

OBJEKTNO PROGRAMIRANJE 2

Oznaka predmeta: OP2
Predavanje broj: 10
Nastavna jedinica: JAVA
Nastavne teme:

Slanje fajla sa servera na klijent. FileDialog. DatagramPacket.
DatagramSocket. URL. URLConnection. Generici.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Eckel B., *Thinking in Java*, 2nd edition, Prentice-Hall, New Jersey 2000.
Cay S. Horstmann and Gary Cornell: *"Core Java, Advanced Features", Vol. 2, Prantice Hall, 2013.*
The Java Tutorial, Sun Microsystems 2001. <http://java.sun.com>
Branko Milosavljević, Vidaković M, *Java i Internet programiranje*, GInT, Novi Sad 2002.

Odabiranje i slanje fajla: server

- U sledećem primeru koristi se klasa `FileDialog` za odabiranje fajla koji server šalje klijentima. Unapredite dati primer.

```
import java.util.*; import java.io.*; import java.net.*;
import java.text.SimpleDateFormat; import java.awt.*;
import java.awt.event.*; import javax.swing.*;
public class TrivialFileServer extends JFrame implements ActionListener {
    public final static int SOCKET_PORT = 12345;
    private JButton odaberifajl;
    private FileDialog filedialog;
    private volatile String strfajl;
    private JTextArea jta;
    public TrivialFileServer() {
        setTitle("T_F_S");      setSize(500, 300);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = this.getContentPane();
        odaberifajl = new JButton("ODABERI FAJL");
        odaberifajl.setBackground(Color.YELLOW);
        odaberifajl.addActionListener(this);
        filedialog = new FileDialog(this, "Odaberite fajl !");
        strfajl= null;
        jta = new JTextArea(30,20);
        Ispisi("ODABERITE FAJL ZA KLIJENTE!");
    }
}
```

Odabiranje i slanje fajla: server

```
c.setLayout(new GridLayout(2,1));
c.add(odaberifajl);      c.add(jta);
setVisible(true);
}
public String Ok(){ return strfajl; }
public void Ispisi(String str){
    Date dt = new Date();
    SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");
    String time1 = sdf.format(dt);
    jta.insert(time1+" -> " +str+"\n", 0);
}
public void actionPerformed(ActionEvent ae) {
    if(ae.getActionCommand().equals("ODABERI FAJL")){
        filedialog.setVisible(true);
        strfajl = filedialog.getFile();
        if(strfajl!= null){
            Ispisi("ODABRAN: "+strfajl);
            this.getContentPane().remove(odaberifajl);
            this.getContentPane().setLayout(new GridLayout(1,1));
            revalidate();
        }
    }
}
```

