Chapter -10

NIO

জাভা IO পদ্ধতিটি একটু ধীর গতির হওয়ায় New I/O (NIO) পদ্ধতির উদ্ভব।

NIO তে Channel এবং Buffer ব্যবহার করা হয়।

Stream এর Data Transfer একমুখী।

কিন্তু Chanel ব্যবহার করে একসাথে ডাটা রীড এবং রাইট করা যায়।

Steam ভিত্রিক I/O এর মৌলিক Data Transfer একক হচ্ছে Byte.

Channel এর মৌলিক Data Transfer একক হচ্ছে Buffer.

Buffer একটি Container.

Buffer:-

Buffer একটি নিদিষ্ট দৈর্ঘের Container. এটি মূলত একটি Object যাতে একটি মেমোরি ব্লক থাকে যেখান হতে Data Write এবং Read করা যায়।

Primitive Data Type এর Buffer Class আলাদা আছে,

* ByteBuffer
* ShortBuffer
* CharBuffer
* IntBuffer
* FloatBuffer
* DoubleBuffer

শুধু Boolean এর জন্য Buffer নেই।

এই সবগুলো ক্লাস Abstract.

এই সবগুলো Class এর সুপার ক্লাস হল Buffer.

বাফার এর গুরুত্বপূর্ন প্রোপ্রাটি হলঃ-

1. Capacity
2. Position
3. Limit
4. Mark

Buffer এর ডাটা ধারন ক্ষমতাকে বলা হয় Capacity.

Buffer এর Size Array এর মত Change পরিবর্তন করা যায় না।

CharBuffer charB=CharBuffer.allocate(10);

ByteBuffer এ একটি মেথড আছে allocateDirect() যার মাধ্যমে সরাসরি Buffer তৈরী করা হয়।

বাফার এর ইলেমেন্ট গুলো ইনিশিয়ালাইজ ‘0’ হয় এবং Position ‘0’ হয়। Limit ও Capacity সমান হয়।

প্রত্যেক Element এর একটি করে Index থাকে।

Buffer এ Position() এবং Position(int newPosition) এই মেথড ব্যবহার করে Position get এবং Set করা যায়।

অনেকসময় Buffer এর একটি Position ‘mark’ করা বা মনে রাখতে হয়। সেক্ষেত্রে mark() ব্যবহার করা হয়। তবে mark() method এর করা হয়। তবে mark এর value Reset করার জন্য Reset() ব্যবহার করা হয়।

তবে mark() এর আগে reset() কল করা যায় না,

ByteBuffer bb=ByteBuffer.allocate(8);

int capacity=bb.capacity();

int limit=bb.limit(0;

int position=bb.position();

try{

bb.reset();

}

catch(InvalidMarkException ex)

{

}

Method of Buffer :

* get()

বর্তমান পজিশন এর ডাটা রিটার্ন করে এবং পজিশন এক বাড়ায়।

* get(int index)

ইনডেক্স অনুযায়ী তার ভালু রিটার্ন করে। এটি পজিশন বাড়ায় না।

* put(byte b)

বর্তমান পজিশন এ ভালু সেট করে এবং পজিশন এক বাড়ায়।

* put(int index,byte b)

index এ ডাটা Store করে।

ByteBuffer bb=ByteBuffer.allocate(8);

System.out.println(“Right after Creation”);

printBuffer(bb);

for(int i=10;i<18;i++)

bb.put((byte)i);

System.out.println(“After populating data”);

printlBuffer(bb);

}

public static void printBuffer(ByteBuffer bb)

{

int limit=bb.limit();

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

System.out.println(bb.get(i)+” “);

}

ByteBuffer এর মত অন্যান্য Buffer গুলোও কাজ করে।

ByteBuffer এ কিছু মেথড আছে যা ব্যবহার করে RawBuffer কে Primitive Type এ পার্সকরা যায়।

* public char getChar()
* public void putChar(char val)
* public char getChar(int position)
* public void putChar(int index,char val)
* public int getInt()
* public void putInt(int val)
* public int getInt(int position)
* public void putInt(int index, int val)

Chanel

1. Chanel এর মাধ্যমে একই সাথে Data Read এবং Write করা যায়।
2. Chanel এ Asynchronously Data Read/Write করা যায়।
3. chanel এ Buffer শুধুমাত্র এর মাধ্যমে Data Reda/Write করতে পারে।
4. Chanel শুধুমাত্র ByteBuffer এ কাজ করে।
5. Chanel একটি Interface.

InputStream is=new FileInputStream(“test.txt”);

ReadableByteChanel rbc=Chanel.newChanel(is);

আবার Chanel হতেও Input Stream পাওয়া যায়,

InputStream is=Chanels.newInputStream(rbc);

সরাসরি Channel ব্যবহারের জন্য এদের প্রতেকটিকে একটি মেথড যুক্ত করা হয়েছে getChannel() এটি একটি FileChannel রিটার্ন করে। FileChannel ReadableByteChannel, WritableByteChannel এই সবগুলো Interface কে Implement করে। এর মাধ্যমে read/write দুইটি ই করা যায়।

FileChannel যদি FileInput হতে আছে তবে read only মোডে ওপেন হবে।

যদি FileOutputStream হতে আসে তবে write only মোডে ওপেন হবে।

Ex:-

FileInputStream fis=new FileInputStream(“text.txt”);

FileChannel ch1=fis.getChannel(); //read only

FileOutputStream fos=new FileOutputStream(“text.txt”);

FileChannel ch2=fos.getChannel(); //write only

Read File:-

flip() method change Buffer write to read mode.

