<http://java67.blogspot.com/2012/08/10-java-coding-interview-questions-and.html>

<http://www.java2novice.com/java-interview-programs/>

1. [Comparator\_Compare](#Comparator_Compare)
2. [float\_double](#float_double)
3. [fibonacci](#fibonacci)
4. [ArmStrong](#ArmStrong)
5. [Java internalization example](#Internalization)
6. [Swap two number](#Swap)
7. [Linked List fine middle node](#LinkedList_middle)
8. [Palindrome number](#Palindrome) (if reverse is equal to number logic: reverse = reverse\*10 + number%10;

number = number/10;

1. [Factorial of given number:](#Factorial)

Comparable and Comparator examples

Comparable for natural ordering

Comparator for comparing the already tested and external objects

1. ComparatorChain chain =  new ComparatorChain();
2. chain.addComparator(new EmployeeNameComparator());
3. chain.addComparator(new EmployeeDOJComparator());
4. // sorting the Employee object
5. Collections.sort(list,chain);

<http://java-espresso.blogspot.com/2011/06/comparable-and-comparator-interfaces.html>

ComparableExample

**package** test;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** ComparableExample {

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

Employee a1= **new** Employee("Nages",195321,73400);

Employee a2= **new** Employee("Vishwa",389293,67200);

Employee a3= **new** Employee("Aishwa",389293,80200);

List<Employee> l = **new** ArrayList<Employee>();

l.add(a1);

l.add(a2);

l.add(a3);

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

System.*out*.println("Name "+"|Id "+"|Salary ");

System.*out*.println("\_\_\_\_\_\_ "+"\_\_\_"+"\_\_\_\_\_\_\_ ");

**for**(Employee e:l){

System.*out*.println(e.getName()+" "+e.getId()+" "+e.getSalary());

}

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

System.*out*.println("Name "+"|Id "+"|Salary ");

System.*out*.println("\_\_\_\_\_\_ "+"\_\_\_"+"\_\_\_\_\_\_\_ ");

Collections.*sort*(l);

**for**(Employee e:l){

System.*out*.println(e.getName()+" "+e.getId()+" "+e.getSalary());

}

}

}

**class** Employee **implements** Comparable<Employee> {

**private** String name;

**private** **int** id;

**private** **double** salary;

**public** Employee(String name, **int** id, **double** salary){

**this**.name=name;

**this**.id=id;

**this**.salary=salary;

}

@Override

**public** **int** compareTo(Employee o) {

**if** (o == **null**) {

**throw** **new** NullPointerException("compareTo: Argument passed is null");

}

**if** (**this**.getClass().equals(o.getClass())) {

Employee obj = (Employee) o;

**return** **this**.getName().compareTo(obj.getName());

//return this.getId()-o.getId();

//return (int) (this.getSalary()-o.getSalary());

} **else** {

**throw** **new** ClassCastException("Objects are not comparable");

}

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

}

**How will the below literal value be internally represented?  
float f = 21.22;**

It will be represented as a double value. Floating point literals are always double by default. If you want a float, you must append an F or f to the literal.

Fibonacci program

**package** test.main;

**import** java.util.Scanner;

/\*\*

\* Java program to calculate and print Fibonacci number using both recursion and Iteration.

\* Fibonacci number is sum of previous two Fibonacci numbers fn= fn-1+ fn-2

\* first 10 Fibonacci numbers are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

\* **@author**

\*/

**public** **class** FibonacciCalculator {

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

// input to print Fibonacci series upto how many numbers

System.*out*.println("Enter number upto which Fibonacci series to print: ");

**int** number = **new** Scanner(System.*in*).nextInt();

System.*out*.println("Fibonacci series upto " + number + " numbers : ");

// printing Fibonacci series upto number

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

System.*out*.print(*fibonacci2*(i) + " ");

}

}

/\*

\* Java program for Fibonacci number using recursion. This program uses tail

\* recursion to calculate Fibonacci number for a given number

\*

\* @return Fibonacci number

\*/

**public** **static** **int** fibonacci(**int** number) {

**if** (number == 1 || number == 2) {

**return** 1;

}

**return** *fibonacci*(number - 1) + *fibonacci*(number - 2); // tail recursion

}

/\*

\* Java program to calculate Fibonacci number using loop or Iteration.

