(setArray);

}

}

33.)merge two array in java?

**package** interviewQ;

**publicclass** MergeArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**a1[] = **newint**[]{1,2,1,4};

**int**a2[] = **newint**[]{2,3,4};

**int**a3[] = **newint**[a1.length+a2.length];

**for**(**int**i=0;i<a1.length;i++){

a3[i] = a1[i];

}

**int**index=a1.length;

**for**(**int**i=0;i<a2.length;i++){

a3[index++] = a2[i];

}

**for**(**int**i=0;i<a3.length;i++){

System.***out***.print(a3[i]+"\t");

}

}

}

34.)how to print an array of element without loop in one statement?

Ans:-Array.toString(arrayObject);

35.)Remove duplicate number from array in java

Simple program

**package** interviewQ;

**publicclass** RemoveDuplicateElementFromArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**arr[] = {10,10,20,20,30,30,50};

**int**len = arr.length;

**int**j=0;

**for**(**int**i=0;i<len-1;i++){

**if**(arr[i]!=arr[i+1]){

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

}

}

arr[j++]=arr[len-1];

**for**(**int**k=0;k<j;k++){

System.***out***.println(arr[k]);

}

}

}

Program with return

**package** interviewQ;

**publicclass** RemoveDuplicateInArray{

**publicstaticint** removeDuplicateElements(**int**arr[], **int**n){

**if** (n==0 || n==1){

**return**n;

}

**int**[] temp = **newint**[n];

**int**j = 0;

**for** (**int**i=0; i<n-1; i++){

**if** (arr[i] != arr[i+1]){

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

}

}

temp[j++] = arr[n-1];

// Changing original array

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

arr[i] = temp[i];

}

**return**j;

}

**publicstaticvoid** main (String[] args) {

**int**arr[] = {10,20,20,30,30,40,50,50};

**int**length = arr.length;

length = *removeDuplicateElements*(arr, length);

//printing array elements

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

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

}

}

Using collection

**package interviewQ;**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.HashSet;**

**import java.util.List;**

**import java.util.Set;**

**public class RemoveDuplicateElementFromArrayUsingCollection {**

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

**// TODO Auto-generated method stub**

**Integer array[] = {10,20,10,30,50,60,30};**

**Set setArray = Collections.EMPTY\_SET;**

**setArray = new HashSet();**

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

**setArray.add(array[i]);**

**}**

**//setArray = Collections.EMPTY\_SET;**

**assert(setArray!=null):"set should not be null";**

**String s =null;**

**System.out.println(setArray);**

**}**

**}**36.)How to avoid null pointer exception?

Ans:-first use literal object then other like below

**if**("vicky".equals(s)){

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

}

Or

String Vicky=”Vicky”;

**if**(vicky.equals(s)){

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

}

37.)Sort using one for loop and what is complexity of it.

**package** interviewQ;

**publicclass** SingleLoopArraySort {

**publicstaticvoid** main(String[] args) {

**int**[] arr = { 5,6,1,3 };

**for** (**int**i = 1; i<arr.length; i++) {

System.***out***.println("arr[i] < arr[i - 1]"+" "+arr[i]+" "+ arr[i - 1]);

**if** (arr[i] <arr[i - 1])

{

arr[i] = arr[i] + arr[i - 1];

arr[i - 1] = arr[i] - arr[i - 1];

arr[i] = arr[i] - arr[i - 1];

i = 0;

}

}

System.***out***.println("sorted Array :");

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

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

}

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

}

}

Inside the for loop it is working on sorting mechanism like given below

e.g.

a=6;

b=2;

a =a+b

b=a-b

a=a-b

Output

arr[i] < arr[i - 1] 6 5

arr[i] < arr[i - 1] 1 6

arr[i] < arr[i - 1] 1 5

arr[i] < arr[i - 1] 5 1

arr[i] < arr[i - 1] 6 5

arr[i] < arr[i - 1] 3 6

arr[i] < arr[i - 1] 5 1

arr[i] < arr[i - 1] 3 5

arr[i] < arr[i - 1] 3 1

arr[i] < arr[i - 1] 5 3

arr[i] < arr[i - 1] 6 5

sorted Array :

1 3 5 6

After understanding

**package** interviewQ;

**publicclass** SingleLoopToSortArray {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**a[]= **newint**[]{2,3,45,23,12,8,1};

**int**len = a.length;

**for**(**int**i=1;i<len;i++){

**if**(a[i]<a[i-1]){

a[i] = a[i]+a[i-1];

a[i-1] = a[i]-a[i-1];

a[i] = a[i]-a[i-1];

i=0;

}

}

**for**(**int**i=0;i<len;i++){

System.***out***.print(a[i]+"\t");

}

}

}

38.)Method name to display the common elements

retainAll()

**package** interviewQ;

**import** java.util.ArrayList;

**import** java.util.List;

**publicclass** RetainAll {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

List<Integer>l1 = **new** ArrayList<Integer>();

l1.add(1);

l1.add(2);

l1.add(3);

List<Integer>l2= **new** ArrayList<Integer>();

l2.add(4);

l2.add(2);

l2.add(3);

System.***out***.println("l1 == "+l1);

System.***out***.println("l2 == "+l2);

List<Integer>l3 = **new** ArrayList<Integer>(l2);

l3.retainAll(l1);

System.***out***.println("l3 == "+l3);

}

}

ContainsAll() will check the all elements present in left side collection elements .if its found will display true otherwise false.

