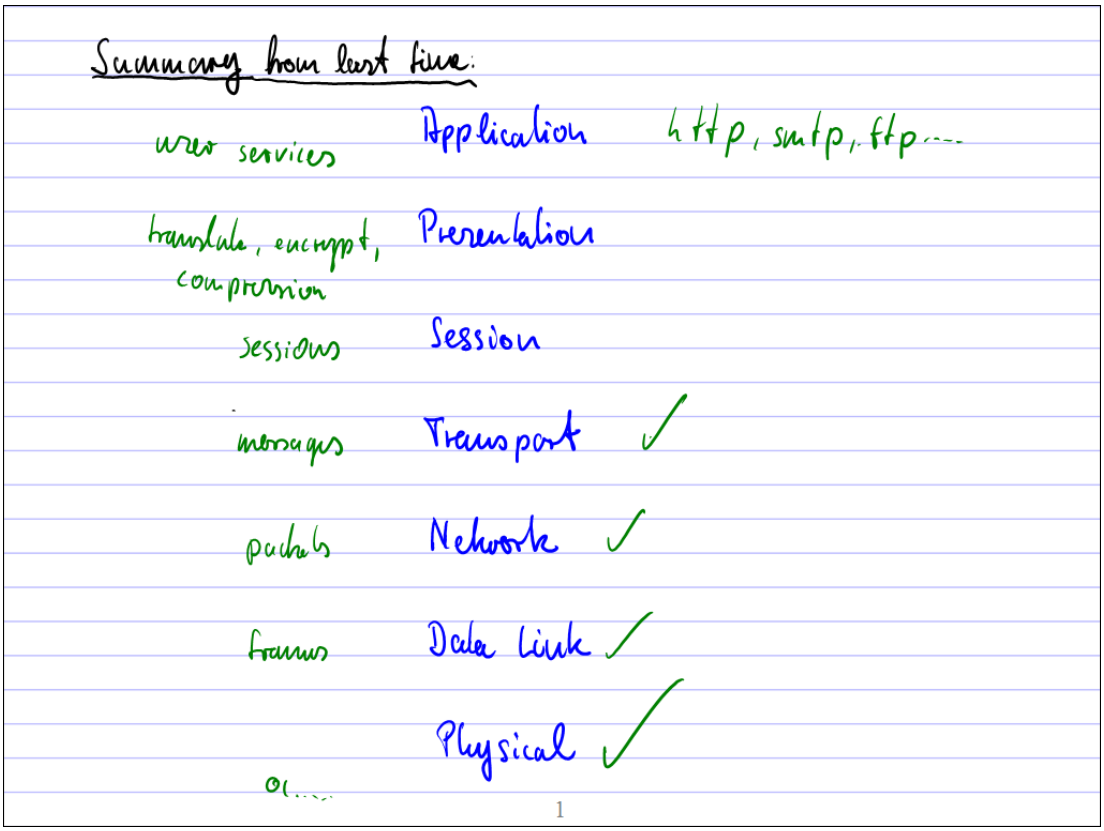
Panel 1



Panel 2

Panel 3

Quiz #2

① How many layers does the OSI Reference model stipulate?

② List all layers of the OSI model in order, starting with the lowest layer.

