



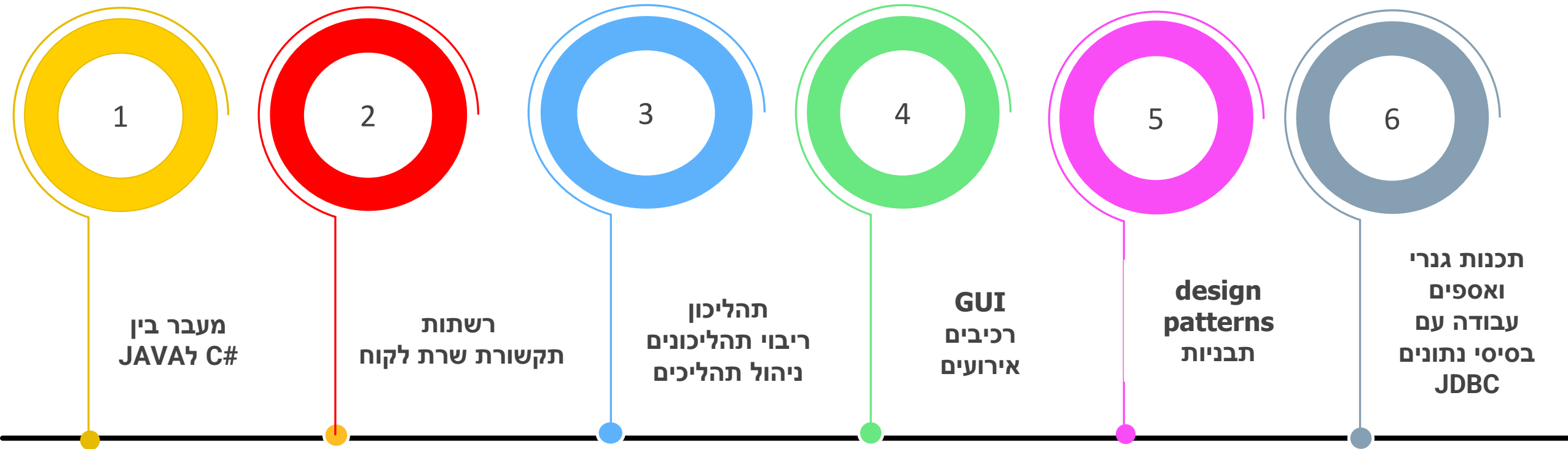
# נושאים מתקדמים בתכנות מונחה עצמים ושפת JAVA

---

ד"ר רינה צביאל-גירשין

יחידה 2

# מפת הקורס



# מבט על על ההרצאה



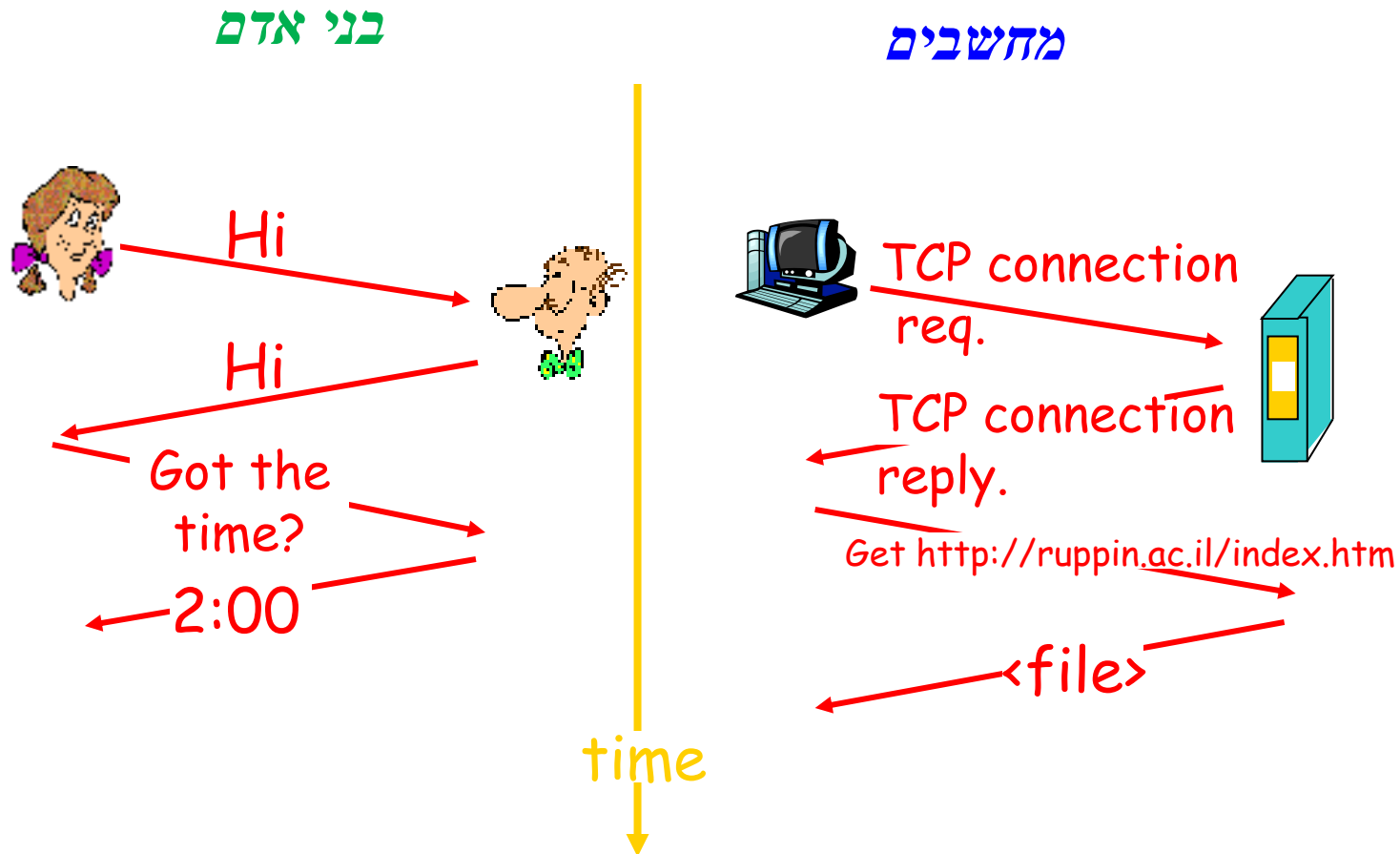
- מושגים מתחום תקשורת
- מודל שרת לקוח
- תקשורת TCP
- לקוח
- שרת
- תקשורת UDP