\*

\* @return Fibonacci number

\*/

**public** **static** **int** fibonacci2(**int** number) {

**if** (number == 1 || number == 2) {

**return** 1;

}

**int** fibo1 = 1, fibo2 = 1, fibonacci = 1;

**for** (**int** i = 3; i <= number; i++) {

fibonacci = fibo1 + fibo2; // Fibonacci number is sum of previous

// two Fibonacci number

fibo1 = fibo2;

fibo2 = fibonacci;

}

**return** fibonacci; // Fibonacci number

}

}

Armstrong:

**package** exam;

**import** java.util.Scanner;

**public** **class** ArmstrongTest {

/\*\*

\* **@param** args

\*/

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

System.*out*.println("input number to test");

**int** number = **new** Scanner(System.*in*).nextInt();

System.*out*.println("Given number is ArmStrong : "+*isArmstrong*(number));

}

**private** **static** **boolean** isArmstrong(**int** number) {

**int** result=0;

**int** org\_num = number;

**if**(number==0)

**return** **false**;

**else**

**while**(number!=0){

**int** remainder = number%10;

result=result+remainder\*remainder\*remainder;

number=number/10;

}

**if**(result==org\_num)

**return** **true**;

**else** **return** **false**;

}

}

**Write a Java program to prevent deadlock in Java ?**

they may ask you to write code which will result in deadlock ?

here is one of my version

**public** **void** method1(){

**synchronized**(String.**class**){

System.out.println("Aquired lock on String.class object");

**synchronized** (Integer.**class**) {

System.out.println("Aquired lock on Integer.class object");

}

}

}

**public** **void** method2(){

**synchronized**(Integer.**class**){

System.out.println("Aquired lock on Integer.class object");

**synchronized** (String.**class**) {

System.out.println("Aquired lock on String.class object");

}

}

}

If method1() and method2() both will be called by two or many threads , there is a good chance of deadlock because **if** thead 1 aquires lock on Sting object **while** executing method1() and thread 2 acquires lock on Integer object **while** executing method2() both will be waiting **for** each other to release lock on Integer and String to proceed further which will never happen.

now interviewer comes to **final** part , one of the most important in my view , How to fix deadlock ? or How to avoid deadlock in Java ?

**if** you have looked above code carefully you may have figured out that real reason **for** deadlock is not multiple threads but the way they access lock , **if** you provide an ordered access then problem will be resolved , here is

the fixed version.

**public** **void** method1(){

**synchronized**(Integer.**class**){

System.out.println("Aquired lock on Integer.class object");

**synchronized** (String.**class**) {

System.out.println("Aquired lock on String.class object");

}

}

}

**public** **void** method2(){

**synchronized**(Integer.**class**){

System.out.println("Aquired lock on Integer.class object");

**synchronized** (String.**class**) {

System.out.println("Aquired lock on String.class object");

}

}

}

Now there would not be any deadlock because both method is accessing lock on Integer and String object in same order . so **if** thread A acquires lock on Integer object , thread B will not proceed until thread A releases Integer lock , same way thread A will not be blocked even **if** thread B holds String lock because now thread B will not expect thread A to release Integer lock to proceed further

Read more: <http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html#ixzz35OTgw2Yv>

### [InternationalizationTest.java](http://www.avajava.com/tutorials/general-java/how-do-i-use-locales-and-resource-bundles-to-internationalize-my-application/InternationalizationTest.java)

package test;

import java.util.Locale;

import java.util.ResourceBundle;

public class InternationalizationTest {

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

ResourceBundle bundle1 = ResourceBundle.getBundle("TestBundle");

displayValues(bundle1);

Locale defaultLocale = Locale.getDefault();

ResourceBundle bundle2 = ResourceBundle.getBundle("TestBundle", defaultLocale);

displayValues(bundle2);

Locale swedishLocale = new Locale("sv", "SE");

ResourceBundle bundle3 = ResourceBundle.getBundle("TestBundle", swedishLocale);

displayValues(bundle3);

Locale nonexistentLocale = new Locale("xx", "XX");

ResourceBundle bundle4 = ResourceBundle.getBundle("TestBundle", nonexistentLocale);

displayValues(bundle4);

}

public static void displayValues(ResourceBundle bundle) {

System.out.println("hello message:" + bundle.getString("my.hello"));

System.out.println("goodbye message:" + bundle.getString("my.goodbye"));

System.out.println("question message:" + bundle.getString("my.question"));

System.out.println();

}

}

**Swap two numbers:**

**package** exam;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.util.Scanner;

**public** **class** SwapTwoNums {

/\*\*

\* **@param** args

Method:1

a = a+ b; //now a is 30 and b is 20

b = a -b; //now a is 30 but b is 10 (original value of a)

a = a -b; //now a is 20 and b is 10, numbers are swapped

Method 2:

a = a\*b; //now a is 18 and b is 3

b = a/b; //now a is 18 but b is 6 (original value of a)

a = a/b; //now a is 3 and b is 6, numbers are swapped

Read more: http://javarevisited.blogspot.com/2013/02/swap-two-numbers-without-third-temp-variable-java-program-example-tutorial.html#ixzz35Vu1LhR2

Method 3:

Using Below:

\*/

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

System.*out*.println("Please enter two numbers");

// int a=0,b=0;

// BufferedReader br = new BufferedReader(new

// InputStreamReader(System.in));

**int**[] array = **new** **int**[2];

**try** {

Scanner in = **new** Scanner(System.*in*); // Import java.util.Scanner for

// it

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

**int** k = in.nextInt();

array[j] = k;