3

Panel 4

Quiz #2 - continued

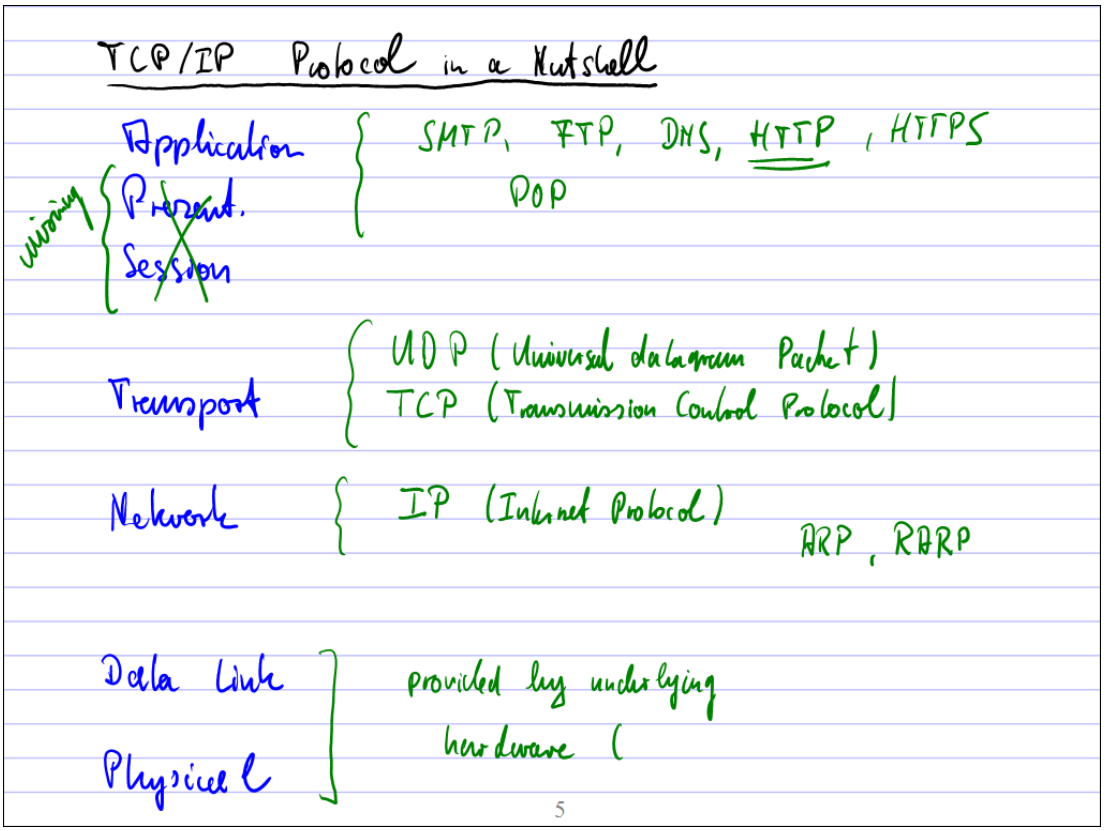
③ Which layer divides groups of bits into frames?

④ Which layer is responsible for routing packets from source to destination?

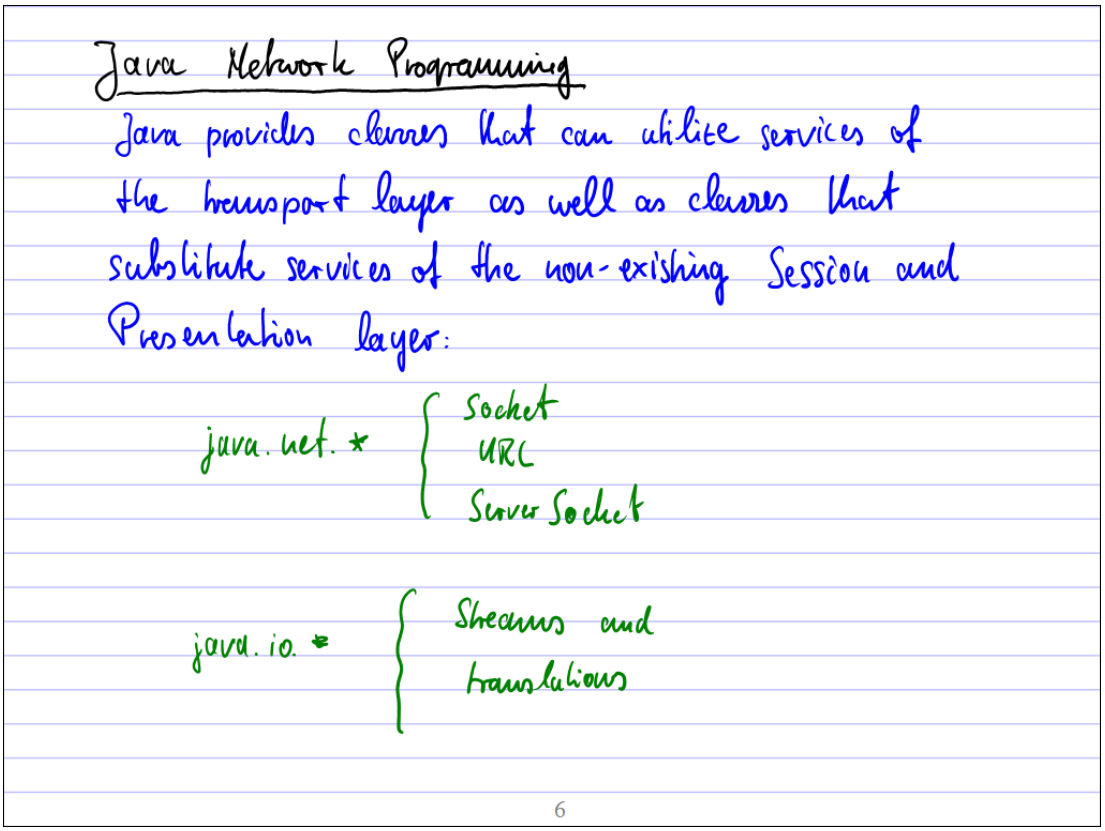
⑤ Which layer is responsible for inserting sequence numbers into packets?

