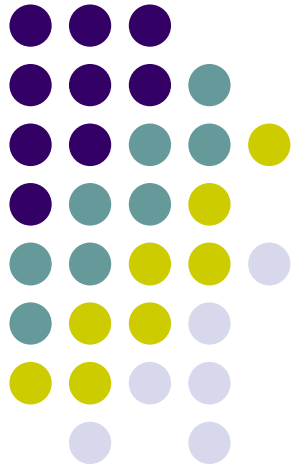
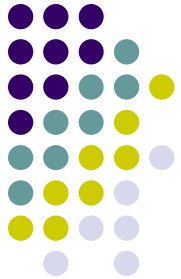
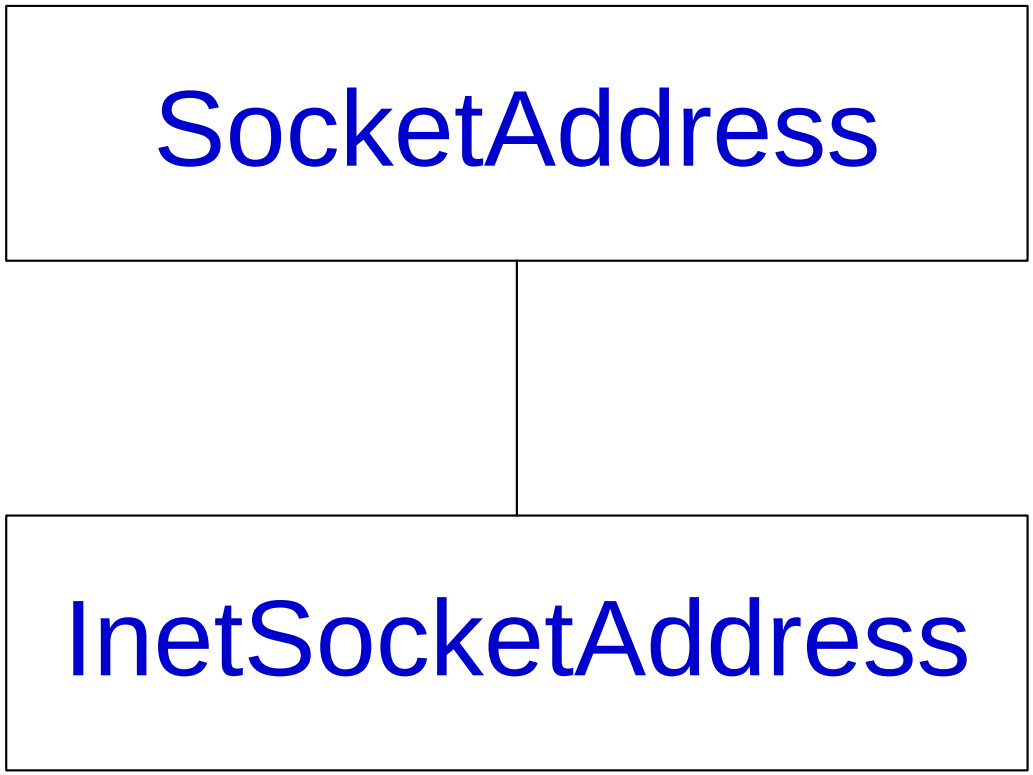
Мрежово програмиране

Използване на сокети в Java



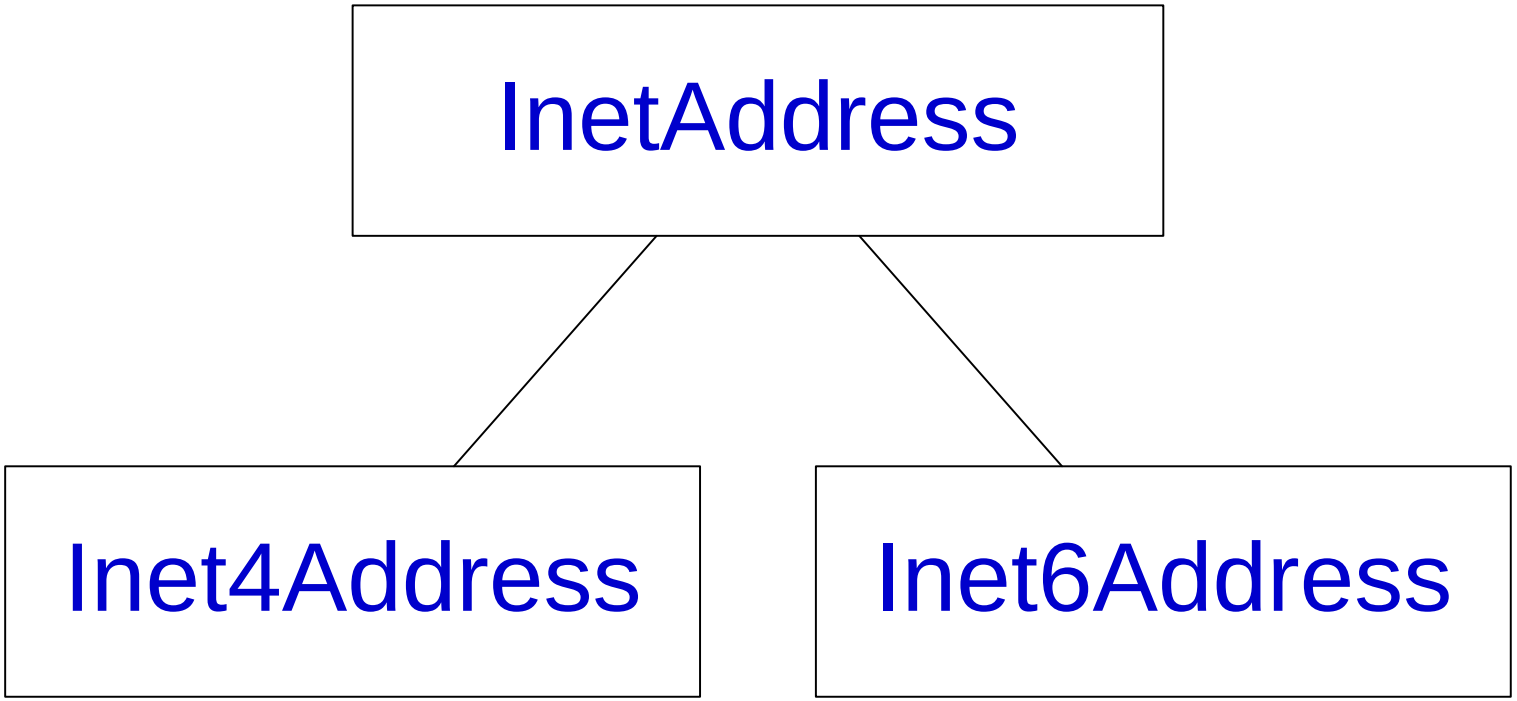


Адрес на сокет

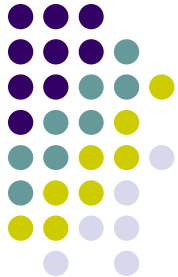


Интернет адрес

Класът **InetAddress** се използва за работа с числов IP адрес или доменно име. Поддържат се адресите на IPv4 и IPv6



