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

a) 1

12

123

12345

**Ans**:

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

**Ans**:

class NumberPattern

{

public static void main(String[] args)

{

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

{

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

{

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

}

System.out.println();

}

}

}

c) x

xxx

xxxxx

xxxxxxx

xxxxx

xxx

x

Note: Shape will be Rhombus.

**Ans:**

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.**

**Ans:**

import java.util.Scanner;

class PrimeCheck

{

public static void main(String args[])

{

int temp;

boolean isPrime=true;

Scanner scan= new Scanner(System.in);

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

int num=scan.nextInt();

scan.close();

for(int i=2;i<=num/2;i++)

{

temp=num%i;

if(temp==0)

{

isPrime=false;

break;

}

}

if(isPrime)

System.out.println(num + " is a Prime Number");

else

System.out.println(num + " is not a Prime Number");

}

}

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

**Ans:**

public class One

{

public static long fib(int n) {

if (n == 0) {

return 0;

} else if (n == 1) {

return 1;

} else {

return fib(n-1) + fib(n-2);

}

}

public static void main(String[] args){

for(int i=0; i < 13; i++) {

System.out.println(i + ": " + fib(i));

}

}

}

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

**Ans**:

import java.util.\*;

class Palindrome

{

public static void main(String args[])

{

String original, reverse = ""; // Objects of String class

Scanner in = new Scanner(System.in);

System.out.println("Enter a string to check if it is a palindrome");

original = in.nextLine();

int length = original.length();

for ( int i = length - 1; i >= 0; i-- )

reverse = reverse + original.charAt(i);

if (original.equals(reverse))

System.out.println("Entered string is a palindrome.");

else

System.out.println("Entered string is not a palindrome.");

}

}

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

**Ans:**

import java.util.Scanner;

class ArmstrongNumber

{

public static void main(String args[])

{

int n, sum = 0, temp, remainder, digits = 0;

Scanner in = new Scanner(System.in);

System.out.println("Input a number to check if it is an Armstrong number");

n = in.nextInt();

temp = n;

while (temp != 0) {

digits++;

temp = temp/10;

}

temp = n;

while (temp != 0) {

remainder = temp%10;

sum = sum + power(remainder, digits);

temp = temp/10;

}

if (n == sum)

System.out.println(n + " is an Armstrong number.");

else

System.out.println(n + " is not an Armstrong number.");

}

static int power(int n, int r) {

int c, p = 1;

for (c = 1; c <= r; c++)

p = p\*n;

return p;

}

}

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

**Ans:**

import java.util.Scanner;

class Factorial

{

public static void main(String args[])

{

int n, c, fact = 1;

System.out.println("Enter an integer to calculate it's factorial");

Scanner in = new Scanner(System.in);

n = in.nextInt();

if ( n < 0 )

System.out.println("Number should be non-negative.");

else

{

for ( c = 1 ; c <= n ; c++ )

fact = fact\*c;

System.out.println("Factorial of "+n+" is = "+fact);

}

}

}

**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.**

**Ans:**

class Char

{

public static void main(String[ ] arg)

{

int i=0;

Scanner sc=new Scanner(System.in);

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

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

switch(ch)

{

case 'a' :

case 'e' :

case 'i' :

case 'o' :

case 'u' :

case 'A' :

case 'E' :

case 'I' :

case 'O' :

case 'U' :i++;

}

if(i==1)

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");

}

}

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

**Ans:**

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: ");

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;

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.**

**Ans:**

public class A {

public static void main(String args[]) {

int a[] = { 1, 2, 3, 4, 5, 6 };

int b[] = new int[a.length];

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

a[i] = b[i];

}

}

}

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

**Ans:**

class One

{

public static void main (String[] args)

{

int arr1[] = {1, 2, 3, 4, 5};

int arr2[] = {1, 2, 3 ,4};

if (arr1.length > arr2.length)