# תקשורת



- **תקשורת** communication היא תהליך של העברת מידע בין שנים (או יותר) צדדים .
- כל אחד מהצדדים יכול להיות
  - שולח/משדר Sender או
  - מקבל/קולט Receiver
- תהליך התקשורת כולל שליחת מסרים/הודעות, קבלה, הבנה ותגובה.

# פרוטוקולי תקשורת



- בערוצי תקשורת דיגיטליים או מחשביים, התקשורת מתבצעת בין מחשבים או מכשירים אחרים, כאשר פרוטוקולים מסוימים מנתיבים את
- אופן ההעברה, האימות והתגובה להודעות שנשלחות ביניהם.
- פרוטוקול תקשורת – אוסף של כללים

# תקשורת אמינה או לא

- ישנם סוגי תקשורת איפה כל דבר שנשלח חייב להתקבל ללא שגיאות

- תקשורת אמינה

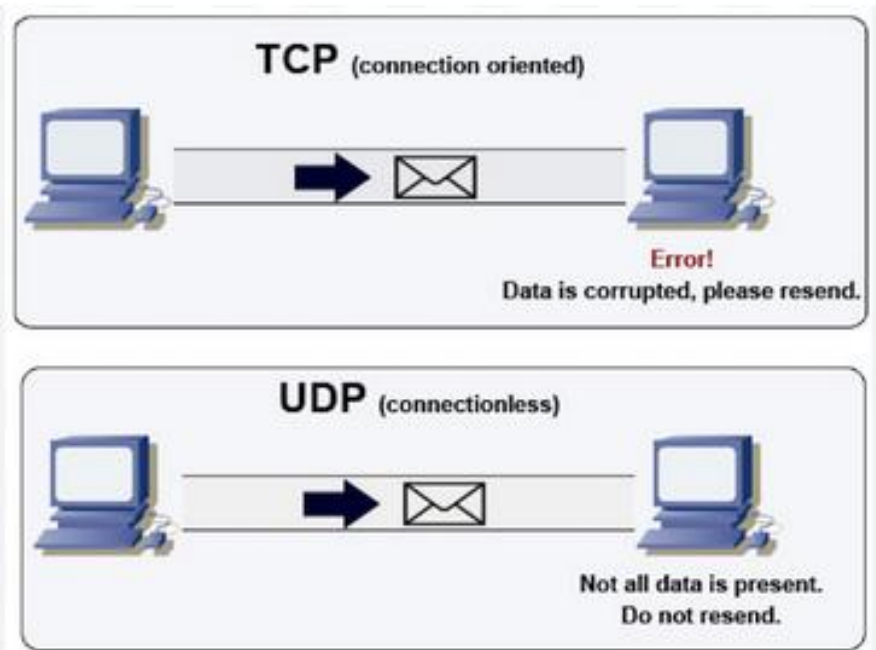
- פרוטוקול TCP

- לחלק מן הישומים תקשורת יכולה להיות לא אמינה

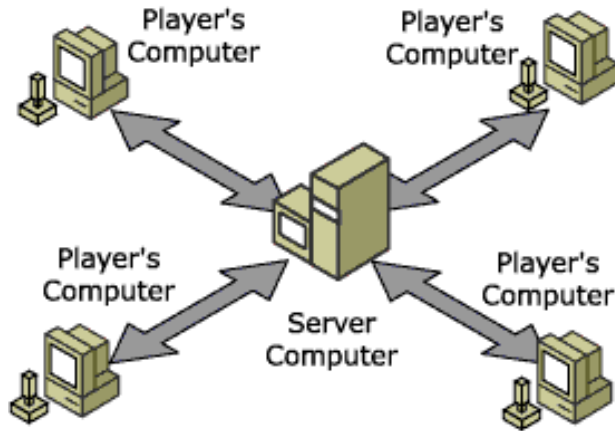
- חלק מן הנתונים יכולים ללכת לאיבוד