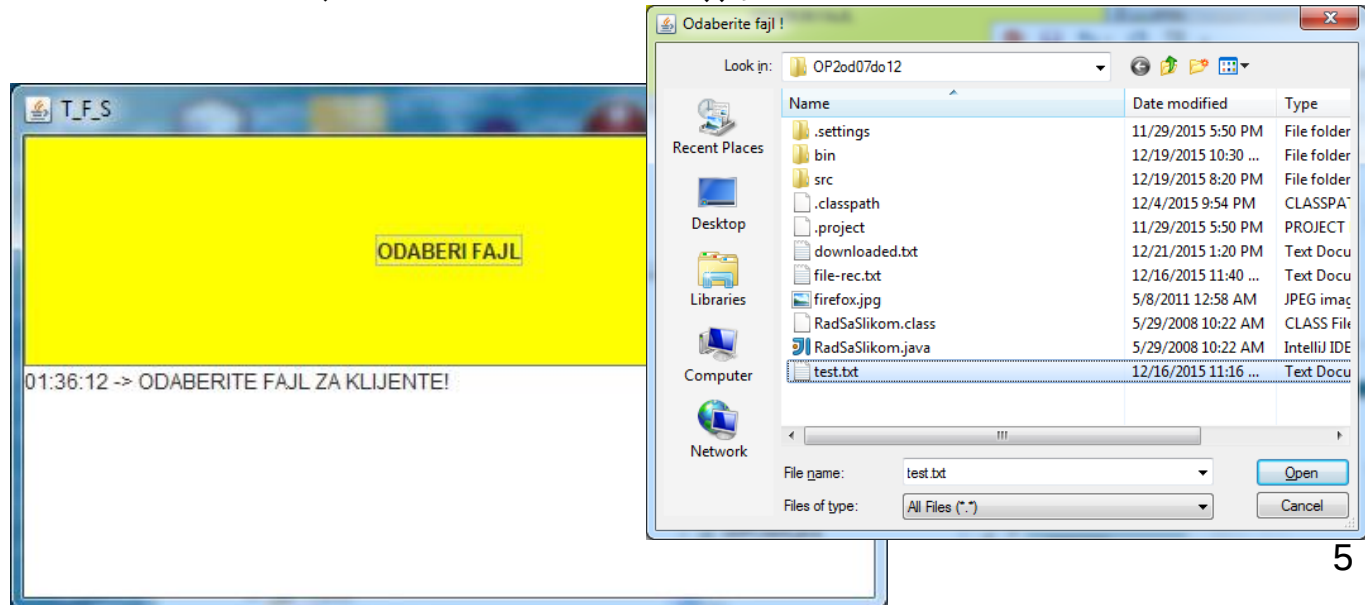
Odabiranje i slanje fajla: server

```
public static void main(String[] args) throws IOException {
    TrivialFileServer tfs = new TrivialFileServer();
    FileInputStream fis = null;
    BufferedInputStream bis = null;
    OutputStream os = null;
    ServerSocket servsock = null;
    Socket sock = null;
    while(tfs.Ok()== null);
    try {
        servsock = new ServerSocket(SOCKET_PORT);
        File myFile = new File(tfs.Ok());
        byte[] mybytearray = new byte[(int) myFile.length()];
        fis = new FileInputStream(myFile);
        bis = new BufferedInputStream(fis);
        bis.read(mybytearray, 0, mybytearray.length);
        while (true) {
            tfs.Ispisi("CEKAM POZIV KLIJENTA ...");
            try {
                sock = servsock.accept();
                tfs.Ispisi("Prihvacen poziv: " + sock);
                // send file
                os = sock.getOutputStream();
            }
        }
    }
}
```

Odabiranje i slanje fajla: server

```
tfs.Ispisi("Saljem " + tfs.Ok() + " (" + mybytearray.length +  
                                                " bytes)");  
  
os.write(mybytearray, 0, mybytearray.length);  
os.flush();  
tfs.Ispisi("Poslato.");  
} finally {  
    if (fis != null) fis.close(); if (bis != null) bis.close();  
    if (os != null) os.close();  
    if (sock != null) sock.close();  
}  
}  
} finally {  
    if (servsock != null) servsock.close();  
}  
}  
}
```

- Sledi kod za klijente. Unapredite primer.

