1.Write 3 different java programs to print the following patterns

                a) 1

                   12

                   123

                   12345

class NumberPattern

{

public static void main(String[] args)

{

for (int i = 1; i <= 5; i++)

{

for (int j = 1; j <= i; j++)

{

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

}

System.out.println();

}

}

}

                b) 54321

                   5432

                   543

                   54

                   5

import java.util.Scanner;

public class MainClass

{

     public static void main(String[] args)

     {

         Scanner sc = new Scanner(System.in);

         //Taking rows value from the user

         System.out.println("How many rows you want in this pattern?");

         int rows = sc.nextInt();

         System.out.println("Here is your pattern....!!!");

         for (int i = 1; i <= rows; i++)

         {

             for (int j = rows; j >= i; j--)

             {

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

             }

             System.out.println();

         }

     }

}

                c)     x

                      xxx

                     xxxxx

                   xxxxxxx

                     xxxxx

                      xxx

                       x

              Note: Shape will be Rhombus.

class Star

{

public static void main(String[] args)

{

int i, j, k;

for(i=1;i<=5;i++)

{

for(j=i;j<5;j++)

{

System.out.print(" ");

}

for(k=1;k<(i\*2);k++)

{

System.out.print("\*");

}

System.out.println();

}

for(i=4;i>=1;i--)

{

for(j=5;j>i;j--)

{

System.out.print(" ");

}

for(k=1;k<(i\*2);k++)

{

System.out.print("\*");

}

System.out.println();

}

}

}

2. Write a java program to take the input from user and determine if it is a prime number or not.

 import java.util.Scanner;

class Primenumber

{

public static void main(String[] args)

{

int no, i, fact=1;

Scanner s=new Scanner(System.in);

System.out.println("Enter any number :");

no=s.nextInt();

if(no==1)

{

System.out.println("Smallest Prime number is 2");

}

for(i=2;i<no;i++)

{

if(no%i==0)

{

System.out.println("Not Prime");

break;

}

}

if(no==i)

{

System.out.println("Prime");

}

}

}

3. Write a java program to display the Fibonacci series till less than 200 using only 2 variables.

public class Fibonacci

{

public static void main(String[] args) {

int n = 200, t1 = 0, t2 = 1;

System.out.print("First " + n + " terms: ");

for (int i = 1; i <= n; ++i)

{

System.out.print(t1 + " + ");

int sum = t1 + t2;

t1 = t2;

t2 = sum;

}

}

}

5.Write Java program to check if a name is palindrome.

import java.util.Scanner;

class Palindrome

{

public static void main(String[] args)

{

int a,no,b,temp=0;

Scanner s=new Scanner(System.in);

System.out.println("Enter any num: ");

no=s.nextInt();

b=no;

while(no>0)

{

a=no%10;

no=no/10;

temp=temp\*10+a;

}

if(temp==b)

{

System.out.println("Palindrome");

}

else

{

System.out.println("not Palindrome");

}

}

}

6.Write Java program to check if a number is Armstrong number or not? (input 153 output true,  123 output false)

  import java.util.Scanner;

class Armstrong

{

public static void main(String[] args)

{

int arm=0,a,b,c,d,no;

Scanner s=new Scanner(System.in);

System.out.println("Enter any num :");

no=s.nextInt();

d=no;

while(no>0)

{

a=no%10;

no=no/10;

arm=arm+a\*a\*a;

}

if(arm==d)

{

System.out.println("Armstrong :");

}

else

{

System.out.println("not Armstrong");

}

}

}

7.How to find factorial of number in Java using iteration?

  import java.util.Scanner;

class Factorial

{

public static void main(String[] args)

{

int no, fect=1;

Scanner s=new Scanner(System.in);

System.out.println("Enter any number :");

no=s.nextInt();

for(int i=1; i<=no; i++)

{

fect=fect\*i;

}

System.out.println("Factorial is :" +fect);

}

}

8.Write a Java code to take a character as a input from user and determine if it is a vowel or a consonant using conditional construct.

class Char

{

void findVowelOrNot(char ch)

{

if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')

{

System.out.println("Entered character "+ch+" is  Vowel");

}

else if((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z'))

System.out.println("Entered character "+ch+" is Consonent");

      else

System.out.println("Not an alphabet");