- פרוטוקול UDP

- ב-JAVA נשתמש במחלקות מוכנות שונות לתקשורת אמינה (*Socket*) ולא אמינה (*DatagramSocket*)

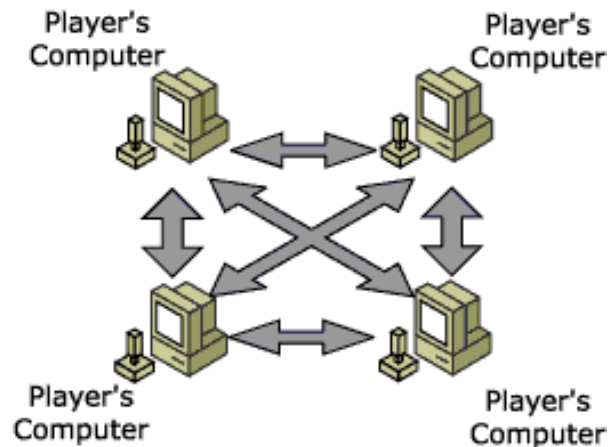


# מודלי תקשורת



## ■ שרת לקוח client server

- שרת מרכזי שלקוחות פונים אליו בבקשות
- שרת קבצים, שרת משחקים, שרת web וכו



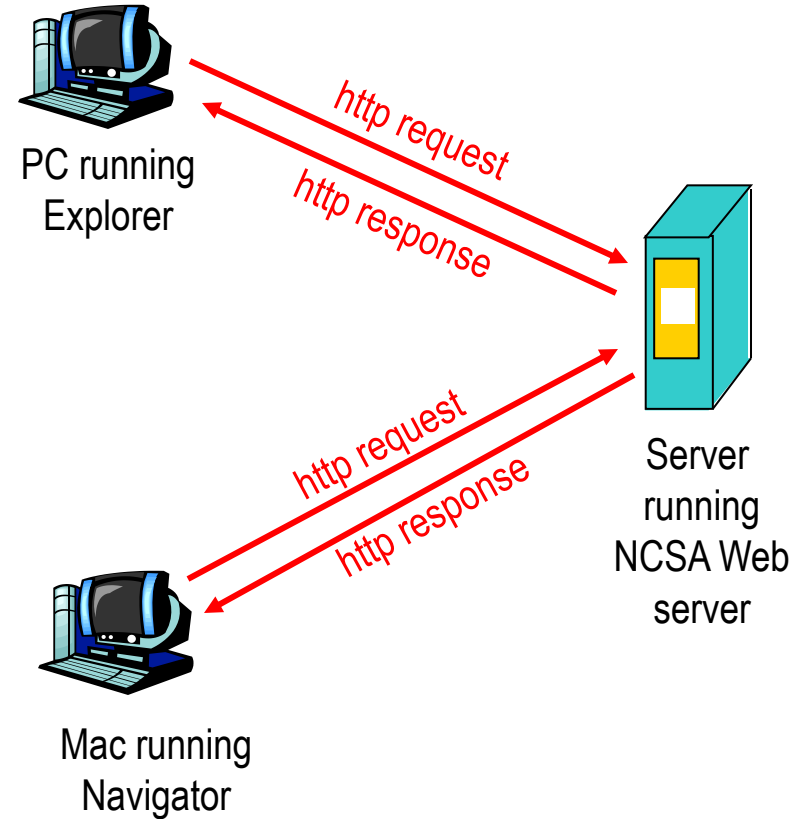
## ■ חבר לחבר P2P peer to peer

- לקוחות מדברים אחד עם השני
- שניהם יכולים להיות בו-זמנית גם שרת וגם לקוח

# The Web: the HTTP protocol

## HTTP: hypertext transfer protocol

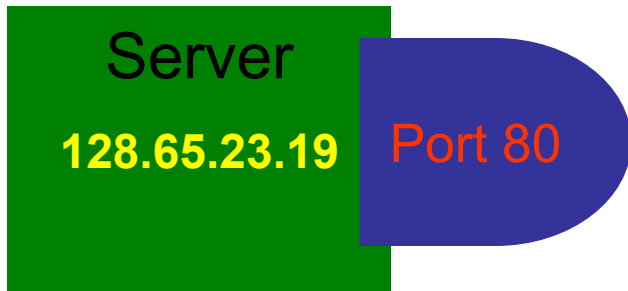
- Web's application layer protocol
- client/server model
  - *client*: browser that requests, receives, "displays" Web objects
  - *server*: Web server sends objects in response to requests





