## Тармак жана сокеттер

java.net пакети тармактык иштөөнү камсыздайт. Java TCP/IP протоколдорун, жогорудагы агымдык киргизүү жана чыгаруу темаларында каралган интерфейстерди кеңейтүү жолу аркылуу ишке ашырат. java.net пакети бир нече тармактык класстарды жана интерфейстерди өзүнө камтыйт. Алардын негизгилерине төмөндө токтолуп кетебиз.

ServerSocket классы клиенттин көрсөтүлгөн порту аркылуу аралыктагы же локалдык программаларына байланышуучу серверлерди түзүү үчүн колдонулат. Клиенттик TCP/IP сокеттер интернет менен компьютердин эки жактуу байланышын түзүү үчүн колдонулат. Java TCP сокеттердин эки түрү бар: Сервер үчүн жана клиент үчүн. ServerSocket классы “угуучу” болуп түзүлгөн. Дайыма иштээр алдында, клиент менен байланышты күтөт. Төмөндө, сокетке жөнөкөй мисал келтирилди:

```
import java.net.*;
import java.io.*;
class Set{
public static void main(String args[]) throws Exception{
int c;
Socket s = new Socket ("internic.net", 43);
InputStream in = s.getInputStream();
OutputStream out = s.getOutputStream();
String str = (args.length == 0 ? "www.kg" : args[0]) + "\n";
byte buf[] = str.getBytes();
out.write(buf);
while ((c=in.read()) != -1){
System.out.println((char)c);
}
s.close();
}
}
```

Заманбап Internet World Wide Web (WWW) системасынын протоколдоруна багытталган. Web протокол бул Web браузерде бириккен файлдардын бардык форматтарын жана жогорку деңгээлдеги эркин протоколдордун жыйындысы деп түшүнсөк болот. Анын негиизнде, бардык форматтарды ыңгайлуу ат берүү мүмкүнчүлүгү пайда болот. Бул принцип Uniform Resoursce Locator (URL) деп аталат. Java да URL классынын бир нече конструктору бар жана алардын ар бири ката болгон учурда MalformedExeception чыгарып салуу механизмине өтүшөт. Төмөндө, URL классынын колдонулушу жана ага колдонулуучу ыкмалар каралды.

```
import java.net.*;
class UR{
public static void main(String args[]) throws MalformedURLException{
URL hp = new URL("http://intl.manas.edu.kg/en/engineering/");
System.out.println("Protocol:"+ hp.getProtocol());
System.out.println("Port:"+ hp.getPort());
System.out.println("Host:"+ hp.getHost());
System.out.println("File:"+ hp.getFile());
System.out.println("Url:"+ hp.toExternalForm());
}
}
```

Жогорудагы кодду компилеп, иштетсек, анда engineering web баракчасы кайсы хостто жайгашкан, порту, протоколу, url, http си жөнүндө маалымат чыгат.

## InetAdress классы

InetAdress классы IP жана домен даректи алып жүрөт. Бул классты колдонуу менен компьютердин домендик дареги (host) менен иш алып барабыз. InetAdress бул класс экени бизге белгилүү, ага объект түзүү үчүн , класстын экземплярын кайтарып берүүчү ыкмаларды колдонобуз.

InetAdress классынын экземплярын түзүү үчүн үч ыкма колдонулат:

1) getLocalHost()

2) getByName()
3) getAllByName()

Бул ыкмалардын жалпы жазылышы төмөнкү түрдө жазылат:

```
Static InetAdress getLocalHost()
throws UnknownHostException()
Static InetAdress getByName(String hostName)
throws UnknownHostException()
Static InetAdress[ ] getAllByName(String hostName)
throws UnknownHostException()
```

getLocalHost() ыкмасы локалдык компьютердин IP дарегин чыгарып берет. Ал эми, getByName() ыкмасы hostName параметри жиберген компьютерлердин атын чыгарып берет. Бул ыкмалар компьютердин атын аныктай албаган учурда UnknownHostException чыгарып салуу механизми иштейт. getAllByName() ыкмасы көрсөтүлгөн атка байланышкан InetAdress обьектилерин массив түрүндө чыгарып берет. Бул ыкма дагы, эгер бул атка байланыштуу бир дагы обьект табылбаса, анда чыгарып салуу механизмине өтөт. Төмөндө, InetAddress классы жана анын үч ыкмасынын колдонулушу көрсөтүлдү. Бул код, www.kg web – сайтынын дарегин жана локалдык компьютердин атын чыгарып берет.

```
import java.net.*;
class Set2{
public static void main(String args[]) throws UnknownHostException{
InetAddress adress = InetAddress.getLocalHost();
System.out.println(adress);
adress = InetAddress.getByName("www.kg");
System.out.println(adress);
InetAddress sw[]= InetAddress.getAllByName("www.kg");
for(int i=0; i<sw.length; i++){
System.out.println(sw[i]);
}
```