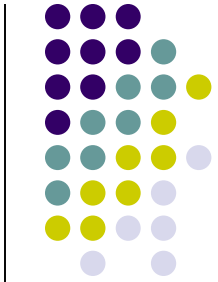
Съществуват два подкласа - **Inet4Address** и **Inet6Address**. В повечето случаи се използва класа **InetAddress**, понеже е приспособен и за двата стила.



Класът InetAddress позволява да се определят имената на хостовете и свързания с тях IP адрес, необходими по време на създаването на мрежовите приложения. Обектите InetAddress се инициализират с използване на статични методи, обявени в класа InetAddress. Основните методи, използвани за инициализация на обектите InetAddress, са:

- public static InetAddress getLocalHost(): Връща обекта InetAddress, който съдържа IP адреса на локалния компютър.
- public static InetAddress getByName(String host): Връща обекта InetAddress, който съдържа IP адреса, съответстващ на името на хоста, предадено на метода като String.
- public static InetAddress[] getAllByName(String host): Връща масив от обекти InetAddress, който съдържа IP адресите, съответстващи на името на хоста, предадени на метода като String.

Статичните методи getLocalHost(), getByName(), getAllByName() и getByAddress() могат да извикат изключителната ситуация UnknownHostException.



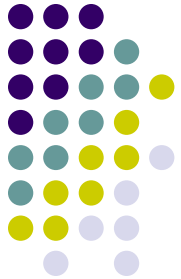
Нестатичните методи, определени в класа InetAddress, могат да бъдат достъпни след инициализацията на обектите InetAddress. Нестатичните методи са определени в класа InetAddress и такива са:

- public boolean equals(Object obj): Връща true, ако обектът InetAddress има същия IP адрес като обекта obj, предаден като параметър.
- public byte[] getAddress(): Връща IP адреса на обекта InetAddress във вид на масив byte.
- public String getHostAddress(): Връща IP адреса на обекта InetAddress като String.
- public String toString(): Връща IP адреса на хоста като string във формат име на хост/IP адрес.

TCP сокети

- Класове
  - Socket – представлява TCP съединението, създава TCP съединение на клиентската страна
  - ServerSocket – създава TCP съединения на сървърната страна

Езикът Java опростява мрежовото програмиране, чрез инкапсулация на функционалността на съединение на TCP сокет в класове, в които класът Socket е предназначен за създаване на клиентския сокет, а класът ServerSocket за създаване на сървърния сокет.



Класът Socket осигурява методи за потоков вход/изход, осигурява изпълнението на операциите четене и запис в сокет.
За създаването на обектите от класа Socket се използват следващите конструктори, определени в класа Socket:

- public Socket (InetAddress IP_address, int port): Създава обект Socket, който се съединява с хоста, зададен от параметрите IP_address и port.
- public Socket (String hostname, int port): Създава обект Socket, който се съединява с хоста, зададен от параметрите име на хост или IP адрес и port, на който сървърът "слуша".

Най-често използваните методи на класа Socket са:

- public InetAddress getInetAddress(): Връща обекта InetAddress, който съдържа IP адреса, с който се свързва обекта Socket.
- public InputStream getInputStream(): Връща входящия поток за обекта Socket.
- public InetAddress getLocalAddress(): Връща обекта InetAddress, съдържащ локалния адрес, с който се свързва обекта Socket.
- public int getPort(): Връща отдалечения порт, с който се свързва обекта Socket.
- public int getLocalPort(): Връща локалния порт, с който се свързва обекта Socket.
- public OutputStream getOutputStream(): Връща изходящия поток на обекта Socket.
- void close(): Затваря обекта Socket.
- public String toString(): Връща IP адрес и номер на порт на сокета на клиента като String.

Конструкторите и метода close() на класа Socket в случай на грешка генерират IOException, което трябва да се проверява и обработва.

ServerSocket е клас, използван от програмите на сървъра за слушане на заявките на клиентите. ServerSocket реално не изпълнява обслужване, но създава обект Socket от името на клиента, чрез който се реализира взаимодействието със сокета на клиента.

За създаването и инициализацията на обектите на ServerSocket се използват следващите конструктори, определени в класа ServerSocket:

- public ServerSocket(int port_number): Създава сокет на сървъра на зададен порт на локалната машина. Клиентите ще използват този порт, за да общуват със сървъра. Ако номерът на порта е 0, тогава сокетът на сървъра се създава на кой да е свободен порт на локалната машина.
- public ServerSocket(int port, int backlog): Създава сокет на сървър на зададен порт на локалната машина. Вторият параметър задава максималния брой съединения за клиентите, които сокетът на сървъра поддържа на зададения порт.
- public ServerSocket(int port, int backlog, InetAddress bindAddr): Създава сокет на сървъра на зададен порт. Третият параметър се използва за създаване на сокет на сървър за хост, свързан с няколко физически линии (multi-homed host). Сокетът на сървъра приема заявки от клиенти само от зададени IP адреси.