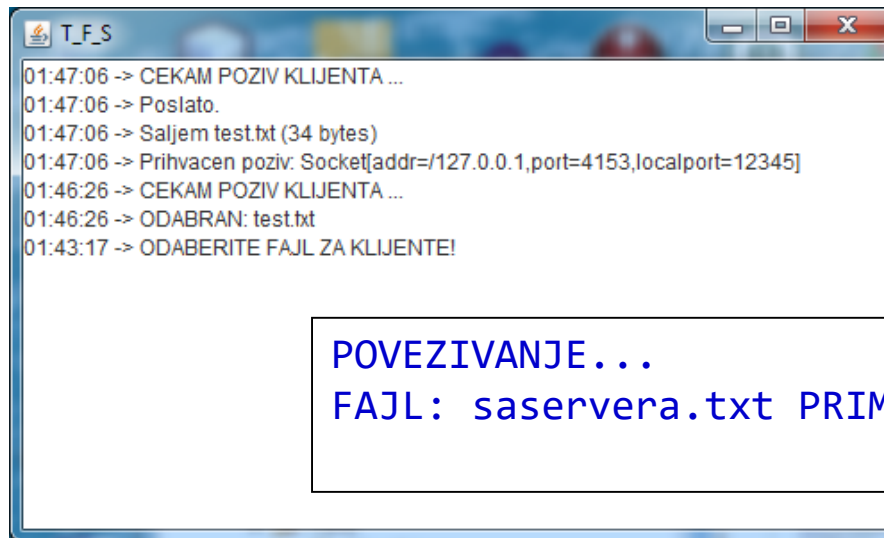


Odabiranje i slanje fajla: klijent

```
import java.io.*; import java.net.*;
public class TrivialFileClient {
    public final static int SOCKET_PORT = 12345;
    public final static String SERVER = "127.0.0.1";
    public final static String FILE_TO_RECEIVED = "saservera.txt";
    public final static int FILE_SIZE = 1000000; //uradite bez ovoga
    public static void main(String[] args) throws IOException {
        int bytesRead;
        int current = 0;
        FileOutputStream fos = null;
        BufferedOutputStream bos = null;
        Socket sock = null;
        try {
            sock = new Socket(SERVER, SOCKET_PORT);
            System.out.println("POVEZIVANJE...");
            // receive file
            byte[] mybytearray = new byte[FILE_SIZE];
            InputStream is = sock.getInputStream();
            fos = new FileOutputStream(FILE_TO_RECEIVED);
            bos = new BufferedOutputStream(fos);
            bytesRead = is.read(mybytearray, 0, mybytearray.length);
            current = bytesRead;
        }
```

Odabiranje i slanje fajla: klijent

```
do {  
    bytesRead = is.read(mybytearray, current, (mybytearray.length -  
                                                    current));  
  
    if (bytesRead > 0) current += bytesRead;  
} while (bytesRead > -1);  
bos.write(mybytearray, 0, current);  
bos.flush();  
System.out.println("FAJL: " + FILE_TO_RECEIVED + " PRIMLJENO (" +  
                    current + " BAJTOVA)");  
} finally {  
    if (fos != null) fos.close();  
    if (bos != null) bos.close(); if (is != null) is.close();  
    if (sock != null) sock.close();  
}  
}
```



Datagram

- TCP garantuje isporuku paketa i čuvanje njihovog redosleda kojim pristižu na odredište.
 - Ova osobina ima i svoju cenu u smislu efikasnosti prenosa.
- Kada navedene osobine nisu potrebne može se koristiti UDP protokol.
 - Ovaj protokol prenosi datagram pakete.
 - Datagram paketi se koriste za realizaciju bes konekcionih (connection-less) paketa.
 - Svaka poruka se prenosi od izvora do odredišta na osnovu podataka sadržanih unutar tog paketa.
 - Svaki paket mora imati adresu odredišta i svaki paket može biti preusmeren drugačije, a može stići i u bilo kom redosledu.
 - Dostava ovakvih paketa nije garantovana.
- Dat je format datagram paketa koji sadrži poruku, (offset), dužinu poruke, odredišni host i odredišni port.

Message	Length	Host	Server Port
---------	--------	------	-------------

DatagramPacket

- Konstruktori klase **DatagramPacket** (za predajnu stranu):

```
DatagramPacket(byte[] buf,  
               int length, InetAddress address, int port);
```

ovaj konstruktor se koristi za kreiranje datagram paketa koji šalje poruku (smeštenu u prvi argument) date dužine na IP adresu i dati broj porta.

- zadaje pomak u baferu gde će se smeštati podaci.

```
DatagramPacket (byte[] buf, int offset,  
               int length, InetAddress ipAdresa, int port)
```

Za prijemnu stranu:

```
DatagramPacket (byte[] buf, int length)  
DatagramPacket (byte[] buf, int offset, int length)
```

- Značajne metode klase **DatagramPacket** su:

<code>byte[]</code>	<code>getData()</code>	vraća bafer podataka.
<code>int</code>	<code>getOffset()</code>	vraća offset.
<code>int</code>	<code>getLength()</code>	vraća dužinu podataka koji se šalju ili dužinu prispelih podataka.
<code>InetAddress</code>	<code>getAddress()</code>	vraća odredišnu adresu.
<code>int</code>	<code>getPort()</code>	vraća broj priključka odredišta.
<code>void</code>	<code>setData(byte[] buf)</code>	postavlja podatke za tekući paket.
<code>void</code>	<code>setLength(int length)</code>	postavlja dužinu tekućeg paketa.