**package** interviewQ;

**import** java.util.ArrayList;

**import** java.util.List;

**publicclass** RetainAll {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

List<Integer>l1 = **new** ArrayList<Integer>();

l1.add(1);

l1.add(2);

l1.add(3);

List<Integer>l2= **new** ArrayList<Integer>();

l2.add(4);

l2.add(2);

l2.add(3);

System.***out***.println("l1 == "+l1);

System.***out***.println("l2 == "+l2);

List<Integer>l3 = **new** ArrayList<Integer>(l2);

l3.retainAll(l1);

l3.add(5);

System.***out***.println("l3 == "+l3);

System.***out***.println(l1.containsAll(l3));

}

}

39.)Write an example of Generic.

40.)what is variable array argument or Vararg in main method?

classTest1

{

    // A method that takes variable number of intger

    // arguments.

    staticvoidfun(int...a)

    {

        System.out.println("Number of arguments: "+ a.length);

        // using for each loop to display contents of a

        for(inti: a)

            System.out.print(i + " ");

        System.out.println();

    }

    // Driver code

    publicstaticvoidmain(String args[])

    {

        // Calling the varargs method with different number

        // of parameters

        fun(100);         // one parameter

        fun(1, 2, 3, 4);  // four parameters

        fun();            // no parameter

    }

}

classTest2

{

    // Takes string as a argument followed by varargs

    staticvoidfun2(String str, int...a)

    {

        System.out.println("String: "+ str);

        System.out.println("Number of arguments is: "+ a.length);

        // using for each loop to display contents of a

        for(inti: a)

            System.out.print(i + " ");

        System.out.println();

    }

    publicstaticvoidmain(String args[])

    {

        // Calling fun2() with different parameter

        fun2("GeeksforGeeks", 100, 200);

        fun2("CSPortal", 1, 2, 3, 4, 5);

        fun2("forGeeks");

    }

}

**Important points:**

* + Vararg Methods can also be overloaded but overloading may lead to ambiguity.
  + Prior to JDK 5, variable length arguments could be handled into two ways : One was using overloading, other was using array argument.
  + There can be only one variable argument in a method.
  + Variable argument (varargs) must be the last argument.

**Erroneous varargs Examples**

* + Specifying two varargs in a single method:
  + void method(String... gfg, int... q)
  + {
  + // Compile time error as there are two
  + // varargs
  + }
  + Specifying varargs as the first parameter of method instead of last one:
  + void method(int... gfg, String q)
  + {
  + // Compile time error as vararg appear
  + // before normal argument

}

41.)Add all number given in number.

**package** interviewQ;

**publicclass** CountNumber {

**publicstaticvoid** main(String[] args) {

// **TODO** Auto-generated method stub

**int**num = 123;

**int**sum = 0;

**int**count = 0;

**int**temp = 0;

**while** (num> 0) {

temp = num % 10;

sum = sum + temp;

num = num / 10;

count++;

}

System.***out***.println("sum of the number=" + sum + " "

+ "total number present in number" + " " + count);

}

}

42.)Prime number between range

**package** interviewQ;

**import** java.util.Scanner;

**publicclass** PrimeFromRange {

**publicstaticvoid** main(String args[]) {

// initialize and declare here.

**int**s1, s2, count = 0, i, j;

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

System.***out***.print("Enter the lower limit : ");

s1 = s.nextInt();

System.***out***.print("Enter the upper limit :");

s2 = s.nextInt();

System.***out***.println("Prime numbers between given range are :");

**for** (i = s1; i<= s2; i++) {

**if**(*prime*(i)){

System.***out***.print(i+"\t");

}

}

}

**privatestaticboolean** prime(**int**i) {

// **TODO** Auto-generated method stub

**int**count=0;

**for**(**int**j = 2; j<i; j++)

{

**if**(i % j == 0)

{

count = 0;

**break**;

}

**else**

{

count = 1;

}

}

**if**(count == 1)

{

**returntrue**;

}

**else**

**returnfalse**;

}

}

43.)Write a qsl query to display 2nd highest paid salary

In MYsql

SELECT MAX(sal) FROM `employee`

WHERE sal NOT IN (SELECT MAX(sal) FROM employee);

In sql

Select RANK ,sal

From(

Select Rownum as RANK,sal

From(

Select DISTINCT sal from emp order by sal desc

)

)

Where RANK=&Rank;

In MYSQL

For nth paid salary

SELECT \* /\*This is the outer query part \*/

FROM Employee Emp1

WHERE (n) = ( /\* Subquery starts here \*/

SELECT COUNT(DISTINCT(Emp2.Sal))

FROM Employee Emp2

WHERE Emp2.Sal > Emp1.Sal)

N can be replaced by number.