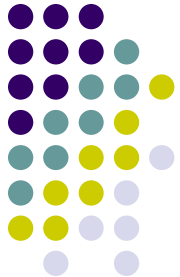
Често използвани методи на класа ServerSocket:

- public InetAddress getInetAddress(): Връща обекта InetAddress, който съдържа адреса на обекта ServerSocket.
- public int getLocalPort(): Връща номера на порта, на който обекта ServerSocket слуша за заявки от клиенти.
- public Socket accept() throws IOException: Принуждава сокета на сървъра да слуша за лог. съединение на клиента и да го приеме. След установяването на лог. съединение на клиента със сървъра методът връща сокета на клиента.
- public void bind(SocketAddress address) throws IOException: Свързва обекта ServerSocket със зададения адрес (IP адрес и порт). В случай на грешка методът извиква изключителната ситуация IOException.
- public void close() throws IOException: Затваря обекта ServerSocket. В случай на грешка методът извиква изключителната ситуация IOException.
- public String toString(): Връща IP адреса и номера на порта на сокета на сървъра като String.

UDP сокети

- Класове (пакет java.net)
  - DatagramPacket – за създаване и получаване на UDP пакет (контейнер с данни)
  - DatagramSocket – създава UDP сокет за получаване и изпращане на UDP пакети (използвани механизми при предаване и получаване на обекти DatagramPacket)

Самата дейтаграма е обект от класа DatagramPacket и използва група методи на класа DatagramSocket

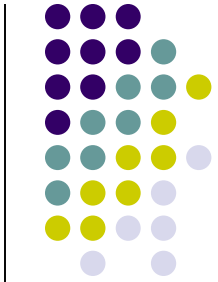


Обектът DatagramPacket е контейнер с данни, състоящ се от дейтаграмни пакети, които се изпращат или приемат по мрежата. Следващите конструктори се използват за инициализация на обектите DatagramPacket:

- public DatagramPacket(byte[] buffer, int buffer_length): Създава обекта DatagramPacket, който приема и съхранява данните в масива byte. Дължината на буфера на масива byte се задава от втория параметър buffer_length.
- public DatagramPacket(byte[] buffer, int buffer_length, InetAddress address, int port): Създава се обект DatagramPacket, който изпраща пакети с данни със зададена дължина. Пакетите с данни се изпращат на компютър със зададен IP адрес и номер на порт, предавани като параметри.

Следващите методи, определени в класа DatagramPacket, могат да се използват след инициализация на обект от класа DatagramPacket:

- public InetAddress getAddress(): Връща обекта InetAddress, който съдържа IP адреса на компютъра, на който се изпраща дейтаграмния пакет или от който се приема дейтаграмния пакет.
- public byte[] getData(): Връща буферен масив byte, който съдържа данните.
- public int getLength(): Връща дължината на буферния масив, който съдържа данните.
- public int getPort(): Връща номера на порта на компютъра, на който се изпраща дейтаграмния пакет или откъдето дейтаграмния пакет се получава.
- public void setAddress(InetAddress address): Установява IP адреса на машината, на която дейтаграмния пакет трябва да се изпрати.
- public void setData(byte[] buffer): Установява масива byte в качеството на данни за пакета.
- public void setPort(int port): Установява номера на порта на отдалечения хост.
- public void setLength(int length): Установява дължината на буфера за обмен.

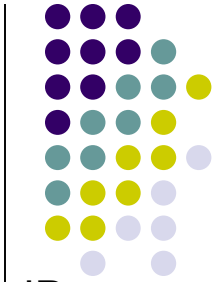


Класът DatagramSocket съдържа функционалността за управлението на обектите DatagramPacket. Обектите DatagramPacket изпращат и получават съхранените данни, като използват обекта DatagramSocket. Следващите конструктори се използват за инициализация на обекта DatagramSocket:

- public DatagramSocket(): Създава обект DatagramSocket и го свързва с някой достъпен порт на локалния компютър.
- public DatagramSocket(int port): Създава обект и го свързва с порт на локалния хост, зададен като параметър.
- public DatagramSocket(int port, InetAddress address): Създава обект и го свързва с порт на зададения хост.

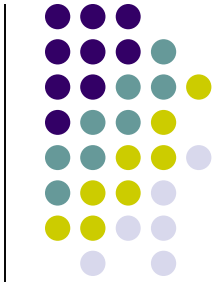
Конструкторът на класа DatagramSocket може да извика изключителната ситуация SocketException

Методите на класа DatagramSocket, които се използват за получаване на информация от обекта DatagramSocket, са:



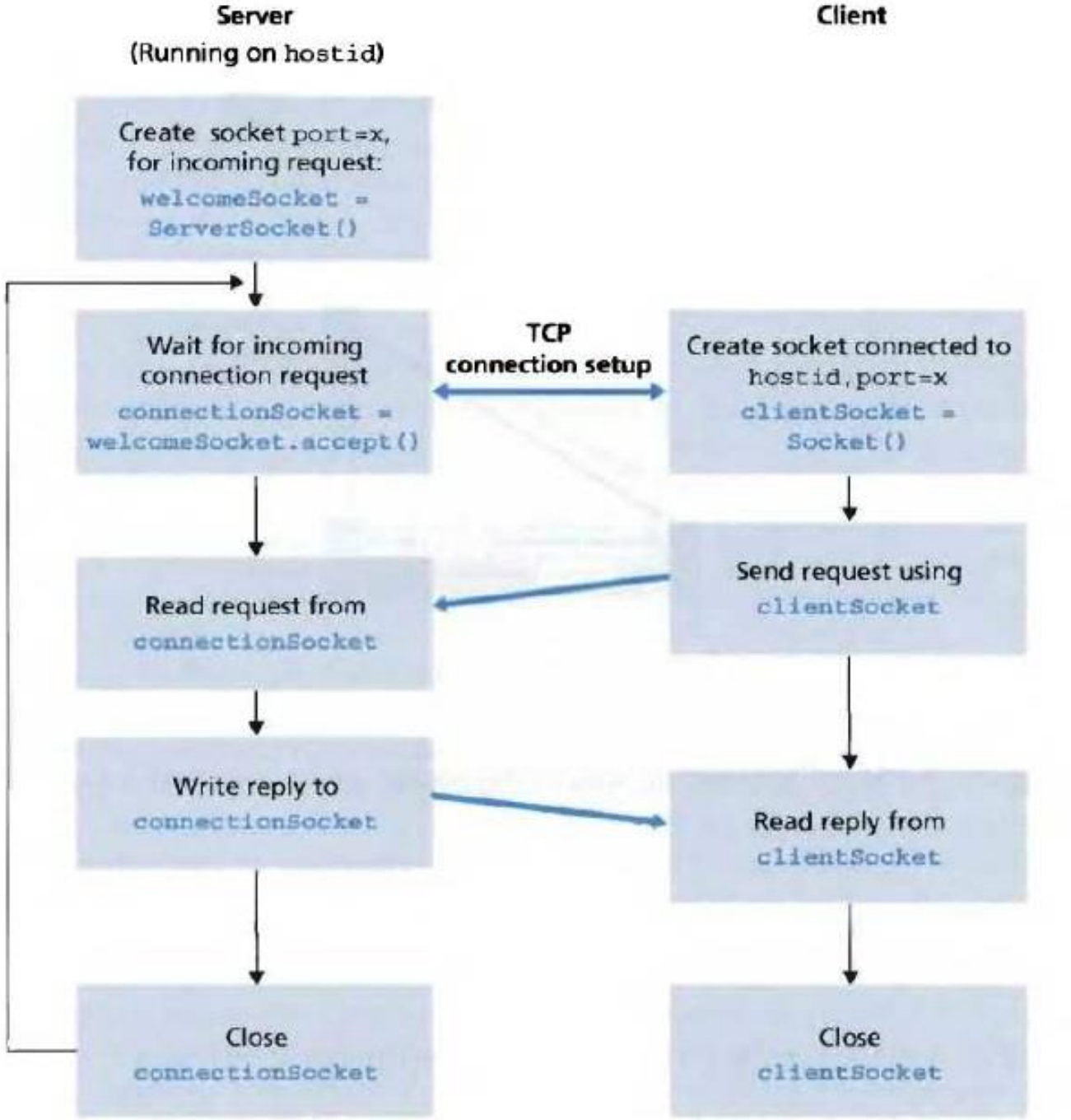
- public InetAddress getInetAddress(): Връща обекта InetAddress, съдържащ IP адрес, с който обекта DatagramSocket се свързва.
- public InetAddress getLocalAddress(): Връща обекта InetAddress, който съдържа IP адреса на локалния хост, с който обекта DatagramSocket се свързва.
- public int getLocalPort(): Връща целочислена стойност, която е номера на порта на локалния хост, с който обекта DatagramSocket се свързва.
- public void bind(SocketAddress address): Свързва обекта DatagramSocket с обекта SocketAddress.
- public void close(): Затваря обекта DatagramSocket.
- public void connect(InetAddress address, int port): Съединява обекта DatagramSocket със зададен IP адрес и порт.
- public void disconnect(): Затваря съединението на обекта DatagramSocket.
- public boolean isBound(): Връща true, ако обектът DatagramSocket е свързан с порт.
- public boolean isClosed(): Връща true, ако обектът DatagramSocket се затваря.
- public boolean isConnected(): Връща true, когато обектът DatagramSocket се съединява с IP адреса.
- public void receive(DatagramPacket packet): Получава дейтаграмния пакет от текущия обект DatagramSocket.
- public void send(DatagramPacket packet): Предава дейтаграмния пакет от текущия обект DatagramSocket.

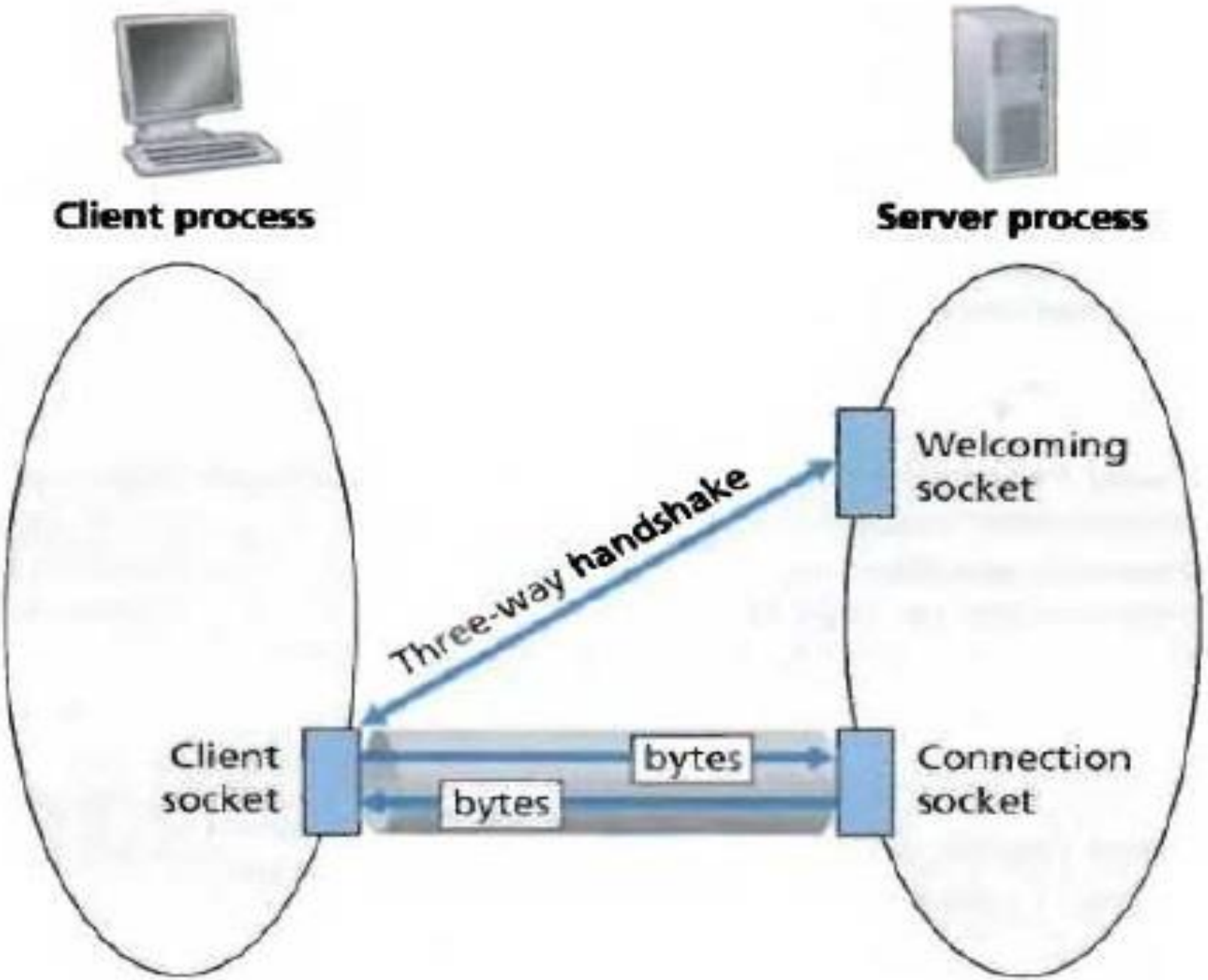
Ще разгледаме пример за приложение клиент-сървър на езика Java