}

} **catch** (Exception e) {

e.printStackTrace();

}

// int[] args= new int[argss.length];

**int** a = array[0];

**int** b = array[1];

System.*out*.println("Before swap two numbers " + a + ":" + b);

a = a ^ b;

b = a ^ b;

a = a ^ b;

System.*out*.println("After swap two numbers " + a + ":" + b);

}

}

**package** exam;

**public** **class** LinkedListTest {

/\*How to find middle element of LinkedList in one pass

Read more: http://javarevisited.blogspot.com/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html#ixzz35Vv77X8R

http://javarevisited.blogspot.com/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html

\*/

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

//creating LinkedList with 5 elements including head

LinkedList linkedList = **new** LinkedList();

LinkedList.Node head = linkedList.head();

linkedList.add( **new** LinkedList.Node("1"));

linkedList.add( **new** LinkedList.Node("2"));

linkedList.add( **new** LinkedList.Node("3"));

linkedList.add( **new** LinkedList.Node("4"));

//finding middle element of LinkedList in single pass

LinkedList.Node current = head;

**int** length = 0;

LinkedList.Node middle = head;

**while**(current.next() != **null**){

length++;

**if**(length%2 ==0){

middle = middle.next();

}

current = current.next();

}

**if**(length%2 == 1){

middle = middle.next();

}

System.*out*.println("length of LinkedList: " + length);

System.*out*.println("middle element of LinkedList : " + middle);

}

}

**class** LinkedList{

**private** Node head;

**private** Node tail;

**public** LinkedList(){

**this**.head = **new** Node("head");

tail = head;

}

**public** Node head(){

**return** head;

}

**public** **void** add(Node node){

tail.next = node;

tail = node;

}

**public** **static** **class** Node{

**private** Node next;

**private** String data;

**public** Node(String data){

**this**.data = data;

}

**public** String data() {

**return** data;

}

**public** **void** setData(String data) {

**this**.data = data;

}

**public** Node next() {

**return** next;

}

**public** **void** setNext(Node next) {

**this**.next = next;

}

**public** String toString(){

**return** **this**.data;

}

}

}

**package** exam;

**import** java.util.Scanner;

/\*\*

\* This Java program takes an input number from command line and integer array

\* and check if number is palindrome or not. A number is called palindrome

\* if number is equal to reverse of number itself.

Read more: http://javarevisited.blogspot.com/2012/12/how-to-check-if-number-is-palindrome-or-not-example.html#ixzz35VwjHMIQ

\*

\* **@author** Javin Paul

\*/

**public** **class** PalindromeTest {

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

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

//int number = scanner.nextInt();

**int**[] numbers = {1, 20, 22, 102, 101, 1221, 13321, 13331, 0, 11};

**for**(**int** number: numbers){

System.*out*.println("Does number : "

+ number +" is a palindrome? " + *isPalindrome*(number));

}

}

**private** **static** **boolean** isPalindrome(**int** number) {

**if**(number == *reverse*(number)){

**return** **true**;

}

**return** **false**;

}

**private** **static** **int** reverse(**int** number){

**int** reverse = 0;

**while**(number != 0){

reverse = reverse\*10 + number%10;

number = number/10;

}

**return** reverse;

}

}

/\*

Output

Does number : 1 is a palindrome? true

Does number : 20 is a palindrome? false

Does number : 22 is a palindrome? true

Does number : 102 is a palindrome? false

Does number : 101 is a palindrome? true

Does number : 1221 is a palindrome? true

Does number : 13321 is a palindrome? false

Does number : 13331 is a palindrome? true

Does number : 0 is a palindrome? true

Does number : 11 is a palindrome? true\*/

**Factorial of a given number:**

**package** exam;

/\*\*

\* Simple Java program to find factorial of a number using recursion and iteration.

\* Iteration will use for loop while recursion will call method itself

\*/

**public** **class** FactorialInJava{

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

//finding factorial of a number in Java using recursion - Example

System.*out*.println("factorial of 5 using recursion in Java is: " + *factorial*(5));

//finding factorial of a number in Java using Iteration - Example

System.*out*.println("factorial of 6 using iteration in Java is: " + *fact*(6));

}

/\*

\* Java program example to find factorial of a number using recursion

\* @return factorial of number

\*/

**public** **static** **int** factorial(**int** number){

//base case

**if**(number == 0){

**return** 1;

}

**return** number\**factorial*(number -1); //is this tail-recursion?

}

/\*

\* Java program example to calculate factorial using while loop or iteration

\* @return factorial of number

\*/

**public** **static** **int** fact(**int** number){

**int** result = 1;

**while**(number != 0){

result = result\*number;

number--;

}

**return** result;

}

}

/\*

Output:

factorial of 5 using recursion in Java is: 120

factorial of 6 using iteration in Java is: 720

Read more: http://javarevisited.blogspot.com/2012/04/java-program-to-find-factorial-of.html#ixzz35VyljCKW\*/