# Chp14 网络编程

## 参考答案

1. 参考答案
   TCP 和 UDP 都是传输层协议，TCP 是有连接（有连接|无连接）的协议，UDP 是无连接（有连接|无连接）的协议。这两种协议中，TCP 协议更安全，而 UDP 协议传输效率更高。

2. ABC

3. AB
   ServerSocket 中没有 getInputStream 和 getOutputStream 方法

4. 参考答案
   一般而言，创建一个 Tcp 客户端，有以下几步：
   1） 创建一个 Socket 对象
   2） 调用 getInputStream 方法和 getOutputStream 方法获得输入输出流
   3） 利用输入输出流，读写数据
   4） 关闭 socket

   创建一个多线程的 Tcp 服务器，有以下几步
   1) 创建 ServerSocket 对象
   2) 调用该对象的 accept 方法，以获取客户端的连接。该方法返回一个 Socket 对象。
   3) 利用返回的对象，创建一个新线程
   4) 在新线程中完成读写操作
   5) 在新线程中调用 Socket 对象的 close 方法

5. 参考答案

```
//Client.java
import java.net.*;
import java.io.*;
public class Client{
    public static void main(String args[])throws Exception{
        Socket s;
        //创建一个到“127.0.0.1: 9000”的 Tcp 连接
        s = new Socket(“127.0.0.1”, 9000);
        //向 Tcp 连接输出“Hello World”并换行
        PrintWriter pw = new PrintWriter(s.getOutputStream());
        pw.println(“Hello World”);
        //从服务器端读入一行文本，并打印出来
```

```
        BufferedReader br = new BufferedReader(
                new InputStreamReader(s.getInputStream()));
        String str = br.readLine();
        System.out.println(str);
        s.close();
    }
}

//Server.java
import java.io.*;
import java.net.*;
public class Server{
    public static void main(String args[]) throws Exception {
        //创建一个服务器端口对象
        ServerSocket ss = new ServerSocket(9000);
        //获得一个客户的连接
        Socket s = ss.accept();
        //读入一行文本
        BufferedReader br = new BufferedReader(
                new InputStreamReader(s.getInputStream()));
        String str = br.readLine();
        //在读入的文本后面加上+“ From Server”
        str += “ From Server”;
        //把处理之后的文本向客户端输出并换行
        PrintWriter pw = new PrintWriter(s.getOutputStream());
        pw.println(“Hello World”);
        //关闭连接
        s.close();
    }
}
```

6. 参考答案
   在 UDP 编程中，表示 UDP 端口的是 DatagramSocket 类，其中发送和接受的方法分别为 send 方法和 receive 方法；
   表示 UDP 数据包的类是 DatagramPacket 类。

7. 参考答案
   URL 编程中，要用到 URL 类的 openConnection 方法获得一个 url 连接，该方法返回值为 URLConnection 类型。
   可以对返回的对象调用 getInputStream 方法，用来为读取 url 上的数据做准备。

8. 输出为 null。应该在调用 PrintWriter 的 println 方法之后使用 flush()刷新缓冲区

9. C

由于调用了 print 方法而不是 println 方法。

由于在读取数据时调用了 readLine 方法，这个方法会直到获得一个换行符才会返回，而如果在写数据时用 print 方法而不用 println 方法，则写数据时不会写入一个换行符。这样会造成 readLine 方法一直等待，因此结果就是没有输出结果。

10. 参考答案

```
//UDPServer.java
import java.io.*;
import java.net.*;

public class UdpServer {

    public static void main(String[] args) throws Exception {
        DatagramSocket socket = new DatagramSocket(9000);
        byte[] bs = new byte[128];
        DatagramPacket packet;
        //创建一个 packet，用 bs 数组来接受数据
        packet = new DatagramPacket(bs, bs.length);
        //接收客户端发送的信息
        socket.reiceve(packet);
        for(int i = 0; i<30; i++){
            bs = "Hello From Server".getBytes();
            DatagramPacket newPacket = new DatagramPacket(
                    bs, 0, bs.length,
                    packet.getAddress(), packet.getPort());
            //向客户端发送数据
            socket.send(newPacket);
        }
        socket.close();
    }
}
//Client
import java.io.*;
import java.net.*;

public class UdpClient {

    public static void main(String[] args) throws Exception{
        DatagramSocket socket = new DatagramSocket();
        byte[] data = "Hello".getBytes();
        DatagramPacket packet = new DatagramPacket(
                data, 0, data.length,
                new InetSocketAddress("127.0.0.1", 9000));
        //发送数据包
```

```
            socket.send(packet);
            data = new byte[128];

            for(int i = 0; i<30; i++){
                //以 data 作为媒介，创建一个新数据包
                packet = new DatagramPacket(data, data.length);
                //接受数据包
                socket.receive(packet);
                String str = new String(
                        packet.getData(), packet.getOffset(),
                        packet.getLength());
                System.out.println(str);
            }

            socket.close();
        }
    }
```

11. 参考 UpperCaseClient.java 和 UpperCaseServer.java

12. 参考 FtpServer.java 和 FtpClient.java