4

Panel 5



Panel 6



Panel 7

Definition: The `Socket` Class

This class is part of the `java.net` package and implements a client socket.[1] Constructing a new `Socket` instance will turn a passive socket on a port into an active socket if a connection to the specified port on the host can be established. The `Socket` class can return input streams to receive information from the host and output streams to send information to the host. The Java API defines `Socket` as follows:

```
public class Socket extends Object
{ // selected constructor
   public Socket(String host, int port)
          throws UnknownHostException, IOException
   // selected methods
   public InetAddress getInetAddress()[2]
   public InetAddress getLocalAddress()
   public int getPort()
   public int getLocalPort()
   public InputStream getInputStream() throws IOException
   public OutputStream getOutputStream() throws IOException
   public void close() throws IOException
}
```

session layer

Definition: The `InetAddress` Class

This class is part of the java.net package and represents an IP address. The Java API defines `InetAddress` as follows:

```
public final class InetAddress extends Object implements Serializable
{  // selected methods
   public String getHostName()[3]
   public String getHostAddress()
   public boolean equals(Object obj)
}
```

utility class

7

Panel 8

Definition: The `URL` Class

The `URL` class represents a Uniform Resource Locator, or URL. The Java API defines the class as follows:

```
public final class URL extends Object implements Serializable
{  // selected constructors
   public URL(String protocol, String host, int port, String file)
          throws MalformedURLException
   public URL(String protocol, String host, String file)
          throws MalformedURLException
   public URL(URL context, String spec)[31]
          throws MalformedURLException
   public URL(String spec)
          throws MalformedURLException
   // selected methods
   public int getPort()
   public String getProtocol()
   public String getHost()
   public String getFile()
   public boolean equals(Object obj)
   public String toString()
   public final InputStream openStream() throws IOException
   public final Object getContent() throws IOException
}
```

utility class

connects to web server!

8

Panel 9

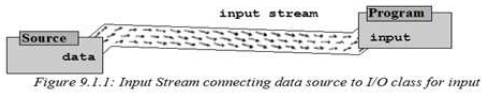


Figure 9.1.1: Input Stream connecting data source to I/O class for input

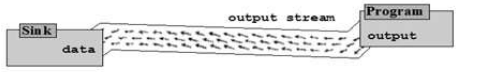
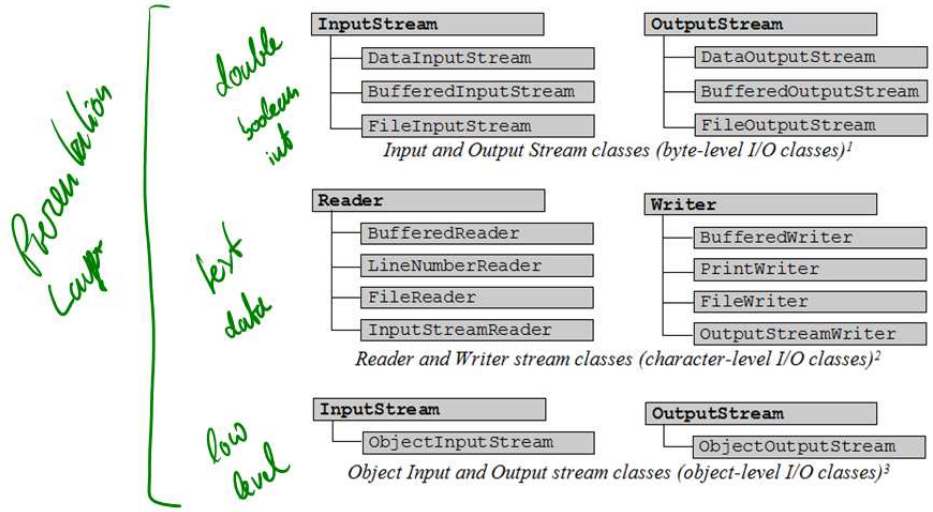


Figure 9.1.2: Output Stream connecting I/O class to data sink for output



Input and Output Stream classes (byte-level I/O classes)[1]

Reader and Writer stream classes (character-level I/O classes)[2]

Object Input and Output stream classes (object-level I/O classes)[3]

9

Panel 10

Characters Translation Classes

Class Name	Description from API
`Writer`	Abstract class for writing to character streams.
`BufferedWriter`	Writes text to a character-output stream, buffering characters.
`PrintWriter`	Print formatted representations of objects to a text-output stream. Methods in this class never throw `I/O` exceptions but clients can check whether errors occurred using `checkError`
`OutputStreamWriter`	A bridge from character streams to byte streams. It translates characters to bytes and writes the bytes to the stream
`FileWriter`	Convenience class for writing character files.