# כתובת IP ופורט

- תקשורת בין מחשבים דורשת לדעת איפה מחשב נמצא ואיזה תהליך מעניין אותנו
- איפה מחשב נמצא - כתובת IP
- מקבלים מספק תקשורת/ספק אינטרנט
- כתובת IP גרסה 4
- 4 גושים בין 0-255
- 128.65.23.19
- איזה תהליך – פורט
- מע"פ OS מקצה פורטים לתקשורת
- במודל שרת לקוח יש פורט קבוע לשרת ופורט שרותי ללקוח

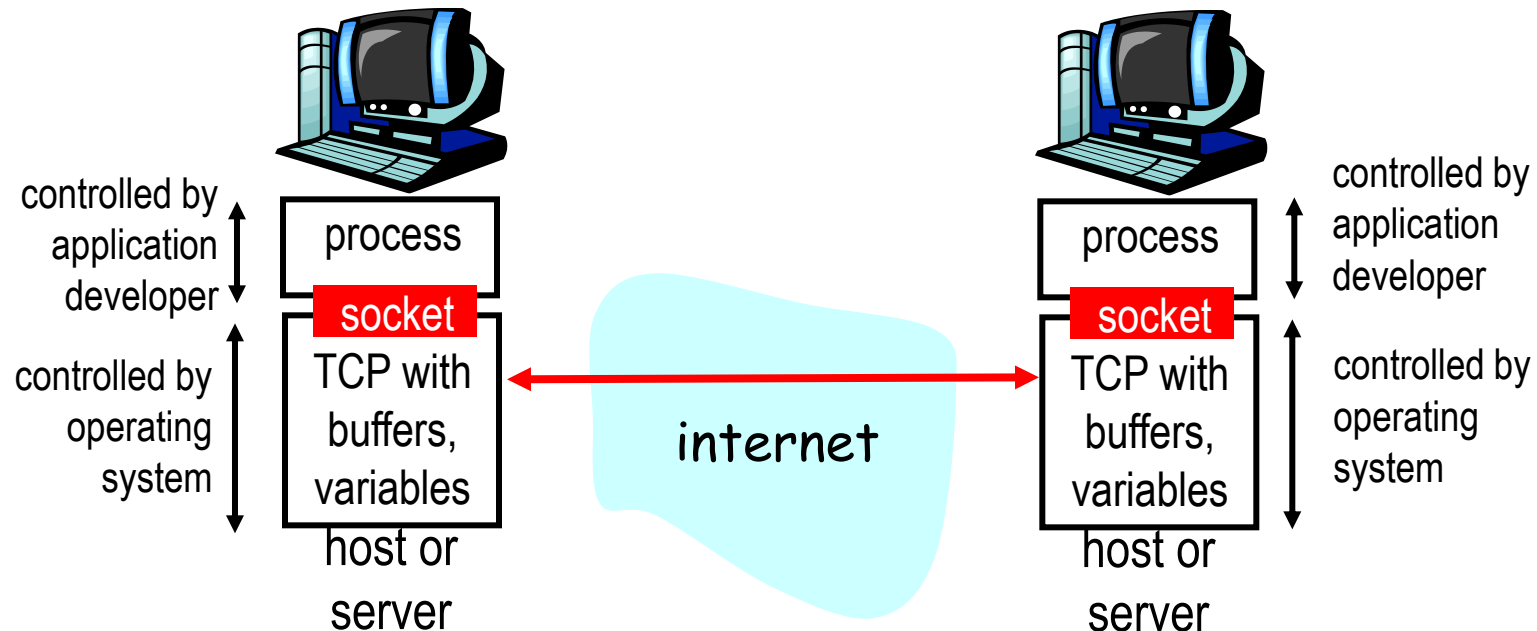


# Socket-programming

Socket: a door between application process and end-end-transport protocol (UDP or TCP)

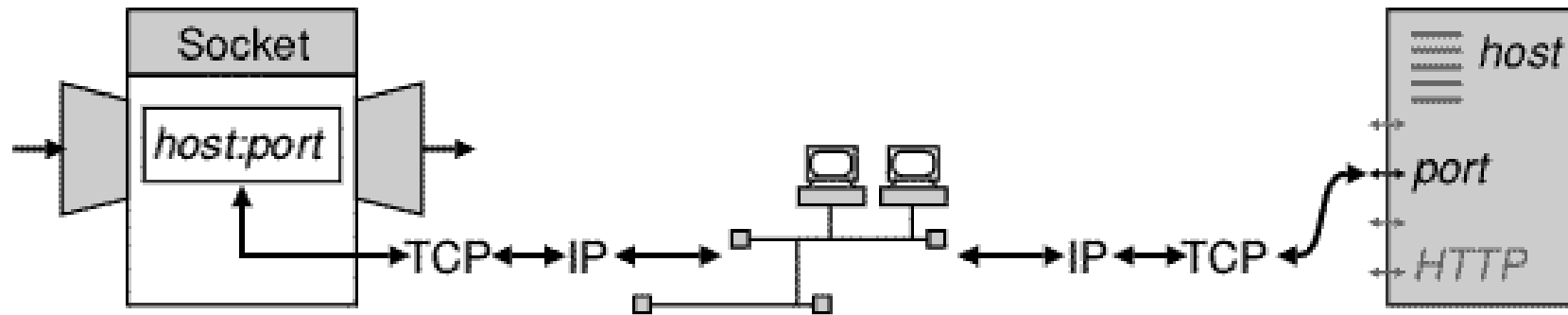
Port number: allows receiving host to determine to which local process the message should be delivered