}}

## САПТАРДЫ ИШТЕТҮҮ

**String** – символдордун өзгөрүүсүз ырааттуулугун камтыйт.

**StringBuffer** – өсүүчү жана кайра жазылуучу символдордун ырааттуулугун камтыйт. Саптарды, символдорду түзүүгө мисал келтирели:

String s1= new String();  //бош сап

Char ch[]={‘j’,’a’,’v’,’a’};

String s2=new String(ch);  //java

String s3=new String(s2); //java

Java да  char() 16 разряддык символду (Unicod) камтыйт. Ал эми ASCII символдорун чыгаруу үчүн String классына byte-массив колдонулат. Мисалы:

char ch2[]={66,67,68};

String s4=new String(ch2);  //B,C,D

String s5=new String(ch2,1,2);   //C D

### Саптын узундугу

Саптын узундугун алуу үчүн **length()** ыкмасын колдонобуз.

**String s6=”abc”;**

**System.out.println(“abc”.length());**

**toString()** ыкмасы – саптык маанилерди кайтарууда жардам берет. Бул ыкманы төмөндөгүдөй колдонсок болот.

```
public class Exz {
        double height;
      double depth;
      double width;
      Exz(double h, double d, double w){
      height=h;
      depth=d;
      width=w;
      }
```

```
        public String toString(){
            return(width+"x"+height+"x"+depth);
        }
      public static void main(String[] args) {
        Exz ob = new Exz(12,3,5);
        System.out.print(ob);
     }
}
```

**charAt()** – ыкмасы String объектинин бир символун алып чыгат.

```
char s1="abc".charAt(2);
System.out.println(s1);  //c
```

Бир символдон ашык чыгарыш керек болсо **getChar()** ыкмасын колдонобуз.

```
String s1= new String("java world");
int a=2;
int b=4;
char buf[]=new char[b-a];
s1.getChars(a,b,buf,0);
System.out.println(buf);  //va
```

## Саптарды салыштыруу

String классы саптарды салыштырууда бир нече ыкмаларды колдонот. Эки сапты бири-бири менен салыштырыш үчүн **equals()** ыкмасын , ал эми саптарды чоң же кичине тамгаларды эске албай салыштыргыбыз келсе **equalsIgnoreCase()** ыкмасын колдонобуз. Төмөндө эки ыкманын колдонулушуна мисал келтирилген:

```
String s1= "java";
String s2= "java";
String s3= "world";
String s4= "WORLD";
System.out.println(s1.equals(s2));  //true
```

```
System.out.println(s3.equals(s4));  //false
System.out.println(s3.equalsIgnoreCase(s4));  //true
```

**startsWith()** ыкмасы String объект көрсөтүлгөн саптан башталабы, ал эми **endsWith()** ыкмасы String объект көрсөтүлгөн саптан менен аяктайбы текшерип true же false маанилерин кайтарат. Мисалы:

```
String s3= "java world";
System.out.println(s3.startsWith("ja"));  //false
System.out.println(s3.endsWith("ld"));  //true
```

### equals() ыкмасы жана == оператору аркылуу салыштыруу

Белгилей кетчүү нерсе, equals() ыкмасы жана == оператору экөө эки башка кызматты аткарат. equals() ыкмасы саптарды салыштырат, а эми == оператору объекттик шилтемелерди салыштырат. Эки шилтеме бир экземплярга(объектке) кайрылып жатабы, ошону салыштырат. Төмөнкү мисалда, ыкмалардын иштеши ачык көрүнүп турат.

```
String s1= "java";
String s2= new String (s1);
System.out.println(s1.equals(s1));  //true
System.out.println(s1==s2);  //false
```