{ // Same as arr1.equals(arr2)

for (int i = 0; i < arr1.length; i++) {

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

}

}

else

for (int i = 0; i < arr2.length; i++) {

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

}

}

}

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

**Ans:**

import java.util.Collections;

import java.util.List;

import java.util.Arrays;

public class ReverseStringArrayExample {

public static void main(String args[]){

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

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

Collections.reverse(list);

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

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.**

**Ans:**

1) Checked: are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

For example, consider the following Java program that opens file at locatiobn “C:\test\a.txt” and prints first three lines of it. The program doesn’t compile, because the function main() uses FileReader() and FileReader() throws a checked exception FileNotFoundException. It also uses readLine() and close() methods, and these methods also throw checked exception IOException

import java.io.\*;

class Main {

public static void main(String[] args) {

FileReader file = new FileReader("C:\\test\\a.txt");

BufferedReader fileInput = new BufferedReader(file);

// Print first 3 lines of file "C:\test\a.txt"

for (int counter = 0; counter < 3; counter++)

System.out.println(fileInput.readLine());

fileInput.close();

}

}

2) Unchecked are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.

In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked

Consider the following Java program. It compiles fine, but it throws ArithmeticException when run. The compiler allows it to compile, because ArithmeticException is an unchecked exception.

class Main {

public static void main(String args[]) {

int x = 0;

int y = 10;

int z = y/x;

}

}

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

**Ans:**

|  |  |  |
| --- | --- | --- |
| No. | throw | throws |
| 1) | Java throw keyword is used to explicitly throw an exception. | Java throws keyword is used to declare an exception. |
| 2) | Checked exception cannot be propagated using throw only. | Checked exception can be propagated with throws. |
| 3) | Throw is followed by an instance. | Throws is followed by class. |
| 4) | Throw is used within the method. | Throws is used with the method signature. |
| 5) | You cannot throw multiple exceptions. | You can declare multiple exceptions e.g. public void method()throws IOException,SQLException. |
| 6) | Example:  void m(){  throw new ArithmeticException("sorry");  } | Example:  void m()throws ArithmeticException{  //method code  } |

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

**Ans:**

You associate exception handlers with a try block by providing one or more catch blocks directly after the try block. No code can be between the end of the try block and the beginning of the first catch block.

try {

} catch (ExceptionType name) {

} catch (ExceptionType name) {

}

Each catch block is an exception handler that handles the type of exception indicated by its argument. The argument type, ExceptionType, declares the type of exception that the handler can handle and must be the name of a class that inherits from the Throwable class. The handler can refer to the exception with name.

The catch block contains code that is executed if and when the exception handler is invoked. The runtime system invokes the exception handler when the handler is the first one in the call stack whose ExceptionType matches the type of the exception thrown. The system considers it a match if the thrown object can legally be assigned to the exception handler's argument.

Example:

public class Testtrycatch2{

public static void main(String args[]){

try{

int data=50/0;

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

System.out.println("rest of the code...");

}

}

**16. Write a note on MultiThreading and MultiTasking**

**Ans:**

Multitasking

The ability to run several programs simultaneously, potentially by utilizing several processors, but predominantly, by time-sharing their resource requirements.

An example is right on your desktop, where you may have a web browser, e-mail client, audio player, word processor, spreadsheet and who knows what else on the air at the same time. They can dance in and out of having the processor to themselves many times per second, because neither of them needs all of it for very long at a time.

Multithreading

The ability to run several functions of a single program simultaneously, predominantly by utilizing several processors, but potentially, by time-sharing their resource requirements.

An example would be a web server, where the responses to all the incoming requests need much of the same program logic and state, but different handles on a few things (network socket, id of caller, whatever else). Sharing the greater bunch of the data pertaining to the program, but having dedicated copies of a small amount of private things, lets threads be spawned and destroyed very quickly, and permits an increase in available processing power to increase the number of requests answered without requiring an additional copies of the server program to be running.