44.)write a quey to find out the odd position records.

select rank,ename,empno,sal,hiredate

from

(

select rownum as rank,e.\* from emp e

)

where mod(rank,2)=0;

45.)write a query to find out the even position records.

select rank,ename,empno,sal,hiredate

from

(

select rownum as rank,e.\* from emp e

)

where mod(rank,2)!=0;

46.)display the only at position 1,6,10,13

Select rank,ename,empno,sal,hiredate

From(

Select rownum as rank, e.\* from emp e

)

Where rank in(1,6,10,12);

46.)write a query for group by clause

a.)waq to list to count all employee in each department

select deptno,count(\*) as number\_of\_employee

from emp

group by deptno

order by deptno desc;

b.)waq to display employee details from each department which has atleast 5 records.

select deptno,count(\*) as number\_of\_employee

from emp

group by deptno

having count(\*)>=5

order by deptno desc;

47.)write a query to find out the senior employee in company with age in each department

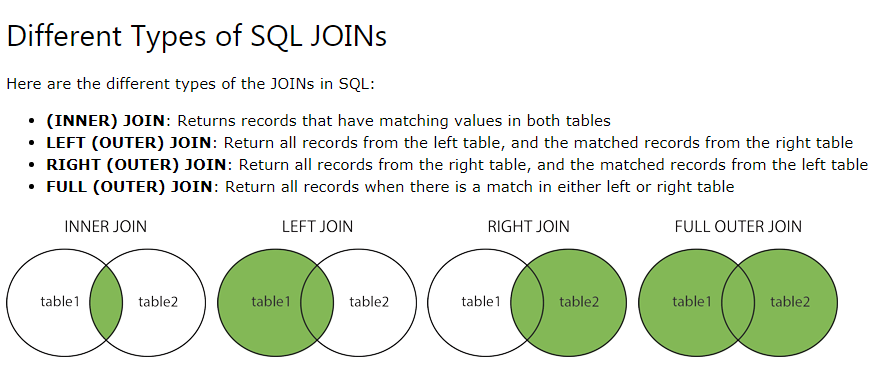
select deptno,min(hiredate) as senior\_employee

from emp

group by deptno

order by deptno desc;

48.)write a query for different join



Inner Join

49.)write a query for inner join

Select empid,ename,sal ,d.loc

From employee e ,dept d

Where e.deptno = d.deprtno

Or

Select empnam, enempid,sal

From emp inner join dept

On emp.deprtno=dept.deprtno

e.g

SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

SELECT map.`res\_statuscode\_mst\_key`,`sys\_code`,`res\_statuscode\_msg`

FROM `res\_statuscode\_map\_mst` map,`res\_statuscode\_mst` mst

WHERE map.`res\_statuscode\_mst\_key`=mst.`res\_statuscode\_mst\_key`;

Join three table

SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName  
FROM ((Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)  
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);

50.)write a query for inner left join

Select Customers.CustName,Order.orderId

From Order left join Customer

On oder.customerid=customers.customerid

Order by customers.customerid asc

51.)Write a query for inner right join

Select Customers.customerId,order.orderId

From Customer right join Order

On Customers.CustomerId=Order.CustomerID

Order by Customers.CustomerId asc;

52.)write Full outer join query

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

50.)WAQ to delete duplicate records from specific table

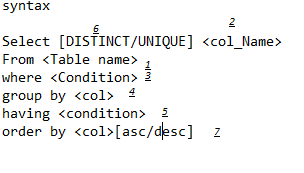
Delete from myemp e

Where rowid>any(select m.rowid from myemp m where

e.empno=m.empno and

e.ename = m.ename)

51.)Describe order of execution of sql query



52.)  Write a Java program to insert an element (specific position) into an array.

Ans- **package** interviewQ;

**import** java.util.Arrays;

**public** **class** Exercise9 {

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

**int**[] arr = {25, 14, 56, 15};

// Insert an element in 3rd position of the array (index->2, value->5)

**int** Index\_position = 2;

**int** newValue = 5;

System.***out***.println("Original Array : "+Arrays.*toString*(arr));

**for**(**int** i=arr.length-1; i > Index\_position; i--){

arr[i] = arr[i-1];

}

arr[Index\_position] = newValue;

System.***out***.println("New Array: "+Arrays.*toString*(arr));

}

}

Original Array : [25, 14, 56, 15]

New Array: [25, 14, 5, 56]

53.) Write a Java program to remove a specific element from an array

54.) Write a Java program to convert an array to ArrayList.

55.) Write a Java program to convert an ArrayList to an array.

56. Write a Java program to test the equality of two arrays.

57.) Write a Java program to compute the average value of an array of integers except the largest and smallest values.

58.)  Write a Java program to remove the duplicate elements of a given array and return the new length of the array.  
Sample array: [20, 20, 30, 40, 50, 50, 50]  
After removing the duplicate elements the program should return 4 as the new length of the array

59.) Write a Java program to segregate all 0s on left side and all 1s on right side of a given array of 0s and 1s

60.)  Write a Java program to replace every element with the next greatest element (from right side) in an given array of integers.

61.)

