Java による IP アドレスの管理　(Z:ドライブ内に「java_net」フォルダを作り、中に記憶のこと)

java では、java.net.InetAddress クラスで IP アドレスを管理します。

このクラスには、コンストラクタが存在せず、次の static メソッドで取得します。

- ローカルの InetAddress 取得メソッド→**getLocalHost**()
- ホスト名から、全ての InetAddress を取得するメソッド→**getAllByName("ホスト名")**
- ホスト名から、一つの InetAddress を取得するメソッド→getByName("ホスト名")
- IP アドレスのバイト列から取得するメソッド→**getByAddress**(IP アドレスバイト配列)

以上で得られた InetAddress のインスタンスより、次の逆の情報も得られます。

- ホスト名取得で使うメソッド（戻り値:String）→getHostName()
- IP アドレスのバイト列取得で使うメソッド（戻り値:byte[]）→getAddress()
- IP アドレスのバイト列取得で使うメソッド（戻り値:String）→getHostAddress()

次のプログラムで確認ください。InetAddress は今後の UDP や TCP のプログラムでも使います。

```
01 import java.net.*;
02 import java.util.Enumeration;
03
04 public class IP {
05    public static void print(InetAddress inet)throws Exception{
06        System.out.print("IP アドレス:" + inet.getHostAddress());
07        String hostName = inet.getHostName();
08        System.out.println("  ホスト名:" + hostName);
09    }
10    public static void printLocal()throws Exception{
11        InetAddress inet  = InetAddress.getLocalHost();//このマシンの IP アドレス情報取得
12        System.out.print("この PC の");
13        print(inet);
14    }
15
16    //引数でホスト名（"localhost" や  NetBIOS 名 や  "proxy.scc01"  や
17    //    "web.scc01" や "google.co.jp"）を指定できます。
18    public static void main(String[] args) throws Exception {
19        printLocal();//このマシンの IP アドレスとホスト名を表示
20        if(args.length == 0) {
21             System.out.println(" コマンドラインで、ホスト名を指定できます。");
22             System.exit(0);//終了
23        }
24        //ホスト名から全ての IP アドレスを表示（内部で、DNS を利用している。）
25        //ホスト名を指定してアドレス群を得る。
26        InetAddress []adrs = InetAddress.getAllByName(args[0]);
27        for(int i = 0; i < adrs.length; i++){
28             System.out.println(adrs[i].getHostAddress());
29        }
30    }
31 }
```

実行例

```
Z:¥ネットワーク java_net>java IP
この PC の IP アドレス:192.168.11.101  ホスト名:TL40-56
コマンドラインで、ホスト名を指定できます。

E:¥１ABK ネットワーク¥java_net>java IP web.scc01
この PC の IP アドレス:192.168.11.101  ホスト名:TL40-56
192.168.10.101

E:¥１ABK ネットワーク¥java_net>
```

ホストによっては、
複数の IP アドレスを持つ。
自身のホスト名でも実験せよ

上記は、課題です。作成のこと

ネットワークに関する Java のクラスは他にもあります。

マシンによっては前述の InetAddress.getLocalHost()で割り当てられる IP アドレスが得られずに、127.0.0.1 が取得されることもあります。

（Java のメソッド InetAddress.getLocalHost() はローカルマシンのホスツファイル情報を参照して、次ぎに DNS を使うため？　なお hosts（ホスツ）とは、TCP/IP 利用のコンピュータにおいて、IP アドレスとホスト名の対応を記述したテキストファイルです。Linux の場合 /etc/hosts、Windows の場合は、c:¥Windows¥system32¥drivers¥etc にあります。）

希望のアドレス場合は、NetworkInterface クラスを使います。これはネットワークインターフェイスカード（NIC）の情報を管理するクラスで、「lo」や「eth0」のような識別名で管理されます。