}

public static void main(String[ ] arg)

{

Char c=new Char();

Scanner sc=new Scanner(System.in);

System.out.println("Enter a character : ");

char in=sc.next( ).charAt(0);

c.findVowelOrNot(in);

}

}

9. Write a switch case java code to create calculator with + - / \* functionalities only.

import java.util.Scanner;

public class Calculator {

public static void main(String[] args) {

Scanner reader = new Scanner(System.in);

System.out.print("Enter two numbers: ");

// nextDouble() reads the next double from the keyboard

double first = reader.nextDouble();

double second = reader.nextDouble();

System.out.print("Enter an operator (+, -, \*, /): ");

char operator = reader.next().charAt(0);

double result;

switch(operator)

{

case '+':

result = first + second;

break;

case '-':

result = first - second;

break;

case '\*':

result = first \* second;

break;

case '/':

result = first / second;

break;

// operator doesn't match any case constant (+, -, \*, /)

default:

System.out.printf("Error! operator is not correct");

return;

}

System.out.printf("%.1f %c %.1f = %.1f", first, operator, second, result);

}

}

10. Write a java code to copy one array into another.

public class Test

{

    public static void main(String[] args)

    {

        int a[] = {1, 8, 3};

        // Create an array b[] of same size as a[]

        int b[] = new int[a.length];

        // Copy elements of a[] to b[]

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

            b[i] = a[i];

        // Change b[] to verify that b[] is different

        // from a[]

        b[0]++;

        System.out.println("Contents of a[] ");

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

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

        System.out.println("\n\nContents of b[] ");

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

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

    }

}

11. Write a java code to compare the length of two arrays and display the longer array.

import java.util.Scanner;

public class JavaProgram

{

public static void main(String args[])

{

int size1, size2, size, i, j, k;

int arr1[] = new int[50];

int arr2[] = new int[50];

int merge[] = new int[100];

Scanner scan = new Scanner(System.in);

System.out.print("Enter Array 1 Size : ");

size1 = scan.nextInt();

System.out.print("Enter Array 1 Elements : ");

for(i=0; i<size1; i++)

{

arr1[i] = scan.nextInt();

}

System.out.print("Enter Array 2 Size : ");

size2 = scan.nextInt();

System.out.print("Enter Array 2 Elements : ");

for(i=0; i<size2; i++)

{

arr2[i] = scan.nextInt();

}

if(arr1.length>arr2.length)

for(i=0; i<size1; i++)

System.out.println(arr1[i]);

else

for(i=0; i<size1; i++)

System.out.println(arr2[i]);

}

}

12. Write a java code to display a reverse String array.

import java.io.\*;

import java.util.Collections;

import java.util.List;

import java.util.Arrays;

public class ReverseStringArrayExample {

        public static void main(String args[]){

                //String array

                String[] strDays = new String[]{"Sunday", "Monday", "Tuesday", "Wednesday"};

                /\*

                 \* There are basically two methods, one is to use temporary array and

                 \* manually loop through the elements of an Array and swap them or to use

                 \* Arrays and Collections classes.

                 \*

                 \* This example uses the second approach i.e. without temp variable.

                 \*

                 \*/

                //first create a list from String array

                List<String> list = Arrays.asList(strDays);

                //next, reverse the list using Collections.reverse method

                Collections.reverse(list);

                //next, convert the list back to String array

                strDays = (String[]) list.toArray();

                System.out.println("String array reversed");

                //print the reversed String array

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

                        System.out.println(strDays[i]);

                }

        }

}

13.   Write the difference between checked and unchecked exception with example code.

Checked exceptions are checked at compile-time. It means if a method is throwing a checked exception then it should handle the exception using [try-catch block](https://beginnersbook.com/2013/04/try-catch-in-java/) or it should declare the exception using [throws keyword](https://beginnersbook.com/2013/04/java-throws/), otherwise the program will give a compilation error.

import java.io.\*;

class Example {

public static void main(String args[])

{

FileInputStream fis = null;

try{

fis = new FileInputStream("B:/myfile.txt");

}catch(FileNotFoundException fnfe){

System.out.println("The specified file is not " +

"present at the given path");

}

int k;

try{

while(( k = fis.read() ) != -1)

{

System.out.print((char)k);

}

fis.close();