Output of the following program

**package** interviewQ;

**public** **class** Output1 {

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

// **TODO** Auto-generated method stub

String name="vicky";

System.***out***.println(10+20+name);

System.***out***.println(name+10+20);

}

}

30vicky

vicky1020

**package** interviewQ;

**public** **class** output2 {

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

// **TODO** Auto-generated method stub

**int** i=0;

**int** j=5;

**for**(;(i<10 && j++);i++){

System.***out***.println(i+" "+j);

}

System.***out***.println(i+" "+j);

}

}

Output

Compile time error

**package** interviewQ;

**public** **class** LenOfTheString {

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

// **TODO** Auto-generated method stub

String str[] = {"Ram","Shyam","vicky"};

System.***out***.println("output of string array"+str.length);

System.***out***.println("Length of first array"+str[0].length());

}

}

output of string array3

Length of first array3

**package** interviewQ;

**public** **class** \_2ndDimen {

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

**int**[][] arrInt = { { 1, 2,3 }, { 3, 4, 5 } };

**int** sum=0;

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

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

System.***out***.print((sum=sum+arrInt[i][j]) + " ");

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

}

System.***out***.println("");

}

}

}

62.)How to create an object of SessionFactory?

SessionFactory factory = new Configuration().config().buildSessionFactory();

Session session = factory.getCurrentSession();

Or Session session = factory.opensession();

63.)Difference between openSession() and getCurrentSession();

Ans:-openSession() of SessionFactory object used to create a fresh session object but we need to explicitly flus and close the object .It is not thread safe so one session object should be created for each request in multithread environment and in web application too

getCurrentsession():-

If we call getCurrentSession() with respect to factory object then it will create an object of session if and only if does noyt exits else use same session present in hibernate context and hibernate internally flush and close the session object.

If you are using hibernate in single threaded environment , you can use getCurrentSession, as it is faster in performance as compare to creating new session each time.

You need to add following property to hibernate.cfg.xml to use getCurrentSession method.

<session-factory>

<!-- Put other elements here -->

<property name="hibernate.current\_session\_context\_class">

thread

</property>

</session-factory>

64.)what is proxy object in hibernate?

Proxy is class created by hibernate internally to help load the object either lazy or early based on value of Proxy object.

For example

If we have cat class , Hibernate create a proxy class which extends cat class.

 LAZY = fetch when needed

 EAGER = fetch immediately

Depends on the value of proxy either true or false, object will be loaded from the child object of parent object.

Eager loading is faster than Lazy loading because it hits the database and store it session next time it wont hits the database. But in case of we can get the Java heap error.

How to solve the Java heap (Permgen)error.

Ans:- The solution to fix this problem is to increase the heap space by specifying the jvm parameter

Codility Test

A *binary gap* within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N.

For example, number 9 has binary representation 1001 and contains a binary gap of length 2. The number 529 has binary representation 1000010001 and contains two binary gaps: one of length 4 and one of length 3. The number 20 has binary representation 10100 and contains one binary gap of length 1. The number 15 has binary representation 1111 and has no binary gaps. The number 32 has binary representation 100000 and has no binary gaps.

Write a function:

class Solution { public int solution(int N); }

that, given a positive integer N, returns the length of its longest binary gap. The function should return 0 if N doesn't contain a binary gap.

For example, given N = 1041 the function should return 5, because N has binary representation 10000010001 and so its longest binary gap is of length 5. Given N = 32 the function should return 0, because N has binary representation '100000' and thus no binary gaps.

Write an efficient algorithm for the following assumptions:

* N is an integer within the range [1..2,147,483,647].

Solution

**package** interviewQ;

**public** **class** GapLenOfBinary {

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

// **TODO** Auto-generated method stub

**int** gap=0;

**long** binary = 1001;

gap = (**int**)*solution*(binary);

System.***out***.println(gap);

}

**public** **static** **int** solution(**long** N) {

// write your code in Java SE 8

System.***out***.println(N);

**int** gap=0;

**int** len =0;

**int** max=0;

**int** index=0;

**int** []lenn=**new** **int**[5];

String s= String.*valueOf*(N);

**if**(Integer.*parseInt*((""+s.charAt(0)).trim())==1 && Integer.*parseInt*((""+s.charAt(s.length()-1)).trim())==1){

**for**(**int** i=1;i<s.length();i++){

**if**(Integer.*parseInt*((""+s.charAt(i)).trim())==0){

len++;

}

**else**{

lenn[index++]=len;

}

}

}

**else** **if**(Integer.*parseInt*((""+s.charAt(0)).trim())==1 && Integer.*parseInt*((""+s.charAt(s.length()-1)).trim())!=1){

**for**(**int** i=1;i<s.length()-1;i++){

**if**(Integer.*parseInt*((""+s.charAt(i)).trim())==0){

len++;

}

**else**{

lenn[index++]=len;

}

}

}

**else** {

max=0;

}

max = lenn[0];

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

**if**(lenn[i]>max){

max=lenn[i];

}

}

**return** max;

}

}

63.)What is singleton design pattern?

a.)Singleton design pattern restrict to initialize instance of class and make sure to create a single object to be used through out the application.

b.)The singleton class must provide single global point of access of instance of class

Where we use singleton?