DatagramSocket

- Konstruktor klase **DatagramSocket**:
`DatagramSocket(int port);`
 - Navedeni konstruktor kreira datagram socket koji koristi dati port .
- Ključne metode klase **DatagramSocket** su:

<code>void send (DatagramPacket dp)</code>	šalje datagram paket na socket.
<code>void receive(DatagramPacket dp)</code>	prihvata datagram paket sa socket-a.
<code>InetAddress getAddress()</code>	vraća odredišnu adresu.
<code>InetAddress getLocalAddress()</code>	vraća lokalnu adresu.
<code>int getPort()</code>	vraća broj priključka odredišta.
<code>int getLocalPort()</code>	vraća broj lokalnog priključka.
- Sledi komunikacioni program koji koristi slanje datagrama.
 - Jednostavan UDP eho serverski program čeka zahtev klijenta.
 - Po prijemu zahteva server klijentu vraća prihvaćenu poruku (datagram).

```
//serverska strana
import java.net.*; import java.io.*;
public class UDPServer {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        if (args.length < 1) {
            System.out.println("Usage: java UDPServer <Port Number>");
            System.exit(1);
        }
        try {
            int socket_no = Integer.valueOf(args[0]).intValue();
            aSocket = new DatagramSocket(socket_no);
            byte[] buffer = new byte[1000];
            while (true) {
                System.out.println("cekam klijenta ...");
                DatagramPacket request =
                    new DatagramPacket(buffer, buffer.length);
                aSocket.receive(request);
                DatagramPacket reply =
                    new DatagramPacket(
                        request.getData(),    request.getLength(),
                        request.getAddress(), request.getPort());
            }
        }
    }
}
```

```
        System.out.println("saljem klijentu ...");
        aSocket.send(reply);
        System.out.println("POSLAO !");
    }
} catch (SocketException e) {
    System.out.println("Socket: " + e.getMessage());
} catch (IOException e) {
    System.out.println("IO: " + e.getMessage());
} finally {
    if (aSocket != null) { aSocket.close(); }
}
}
}

//----- klijentska strana -----
import java.net.*; import java.io.*;
public class UDPClient {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        if (args.length < 3) {
            System.out.println(
                "Usage: java UDPClient <message > <Host name > <Port number >");
            System.exit(1);
        }
    }
}
```

```
try {
    aSocket = new DatagramSocket();
    byte[] m = args[0].getBytes();
    InetAddress aHost = InetAddress.getByName(args[1]);
    int serverPort = Integer.valueOf(args[2]).intValue();
    DatagramPacket request =
        new DatagramPacket(m, args[0].length(), aHost, serverPort);
    aSocket.send(request);
    System.out.println("poslao !");
    byte[] buffer = new byte[1000];
    DatagramPacket reply = new
        DatagramPacket(buffer, buffer.length);
    aSocket.receive(reply);
    System.out.println("Server vraca: " +
        new String(reply.getData()));
}
catch (SocketException e) {
    System.out.println("Socket: " + e.getMessage());
}
catch (IOException e) {
    System.out.println("IO: " + e.getMessage());
}
```

```
finally {  
    if (aSocket != null) {  
        aSocket.close();  
    }  
}  
}  
}
```

- Izlaz:

serverska strana java UDPServer 12345	klijentska strana java UDPClient "Pozdrav od klijenta!" 127.0.0.1 12345
cekam klijenta ...	
	poslao !
saljem klijentu ...	
POS LAO ! cekam na klijenta ...	Server vraca: Pozdrav od klijenta !