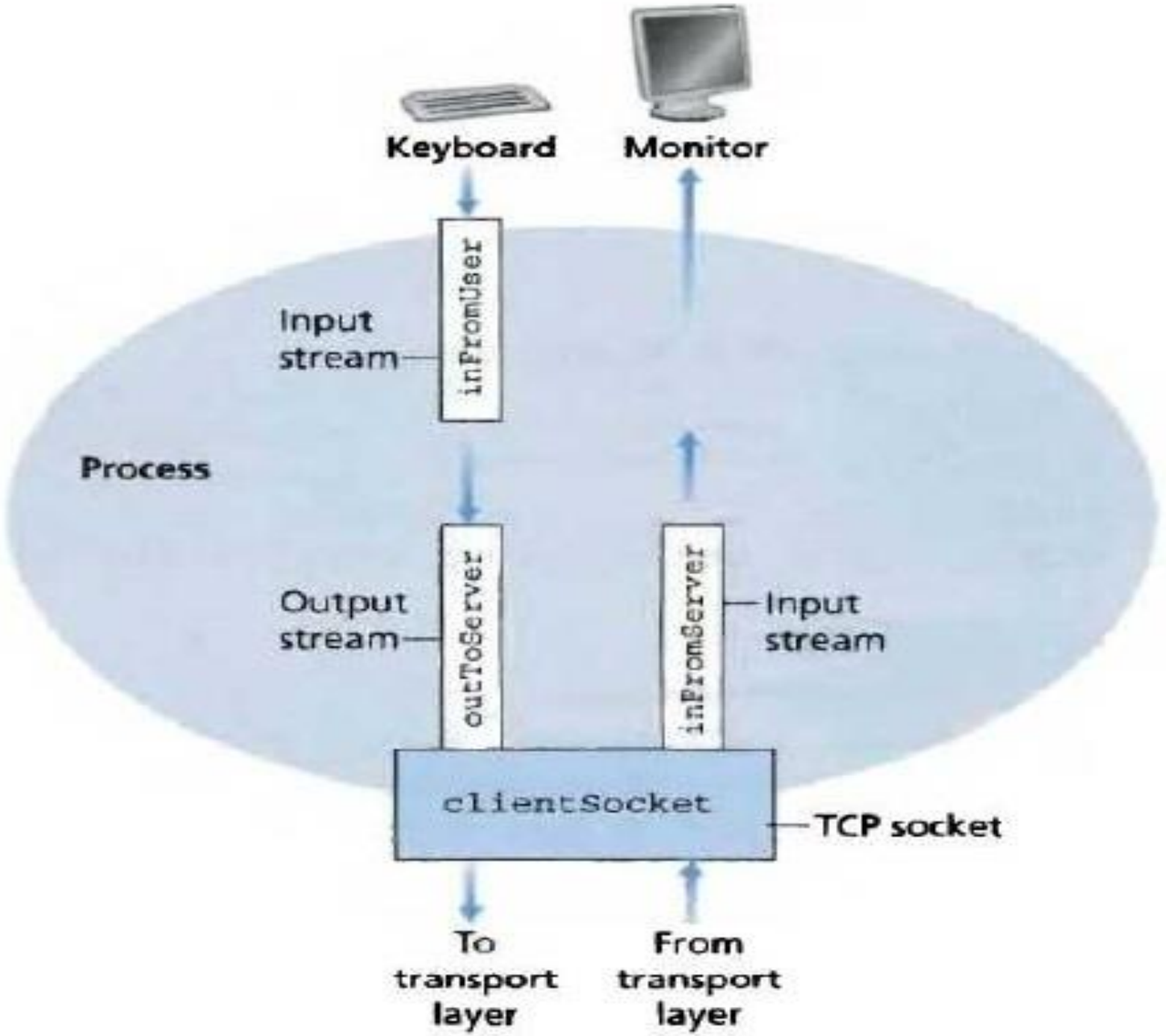


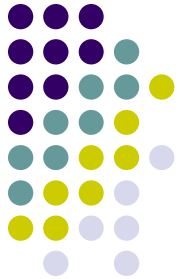
Приложението функционира по следния алгоритъм:

1. Клиентът прочита от стандартното устройство за въвеждане (клавиатура) низ от символи и изпраща този низ през свой сокет.
2. Сървърът приема низа през свой сокет.
3. Сървърът преобразува всички символи на низа в горен регистър (в главни).
4. Сървърът изпраща модифицирания низ на клиента.
5. Клиентът получава низа и го отпечатва на стандартното устройство за извеждане (дисплея).


Server
(Running on hostid)
Client
Create socket port=x, for incoming request:
welcomeSocket = ServerSocket()
Wait for incoming connection request
connectionSocket = welcomeSocket.accept()
TCP connection setup
Create socket connected to hostid, port=x
clientSocket = Socket()
Send request using clientSocket
Read request from connectionSocket
Write reply to connectionSocket
Read reply from clientSocket
Close connectionSocket
Close clientSocket

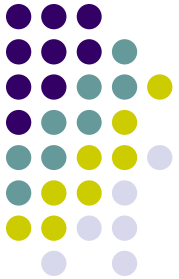

Client process
Server process
Welcoming socket
Three-way handshake
Client socket
bytes
bytes
Connection socket


Keyboard
Monitor
Input stream
inFromUser
Process
Output stream
outToServer
inFromServer
Input stream
clientSocket
TCP socket
To transport layer
From transport layer

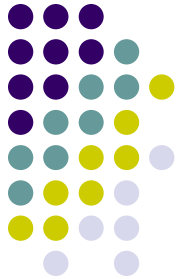


Първата стъпка при клиента се състои в установяването на логическо съединение между клиента и сървъра. За установяването на това съединение между клиента и сървъра е необходимо да се създаде обект Socket. За създаването на клиентското приложение за TCP сокет е необходимо да се изпълнят следните задачи:
1. Да се създаде сокет на клиента, като се използва обекта Socket.
2. Да се чете и записва в сокета.
3. Да се затвори съединението.