We use the singleton for logging , driver object ,caching ,Thread pool etc or when we want to have a single object to be accessed through out the application.

To implement Singleton pattern, we have different approaches but all of them have following common concepts.

* Private constructor to restrict instantiation of the class from other classes.
* Private static variable of the same class that is the only instance of the class.
* Public static method that returns the instance of the class, this is the global access point for outer world to get the instance of the singleton class.

Different way of creating singleton object

1. [**Eager initialization**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#eager-initialization)
2. [**Static block initialization**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#static-block-initialization)
3. [**Lazy Initialization**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#lazy-initialization)
4. [**Thread Safe Singleton**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#thread-safe-singleton)

[**Eager initialization**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#eager-initialization)

Public class MySingletone{

Private static MySingleton singletone = new MySingleton();

Private MySingletone(){

}

Public static MySingletone getInstance(){

Return singletone

}

3.[**Lazy Initialization**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#lazy-initialization)

Public class MySingletone{

Private static MySingleton singletone;

Private MySingletone(){

}

Public static MySingletone getInstance(){

If(singletone==null){

singletone = new MySingleton();

return singletone;

else

return singletone;

}

Return singletone

}

**2.**[**Static block initialization**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#static-block-initialization)

**When one class only use instance of singletone then we use static singletone**

public class MySingleton{

private static MySingletone singletone ;

private MySingletone(){

}

Static{

Try{

singletone = new Singletone();

}catch(Excetion e){

throw new RuntimeException("Exception occured in creating singleton instance");

}

}

}

**4.**[**Thread Safe Singleton**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples#thread-safe-singleton)

Public class MySIngletone

{

Private static MySingletone singletone;

Private MySingletone(){}

Public static synchronized MySingletone getInstance(){

If(singletone==null){

Singletone = new MySingletone();

Return Singletone

}

Else{

Return Singletone

}

}

}

5.Synchronized with cloneable

Public class MySIngletone implements cloneable

{

Private static MySingletone singletone;

Private MySingletone(){}

Public static synchronized MySingletone getInstance(){

If(singletone==null){

Singletone = new MySingletone();

Return Singletone

}

Else{

Return Singletone

}

}

public Object clone()throws CloneNotSupportedException{

return throw CloneNotSupportedException(“Instance can not be created”);

}

}

Factory Design pattern

Factory design pattern is used when we have a super class with multiple sub-classes and based on input, we need to return one of the sub-class. This pattern take out the responsibility of instantiation of a class from client program to the factory class.

**package** com.hcl.pojopoji;

**public** **interface** Person {

**abstract** **public** **void** greatings(String name);

}

**package** com.hcl.pojopoji;

**public** **class** Male **implements** Person {

@Override

**public** **void** greatings(String name) {

// **TODO** Auto-generated method stub

System.***out***.println("Hi Mr="+name);

}

}

**package** com.hcl.pojopoji;

**public** **class** Female **implements** Person {

@Override

**public** **void** greatings(String name) {

// **TODO** Auto-generated method stub

System.***out***.println("Hi Mr="+name);

}

}

**package** com.factorymethod;

**import** com.hcl.pojopoji.Female;

**import** com.hcl.pojopoji.Male;

**import** com.hcl.pojopoji.Person;

**public** **class** PersonFactoryPattern {

**public** **static** Person getInstance(String name){

**if**(name.equalsIgnoreCase("male")){

**return** **new** Male();

}

**else**{

**return** **new** Female();

}

}

}

**package** main.claus;

**import** com.factorymethod.PersonFactoryPattern;

**import** com.hcl.pojopoji.Person;

**public** **class** TestFactory {

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

// **TODO** Auto-generated method stub

String gender="Male";

PersonFactoryPattern p = **new** PersonFactoryPattern();

Person person = p.*getInstance*(gender);

person.greatings("Vicky");

}

}

64.)Prototype design pattern

Prototype design pattern is used when object creation is costly affair and required a lot of resources.

Prototype provides a mechanism to copy the object into new object and then modify the object accordingly based on needs.

Protoype uses java cloning to copy an object.

Example of Protype design pattern.

Suppose we loaded the data from database into object and now we required a lot of manipulation in different place. So we should copy the object then we should modify the object instead of modification on original object.it performs the deep copy.

Program of prototype

package com.model;

import java.util.ArrayList;

import java.util.List;

public class Employees implements Cloneable{

private List<String> empList;

public Employees(){

empList = new ArrayList<String>();

}

public Employees(List<String> list){

this.empList=list;

}

public void loadData(){

//read all employees from database and put into the list

empList.add("Pankaj");

empList.add("Raj");

empList.add("David");

empList.add("Lisa");

}

public List<String> getEmpList() {

return empList;

}

@Override

public Object clone() throws CloneNotSupportedException{

List<String> temp = new ArrayList<String>();

for(String s : this.getEmpList()){

temp.add(s);

}

return new Employees(temp);

}

}

**package** com.model;

**import** java.util.List;

**public** **class** PrototypeDesignpattern {

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

Employees emps = **new** Employees();

emps.loadData();

//Use the clone method to get the Employee object

Employees empsNew = (Employees) emps.clone();

Employees empsNew1 = (Employees) emps.clone();

List<String> list = empsNew.getEmpList();

list.add("John");

List<String> list1 = empsNew1.getEmpList();

list1.remove("Pankaj");

System.***out***.println("emps List: "+emps.getEmpList());

System.***out***.println("empsNew List: "+list);

System.***out***.println("empsNew1 List: "+list1);

}

}

65.)Façade design pattern

Façade design pattern whch make the client easy to interact with system.