These are largely two sides of the same coin, the difference in vocabulary is mainly down to entire programs (processes) being a larger unit of stored state, with a correspondingly higher workload required to shift it around between processors and/or memory.

Both of them require some O/S scheduling mechanism to keep track of which process/thread goes next, goes where and goes when, but the differences in the cost of manipulating processes and threads means that the best policy for one isn't necessarily good for the other. Hence, they get different names, and can be discussed as different things, within a context implied by the word chosen presently.

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

**Ans:**

Deque is a linear collection that supports element insertion and removal at both ends. The name deque is short for "double ended queue" and is usually pronounced "deck". Most Deque implementations place no fixed limits on the number of elements they may contain, but this interface supports capacity-restricted deques as well as those with no fixed size limit.

This interface defines methods to access the elements at both ends of the deque. Methods are provided to insert, remove, and examine the element. Each of these methods exists in two forms: one throws an exception if the operation fails, the other returns a special value (either null or false, depending on the operation). The latter form of the insert operation is designed specifically for use with capacity-restricted Deque implementations; in most implementations, insert operations cannot fail.

Example:

import java.util.\*;

public class ArrayDequeExample {

public static void main(String[] args) {

//Creating Deque and adding elements

Deque<String> deque = new ArrayDeque<String>();

deque.add("Ravi");

deque.add("Vijay");

deque.add("Ajay");

//Traversing elements

for (String str : deque) {

System.out.println(str);

}

}

}

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

**Ans:**

Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces and methods. Much like the more familiar formal parameters used in method declarations, type parameters provide a way for you to re-use the same code with different inputs. The difference is that the inputs to formal parameters are values, while the inputs to type parameters are types.

Code that uses generics has many benefits over non-generic code:

Stronger type checks at compile time.

A Java compiler applies strong type checking to generic code and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find.

Elimination of casts.

The following code snippet without generics requires casting:

List list = new ArrayList();

list.add("hello");

String s = (String) list.get(0);

When re-written to use generics, the code does not require casting:

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

list.add("hello");

String s = list.get(0); // no cast

**19. Write a short note on Map Interface.**

**Ans:**

A Map is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value. It models the mathematical function abstraction. The Map interface includes methods for basic operations (such as put, get, remove, containsKey, containsValue, size, and empty), bulk operations (such as putAll and clear), and collection views (such as keySet, entrySet, and values).

The Java platform contains three general-purpose Map implementations: HashMap, TreeMap, and LinkedHashMap. Their behavior and performance are precisely analogous to HashSet, TreeSet, and LinkedHashSet, as described in The Set Interface section.

**20. Write the difference between LinkedList and ArrayList.**

**Ans:**

1. Implementation : ArrayList is the resizable array implementation of list interface , while LinkedList is the Doubly-linked list implementation of the list interface.

2. Performance : Performance of ArrayList and LinkedList depends on the type of operation

a. get(int index) or search operation : ArrayList get(int index) operation runs in constant time i.e O(1) while LinkedList get(int index) operation run time is O(n) .

The reason behind ArrayList being faster than LinkedList is that ArrayList uses index based system for its elements as it internally uses array data structure , on the other hand ,

LinkedList does not provide index based access for its elements as it iterates either from the beginning or end (whichever is closer) to retrieve the node at the specified element index.

b. insert() or add(Object) operation : Insertions in LinkedList are generally fast as compare to ArrayList.

In LinkedList adding or insertion is O(1) operation . While in ArrayList, if array is full i.e worst case, there is extra cost of resizing array and copying elements to the new array , which makes runtime of add operation in ArrayList O(n) , otherwise it is O(1) .

c. remove(int) operation : Remove operation in LinkedList is generally same as ArrayList i.e. O(n).

In LinkedList , there are two overloaded remove methods. one is remove() without any parameter which removes the head of the list and runs in constant time O(1) .