}catch(IOException ioe){

System.out.println("I/O error occurred: "+ioe);

}

}

}

Unchecked exceptions are not checked at compile time. It means if your program is throwing an unchecked exception and even if we didn’t handle/declare that exception, the program won’t give a compilation error. Most of the times these exception occurs due to the bad data provided by user during the user-program interaction.

class Example {

public static void main(String args[]) {

try{

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

System.out.println(arr[7]);

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("The specified index does not exist " +

"in array. Please correct the error.");

}

}

}

14.   Write the difference between throw and throws with example code

1. **Throws clause** is used to declare an exception, which means it works similar to the try-catch block. On the other hand **throw** keyword is used to throw an exception explicitly.

2. If we see syntax wise than **throw** is followed by an instance of Exception class and **throws** is followed by exception class names.  
For example:

throw new ArithmeticException("Arithmetic Exception");

and

throws ArithmeticException;

3. Throw keyword is used in the method body to throw an exception, while throws is used in method signature to declare the exceptions that can occur in the statements present in the method.

For example:  
**Throw:**

...

void myMethod() {

try {

//throwing arithmetic exception using throw

throw new ArithmeticException("Something went wrong!!");

}

catch (Exception exp) {

System.out.println("Error: "+exp.getMessage());

}

}

...

Throws:

...

//Declaring arithmetic exception using throws

void sample() throws ArithmeticException{

//Statements

}

...

4. We can throw one exception at a time but we can handle multiple exceptions by declaring them using throws keyword.  
For example:  
**Throw:**

void myMethod() {

//Throwing single exception using throw

throw new ArithmeticException("An integer should not be divided by zero!!");

}

..

Throws:

//Declaring multiple exceptions using throws

void myMethod() throws ArithmeticException, NullPointerException{

//Statements where exception might occur

}

Example:

public class Example1{

void checkAge(int age){

if(age<18)

throw new ArithmeticException("Not Eligible for voting");

else

System.out.println("Eligible for voting");

}

public static void main(String args[]){

Example1 obj = new Example1();

obj.checkAge(13);

System.out.println("End Of Program");

}

}

public class Example1{

int division(int a, int b) throws ArithmeticException{

int t = a/b;

return t;

}

public static void main(String args[]){

Example1 obj = new Example1();

try{

System.out.println(obj.division(15,0));

}

catch(ArithmeticException e){

System.out.println("You shouldn't divide number by zero");

}

}

}

15.   Write a note or nested try…catch block with example code

class Excep6{

 public static void main(String args[]){

  try{

    try{

     System.out.println("going to divide");

     int b =39/0;

    }catch(ArithmeticException e){System.out.println(e);}

    try{

    int a[]=new int[5];

    a[5]=4;

    }catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}

    System.out.println("other statement);

  }catch(Exception e){System.out.println("handeled");}

  System.out.println("normal flow..");

 }

}

16.   Write a note on MultiThreading and MultiTasking

1) In multitasking, several programs are executed concurrently e.g. Java compiler and a Java IDE like [Netbeans](http://java67.blogspot.sg/2013/02/how-to-connect-mysql-database-from-java.html) or [Eclipse](http://java67.blogspot.com/2014/04/how-to-make-executable-jar-file-in-Java-Eclipse.html), while in multi-threading multiple threads execute either same or different part of program multiple times at the same time.

2) Multi-threading is more granular than multi-tasking. In multi-tasking,  CPU switches between multiple programs to complete their execution in real time, while in multi-threading CPU switches between multiple threads of the same program. Remember, switching between multiple processes has more context switching cost than switching between multiple threads of the same program.  
  
3) Process are heavyweight as compared to threads, they require their own address space, which means multi-tasking is heavy compared to multithreading. Inter-process communication is expensive and limited and context switching from one process to another is expensive and limited

17.   Write a short note on Deque and give example code.

The java.util.Deque interface is a subtype of the [java.util.Queue](https://www.geeksforgeeks.org/queue-interface-java/) interface. The Deque is related to the double-ended queue that supports addition or removal of elements from either end of the data structure, it can be used as a [queue (first-in-first-out/FIFO)](https://www.geeksforgeeks.org/queue/) or as a [stack (last-in-first-out/LIFO)](https://www.geeksforgeeks.org/stack/).

import java.util.\*;

public class DequeExample

{

    public static void main(String[] args)

    {

        Deque deque = new LinkedList<>();

        // We can add elements to the queue in various ways

        deque.add("Element 1 (Tail)"); // add to tail

        deque.addFirst("Element 2 (Head)");

        deque.addLast("Element 3 (Tail)");

        deque.push("Element 4 (Head)"); //add to head

        deque.offer("Element 5 (Tail)");

        deque.offerFirst("Element 6 (Head)");

        deque.offerLast("Element 7 (Tail)");

        System.out.println(deque + "\n");

        // Iterate through the queue elements.