- Текст на програмата TCPClient.java:

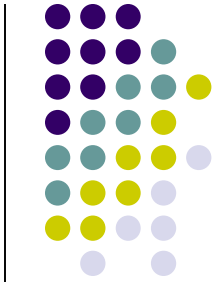


```
import java.io.*;
import java.net.*;
        class TCPClient {
public static void main(String argv[]) throws Exception
        {
String sentence;
String modifiedSentence;
BufferedReader inFromUser =
     new BufferedReader(new InputStreamReader(System.in));
Socket clientSocket = new Socket ("hostname", 6789);
DataOutputStream outToServer =
      new DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer = new BufferedReader(
      new InputStreamReader(clientSoket.getInputStream()));
sentence = inFromUser.readLine();
outToServer.writeBytes(sentence + '\n');
modifiedSentence = inFromServer.readLine();
System.out.println("FROM SERVER: " + modifiedSentence);
clientSocket.close();
}
}
```

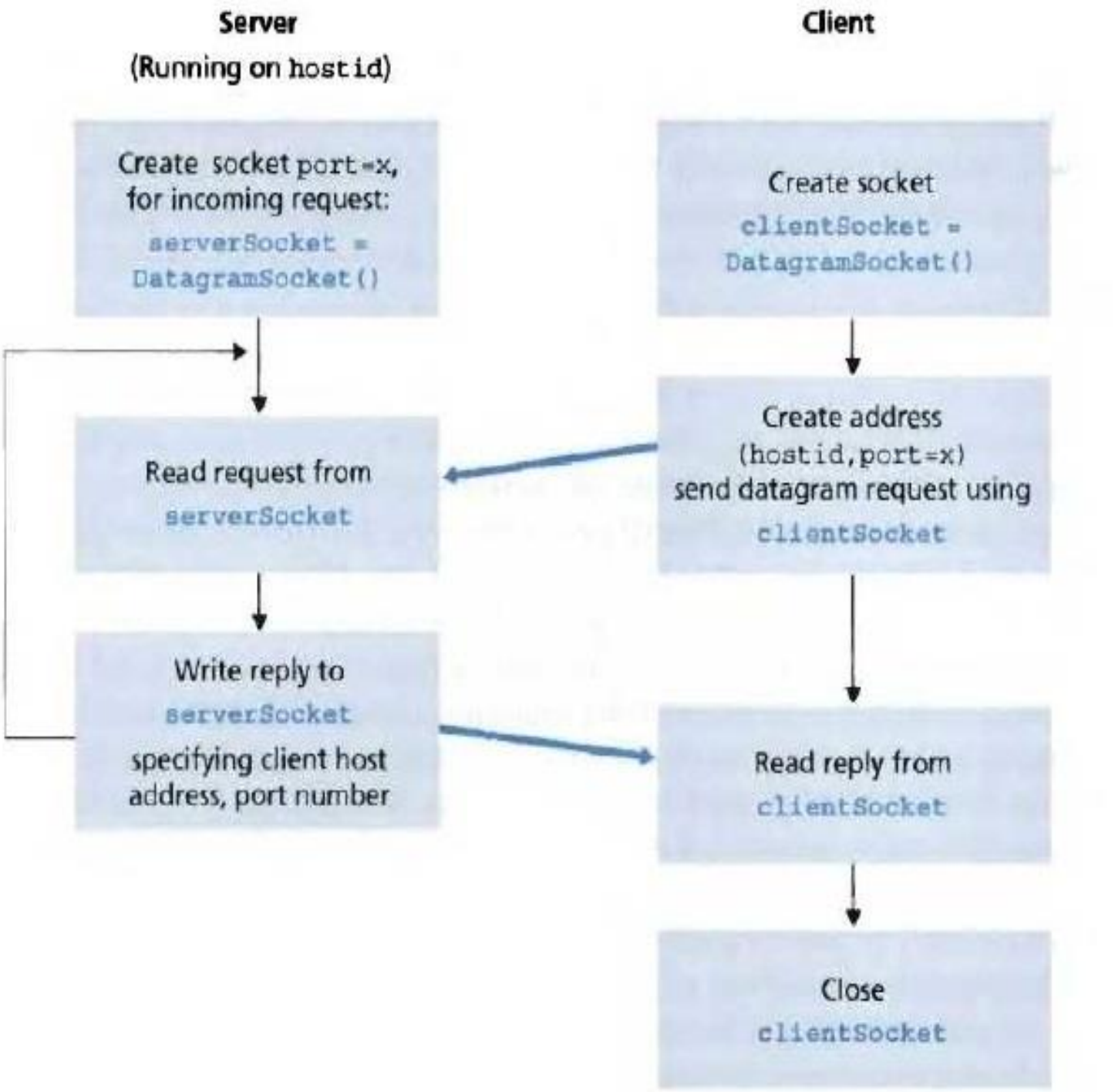


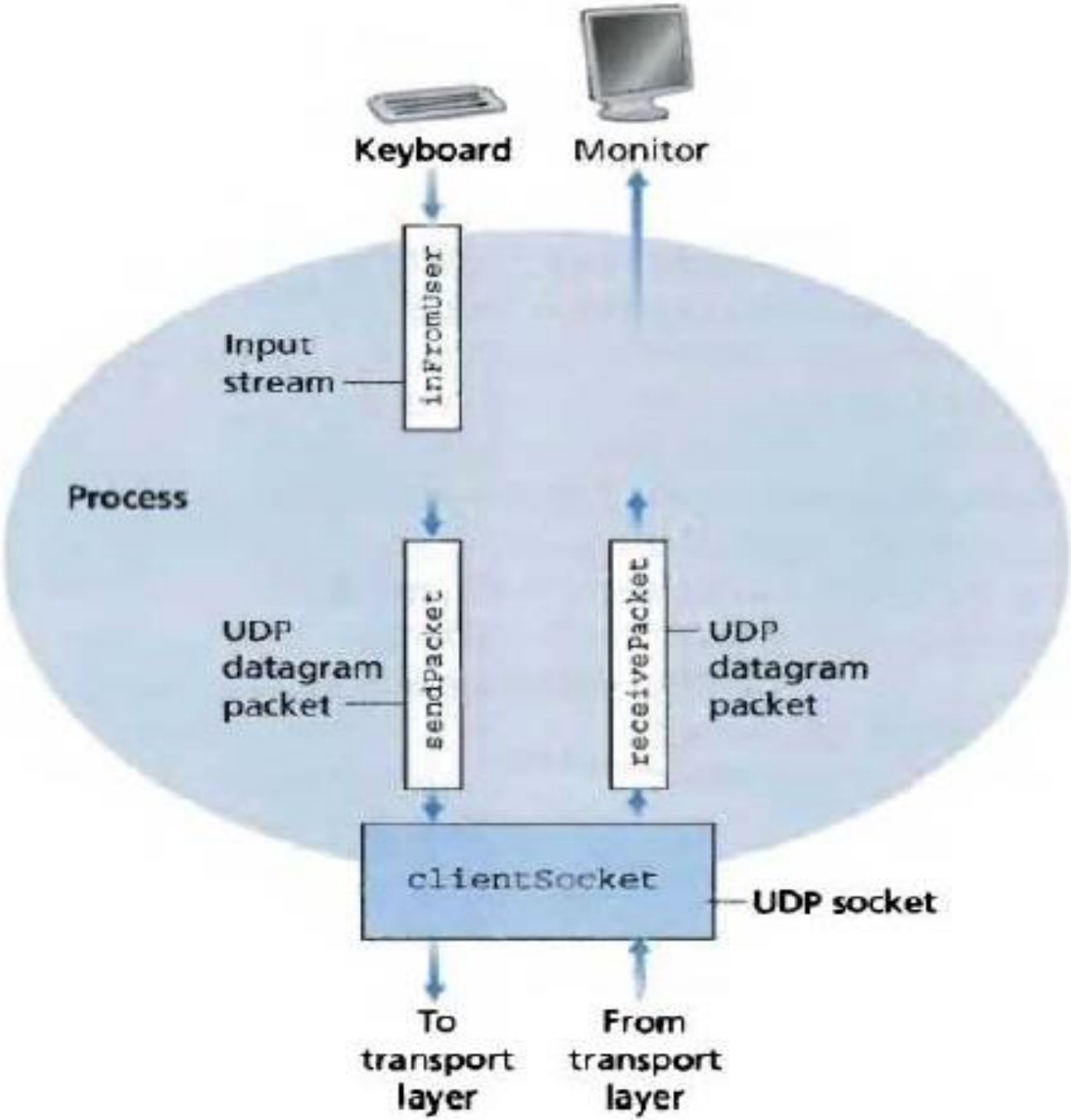
За създаването на сървърното приложение за TCP сокет е необходимо да се изпълнят следващите стъпки:
• Да се създаде обект сокет на сървъра ServerSocket.
• Да се слуша за заявки от клиента за съединение.
• Да се създаде поток от съединения за заявките на клиентите и тяхното обслужване.

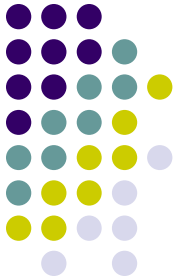
- Текст на програмата TCPServer.java:



```