Façade design pattern provides the unified interface to a set of interfaces in a system.

For example

We have requirement to interact with many databases like mysql,sql,hbase etc to generate the report in different format. So we use different interfaces(helper interface) for different databases .But what will happended if complexity of system get increased or interface behavior is so confusing in this case façade provided wrapper interface on top of existing interface to helo client application.

Example

**package** com.hc.database;

**import** java.sql.Connection;

**public** **class** MySqlHelper {

**public** **static** Connection getMySqlDBConnection(){

//get MySql DB connection using connection parameters

**return** **null**;

}

**public** **void** generateMySqlPDFReport(String tableName, Connection con){

//get data from table and generate pdf report

}

**public** **void** generateMySqlHTMLReport(String tableName, Connection con){

//get data from table and generate pdf report

}

}

**package** com.hc.database;

**import** java.sql.Connection;

**public** **class** OracleHelper {

**public** **static** Connection getOracleDBConnection(){

//get Oracle DB connection using connection parameters

**return** **null**;

}

**public** **void** generateOraclePDFReport(String tableName, Connection con){

//get data from table and generate pdf report

}

**public** **void** generateOracleHTMLReport(String tableName, Connection con){

//get data from table and generate pdf report

}

}

We can create a Facade pattern interface like below. Notice the use of [Java Enum](https://www.journaldev.com/716/java-enum) for **type safety.**

HelperFacade.java

**package** com.hc.facade;

**import** java.sql.Connection;

**import** com.hc.database.MySqlHelper;

**import** com.hc.database.OracleHelper;

**public** **class** HelperFacade {

**public** **static** **void** generateReport(DBTypes dbType, ReportTypes reportType,

String tableName) {

Connection con = **null**;

**switch** (dbType) {

**case** ***MYSQL***:

con = MySqlHelper.*getMySqlDBConnection*();

MySqlHelper mySqlHelper = **new** MySqlHelper();

**switch** (reportType) {

**case** ***HTML***:

mySqlHelper.generateMySqlHTMLReport(tableName, con);

**break**;

**case** ***PDF***:

mySqlHelper.generateMySqlPDFReport(tableName, con);

**break**;

}

**break**;

**case** ***ORACLE***:

con = OracleHelper.*getOracleDBConnection*();

OracleHelper oracleHelper = **new** OracleHelper();

**switch** (reportType) {

**case** ***HTML***:

oracleHelper.generateOracleHTMLReport(tableName, con);

**break**;

**case** ***PDF***:

oracleHelper.generateOraclePDFReport(tableName, con);

**break**;

}

**break**;

}

}

**public** **static** **enum** DBTypes {

***MYSQL***, ***ORACLE***;

}

**public** **static** **enum** ReportTypes {

***HTML***, ***PDF***;

}

}

66.)Adaptor design pattern

Adaptore design patter is structural design pattern used to work with unrelated interfaces. The object that join these unrelated interface is called Adaptor.

One of the great real life example of Adaptor design pattern is mobile charger.

Mobile battery need 3 volts to charge but generally socket produces either 120v (in us) or 240v(in india) .here mobile charge behaves as adaptor in the middle wall socket and moble charging socket.

We will try to implement multi-adapter using adapter design pattern .

So first of all we will have two classes – **Volt** (to measure volts) and **Socket** (producing constant volts of 120V).

**package** com.voltmeasurement;

**public** **class** Volt {

**private** **int** volts;

**public** Volt(**int** v){

**this**.volts=v;

}

**public** **int** getVolts() {

**return** volts;

}

**public** **void** setVolts(**int** volts) {

**this**.volts = volts;

}

}

**package** com.voltmeasurement;

**public** **class** Socket {

**public** Volt getVolt(){

**return** **new** Volt(120);

}

}

Now we want to build an adapter that can produce 3 volts, 12 volts and default 120 volts. So first of all we will create an adapter interface with these methods.

**package** com.adaptor;

**import** com.voltmeasurement.Volt;

**public** **interface** SocketAdaptor {

**public** Volt get120Volt();

**public** Volt get12Volt();

**public** Volt get3Volt();

}

**Two Way Adapter Pattern**

While implementing Adapter pattern, there are two approaches – class adapter and object adapter – however both these approaches produce same result.