Socket = IP address + port number



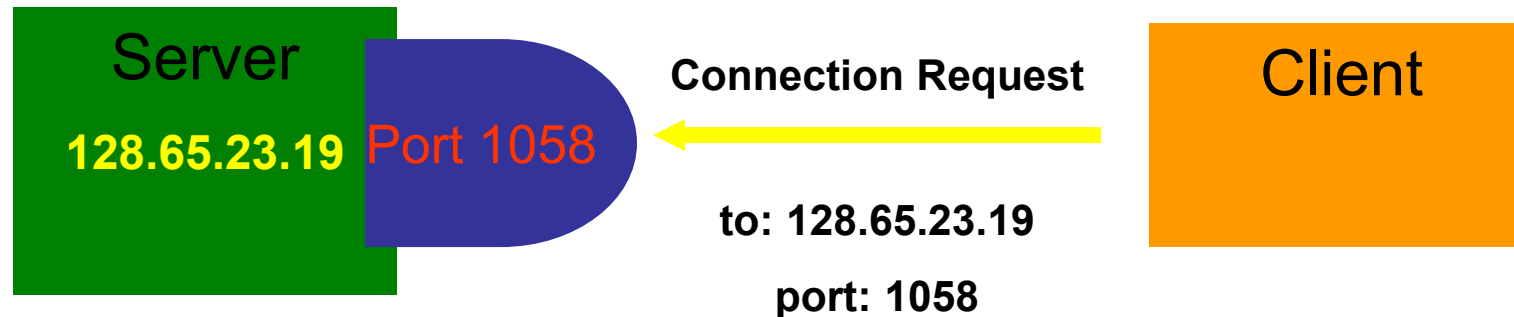
# מה זה שקע socket

- **שקע Socket** הינו קצה אחד של קישור תקשורת דו-כיוונית בין שני תוכניות (ופורטים)
- נוצר בכל צד (בכל תוכנית) - גם לשרת וגם ללקוח וגם לPeer
- תקשורת מתבצעת על ידי משלוח הודעות משקע לשקע



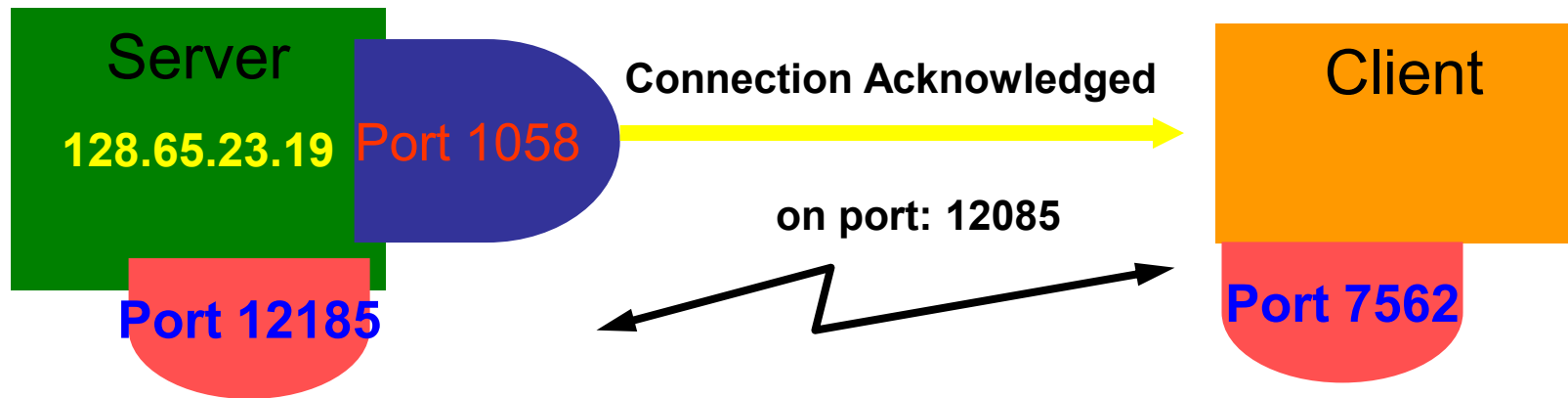
# מנגנון תקשורת

- שרת Server מאזין בלולאה אינסופית לפורט שאליו מגיעות בקשות מלקוחות
- לקוח Client פונה לשרת
  - בקשה להתחבר
  - בקשה לבצע שרות