以下にこのクラスを使って、各種 IP アドレスの情報を表示するプログラムを示します。

内容的に、難しいので、余裕のない方は、前ページの作品だけでよい。

```
01 import java.net.*;
02 import java.util.Enumeration;
03
04 //ネットワークインターフェイスカード（NIC）のインターフェイスを利用
05 public class NetInfo {
06   public static InetAddress getInetAddress() throws Exception {
07       InetAddress rtnInet = null;
08       Enumeration <NetworkInterface> netSet =
09                NetworkInterface.getNetworkInterfaces();
10       while(netSet.hasMoreElements()){
11           NetworkInterface netinterface =
12               (NetworkInterface) netSet.nextElement();
13           Enumeration <InetAddress> inetSet =
14                netinterface.getInetAddresses();
15           //if( inetSet.hasMoreElements() == false) continue;
16           System.out.println(netinterface.getName());//ネットワーク識別名
17           while(inetSet.hasMoreElements()){
18               InetAddress inet = (InetAddress) inetSet.nextElement();
19               System.out.print("¥t" + inet.getClass().getName());
20               System.out.println(inet);
21               rtnInet = inet;
22           }
23       }
24       return rtnInet;
25   }
26   public static void main(String[] args) throws Exception {
27       InetAddress inet = getInetAddress();
28       System.out.println(inet);
29       IP.print(inet);
30   }
31 }
```

余裕ある方は、希望の IP アドレスが戻るように検討して、変更せよ。

実行例→

IP アドレスが無いネットワーク識別名は表示しないように変更せよ

```
Z:¥ネットワーク java_net>java NetInfo
lo
        java.net.Inet4Address/127.0.0.1
        java.net.Inet6Address/0:0:0:0:0:0:0:1
net0
net1
net2
ppp0
eth0
eth1
eth2
ppp1
net3
eth3
        java.net.Inet6Address/fe80:0:0:0:9dd1:e54b:3589:d23c%11
net4
        java.net.Inet4Address/192.168.11.101
```

文字列をバイト列に変換して、それが書き込まれるファイルを作成し、読み取る確認

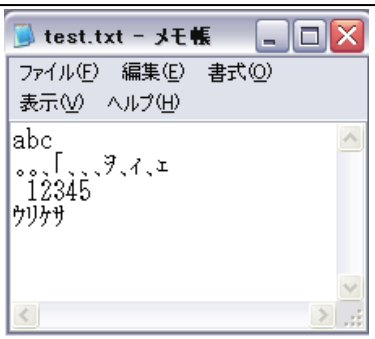
次のプログラムは、
右の枠内のデータを書き込んだファイルを
作成します。

```
abc
  あいうえお
 12345
```

この時文字列を、Linux の OS などでよく使われるキャラクタセットの"EUC_JP"でバイト列で取得し、それで出力しています。（このバイト列の取得に **getBytes** メソッドを使っています。）

```
class EucWrite{
    public static void main(String[] arg) throws Exception{
        String str = "abc¥r¥n  あいうえお¥r¥n 12345¥r¥n";
        //EUC のファイル  test.txt を作成
        FileOutputStream fos = new FileOutputStream("Z:¥¥net¥¥euc.txt");
        byte[] a = str.getBytes("EUC_JP");//"EUC_JP"のキャラクタセット使用
        fos.write(a);
        fos.close();
    }
}
```

このように作成したファイルを、
Windows 用のメモ帳で見ても右のように正しく見えません。
ですが、バイナリとして Linux へコピーすれば
Linux で正しく見えます。
これが、『**文字化け**』という現象です。
Linux では文字コードに「EUC」を使い、Windows では
「ShiftJIS」を使うためです。同じ文字でも違うコード番号



になるためです。ネットワークのように、異なるオペレーションシステム間で、文字列通信を行う場合は、相手が何の**キャラクタセット**で送ったかを知り、それに対応した読み取りが必要になります。上記ファイルであれば、次のプログラムで正しく表示できます。

```
class EucRead{
  public static void main(String[] arg) throws Exception       {
      File file = new File("Z:¥¥net¥¥euc.txt");//ファイル情報を取得
      int size = (int)file.length(); //ファイルサイズ長を取得  byte 単位
      byte[] array = new byte[size];//ファイルサイズと一致するバイト配列を用意

      //ファイルから読むための入り口を管理するインスタンス取得
      InputStream is = new FileInputStream(file);
      is.read(array);//一括読み込み
      is.close();//入り口を閉じる

      //バイト列から文字列へ変換して表示
      String s = new String(array,"EUC_JP");//"EUC_JP"のキャラクタセット使用
      System.out.print(s);
  }
}
```

上記の,**"EUC_JP"**を、**"MS932"**変更すると、メモ帳表示と同じような『**文字化け**』が確認できるでしょう。（**"MS932"**は、Windows で使われるキャラクタセットです。）

問題　**ソースファイル名を** MS932Write.java として作成ください。
プリント１の実行例①の表示部分を、ipinfo.txt の名前のファイルに、**"MS932"**のキャラクタセットで書き込むプログラムを作成し、実行させなさい。