1. Class Adapter – This form uses [java inheritance](https://www.journaldev.com/644/inheritance-java-example) and extends the source interface, in our case Socket class.
2. Object Adapter – This form uses [Java Composition](https://www.journaldev.com/1325/composition-in-java-example) and adapter contains the source object

**Adapter Design Pattern – Class Adapter**

Here is the class adapter approach implementation of our adapter.

package com.adaptor;

import com.voltmeasurement.Socket;

import com.voltmeasurement.Volt;

//Using inheritance for adapter pattern

public class SocketClassAdapterImpl extends Socket implements SocketAdaptor{

@Override

public Volt get120Volt() {

return getVolt();

}

@Override

public Volt get12Volt() {

Volt v= getVolt();

return convertVolt(v,10);

}

@Override

public Volt get3Volt() {

Volt v= getVolt();

return convertVolt(v,40);

}

private Volt convertVolt(Volt v, int i) {

return new Volt(v.getVolts()/i);

}

}

Or

**package** com.adaptor;

**import** com.voltmeasurement.Socket;

**import** com.voltmeasurement.Volt;

**public** **class** SocketObjectAdaptorImpl **implements** SocketAdaptor{

//Using Composition for adapter pattern

**private** Socket sock = **new** Socket();

@Override

**public** Volt get120Volt() {

**return** sock.getVolt();

}

@Override

**public** Volt get12Volt() {

Volt v= sock.getVolt();

**return** convertVolt(v,10);

}

@Override

**public** Volt get3Volt() {

Volt v= sock.getVolt();

**return** convertVolt(v,40);

}

**private** Volt convertVolt(Volt v, **int** i) {

**return** **new** Volt(v.getVolts()/i);

}

}

**Adapter Design Pattern – Object Adapter Implementation**

Here is the Object adapter implementation of our adapter.

Notice that both the adapter implementations are almost same and they implement the **SocketAdapterinterface**. The adapter interface can also be an [**abstract class**](https://www.journaldev.com/1582/abstract-class-in-java).

Test

**import** com.adaptor.SocketAdaptor;

**import** com.adaptor.SocketClassAdapterImpl;

**import** com.adaptor.SocketObjectAdaptorImpl;

**import** com.voltmeasurement.Volt;

**public** **class** AdapterPatternTest {

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

*testClassAdapter*();

*testObjectAdapter*();

}

**private** **static** **void** testObjectAdapter() {

SocketAdaptor sockAdapter = **new** SocketObjectAdaptorImpl();

Volt v3 = *getVolt*(sockAdapter,3);

Volt v12 = *getVolt*(sockAdapter,12);

Volt v120 = *getVolt*(sockAdapter,120);

System.***out***.println("v3 volts using Object Adapter="+v3.getVolts());

System.***out***.println("v12 volts using Object Adapter="+v12.getVolts());

System.***out***.println("v120 volts using Object Adapter="+v120.getVolts());

}

**private** **static** **void** testClassAdapter() {

SocketAdaptor sockAdapter = **new** SocketClassAdapterImpl();

Volt v3 = *getVolt*(sockAdapter,3);

Volt v12 = *getVolt*(sockAdapter,12);

Volt v120 = *getVolt*(sockAdapter,120);

System.***out***.println("v3 volts using Class Adapter="+v3.getVolts());

System.***out***.println("v12 volts using Class Adapter="+v12.getVolts());

System.***out***.println("v120 volts using Class Adapter="+v120.getVolts());

}

**private** **static** Volt getVolt(SocketAdaptor sockAdapter, **int** i) {

**switch** (i){

**case** 3: **return** sockAdapter.get3Volt();

**case** 12: **return** sockAdapter.get12Volt();

**case** 120: **return** sockAdapter.get120Volt();

**default**: **return** sockAdapter.get120Volt();

}

}

}

Remarks:-Java Composition is called has-a relationship which is type of inheritance.

Part-of(composite)

Where we should use has-a(Aggerregate) and is-a(inheritance) to extend the feature of the class.

* If The new class **is** more or less as the original class. Use inheritance. The new class is now a subclass of the original class.
* If the new class must **have** the original class. Use aggregation. The new class has now the original class as a member.

67.)Delegate pattern

68.)Entity pattern

69.)Example of creating an object of class without new operator using example.

**package** interviewQ;

**class** Simple {

**void** message() {

System.***out***.println("Hello Java");

}

}

**class** Test2 {

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

**try** {

Class c = Class.*forName*("interviewQ.Simple");

Simple s = (Simple) c.newInstance();

s.message();

System.***out***.println(s.getClass().getName());

} **catch** (Exception e) {

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

}

}

}

Output

Hello Java

interviewQ.Simple

give the output of given below program

**package** interviewQ;

**class** Simple {

**void** message() {

System.***out***.println("Hello Java");

}

}

**class** Test2 {

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

**try** {

Class c = Class.*forName*("Simple");

Simple s = (Simple) c.newInstance();

s.message();

System.***out***.println(s.getClass().getName());

} **catch** (Exception e) {

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

}

}

}

output

java.lang.ClassNotFoundException: Simple

**public** Example(**int** a) {

System.***out***.println(a);

}

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

// **TODO** Auto-generated method stub

**try**{

Example e = **new** Example(**new** Integer(10));

Example e1 = **new** Example(**new** Integer(**new** Integer(12)));

Example e2= **new** Example(**new** Integer(**null**));

Example e3 = **new** Example(**new** Integer(**null**));

}**catch**(Exception e){

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

}

}

}

Output

10

12

java.lang.NumberFormatException: null

68.) how to integrate the spring and hibernate?