        System.out.println("Standard Iterator");

        Iterator iterator = deque.iterator();

        while (iterator.hasNext())

            System.out.println("\t" + iterator.next());

        // Reverse order iterator

        Iterator reverse = deque.descendingIterator();

        System.out.println("Reverse Iterator");

        while (reverse.hasNext())

            System.out.println("\t" + reverse.next());

        // Peek returns the head, without deleting

        // it from the deque

        System.out.println("Peek " + deque.peek());

        System.out.println("After peek: " + deque);

        // Pop returns the head, and removes it from

        // the deque

        System.out.println("Pop " + deque.pop());

        System.out.println("After pop: " + deque);

        // We can check if a specific element exists

        // in the deque

        System.out.println("Contains element 3: " +

                        deque.contains("Element 3 (Tail)"));

        // We can remove the first / last element.

        deque.removeFirst();

        deque.removeLast();

        System.out.println("Deque after removing " +

                            "first and last: " + deque);

    }

}

18.   Write a short note on Generics an all types of Parameters used in Generics with example code.

Generics was added in Java 5 to provide compile-time type checking and removing risk of ClassCastException that was common while working with collection classes. The whole collection framework was re-written to use generics for type-safety. Let’s see how generics help us using collection classes safely.

List list = new ArrayList();

list.add("abc");

list.add(new Integer(5)); //OK

for(Object obj : list){

//type casting leading to ClassCastException at runtime

String str=(String) obj;

}

19.   Write a short note on Map Interface.

A map contains values on the basis of key i.e. key and value pair. Each key and value pair is known as an entry. Map contains only unique keys.

Map is useful if we have to search, update or delete elements on the basis of key.

import java.util.\*;

class MapInterfaceExample{

 public static void main(String args[]){

  Map<Integer,String> map=new HashMap<Integer,String>();

  map.put(100,"Amit");

  map.put(101,"Vijay");

  map.put(102,"Rahul");

  for(Map.Entry m:map.entrySet()){

   System.out.println(m.getKey()+" "+m.getValue());

  }

 }

}

1. Write the difference between LinkedList and ArrayList.

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| 1) ArrayList internally uses dynamic array to store the elements. | LinkedList internally uses doubly linked list to store the elements. |
| 2) Manipulation with ArrayList is slow because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is faster than ArrayList because it uses doubly linked list so no bit shifting is required in memory. |
| 3) ArrayList class can act as a list only because it implements List only. | LinkedList class can act as a list and queue both because it implements List and Deque interfaces. |
| 4) ArrayList is better for storing and accessing data. | LinkedList is better for manipulating data. |

1. Write a note on Dynamic array in java.

The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed.

Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that we must know in advance how many elements an array will hold.

Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

1. What is the purpose of the System class?

The java.lang.System class contains several useful utilities for mostly used operation. Systemclass cannot be instantiated.

Facilities provided by System:

* standard output
* error output streams
* standard input and access to externally defined properties and environment variables.
* A utility for quickly copying particular portion of an array.
* used to loading files and libraries.

Standard Fields of System class are:

* static PrintStream err -- "standard" error output stream.
* static InputStream in -- "standard" input stream.
* static PrintStream out -- "standard" output stream

1. Which is the abstract parent class of FileWriter ?
   * 1. OutputStreamWriter
2. Which class is used to read streams of characters from a file?

[FileReader](https://docs.oracle.com/javase/8/docs/api/java/io/FileReader.html) and [FileWriter](https://docs.oracle.com/javase/8/docs/api/java/io/FileWriter.html).

1. Which class is used to read streams of raw bytes from a file?

FileInputStream

1. What are the differences between FileInputStream/FileOutputStream and RandomAccessFile

The File class encapsulates the files and directories of the local file system.  
  