Klasa URL

- URL (Uniform Resource Locator) predstavlja jedinstveno ime dodeljeno svakom resursu na Internetu. Java ima podršku za URL adresni pristup resursima Interneta.
- Delovi URL adrese (`http://www.viser.edu.rs:80/index.html`) su kao što sledi:
 - protokol odvojen je dvotačkom od ostatka URL adrese.
 - računar IP adresa računara, levi graničnik je dvostruka kosa crta, a desni graničnik je kosa crta ili dvotačka ako se navodi port.
 - port neobavezan parametar. Levi graničnik je dvotačka a desni graničnik je kosa crta.
 - navigacija do fajla u datom primeru `index.htm`
- URL klasa može izazvati izuzetak `MalformedURLException`.
- Konstruktori URL klase su kao što sledi:
`URL(String urlString)`
`URL(String protokol, String racunar, int port, String putanja)`
`URL(String protokol, String racunar,                      String putanja)`

Klasa URL

- Sledi primer koji ispisuje svojstva stranice Osborne (http://www.osborne/download) :

```
import java.net.*;
class URLDemo {
    public static void main(String args[])
        throws MalformedURLException {
        URL hp = new URL("http://www.osborne/download");
        System.out.println("Protokol:\t" + hp.getProtocol());
        System.out.println("Racunar:\t" + hp.getHost());
        System.out.println("Prikljucak:\t" + hp.getPort());
        System.out.println("Datoteka:\t" + hp.getFile());
        System.out.println("URL:\t" + hp.toExternalForm());
    }
}
```

- Izlaz:

Protokol:	http
Racunar:	www.osborne
Prikljucak:	-1
Datoteka:	/download
URL:	http://www.osborne/download

Klasa URLConnection

- Klasa URLConnection (u java.net paketu) koristi se za pristup sadržaju udaljenih resursa. Ovim se mogu proveriti svojstva udaljenih resursa.
- Sledi primer koji koristi metodu openConnection klase URL da bi se kreirao objekat URLConnection. Program uspostavlja HTTP vezu sa lokacijom `http://www.w3.org/`, lista vrednosti zaglavlja i učitava sadržaj.

```
import java.net.*;
import java.io.*;
import java.util.Date;

class URLConnectionPrimer {
    public static void main(String args[]) throws Exception {
        int c;
        URL hp = new URL("http://www.w3.org/");
        URLConnection hpCon = hp.openConnection();
        System.out.println("Datum: " + new Date(hpCon.getDate()));
        System.out.println("Vrsta sadržaja: " +
                           hpCon.getContentType());
        System.out.println("Rok trajanja: " +
                           hpCon.getExpiration());
        System.out.println("Vreme poslednje izmene: " +
                           new Date(hpCon.getLastModified()));
        int len = hpCon.getContentLength();
        System.out.println("Dužina sadržaja: " + len);
    }
}
```

Klasa URLConnection

```
if (len > 0) {
    System.out.println("Sadržaj:");
    InputStream input = hpCon.getInputStream();
    int i = len; //ako bi trebalo za nesto drugo
    while (((c = input.read()) != -1) && (--i > 0)) {
        System.out.print((char) c);
    }
    input.close();
}
else {
    System.out.println("Nema dostupnih podataka"); }
}
```

Datum: Sun Dec 20 14:16:29 CET 2015
Vrsta sadržaja: text/html; charset=utf-8
Rok trajanja: 1450617989000
Vreme poslednje izmene: Sat Dec 19 10:20:13 CET 2015
Dužina sadržaja: 37845
Sadržaj:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<!-- Generated from data/head-home.php, ../../smarty/{head.tpl} -->
<head> ...

Klasa URLConnection

- Klasa URLConnection poseduje getInputStream i getOutputStream metode slične metodama getInputStream i getOutputStream klase Socket.
- Sledeći primer demonstrira korišćenje klase URLConnetion i URLEncoder za slanje upita na pretraživač Yahoo. Program kreira kodirani upit koji će koristiti web aplikacija a onda šalje i prima podatke u sprezi sa Yahoo pretraživačem.
- Potrebna je klasa za URL kodiranje ne-ASCII karaktera.