File inputFile=new File(“hello.txt”);

if(! inputFile.exists())

{

System.out.println(“File doesn’t exits”);

Return;

}

FileInputStream fis=new FileInputStream(inputFile);

Filechannel fileChannel=fis.getChannel();

ByteBuffer buffer=ByteBuffer.allocate(1024);

While(fileChannel.read(buffer)>0)

{

buffer.flip();

while(buffer.hasRemaining())

{

byte b=buffer.get();

System.out.println((char)b);

}

buffer.clear();

}

fileChannel.close();

}

catch(Exception e)

{

}

Bytebuffer এর একটি method hasRemaining() যা true return করে যদি যতক্ষণ পর্জন্ত Buffer এ অন্তত একটি byte অবশিষ্ট আছে।

File write:-

File outputFile=new File(“hello.txt”);

String text=”Bangladesh”;

FileOutputStream fos=new FileOutputStream(outputFile);

FileChannel fileChannel=fos.getChannel();

byte[] bytes=text.getBytes();

ByteBuffer buffer=ByteBuffer.wrap(bytes);

fileChannel.close();

try-with-resource:-

প্রোগ্রাম এর কাজ শেষ হয়ে গেলে যে অবজেক্ট গুলো close করে দেওয়া অত্যাবশ্যক যেগুলোকে Resource বলে।

try-with-resource যোগ করা হয়েছে java 7 এ।

try(FileInputStream fis=new FileInputStream(“file.txt”))

{

}

এটি try-with-resource. যখন try block এর কাজ শেষ হবে তখন এই Resource টি close হয়ে যাবে।

FileInputStream ক্লাশ টি AutoCloseable Interface কে implement করে। যেসব class এই interface কে implement করে তাদের try-with-resource এ ব্যবহার করা যাবে।

Ex:-

try(FileChannel fileChannel=new FileOutputStream(outputFile).getChannel())

{

}

File Read and Write at the Time:-

FileChannel ব্যবহার করে একটি File এর Content অন্য File এ কপি করতে transferTo() ব্যবহৃত হয়।

Ex:-

File source=new File(“hello.txt”);

File sink=new File(“hello2.txt”);

copy(source,sink);

}

public static void copy(File source, File sink)

{

if(!source.exists() || sink.exists())

{

System.out.println(“File are not exists”);

return;

}

try(FileChannel src=new fileInputStream(source).getChannel(); FileChannel dst=new FileOutputStream(sink).getChannel())

{

src.transferTo(0,src,size(),dst);

}

catch(Exception e)

}

}

}

Character Set:-

java.io.charset.Charset ক্লাস টি ক্যারেক্টার সেট এবং বিভিন্ন এনকোডিং স্কিমকে রিপ্রেজেন্ট করে।

বাংলা বর্ণ রীড করার নিয়মঃ

* বাংলা বর্ণ বা ক্যারেক্টার গুলো এক বাইট দ্বারা প্রকাশ করা যায় না।
* Text File টি যে পদ্ধটিতে এনকোডিং করা হয়েছিল, সেই এনকোডিং স্কীম ব্যবহার করে ডিকোডিং করা হয় নি।

Ex:

Charset cs=Charset.forName(“UTF-8”);

CharBuffer cb=CharBuffer.wrap(“ ক্যারেক্টার”);

ByteBuffer encodeData=cs.encode(cb);

CharBuffer decodeData=cs.decode(encodeData);

Write file:-

File outputFile=new File(“hello.txt”);

String text=” ক্যারেক্টার”;

Charset cs=Charset.forName(“UTF-8”);

try(FileChannel fileChannel=new FileOutputStream(outputFile).getChannel())

{

CharBuffer charBuffer=CharBuffer.wrap(text);

ByteBuffer encodeBuffer=cs.encode(charBuffer);

fileChannel.write(encodeBuffer);

}

Read File:-

File inputFile=new File(“hello.txt”);

try(FileChannel fileChannel=new FileInputStream(inputFile).getChannel())

{

ByteBuffer buffer=ByteBuffer.allocate(1024);

Charset cs=charset.forName(“UTf-8”);

while(charBuffer.hasRemaining())

{

char achar =charBuffer.get();

System.out.println(achar);

}

Buffer.clear();

}

জাভাতে সবগুলো এনকোডিং স্কিম দেখার প্রগ্রামঃ-

Map<String, Charset> map=Charset.availableCharset();

Set<String> keys=map.keySet();

int count=keys.size();

System.out.println(“Available keys :”+count);

for(String ch:keys)

{

System.out.println(ch);

}