- The java.io.RandomAccessFile class implements a random access file.  
  
- Random access file offers a seek feature that can go directly to a particular position.  
  
- Unlike the input and output stream classes in java.io, RandomAccessFile is used for both reading and writing files.  
  
- RandomAccessFile does not inherit from InputStream or OutputStream. It implements the DataInput and DataOutput interfaces.

1. Write a note on Channels and Buffer with example.
2. Java NIO Buffers are used when interacting with NIO Channels. Data is read from channels into buffers, and written from buffers into channels.

A buffer is essentially a block of memory into which we can write data, which we can then later read again. This memory block is wrapped in a NIO Buffer object, which provides a set of methods that makes it easier to work with the memory block.

28.   What is the difference between System.out ,System.err and System.in?

## System.in

System.in is an [**InputStream**](http://tutorials.jenkov.com/java-io/inputstream.html) which is typically connected to keyboard input of console programs. System.in is not used as often since data is commonly passed to a command line Java application via command line arguments, or configuration files. In applications with GUI the input to the application is given via the GUI. This is a separate input mechanism from Java IO.

**System.out**

System.out is a [**PrintStream**](http://tutorials.jenkov.com/java-io/printstream.html). System.out normally outputs the data you write to it to the console. This is often used from console-only programs like command line tools. This is also often used to print debug statements of from a program (though it may arguably not be the best way to get debug info out of a program).

**System.err**

System.err is a [**PrintStream**](http://tutorials.jenkov.com/java-io/printstream.html). System.err works like System.out except it is normally only used to output error texts. Some programs (like Eclipse) will show the output to System.err in red text, to make it more obvious that it is error text.

1. What is the purpose of the System class?

Same as above

1. Which is the abstract parent class of FileWriter ?

Same as above

1. Which class is used to read streams of characters from a file?

Same as above

1. Which class is used to read streams of raw bytes from a file?

Same as above

1. What are the differences between FileInputStream/FileOutputStream and RandomAccessFile

Same as above

1. Write a note on Channels and Buffer with example

Same as above

1. Write a note on PreparedStatement and ResultSetMetaData interfaces with code snippets.

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

String sql="insert into emp values(?,?,?)";

Example:

import java.sql.\*;

class InsertPrepared{

public static void main(String args[]){

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

PreparedStatement stmt=con.prepareStatement("insert into Emp values(?,?)");

stmt.setInt(1,101);//1 specifies the first parameter in the query

stmt.setString(2,"Ratan");

int i=stmt.executeUpdate();

System.out.println(i+" records inserted");

con.close();

}catch(Exception e){ System.out.println(e);}

}

}

ResultSet: The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.

By default, ResultSet object can be moved forward only and it is not updatable.

But we can make this object to move forward and backward direction by passing either TYPE\_SCROLL\_INSENSITIVE or TYPE\_SCROLL\_SENSITIVE in createStatement(int,int) method as well as we can make this object as updatable by:

Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,

                     ResultSet.CONCUR\_UPDATABLE);

import java.sql.\*;

class FetchRecord{

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

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

Statement stmt=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_UPDATABLE);

ResultSet rs=stmt.executeQuery("select \* from emp765");

//getting the record of 3rd row

rs.absolute(3);

System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

con.close();

}}

1. Write a note on DDL, DML, DQL, DDL with code snippets.

**Data Definition Language (DDL)**  
The commands of SQL that are used to create database objects, alter the structure of the database objects and delete database objects from database are collectively called as DDL. Examples include Create, Alter , Drop, Truncate, Rename and Comment Commands.  
  
**Create**Create command is used to create database and its Objects like tables, index, stored procedure, views , triggers, functions and etc.

**Alter**Alter command is used to create database and its Objects.

**Drop**Drop command is used to delete objects from database.  
  
**Truncate**Trunctae Table command is used to remove all records from a table, including all spaces allocated for records are removed.  
  
**Rename**It is used to rename the objects.  
  
**Comment**// -> Single line Comments, /\* --Multi Line Comments-- \*/ used to comment the sql statements.   
  
**Data Manipulation Language (DML)**The commands of SQL that are used to insert data into the database, modify the data of the database and to delete data from the database are collectively called as DML. Examples include Insert, Update and Delete.  
  