## מנגנון תקשורת 2

- אם שרת Server מקבל את השיחה אז הוא (OS) מקצה פורט חדש לתקשורת עם לקוח ספציפי
- וגם אצל לקוח נפתח פורט לתקשורת עם השרת



# שרת לקוח פשוט



- נשתמש במחלקה

- `ServerSocket` ליצירת השרת ו`Socket` עבור לקוח

- יש לצרף חבילה `java.net.*`

```
import java.net.*
```

- שרת `server` `java.net.ServerSocket` class

- מקבל בקשה מלקוח להתחבר

- מייצר שקע לתקשורת

- לקוח `client` `java.net.Socket` class

- פונה לשרת

client

# מחלקת Socket

- מחלקה ליצירת שקע מסוג client לקוח Socket

- בנאי מקבל כתובת IP או שם השרת ופורט ששרת פתח לתקשורת

public Socket(String host, int port) throws UnknownHostException, IOException

- מתודות חשובות

public InputStream getInputStream() throws IOException

- מקבלת נתונים מן השרת

public OutputStream getOutputStream() throws IOException

- שולחת נתונים לשרת

public void close() throws IOException

- סוגרת את הקשר



server

# מחלקת ServerSocket

- מחלקה ליצירת שקע מסוג server שרת ServerSocket
- ממתין לבקשות של לקוחות לפורט מסוים
- בנאי מקבל פורט

public ServerSocket(int port) throws IOException

- לאחר מכך מתודות חשובות

public Socket accept() throws IOException

- ממתינה לבקשה מהלקוח, מייצרת חיבור ומחזירה שקע בצד שרת עבור לקוח ספציפי

public void close() throws IOException

- סוגרת את הקשר



# תקשורת ו-IO



- זרימת נתונים יכולה להיות בתווים/אותיות או בביטים/ביתים
- Character streams: **Readers** and **Writers**
- Byte streams: **InputStream** and **OutputStream**
- לכן יש לצרף **java.io**

# קלט בסיסי



■ **InputStream** הינה מחלקה אבסטראקטית

- **read()** int or bytes in the current machine's byte format
- **available()** returns the number of bytes available for reading
- **skip()** skips a number of bytes

■ נשתמש במחלקות ספציפיות

**BufferedInputStream** ■

**FilterInputStream** ■

**DataInputStream** ■

**FileInputStream** ■

# Basic Output



■ מחלקה אבסטרקטית דומה ל - `InputStream` היא `OutputStream`

- `write()` int or bytes to destination
- `flush()` buffered data

■ נשתמש במחלקות ספציפיות

`BufferedOutputStream` ■

`DataOutputStream` ■

`ObjectOutputStream` ■

`FilterOutputStream` ■

`FileOutputStream` ■

`PipeOutputStream` ■

# דוגמה



- נכתוב שרת פשוט שמאזין לפורט 5000
- נכתוב לקוח שיתחבר לפורט 5000 בשרת

## ■ הרצה:

1. קודם מריצים שרת
2. רק אחר כך לקוח

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
```

```
public class SimpleServer
```

```
{
```

```
    public static void main(String[] args) throws IOException
```

```
    {
```

```
        try (ServerSocket serverSocket = new ServerSocket(5000))
```

```
        {
```

```
            System.out.println("Server is listening on port 5000");
```

```
            Socket socket = serverSocket.accept();
```

```
            System.out.println("New client connected");
```

```
        } catch (IOException e)
```

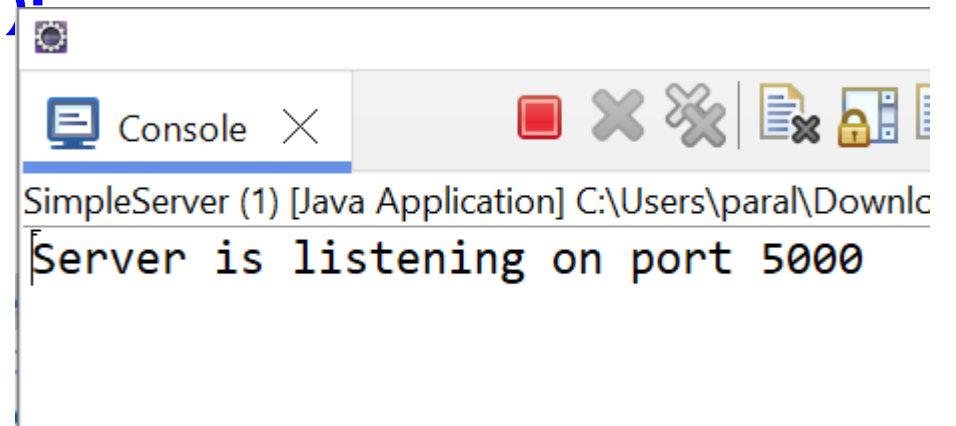
```
        { e.printStackTrace(); }
```