import java.io.*;
import java.net.*;
class TCPServer {
   public static void main(String argv[]) throws Exception
    {
String clientSentence;
String capitalizedSentence;
ServerSocket welcomeSocket = new ServerSocket (6789);
while (true) {
   Socket connectionSocket = welcomeSocket.accept();
BufferedReader inFromClient = new BufferedReader(new
     InputStreamReader(connectionSocket.getInputStream()));
DataOutputStream outToClient =
   new DataOutputStream(connectionSocket.getOutputStream());
clientSentence = inFromClient.readLine();
capitalizedSentence = clientSentence.toUpperCase() + '\n';
outToClient.writeBytes(capitalizedSentence);
}
}
}
```


Server
(Running on hostid)
Client
Create socket port=x, for incoming request: serverSocket = DatagramSocket()
Create socket clientSocket = DatagramSocket()
Read request from serverSocket
Create address (hostid, port=x) send datagram request using clientSocket
Write reply to serverSocket specifying client host address, port number
Read reply from clientSocket
Close clientSocket

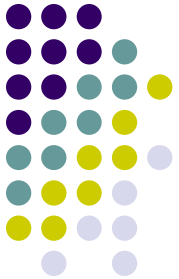

Keyboard
Monitor
inFromUser
Input stream
Process
sendPacket
UDP datagram packet
receivePacket
UDP datagram packet
clientSocket
UDP socket
To transport layer
From transport layer