**Insert**To insert date into a table.  
  
**Update**To update the existing data in a table.  
  
**Delete**delete all records from a table.   
  
**Data Query Language (DQL)**  
The commands of SQL that are used to retrieve data from the database are collectively called as DQL. So all Select statements comes under DQL.  
  
**Select**To retreive data from the database table.   
  
**Data Control Language (DCL)**  
The commands of SQL that are used to control the access to data stored in the database are collectively called as DCL and examples include Grant and Revoke.  
  
**Grant**All users access previleges to database.  
  
**Revoke**  
Withdraw users access previleges given by using the Grant command

1. Write a note on HTML , CSS and Javascript.

HTML is the standard markup language for creating Web pages.

* HTML stands for Hyper Text Markup Language
* HTML describes the structure of Web pages using markup
* HTML elements are the building blocks of HTML pages
* HTML elements are represented by tags
* HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
* Browsers do not display the HTML tags, but use them to render the content of the page

**CSS** stands for **C**ascading **S**tyle **S**heets.

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

* **Inline** - by using the style attribute in HTML elements
* **Internal** - by using a <style> element in the <head> section
* **External** - by using an external CSS file

JavaScript is a very powerful **client-side scripting language**. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, we can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and[Mobile](https://www.guru99.com/mobile-testing.html)application development. The language was initially called LiveScript and was later renamed JavaScript. There are many programmers who think that JavaScript and[Java](https://www.guru99.com/java-tutorial.html)are the same. In fact, **JavaScript and Java are very much unrelated. Java is a very complex programming language whereas JavaScript is only a scripting language**.

1. Write a code to fetch the data from H2 and put it in any collection object and display it.

public class Person {

private String name;

private String jobtitle;

private int frequentflyer;

public int getFrequentflyer() {

return frequentflyer;

}

public void setFrequentflyer(int frequentflyer) {

this.frequentflyer = frequentflyer;

}

public String getJobtitle() {

return jobtitle;

}

public void setJobtitle(String jobtitle) {

this.jobtitle = jobtitle;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

public class Main {

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

ArrayList<Person> personlist = new ArrayList<Person>();

//List<Person> personlist = new List<Person>();

try {

// Step 1: Load the JDBC driver. jdbc:mysql://localhost:3306/travel

Class.forName("org.h2.Driver");

// Step 2: Establish the connection to the database.

String url = " jdbc:h2:~/test";

Connection conn = DriverManager.getConnection(url, "sa", "admin");

Statement st = conn.createStatement();

ResultSet srs = st.executeQuery("SELECT \* FROM person");

while (srs.next()) {

Person person = new Person();

person.setName(srs.getString("name"));

person.setJobtitle(srs.getString("jobtitle"));

person.setFrequentflyer(srs.getInt("frequentflyer"));

personlist.add(person);

}

System.out.println(personlist.size());

System.out.println(personlist.get(1).getName());

System.out.println(personlist.get(2).getName());

System.out.println(personlist.get(3).getName());

System.out.println(personlist.get(4));

//System.out.println(namelist.);

} catch (Exception e) {

System.err.println("Got an exception! ");

System.err.println(e.getMessage());

}

}

}

1. Describe the different approaches of String processing.

Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings.

Creating Strings

The most direct way to create a string is to write −

String greeting = "Hello world!";

Whenever it encounters a string literal in your code, the compiler creates a String object with its value in this case, "Hello world!'.

As with any other object, you can create String objects by using the new keyword and a constructor. The String class has 11 constructors that allow you to provide the initial value of the string using different sources, such as an array of characters.

Example

public class StringDemo {

public static void main(String args[]) {

char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };

String helloString = new String(helloArray);

System.out.println( helloString );

}

}

1. What is the difference between System.out ,System.err and System.in?

Same As Above

1. What is the purpose of the System class?

Same As Above

1. Which is the abstract parent class of FileWriter ?

Same As Above

43.   Which class is used to read streams of characters from a file?

Same As Above

44.   Which class is used to read streams of raw bytes from a file?

Same As Above

45.   What are the differences between FileInputStream/FileOutputStream and RandomAccessFile

Same As Above

46.   Write a note on Channels and Buffer with example.

  Same As Above