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
public class QueryStringFormatter {
    private String queryEngine;
    private StringBuilder query = new StringBuilder();
    public QueryStringFormatter(String queryEngine) {
        this.queryEngine = queryEngine;
    }
    public String getEngine() { return this.queryEngine; }
    public void addQuery(String queryKey, String queryValue)
        throws Exception {
        query.append(queryKey + "="
            + URLEncoder.encode(queryValue, "UTF-8") + "&");
    }
    public String getQueryString() { return "?" + query.toString(); } }
```

Klasa URLConnection

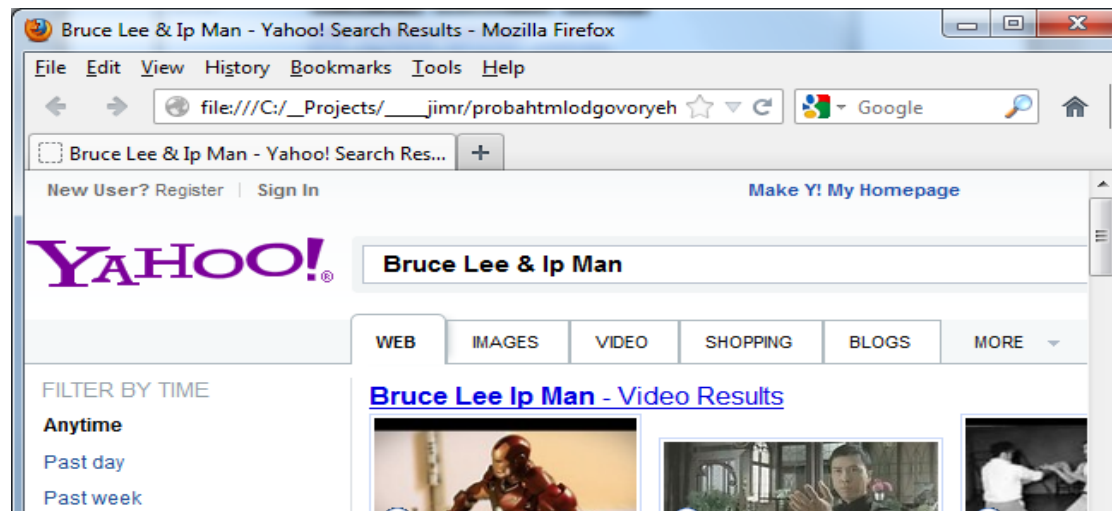
```
import java.io.*; import java.net.*;
public class UpitZaYahoo {
    private String searchEngine;
    public UpitZaYahoo(String searchEngine) {this.searchEngine = searchEngine;}
    public void doSearch(String queryString) {
        try {
            // otvaranje url konekcije
            URL url = new URL(searchEngine);
            URLConnection connection = url.openConnection();
            connection.setDoOutput(true);
            // slanje upita pretrazivacu
            PrintStream ps = new PrintStream(connection.getOutputStream());
            ps.println(queryString);
            ps.close();
            // citanje i ispisivanje rezultata
            DataInputStream input =
                new DataInputStream(connection.getInputStream());
            BufferedReader lines = new BufferedReader(new InputStreamReader(input,
                                                                    "UTF-8"));

            String inputLine = null;
            while ((inputLine = lines.readLine()) != null) {
                System.out.println(inputLine);
            } } catch (Exception e) { e.printStackTrace(); } }
```

Klasa URLConnection

```
public static void main(String[] args) throws Exception {  
    QueryStringFormatter formatter =  
        new QueryStringFormatter("http://search.yahoo.com/search");  
    formatter.addQuery("newwindow", "1");  
    formatter.addQuery("q", "Bruce Lee & Ip Man");  
    // pretrazivanje pomocu yahoo-a  
    UpitZaYahoo search = new UpitZaYahoo(formatter.getEngine());  
    search.doSearch(formatter.getQueryString());  
}  
}
```

- Odgovor sadrži kompletnu html stranicu. Ako se snimi kao html i startuje dobija se:



Generici

- Posmatra se primer:

```
List list = new ArrayList();  
list.add(new Integer(2));  
list.add("a String");
```

sada je uzimanje podataka sa kastovanjem:

```
Integer integer = (Integer) list.get(0);  
String string = (String) list.get(1);
```

- Java generici omogućuju da se postavi tip koji kolekcija prihvata čime dalje nije potrebno kastovanje uzete vrednosti iz kolekcije:

```
List<String> strings = new ArrayList<String>();  
//ili diamond operator List<String> strings = new ArrayList<>();  
strings.add("a String"); ...  
//nema kastovanja  
String aString = strings.get(0);  
for(String aString : strings){  
    System.out.println(aString);  
}  
Iterator<String> iterator = strings.iterator();  
while(iterator.hasNext()){  
    String aString = iterator.next();  
}
```

Generički skupovi, generičke mape

- Generički skupovi:

```
Set<String> set = new HashSet<String>();
String string1 = "a string";
set.add(string1);...
Iterator<String> iterator = set.iterator();
while(iterator.hasNext()){ String aString = iterator.next(); }
for(String aString : set){ System.out.println(aString); }
```

- Generičke mape:

```
Map<Integer, String> map = new HashMap<Integer, String>();
Integer key1 = new Integer(123);
String value1 = "value 1";
map.put(key1, value1);
...
String value1_1 = map.get(key1);
Iterator<Integer> keyIterator = map.keySet().iterator();
while(keyIterator.hasNext()){
    Integer aKey = keyIterator.next();
    String aValue = map.get(aKey);
}
Iterator<String> valueIterator = map.values().iterator();
while(valueIterator.hasNext()){
    String aString = valueIterator.next();
}
```

Generičke mape, generičke klase

- Generičke mape:

```
Map<Integer, String> map = new HashMap<Integer, String>();  
//... add key, value pairs to the Map  
for(Integer aKey : map.keySet()) {  
    String aValue = map.get(aKey);  
    System.out.println("" + aKey + ":" + aValue);  
}  
for(String aValue : map.values()) {  
    System.out.println(aValue);  
}
```

- Generičke klase:

```
public class GenericFactory<T> {  
    Class theClass = null;  
    public GenericFactory(Class theClass) {  
        this.theClass = theClass;  
    }  
    public T createInstance()  
        throws IllegalAccessException, InstantiationException {  
        return (T) this.theClass.newInstance();  
    }  
}
```

Generički metodi

- Poziv bi bio:

```
GenericFactory<MyClass> factory =  
    new GenericFactory<MyClass>(MyClass.class);  
MyClass myClassInstance = factory.createInstance();
```

- Generički metodi:

```
public static <T> T addAndReturn(T element, Collection<T> collection){  
    collection.add(element);  
    return element;  
}
```

```
String stringElement = "stringElement";  
List<String> stringList = new ArrayList<String>();  
String theElement = addAndReturn(stringElement, stringList);  
Integer integerElement = new Integer(123);  
List<Integer> integerList = new ArrayList<Integer>();  
Integer theElement = addAndReturn(integerElement, integerList);  
String stringElement = "stringElement";  
List<Object> objectList = new ArrayList<Object>();  
Object theElement = addAndReturn(stringElement, objectList);
```

Generički metodi

- Kreiranje instance klase preko metode:

```
public static <T> T getInstance(Class<T> theClass)
    throws IllegalAccessException, InstantiationException {
    return theClass.newInstance();
}...
String string    = getInstance(String.class);
MyClass myClass = getInstance(MyClass.class);
```

- Korišćenje npr. kod komunikacije sa bazom podataka:

```
public static <T> T read(Class<T> theClass, String sql)
    throws IllegalAccessException, InstantiationException {
    //execute SQL.
    T obj = theClass.newInstance();
    //set properties via reflection.
    return obj;
}
```

pozivi bi bili npr:

```
Driver employee= read(Driver.class, "select * from drivers where id=1");
Vehicle vehicle= read(Vehicle.class,"select * from vehicles where id=1");
```

Kreiranje iterabilne klase

- Korišćenje generika za kreiranje kolekcije.