The other overloaded remove method in LinkedList is remove(int) or remove(Object) which removes the Object or int passed as parameter . This method traverses the LinkedList until it found the Object and unlink it from the original list . Hence this method run time is O(n).

While in ArrayList remove(int) method involves copying elements from old array to new updated array , hence its run time is O(n).

3. Reverse Iterator : LinkedList can be iterated in reverse direction using descendingIterator() while there is no descendingIterator() in ArrayList , so we need to write our own code to iterate over the ArrayList in reverse direction.

4. Initial Capacity : If the constructor is not overloaded , then ArrayList creates an empty list of initial capacity 10 , while LinkedList only constructs the empty list without any initial capacity.

5. Memory Overhead : Memory overhead in LinkedList is more as compared to ArrayList as node in LinkedList needs to maintain the addresses of next and previous node. While in ArrayList each index only holds the actual object(data).

**21. Write a note on Dynamic array in java.**

**Ans:**

In Java, "normal" arrays are fixed-size. You have to give them a size and can't expand them or contract them. To change the size, you have to make a new array and copy the data you want - which is inefficient and a pain for you. Fortunately, there are all kinds of built-in classes that implement common data structures and other useful tools too.

Example : ArrayList

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

import java.util.\*;

public class ArrayListDemo {

public static void main(String args[]) {

// create an array list

ArrayList al = new ArrayList();

System.out.println("Initial size of al: " + al.size());

// add elements to the array list

al.add("C");

al.add("A");

al.add("E");

al.add("B");

al.add("D");

al.add("F");

al.add(1, "A2");

System.out.println("Size of al after additions: " + al.size());

// display the array list

System.out.println("Contents of al: " + al);

// Remove elements from the array list

al.remove("F");

al.remove(2);

System.out.println("Size of al after deletions: " + al.size());

System.out.println("Contents of al: " + al);

}

}

**22. What is the purpose of the System class?**

**Ans:**

Facilities provided by System:

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

**23. Which is the abstract parent class of FileWriter ?**

**Ans:**

OutputStreamWriter.

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

**Ans:**

The Java FileReader class makes it possible to read the contents of a file as a stream of characters.

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

**Ans :**

FileInputStream reads raw bytes from a file. The read methods in this class return a byte of data read from a file.

**26. What are the differences between FileInputStream/FileOutputStream and RandomAccessFile**

RandomAccessFile treats the file as an array of bytes where it has the internal pointer. The fact that it treats it like a large array of bytes is what is unique about this class. FileInputStream however just reads the stream and returns the data. It is more suited to reading raw data like images etc. It does not treat the file as a large array, it just keeps tabs of where in the file it has read so far. With FileInputStream you would actually have to read the data and place it into an array to get the same style of access as RandomAccessFile.

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

**27. Write a note on Channels and Buffer with example.**

**Ans:**

Java NIO **Buffers** are used when interacting with NIO Channels. As you know, data is read from channels into buffers, and written from buffers into channels.

A buffer is essentially a block of memory into which you can write data, which you 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.

When you write data into a buffer, the buffer keeps track of how much data you have written. Once you need to read the data, you need to switch the buffer from writing mode into reading mode using the flip() method call. In reading mode the buffer lets you read all the data written into the buffer.

Once you have read all the data, you need to clear the buffer, to make it ready for writing again. You can do this in two ways: By calling clear() or by calling compact(). The clear() method clears the whole buffer. The compact() method only clears the data which you have already read. Any unread data is moved to the beginning of the buffer, and data will now be written into the buffer after the unread data.

Java NIO **Channels** are similar to streams with a few differences:

1. You can both read and write to a Channels. Streams are typically one-way (read or write).
2. Channels can be read and written asynchronously.
3. Channels always read to, or write from, a Buffer.