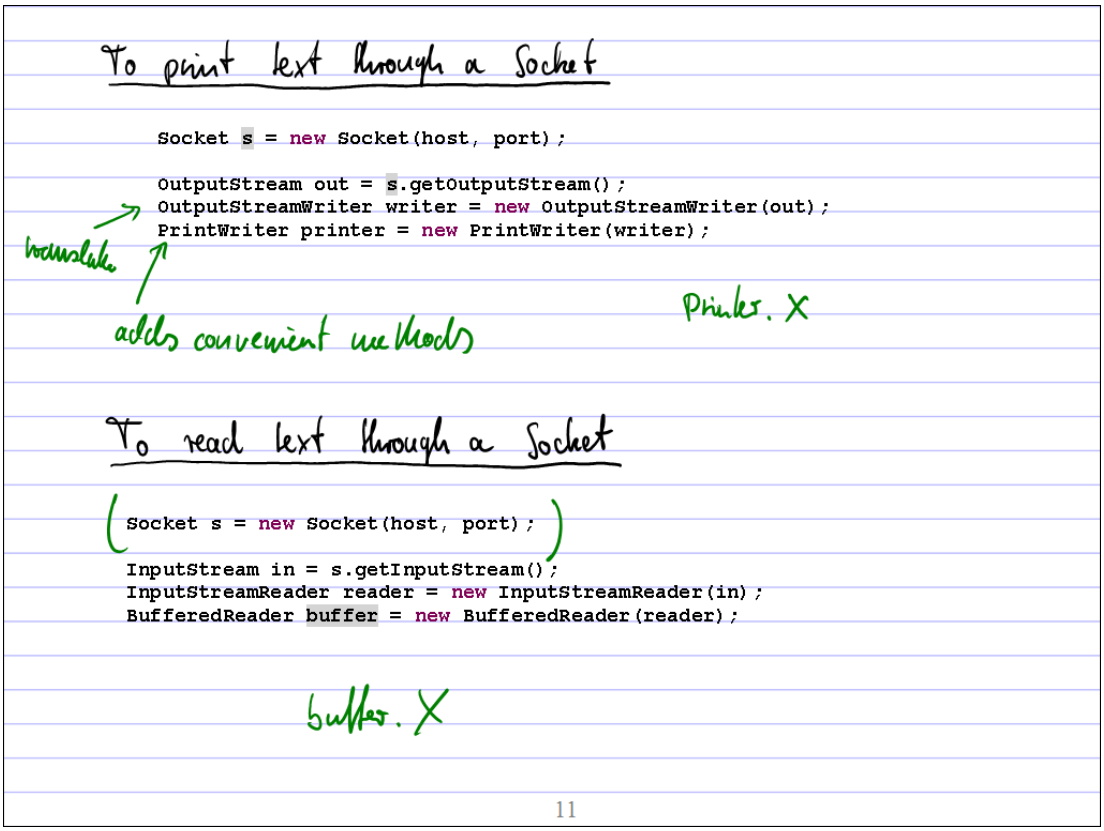
Classes for writing character-based data to streams

Class Name	Description from API
`Reader`	Abstract class for reading character streams.
`BufferedReader`	Reads text from a character-input stream, buffering characters as necessary.
`LineNumberReader`	A buffered character-input stream that keeps track of line numbers.
`InputStreamReader`	A bridge from byte streams to character streams: It reads bytes and translates them into characters.
`FileReader`	Convenience class for reading character files.

Classes for reading character-based data from streams

10

Panel 11



Panel 12

```
public static void grab(String host, int port)
    throws IOException
{
    Socket sock = new Socket(host, port);
    System.out.println("Connected to: " + host);
    System.out.println(" on port " + port);

    OutputStream out = sock.getOutputStream();
    OutputStreamWriter writer = new OutputStreamWriter(out);
    PrintWriter printer = new PrintWriter(writer);

    printer.println("GET /Styles/default.css HTTP/1.0");
    printer.println();
    printer.flush();

    InputStream in = sock.getInputStream();
    InputStreamReader reader = new InputStreamReader(in);
    BufferedReader buffer = new BufferedReader(reader);

    String line = buffer.readLine();
    while (line != null)
    {
        System.out.println(line);
        line = buffer.readLine();
    }
    System.out.println("Done.");

    printer.close();
    sock.close();
}
```

12

Panel 13