# UNIT 9 NON BLOCKING I/O
# LH - 3HRS

PRESENTED BY: ER. SHARAT MAHARJAN
NETWORK PROGRAMMING
PRIME COLLEGE, NAYABAZAAR

# CONTENT (LH - 3HRS)

9.1 An example client and server

9.2 Buffers: Creating Buffers, filling and draining, bulk methods, data conversion, view buffers, compacting buffers, Duplicating buffers, Slicing Buffers, Marking and Resetting, Object Methods.

9.3 Channels: Sockets Channel, ServerSocketChannel, The Channels Class, Asynchronous Channels, Socket Options

9.4 Readiness Selection: The Selectors Class, The Selection Key Class

# 9.1 An example client and server

- Non-blocking I/O in Java is part of the New I/O (NIO) library introduced in Java 1.4.
- It allows for high scalability and better resource management by enabling a single thread to manage multiple channels of input and output.

https://github.com/Sharatmaharjan/Np/blob/main/code/NonBlockingIO.java

# 9.2 Buffers

a. Creating Buffers

```
ByteBuffer buffer = ByteBuffer.allocate(256);
```

b. Filling and Draining

```
// Filling the buffer
buffer.put("Hello".getBytes());
// Draining the buffer
buffer.flip();
byte[] bytes = new byte[buffer.remaining()];
buffer.get(bytes);
System.out.println(new String(bytes));
```

c. Bulk Methods

```
ByteBuffer buffer = ByteBuffer.allocate(256);

byte[] data = "Some data".getBytes();

buffer.put(data);


// Bulk get

buffer.flip();

byte[] receivedData = new byte[buffer.remaining()];

buffer.get(receivedData);

System.out.println(new String(receivedData));
```

d. Data Conversion

```
ByteBuffer buffer = ByteBuffer.allocate(256);
buffer.putInt(42);
buffer.flip();
int number = buffer.getInt();
System.out.println(number);
```

e. View Buffers

```
ByteBuffer buffer = ByteBuffer.allocate(256);
CharBuffer charBuffer = buffer.asCharBuffer();
charBuffer.put("Hello");
```

f. Compacting Buffers

```
buffer.compact();
```

g. Duplicating Buffers

```
ByteBuffer duplicateBuffer = buffer.duplicate();
```

h. Slicing Buffers

```
ByteBuffer slice = buffer.slice();
```

i. Marking and Resetting

```
buffer.mark();
buffer.put("Marked position".getBytes());
buffer.reset();
```

j. Object Methods

```
System.out.println(buffer.position());

System.out.println(buffer.limit());

System.out.println(buffer.capacity());
```

# 9.3 Channels:

a. SocketChannel

```
SocketChannel socketChannel = SocketChannel.open(new InetSocketAddress("localhost", 8080));
```

b. ServerSocketChannel

```
ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();

serverSocketChannel.bind(new InetSocketAddress(8080));
```

c. The Channels Class

The Channels class provides utility methods to interact with channels. For example, converting a stream to a channel:

```
ReadableByteChannel rbc = Channels.newChannel(System.in);

WritableByteChannel wbc = Channels.newChannel(System.out);
```

d. Asynchronous Channels

```
AsynchronousSocketChannel asyncChannel = AsynchronousSocketChannel.open();

asyncChannel.connect(new InetSocketAddress("localhost", 8080));
```

e. Socket Options

```
SocketChannel socketChannel = SocketChannel.open();

socketChannel.setOption(StandardSocketOptions.SO_KEEPALIVE, true);
```

# 9.4 Readiness Selection

a. The Selectors Class

```
Selector selector = Selector.open();
```

b. The SelectionKey Class

```
SelectionKey key = socketChannel.register(selector, SelectionKey.OP_READ);
if (key.isReadable()) {
   // Handle read
}
```

# THANK YOU FOR YOUR ATTENTION