```
    }
```

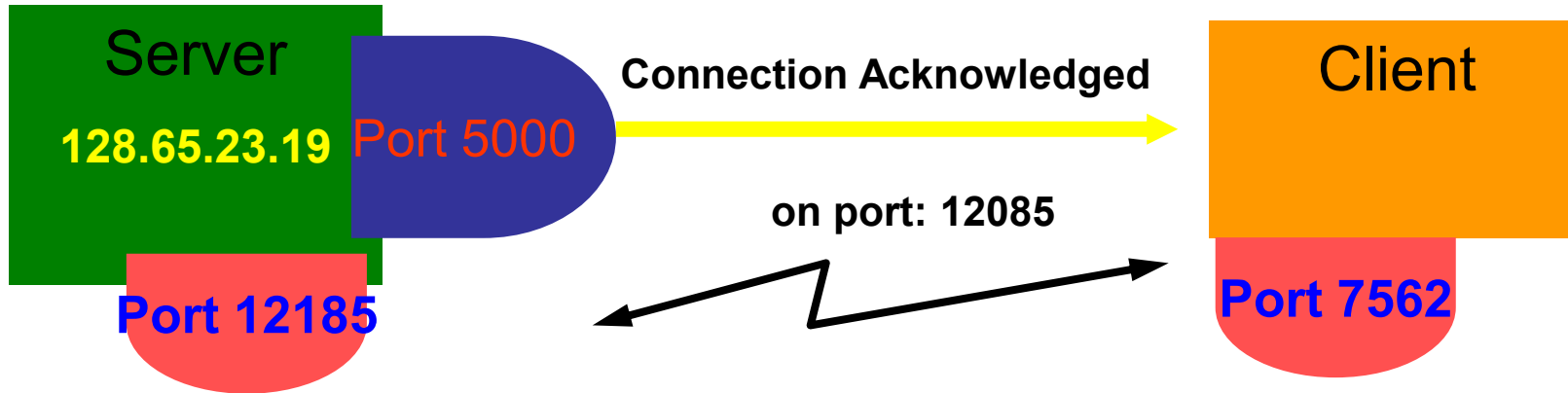
```
} //end of class
```

# Server

Port to listen



# accept()



- שרת ממתין לבקשת התחברות
- מקבל בקשה להתחבר accept ומייצר שקע לתקשורת
- Listens for a connection to be made to this socket and accepts it.
- בסוף תהליך תקשורת רצוי לסגור את הקשר

# Client

```
import java.io.IOException;  
import java.net.Socket;
```

```
public class SimpleClient
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        try (Socket socket = new Socket("localhost", 5000))
```

```
        {
```

```
            System.out.println("Connected to the server");
```

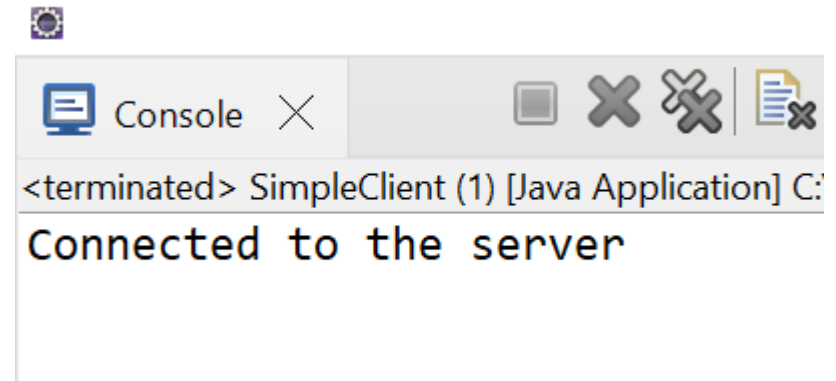
```
        } catch (IOException e)
```

```
        {    e.printStackTrace();    }
```

```
    }
```

```
} //end of class
```

Server IP address/name and port  
127.0.0.1/localhost



# הרצה

```
SimpleServer (1) [Java Application] C:\Users\para\Downl  
Server is listening on port 5000
```

```
<terminated> SimpleClient (1) [Java Application] C:  
Connected to the server
```

```
<terminated> SimpleServer (1) [Java Application] C:\Users\p  
Server is listening on port 5000  
New client connected
```

איך מריצים גם שרת וגם לקוח?