Кээ бир учурда эки сапты салыштаруу эле жетиштүү эмес. Эки саптын кайсынысы чоң, кичине же барабар экенин билүүгө туура келет. Ал учурда compareTo() ыкмасын колдонсок болот. Бул ыкма чоң жана кичине тамгаларды эске алат (ASCII таблицасы боюнча ирээттейт). Мисалы:

```
String massiv[]={"Bir","Tort","Alty","On","Oneki" };
for(int i=0; i<massiv.length; i++){
   for(int j=i+1; j<massiv.length; j++){
      if(massiv[j].compareTo(massiv[i])<0){
         String a=massiv[i];
         massiv[i]=massiv[j];
```

```
            massiv[j]=a;
          }
        }
        System.out.println(massiv[i]);
    }
```

String классы саптардын ичинен символдорду издөө үчүн эки ыкманы колдонот. **indexOf()** ыкмасы көрсөтүлгөн символдун сапта биринчи кездешкен маанисин, ал эми **lastIndexOf()** ыкмасы көрсөтүлгөн символдун сапта акыркы кездешкен маанисин int тибинде кайтарат. Эки ыкмага тең баштапкы маани берсе болот. Саптан берилген мааниден баштап символду издөө аткарылат. Жогорудагы айтылган операциялар, мисалда көрсөтүлүп турат:

```
String s1= "java";
System.out.println(s1.indexOf('a')); //1
System.out.println(s1.lastIndexOf('a'));//3
System.out.println(s1.lastIndexOf('a', 1));//1
```

Эки сапты бири-бирине кошуш үчүн **concat()** ыкмасын колдонобуз. Бул ыкма конкатенация операциясын жүргүзөт. Мисалы:

```
String s1= "java";
String s2= "world".concat(s1);
```

Жогорку ыкма төмөндө көрсөтүлгөн бизге тааныш конкатенация операциясы менен барабар.

```
String s1= "java";
String s2= "world"+s1;
```

**replace() ыкмасы** саптагы бир аныкталган смиволду башка символго алмаштырууда колдонулат. Төмөнкү мисалда 'w' тамгасын 'd' тамгасына алмаштыруу көрсөтүлгөн.

```
String s1= "java";
String s2= "world".replace('w', 'd');
System.out.println(s2); //dorld
```

**trim()** ыкмасы саптын башында жана акыркындагы бош аттамак бар болсо өчүрүп, баштапкы саптын көчүрмөсүн кайтарат. Төмөнкү мисалда өлкөнүн атын жазса, анын борбор шаарын чыгарып берет. Эгер кокус башында жана акыркындагы бош аттамак бар болсо өчүрүлөт. Бул ыкма ушул сыяктуу учурларда абдан жардамы тийет.

```
import java.io.*;
import java.util.Scanner;
public class TRim {
public static void main(String[] args) throws IOException{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String s1;
System.out.println("Chyguu uchun stop dep jazinyz ");
System.out.println("Olkonu kirgiz ");
do{
s1=br.readLine();
s1=s1.trim();
if(s1.equals("Kyrgystan")){
   System.out.println("  Bishkek");
}
else if(s1.equals("Uzbekiston")){
   System.out.println("Toshkent  ");
}
}while(!s1.equals("Stop"));
}
}
```

## Саптык операциялар

**insert()** - ыкмасы сапты башка сапка кошот.

```
StringBuffer s1=new StringBuffer("java");
```

```
s1.insert(4," world");
System.out.println(s1);  //java world
```

**reverse()** – ыкмасы саптын символдорунун тартибин аягынан башын көздөй чыгарат.

```
StringBuffer s1=new StringBuffer("java");
s1.reverse(); //avaj
System.out.println(s1);  //java world
```

**delete()** – ыкмасын саптагы символдорду көрсөтүлгөн символдон баштап, аныкталган символго чейин өчүрүүдө колдонобуз.

```
StringBuffer s1=new StringBuffer("java");
s1.delete(2,3); //jaa
System.out.println(s1);  //java world
```

**deleteChar()** – ыкмасы бир символду өчүрүүдө колдонобуз.

```
StringBuffer s1=new StringBuffer("java");
s1.deleteChar(0) //ava
```

**replace()** – ыкмасын саптагы символдорду көрсөтүлгөн символдон баштап, аныкталган символго чейин жазылган сөз менен алмаштырат.

```
StringBuffer s1=new StringBuffer("world is good");
s1.replace(0,5," java");
System.out.println(s1);
```