**Example**

package com.javapapers.java.nio;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.nio.channels.SeekableByteChannel;

import java.nio.charset.Charset;

import java.nio.file.FileSystems;

import java.nio.file.Files;

import java.nio.file.OpenOption;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.HashSet;

import java.util.Set;

public class JavaNIOReadWriteChannels {

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

byteChannelRead();

String string = "foo bar";

byte[] byteArray = string.getBytes();

ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray);

byteChannelWrite(byteBuffer);

fileChannelRead();

fileChannelWrite(byteBuffer);

}

public static void byteChannelRead() throws IOException {

Path filePath = FileSystems.getDefault().getPath(".", "temp.txt");

SeekableByteChannel byteChannel = Files.newByteChannel(filePath);

ByteBuffer byteBuffer = ByteBuffer.allocate(10);

Charset charset = Charset.forName("US-ASCII");

while (byteChannel.read(byteBuffer) > 0) {

byteBuffer.rewind();

System.out.print(charset.decode(byteBuffer));

byteBuffer.flip();

}

}

public static void byteChannelWrite(ByteBuffer byteBuffer)

throws IOException {

Set options = new HashSet();

options.add(StandardOpenOption.CREATE);

options.add(StandardOpenOption.APPEND);

Path file = Paths.get("./byByteChannel.txt");

SeekableByteChannel byteChannel = Files.newByteChannel(file, options);

byteChannel.write(byteBuffer);

}

public static void fileChannelRead() throws IOException {

RandomAccessFile randomAccessFile = new RandomAccessFile("./temp.txt",

"rw");

FileChannel fileChannel = randomAccessFile.getChannel();

ByteBuffer byteBuffer = ByteBuffer.allocate(512);

Charset charset = Charset.forName("US-ASCII");

while (fileChannel.read(byteBuffer) > 0) {

byteBuffer.rewind();

System.out.print(charset.decode(byteBuffer));

byteBuffer.flip();

}

fileChannel.close();

randomAccessFile.close();

}

public static void fileChannelWrite(ByteBuffer byteBuffer)

throws IOException {

Set options = new HashSet();

options.add(StandardOpenOption.CREATE);

options.add(StandardOpenOption.APPEND);

Path path = Paths.get("./byFileChannel.txt");

FileChannel fileChannel = FileChannel.open(path, options);

fileChannel.write(byteBuffer);

fileChannel.close();

}

}

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

**Ans:**

System.out is "standard output" (stdout) and System.err is "error output" (stderr). Along with System.in (stdin), these are the three standard I/O streams in the Unix model.

**29. What is the purpose of the System class?**

**Ans:**

Facilities provided by System:

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

**30. Which is the abstract parent class of FileWriter ?**

**Ans:**

OutputStreamWriter.

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

**Ans:**

The Java FileReader class makes it possible to read the contents of a file as a stream of characters.

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

**Ans :**

FileInputStream reads raw bytes from a file. The read methods in this class return a byte of data read from a file.

**33. What are the differences between FileInputStream/FileOutputStream and RandomAccessFile**

**Ans:**

RandomAccessFile treats the file as an array of bytes where it has the internal pointer. The fact that it treats it like a large array of bytes is what is unique about this class. FileInputStream however just reads the stream and returns the data. It is more suited to reading raw data like images etc. It does not treat the file as a large array, it just keeps tabs of where in the file it has read so far. With FileInputStream you would actually have to read the data and place it into an array to get the same style of access as RandomAccessFile.

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

**34. Write a note on Channels and Buffer with example.**

**Ans:**

Java NIO **Buffers** are used when interacting with NIO Channels. As you know, data is read from channels into buffers, and written from buffers into channels.

A buffer is essentially a block of memory into which you can write data, which you 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.

When you write data into a buffer, the buffer keeps track of how much data you have written. Once you need to read the data, you need to switch the buffer from writing mode into reading mode using the flip() method call. In reading mode the buffer lets you read all the data written into the buffer.