```
t java.net.Socket;  
c class SimpleClient  
  
public static void main(String[] args)  
{  
    try {  
        Socket s = new Socket("localhost", 5000);  
        System.out.println("Connected to the server");  
    } catch (IOException e)  
    {  
        e.printStackTrace();  
    }  
}
```



# משימה



- יש לשכת קוד השרת כך שיריץ לולאה
- While(true)
- יש לקרוא למחלקה SimpleServerLoop

# java.net



## ■ חריגות חשובות

- BindException
- ConnectException
- MalformedURLException
- NoRouteToHostException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

## ■ מחלקות חשובות

- ServerSocket ■
- Socket ■
- InetAddress ■
- DatagramPacket ■
- DatagramSocket ■

# דוגמה



- ניצור אפליקצית שרת לקוח מסוג הד ECHO
- לקוח יפנה לשרת לצורך התחברות
- כל מילה שלקוח שולח שרת ישלח בחזרה

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

```
public class SimpleServer
```

```
{
```

```
    public static void main(String[] args) throws IOException
```

```
    {
```

```
        ServerSocket serverSocket = new ServerSocket(1234);
```

```
        System.out.println("Server is listening on port 1234");
```

```
        Socket clientSocket = serverSocket.accept();
```

```
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
```

```
        BufferedReader in =
```

```
            new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

```
        String inputLine;
```

```
        while ((inputLine = in.readLine()) != null)
```

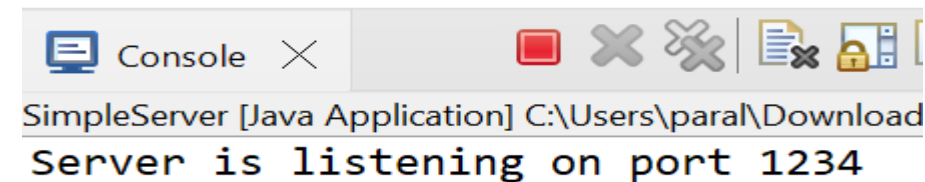
```
        {    System.out.println("Received: " + inputLine);
```

```
            out.println("Echo: " + inputLine);    }
```

```
        in.close();    out.close();    clientSocket.close();    serverSocket.close();
```

```
    }} //end of class
```

# Echo Server



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

# Echo Client

```
public class SimpleClient
```

```
{
```

```
    public static void main(String[] args) throws IOException
```

```
{
```

```
    Socket socket = new Socket("127.0.0.1", 1234);
```

```
    PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
```

```
    BufferedReader in =
```

```
        new BufferedReader(new InputStreamReader(socket.getInputStream()));
```

```
    BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
```

```
    String userInput;
```

```
    while ((userInput = stdIn.readLine()) != null)
```

```
    {    out.println(userInput);
```

```
        System.out.println("Server says you sent: " + in.readLine());
```

```
    }
```

```
    out.close();    in.close();    socket.close();
```

```
}}//end of class
```



```
SimpleServer [Java Application] C:\Users\para\Downloads\eclipse-java-2020-  
Server is listening on port 1234  
Received: hello server  
Received: thank you
```

```
SimpleClient [Java Application] C:\Users\para\Downloads\eclipse-java-2020-  
hello server  
Server says you sent: Echo: hello server  
thank you
```

```
SimpleServer [Java Application] C:\Users\para\Downloads\eclipse-java-2020-  
Server is listening on port 1234  
Received: hello server  
Received: thank you
```

```
SimpleClient [Java Application] C:\Users\para\Downloads\eclipse-java-2020-  
hello server  
Server says you sent: Echo: hello server  
thank you  
Server says you sent: Echo: thank you
```

# הרצה לאחר סגירת שרת

```
Console X
SimpleServer [Java Application] C:\Users\para\Downloads\
Server is listening on port 1234
Received: hello server
Received: thank you
```

```
SimpleClient [Java Application] C:\Users\para\Downloads\
hello server
Server says you sent: Echo: hello server
thank you
Server says you sent: Echo: thank you
what is your name
```

```
Console X
<terminated> SimpleServer [Java Application] C:\Users\para\Downloads\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.e
Server is listening on port 1234
Received: hello server
Received: thank you

Console X
<terminated> SimpleClient [Java Application] C:\Users\para\Downloads\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.e
hello server
Server says you sent: Echo: hello server
thank you
Server says you sent: Echo: thank you
what is your name
Exception in thread "main" java.net.SocketException: Connection reset
    at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:323)
    at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)
    at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:803)
    at java.base/java.net.Socket$SocketInputStream.read(Socket.java:981)
    at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:297)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:339)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:181)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:161)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:321)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:394)
    at SimpleClient.main(SimpleClient.java:16)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(NativeMethodAccessorImpl.java:62)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.base/java.lang.reflect.Method.invoke(Method.java:566)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(NativeMethodAccessorImpl.java:62)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.base/java.lang.reflect.Method.invoke(Method.java:566)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:111)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTestRunner.java:76)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.access$9$1(RemoteTestRunner.java:117)
    at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner$9.run(RemoteTestRunner.java:1043)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at java.base/java.util.concurrent.ThreadPooledBoundedQueue$Worker.run(ThreadPooledBoundedQueue.java:632)
    at java.base/java.lang.Thread.run(Thread.java:833)
```



## מגבלות לרגע זה

- כרגע שרת מאזין ללקוח אחד.
  - לקוח שמתחבר "ראשון" חוסם תקשורת עם שאר הלקוחות.
  - בהמשך נרצה ששרת אחת "ישרת" הרבה לקוחות.
- multithreading