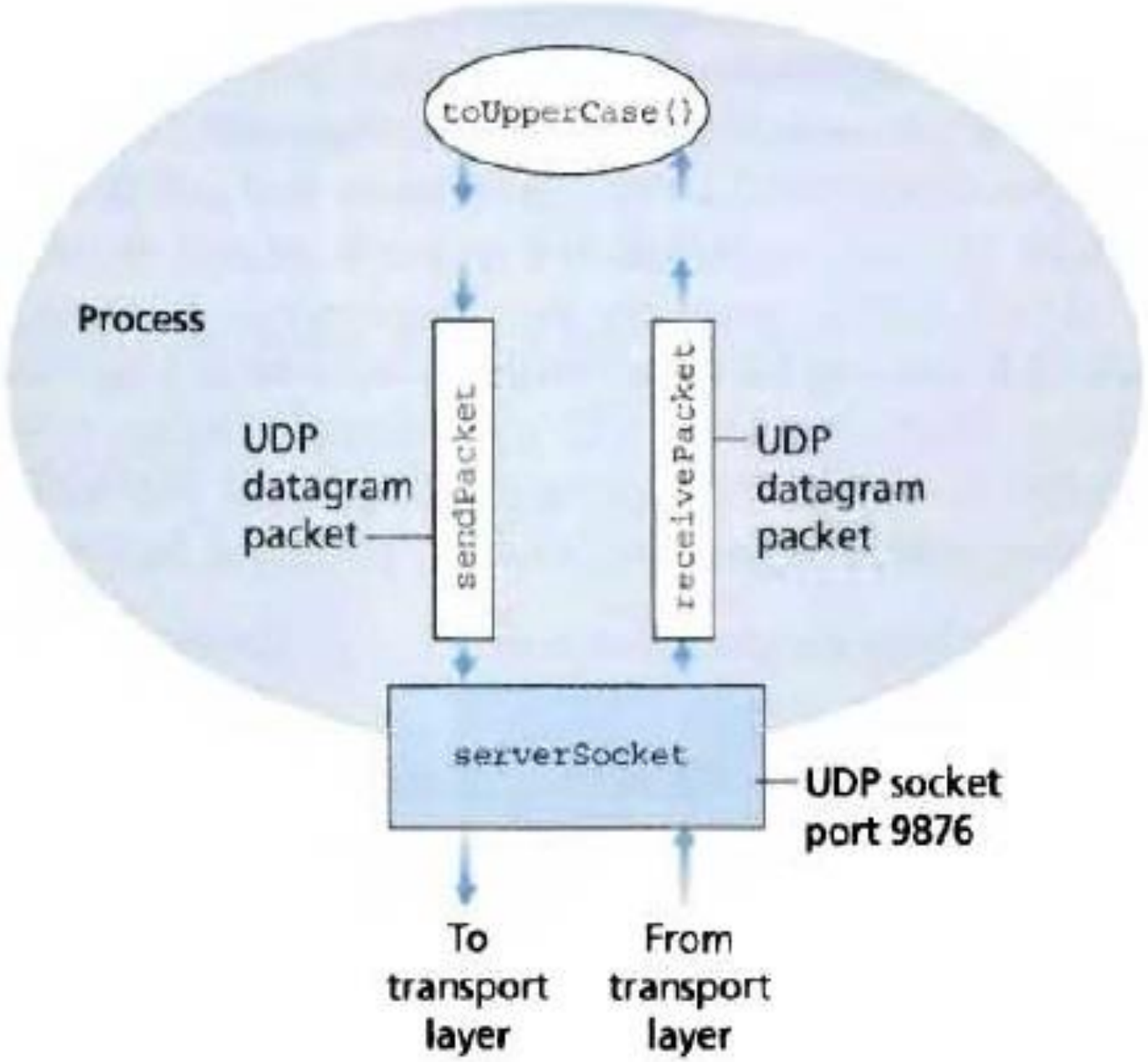


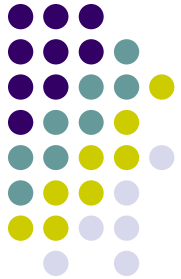
За създаването на UDP клиент се изпълняват следните стъпки:
1. Създава се сокет, използващ обект от класа DatagramSocket за създаване на съединение със сървъра.
2. Създава се обект от класа DatagramPacket и се използва метода send() за изпращане на съобщения към сървъра.
3. Създава се обект от класа DatagramPacket и се използва метода receive() за получаването на съобщенията, изпратени от сървъра.

- Текст на програмата UDPClient.java:

```
import java.io.*;
import java.net.*;
class UDPClient {
  public static void main(String args[]) throws Exception
  {
   BufferedReader inFromUser =
     new BufferedReader(new InputStreamReader (System.in));
   DatagramSocket clientSocket = new DatagramSocket();
   InetAddress IPAddress = InetAddress.getByName("hostname");
   byte[ ] sendData = new byte[1024];
   byte[ ] receiveData = new byte[1024];
   String sentence = inFromUser.readLine();
   sendData = sentence.getBytes();
   DatagramPacket sendPacket =
     new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
   clientSocket.send(sendPacket);
   DatagramPacket receivePacket =
     new DatagramPacket(receiveData, receiveData.length);
   clientSocket.receive(receivePacket);
   String modifiedSentence = new String(receivePacket.getData());
   System.out.println("FROM SERVER:" + modifiedSentence);
   clientSocket.close();
   }
}
```

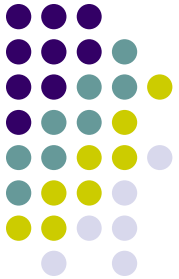



toUpperCase()
Process
UDP
datagram
packet
sendPacket
receivePacket
UDP
datagram
packet
serverSocket
UDP socket
port 9876
To
transport
layer
From
transport
layer

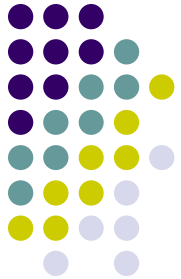


За създаването на UDP сървър се изпълняват следните стъпки:
1. Създава се сокет, като се използва обекта DatagramSocket.
2. Създава се обект от класа DatagramPacket, като се използва метода receive() за получаването на съобщенията от клиента.
3. Създава се обект от класа DatagramPacket и се използва метода send() за изпращане на съобщение на клиента.

- Текст на програмата UDPServer.java, представляваща сървърната част на приложението:



```
import java.io.*;
import java.net.*;
class UDPServer {
  public static void main(String args[]) throws Exception
     {
      DatagramSocket serverSocket = new DatagramSocket(9876);
       byte[ ] receiveData = new byte[1024];
       byte[ ] sendData = new byte[1024];
    while(true)
      {
      DatagramPacket receivePacket =
         new DatagramPacket(receiveData, receiveData.length);
      serverSocket.receive(receivePacket);
      String sentence = new String(receivePacket.getData());
      InetAddress IPAddress = receivePacket.getAddress();
      int port = receivePacket.getPort();
      String capitalizedSentence = sentence.toUpperCase();
      sendData = capitalizedSentence.getBytes();
      DatagramPacket sendPacket =
         new DatagramPacket(sendData, sendData.length, IPAddress, port);
      serverSocket.send(sendPacket);
     }
   }
}
```

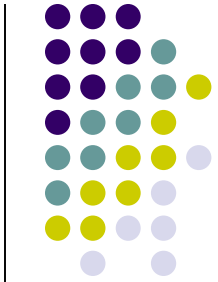


- С TCP протокол се работи по-лесно, отколкото с UDP в езика Java;
- Каналите за потоково предаване добре съответстват на възможностите на Java;
- Java скрива детайлите на мрежовото взаимодействие и опростява мрежовите интерфейси.

API

- набор от готови класове, функции, структури и константи, предоставяни на приложението (библиотеки, услуги) за използване от външни програмни продукти
- използват се от програмистите за създаване на най-различни приложения

Причини за успеха на сокетите: *простота и универсалност*



- След като се напише библиотеката със сокетите, тогава могат да се скрият всички сложности зад набор от прости интерфейси;
- След като се напише веднъж класа, реализиращ базовите функции за работа със сокети, тогава вече можем да преминем към решаването на алгоритмичните задачи.