Once you have read all the data, you need to clear the buffer, to make it ready for writing again. You can do this in two ways: By calling clear() or by calling compact(). The clear() method clears the whole buffer. The compact() method only clears the data which you have already read. Any unread data is moved to the beginning of the buffer, and data will now be written into the buffer after the unread data.

Java NIO **Channels** are similar to streams with a few differences:

1. You can both read and write to a Channels. Streams are typically one-way (read or write).
2. Channels can be read and written asynchronously.
3. Channels always read to, or write from, a Buffer.

**Example**

package com.javapapers.java.nio;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.nio.channels.SeekableByteChannel;

import java.nio.charset.Charset;

import java.nio.file.FileSystems;

import java.nio.file.Files;

import java.nio.file.OpenOption;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.HashSet;

import java.util.Set;

public class JavaNIOReadWriteChannels {

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

byteChannelRead();

String string = "foo bar";

byte[] byteArray = string.getBytes();

ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray);

byteChannelWrite(byteBuffer);

fileChannelRead();

fileChannelWrite(byteBuffer);

}

public static void byteChannelRead() throws IOException {

Path filePath = FileSystems.getDefault().getPath(".", "temp.txt");

SeekableByteChannel byteChannel = Files.newByteChannel(filePath);

ByteBuffer byteBuffer = ByteBuffer.allocate(10);

Charset charset = Charset.forName("US-ASCII");

while (byteChannel.read(byteBuffer) > 0) {

byteBuffer.rewind();

System.out.print(charset.decode(byteBuffer));

byteBuffer.flip();

}

}

public static void byteChannelWrite(ByteBuffer byteBuffer)

throws IOException {

Set options = new HashSet();

options.add(StandardOpenOption.CREATE);

options.add(StandardOpenOption.APPEND);

Path file = Paths.get("./byByteChannel.txt");

SeekableByteChannel byteChannel = Files.newByteChannel(file, options);

byteChannel.write(byteBuffer);

}

public static void fileChannelRead() throws IOException {

RandomAccessFile randomAccessFile = new RandomAccessFile("./temp.txt",

"rw");

FileChannel fileChannel = randomAccessFile.getChannel();

ByteBuffer byteBuffer = ByteBuffer.allocate(512);

Charset charset = Charset.forName("US-ASCII");

while (fileChannel.read(byteBuffer) > 0) {

byteBuffer.rewind();

System.out.print(charset.decode(byteBuffer));

byteBuffer.flip();

}

fileChannel.close();

randomAccessFile.close();

}

public static void fileChannelWrite(ByteBuffer byteBuffer)

throws IOException {

Set options = new HashSet();

options.add(StandardOpenOption.CREATE);

options.add(StandardOpenOption.APPEND);

Path path = Paths.get("./byFileChannel.txt");

FileChannel fileChannel = FileChannel.open(path, options);

fileChannel.write(byteBuffer);

fileChannel.close();

}

}

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

**Ans:**

**PreparedStatement**

The main feature of a PreparedStatement object is that, unlike a Statement object, it is given a SQL statement when it is created. The advantage to this is that in most cases, this SQL statement is sent to the DBMS right away, where it is compiled. As a result, the PreparedStatement object contains not just a SQL statement, but a SQL statement that has been precompiled. This means that when the PreparedStatement is executed, the DBMS can just run the PreparedStatement SQL statement without having to compile it first.

Although PreparedStatement objects can be used for SQL statements with no parameters, you probably use them most often for SQL statements that take parameters. The advantage of using SQL statements that take parameters is that you can use the same statement and supply it with different values each time you execute it.

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();

**ResultSetMetaData**

An object that can be used to get information about the types and properties of the columns in a ResultSet object. The following code fragment creates the ResultSet object rs, creates the ResultSetMetaData object rsmd, and uses rsmd to find out how many columns rs has and whether the first column in rs can be used in a WHERE clause.

ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM TABLE2");

ResultSetMetaData rsmd = rs.getMetaData();

int numberOfColumns = rsmd.getColumnCount();

boolean b = rsmd.isSearchable(1);

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

**Ans:**

* DDL (Data Definition Language)

Defining Database Structures

Data Definition Language, DDL, is the part of SQL that allows a database user to create and restructure database objects, such as the creation or the deletion of a table.

Some of the most fundamental DDL commands discussed during following hours include the following:

CREATE TABLE

Eg. CREATE TABLE employee

( id number(5),

name char(20),

dept char(10),

age number(2),

salary number(10),

location char(10)

);

ALTER TABLE

Eg: ALTER TABLE employee ADD experience number(3);

DROP TABLE

Eg: DROP TABLE employee;

* DML (Data Manipulation Language)

Manipulating Data

Data Manipulation Language, DML, is the part of SQL used to manipulate data within objects of a relational database.

There are three basic DML commands:

INSERT

UPDATE

DELETE

* DQL (Data Query Language)

Selecting Data

Though comprised of only one command, Data Query Language (DQL) is the most concentrated focus of SQL for modern relational database users. The base command is as follows:

SELECT

This command, accompanied by many options and clauses, is used to compose queries against a relational database. Queries, from simple to complex, from vague to specific, can be easily created.

**37. Write a note on HTML , CSS and Javascript.**

**Ans:**

The definition of HTML is HyperText Markup Language.

1. HyperText is the method by which you move around on the web — by clicking on special text called hyperlinks which bring you to the next page. The fact that it is hyper just means it is not linear — i.e. you can go to any place on the Internet whenever you want by clicking on links — there is no set order to do things in.
2. Markup is what HTML tags do to the text inside them. They mark it as a certain type of text (italicised text, for example).
3. HTML is a Language, as it has code-words and syntax like any other language.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

JavaScript, often abbreviated as JS, is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine.

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

**Ans:**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class H2jdbcReadDemo {

static final String JDBC\_DRIVER = "org.h2.Driver";

static final String DB\_URL = "jdbc:h2:~/test";

static final String USER = "sa";

static final String PASS = "";

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

Class.forName(JDBC\_DRIVER);

System.out.println("Connecting to database...");

conn = DriverManager.getConnection(DB\_URL,USER,PASS);

System.out.println("Connected database successfully...");

stmt = conn.createStatement();

String sql = "SELECT id, first, last, age FROM Registration";

ResultSet rs = stmt.executeQuery(sql);

while(rs.next()) {

int id = rs.getInt("id");

int age = rs.getInt("age");

String first = rs.getString("first");

String last = rs.getString("last");

System.out.print("ID: " + id);

System.out.print(", Age: " + age);

System.out.print(", First: " + first);

System.out.println(", Last: " + last);

}

rs.close();

} catch(SQLException se) {

se.printStackTrace();

} catch(Exception e) {

e.printStackTrace();

} finally {

try {

if(stmt!=null) stmt.close();

} catch(SQLException se2) {

}

try {

if(conn!=null) conn.close();

} catch(SQLException se) {

se.printStackTrace();

}

}

System.out.println("Goodbye!");

}

}

**39. Describe the different approaches of String processing.**

**Ans:**

1. Using String Class

The java.lang.String class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.

1. Using StringBuilder Class

Java StringBuilder class is used to create mutable (modifiable) string. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.

1. Using StringBuffer Class

Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

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

**Ans:**

System.out is "standard output" (stdout) and System.err is "error output" (stderr). Along with System.in (stdin), these are the three standard I/O streams in the Unix model.

**41. What is the purpose of the System class?**

**Ans:**

Facilities provided by System:

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

**42. Which is the abstract parent class of FileWriter ?**

**Ans:**

OutputStreamWriter.

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

**Ans:**

The Java FileReader class makes it possible to read the contents of a file as a stream of characters.