Spring 4.0 can support both hibernate 3.0 and hibernate 4.0.

We can integrate the hibernate using spring configuration file.example is given below

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

destroy-method="close">

<property name="driverClassName" value="com.mysql.jdbc.Driver" />

<property name="url" value="jdbc:mysql://localhost:3306/TestDB" />

<property name="username" value="pankaj" />

<property name="password" value="pankaj123" />

</bean>

<!-- Hibernate 3 XML SessionFactory Bean definition-->

<!-- <bean id="hibernate3SessionFactory"

class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="mappingResources">

<list>

<value>person.hbm.xml</value>

</list>

</property>

<property name="hibernateProperties">

<value>

hibernate.dialect=org.hibernate.dialect.MySQLDialect

</value>

</property>

</bean> -->

<!-- Hibernate 3 Annotation SessionFactory Bean definition-->

<bean id="hibernate3AnnotatedSessionFactory"

class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="annotatedClasses">

<list>

<value>com.journaldev.model.Person</value>

</list>

</property>

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>

<prop key="hibernate.current\_session\_context\_class">thread</prop>

<prop key="hibernate.show\_sql">false</prop>

</props>

</property>

</bean>

<bean id="personDAO" class="com.journaldev.dao.PersonDAOImpl">

<property name="sessionFactory" ref="hibernate3AnnotatedSessionFactory" />

</bean>

</beans>

70.) @Transactional and without @Transactional

public class PersonDAOImpl implements PersonDAO {

private SessionFactory sessionFactory;

public void setSessionFactory(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

@Override

public void save(Person p) {

Session session = this.sessionFactory.openSession();

Transaction tx = session.beginTransaction();

session.persist(p);

tx.commit();

session.close();

}

@SuppressWarnings("unchecked")

@Override

public List<Person> list() {

Session session = this.sessionFactory.openSession();

List<Person> personList = session.createQuery("from Person").list();

session.close();

return personList;

}

}

Using @Transactional

package com.journaldev.spring.jdbc.dao;

import com.journaldev.spring.jdbc.model.Customer;

public interface CustomerDAO {

public void create(Customer customer);

}

package com.journaldev.spring.jdbc.dao;

import javax.sql.DataSource;

import org.springframework.jdbc.core.JdbcTemplate;

import com.journaldev.spring.jdbc.model.Customer;

public class CustomerDAOImpl implements CustomerDAO {

private DataSource dataSource;

public void setDataSource(DataSource dataSource) {

this.dataSource = dataSource;

}

@Override

public void create(Customer customer) {

String queryCustomer = "insert into Customer (id, name) values (?,?)";

String queryAddress = "insert into Address (id, address,country) values (?,?,?)";

JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

jdbcTemplate.update(queryCustomer, new Object[] { customer.getId(),

customer.getName() });

System.out.println("Inserted into Customer Table Successfully");

jdbcTemplate.update(queryAddress, new Object[] { customer.getId(),

customer.getAddress().getAddress(),

customer.getAddress().getCountry() });

System.out.println("Inserted into Address Table Successfully");

}

}

Notice that CustomerDAO implementation is not taking care of transaction management. This way we are achieving separation of concerns because sometimes we get DAO implementations from third party and we don’t have control on these classes

Let’s create a Customer Service that will use the CustomerDAO implementation and provide transaction management when inserting records in the customer and address tables in a single method.

package com.journaldev.spring.jdbc.service;

import com.journaldev.spring.jdbc.model.Customer;

public interface CustomerManager {

public void createCustomer(Customer cust);

}

package com.journaldev.spring.jdbc.service;

import org.springframework.transaction.annotation.Transactional;

import com.journaldev.spring.jdbc.dao.CustomerDAO;

import com.journaldev.spring.jdbc.model.Customer;

public class CustomerManagerImpl implements CustomerManager {

private CustomerDAO customerDAO;

public void setCustomerDAO(CustomerDAO customerDAO) {

this.customerDAO = customerDAO;

}

@Override

@Transactional

public void createCustomer(Customer cust) {

customerDAO.create(cust);

}

}

If you notice the CustomerManager implementation, it’s just using CustomerDAO implementation to create the customer but provide declarative transaction management through annotating createCustomer() method with @Transactional annotation. That’s all we need to do in our code to get the benefits of Spring transaction management.

@Transactional annotation can be applied over methods as well as whole class. If you want all your methods to have transaction management features, you should annotate your class with this annotation. Read more about annotations at Java Annotations Tutorial.

The only part remaining is wiring spring beans to get spring transaction management example to work.

References:

<https://app.codility.com/candidate-faq/>

<https://simpleprogrammer.com/programming-interview-questions/>

<https://www.javacodegeeks.com/2012/06/avoid-null-pointer-exception-in-java.html>

<https://www.w3resource.com/java-exercises/array/index.php>