```
public class MyCollection<E> implements Iterable<E>{
    public Iterator<E> iterator() {
        return new MyIterator<E>();
    }
}

public class MyIterator <T> implements Iterator<T> {
    public boolean hasNext() {
        //implement...
    }
    public T next() {
        //implement...
    }
    public void remove() {
        //implement...
    }
}

public static void main(String[] args) {
    MyCollection<String> stringCollection = new MyCollection<String>();
    ...
    for(String string : stringCollection){ ... }
}
```

Nasleđivanje i generici

- Lista tipova nepoznatog tipa:

```
public class A { }  
public class B extends A { }  
public class C extends A { }  
    public static void processElements(List<?> elements){  
        for(Object o : elements){  
            System.out.println(o);  
        }  
    }...  
List<A> listA = new ArrayList<A>();  
processElements(listA);
```

- Lista instanci klase ili njene podklase

```
    public static void processElements(List<? extends A> elements){  
        for(A a : elements){  
            System.out.println(a.getValue());  
        }  
    }...  
List<A> listA = new ArrayList<A>();  
processElements(listA);  
List<B> listB = new ArrayList<B>();  
processElements(listB);  
List<C> listC = new ArrayList<C>();  
processElements(listC);
```

Nasleđivanje i generici

- Lista instanci klase ili superklase:

```
public static void insertElements(List<? super A> list){  
    list.add(new A());  
    list.add(new B());  
    list.add(new C());  
}
```

```
List<A> listA = new ArrayList<A>();  
insertElements(listA);  
List<Object> listObject = new ArrayList<Object>();  
insertElements(listObject);
```

sada je kastovanje na Object:

```
Object object = list.get(0);
```

Primer generičke metode

```
public class GenericMethodTest {  
    public static <E> void printArray(E[] inputArray) {  
        for (E element : inputArray) { System.out.printf("%s ", element); }  
        System.out.println();  
    }  
    public static void main(String args[]) {  
        Integer[] intArray = { 1, 2, 3, 4, 5 };  
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };  
        Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };  
        System.out.println("Array integerArray contains:");  
        printArray(intArray); // pass an Integer array  
        System.out.println("\nArray doubleArray contains:");  
        printArray(doubleArray); // pass a Double array  
        System.out.println("\nArray characterArray contains:");  
        printArray(charArray); // pass a Character array  
    }  
}
```

```
Array integerArray contains:  
1 2 3 4 5  
Array doubleArray contains:  
1.1 2.2 3.3 4.4  
Array characterArray contains:  
H E L L O
```

Primer generičke metode

```
public class MaximumTest{
    // najveći od tri
    public static <T extends Comparable<T>> T maximum(T x, T y, T z) {
        T max = x;
        if ( y.compareTo( max ) > 0 ){ max = y; }
        if ( z.compareTo( max ) > 0 ){ max = z; }
        return max;
    }
    public static void main( String args[] ) {
        System.out.printf( "Max of %d, %d and %d is %d\n\n",
                           3, 4, 5, maximum( 3, 4, 5 ) );
        System.out.printf( "Maxm of %.1f,%.1f and %.1f is %.1f\n\n",
                           6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ) );
        System.out.printf( "Max of %s, %s and %s is %s\n", "pear",
                           "apple", "orange", maximum( "pear", "apple", "orange" ) );
    }
}
```

Max of 3, 4 and 5 is 5

Maxm of 6.6,8.8 and 7.7 is 8.8

Max of pear, apple and orange is pear

Primer generičke klase

```
public class Box<T> {  
    private T t;  
    public void add(T t) {  
        this.t = t;  
    }  
    public T get() {  
        return t;  
    }  
    public static void main(String[] args) {  
        Box<Integer> integerBox = new Box<Integer>();  
        Box<String> stringBox = new Box<String>();  
        integerBox.add(new Integer(10));  
        stringBox.add(new String("Hello World"));  
        System.out.printf("Integer Value :%d\n\n", integerBox.get());  
        System.out.printf("String Value :%s\n", stringBox.get());  
    }  
}
```

Integer Value :10

String Value :Hello World