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

**Ans :**

FileInputStream reads raw bytes from a file. The read methods in this class return a byte of data read from a file.

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

**Ans:**

RandomAccessFile treats the file as an array of bytes where it has the internal pointer. The fact that it treats it like a large array of bytes is what is unique about this class. FileInputStream however just reads the stream and returns the data. It is more suited to reading raw data like images etc. It does not treat the file as a large array, it just keeps tabs of where in the file it has read so far. With FileInputStream you would actually have to read the data and place it into an array to get the same style of access as RandomAccessFile.

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

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

**Ans:**

Java NIO **Buffers** are used when interacting with NIO Channels. As you know, data is read from channels into buffers, and written from buffers into channels.

A buffer is essentially a block of memory into which you can write data, which you 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.

When you write data into a buffer, the buffer keeps track of how much data you have written. Once you need to read the data, you need to switch the buffer from writing mode into reading mode using the flip() method call. In reading mode the buffer lets you read all the data written into the buffer.

Once you have read all the data, you need to clear the buffer, to make it ready for writing again. You can do this in two ways: By calling clear() or by calling compact(). The clear() method clears the whole buffer. The compact() method only clears the data which you have already read. Any unread data is moved to the beginning of the buffer, and data will now be written into the buffer after the unread data.

Java NIO **Channels** are similar to streams with a few differences:

1. You can both read and write to a Channels. Streams are typically one-way (read or write).
2. Channels can be read and written asynchronously.
3. Channels always read to, or write from, a Buffer.

**Example**

package com.javapapers.java.nio;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.nio.channels.SeekableByteChannel;

import java.nio.charset.Charset;

import java.nio.file.FileSystems;

import java.nio.file.Files;

import java.nio.file.OpenOption;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.HashSet;

import java.util.Set;

public class JavaNIOReadWriteChannels {

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

byteChannelRead();

String string = "foo bar";

byte[] byteArray = string.getBytes();

ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray);

byteChannelWrite(byteBuffer);

fileChannelRead();

fileChannelWrite(byteBuffer);

}

public static void byteChannelRead() throws IOException {

Path filePath = FileSystems.getDefault().getPath(".", "temp.txt");

SeekableByteChannel byteChannel = Files.newByteChannel(filePath);

ByteBuffer byteBuffer = ByteBuffer.allocate(10);

Charset charset = Charset.forName("US-ASCII");

while (byteChannel.read(byteBuffer) > 0) {

byteBuffer.rewind();

System.out.print(charset.decode(byteBuffer));

byteBuffer.flip();

}

}

public static void byteChannelWrite(ByteBuffer byteBuffer)

throws IOException {

Set options = new HashSet();

options.add(StandardOpenOption.CREATE);

options.add(StandardOpenOption.APPEND);

Path file = Paths.get("./byByteChannel.txt");

SeekableByteChannel byteChannel = Files.newByteChannel(file, options);

byteChannel.write(byteBuffer);

}

public static void fileChannelRead() throws IOException {

RandomAccessFile randomAccessFile = new RandomAccessFile("./temp.txt",

"rw");

FileChannel fileChannel = randomAccessFile.getChannel();

ByteBuffer byteBuffer = ByteBuffer.allocate(512);

Charset charset = Charset.forName("US-ASCII");

while (fileChannel.read(byteBuffer) > 0) {

byteBuffer.rewind();

System.out.print(charset.decode(byteBuffer));

byteBuffer.flip();

}

fileChannel.close();

randomAccessFile.close();

}

public static void fileChannelWrite(ByteBuffer byteBuffer)

throws IOException {

Set options = new HashSet();

options.add(StandardOpenOption.CREATE);

options.add(StandardOpenOption.APPEND);

Path path = Paths.get("./byFileChannel.txt");

FileChannel fileChannel = FileChannel.open(path, options);

fileChannel.write(byteBuffer);

fileChannel.close();

}

}