Introduction to Java.net Package

CGS 3416 Java for Non Majors

Package Overview

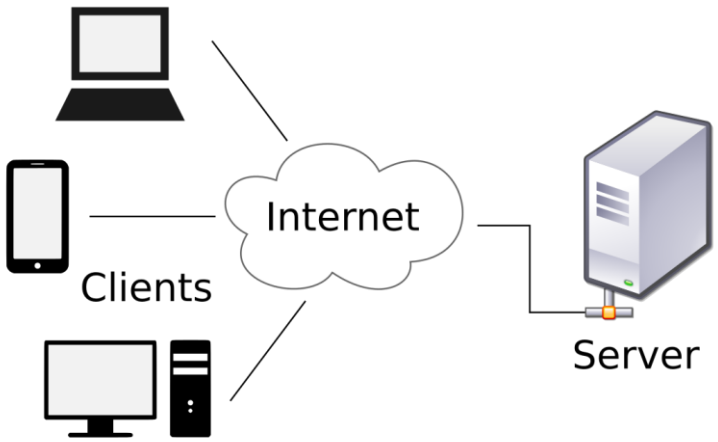
- The package java.net contains class and interfaces that provide powerful infrastructure for implementing networking applications.
- It can be roughly divided into *Low Level API* and *High Level API*
- *Low Level API* - deals with the following abstractions:
  - *Addresses*, which are networking identifiers, like IP addresses.
  - *Sockets*, which are basic bidirectional data communication mechanisms.
  - *Interfaces*, which describe network interfaces.
- *High Level API* - deals with the following abstractions:
  - *URIs*, which represent Universal Resource Identifiers.
  - *URLs*, which represent Universal Resource Locators.
  - *Connections*, which represents connections to the resource pointed to by *URLs*.
- More information can be found at following link:
  - https://docs.oracle.com/javase/9/docs/api/java/net/package-summary.html

Networking Basic

- IP Addresses – numerical label assigned to each device connected to a computer network that uses the Internet Protocol
  - Internet Protocol Version 4 (32 bits), i.e., 192.168.0.0
  - Internet Protocol Version 6 (128 bits), i.e., FE80::0202:B3FF:FE1E:8329
- URI – Uniform Resource Identifier, a sequence of characters that identifies a logical or physical resource
- URL – Uniform Resource Locator, a reference to a web resource that specifies its location on a network, a special kind of URI
  - For example: https://www.google.com/
- DNS – Domain Name System, a system contains a database of public IP address and associated hostnames, can be viewed as a phone book
- Protocol – rules or scheme that facilitate communication between machines
  - Examples: HTTP - HyperText Transfer Protocol, FTP – File Transfer Protocol

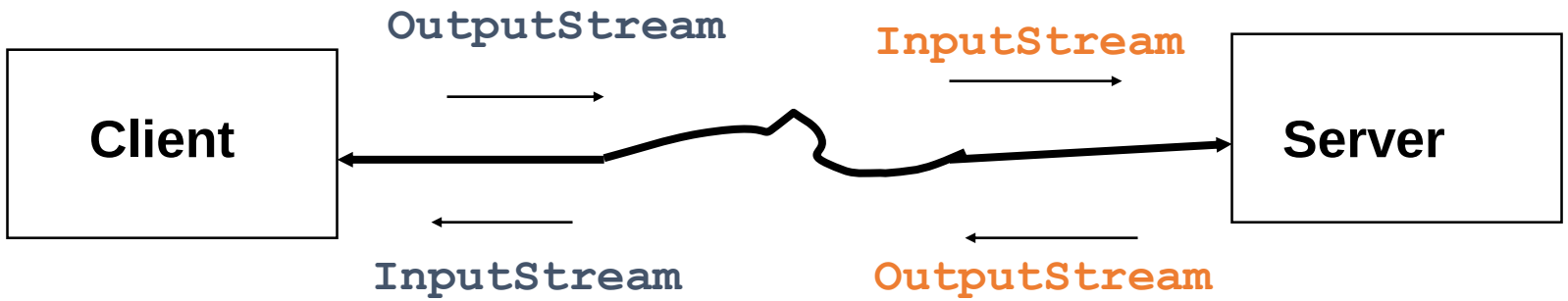
Networking Basic

- Port Number – a number that identifies a specific process or a type of network service within a host (range from 0 to 65535)
  - For instance: FTP – 21, SSH – 22, HTTP – 80, HTTPS – 443
- IP Address + Port Number = “Phone Number” for a service
  - For example: scheme://host:port - http://127.0.0.1:8080/
- Client-Server interaction
  - Communication between hosts is two-way, but usually those two hosts take different roles (i.e., client and server) and those roles can be interchangeable
  - Server registers on a known port with the host IP address
  - Server waits for client to make request through listen for incoming connections
  - Client sends request, wait for server respond then establish connection
  - Server respond with request resource

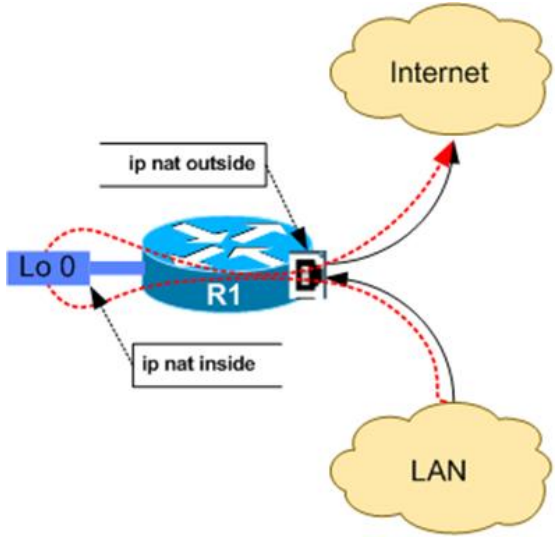


Networking Basic

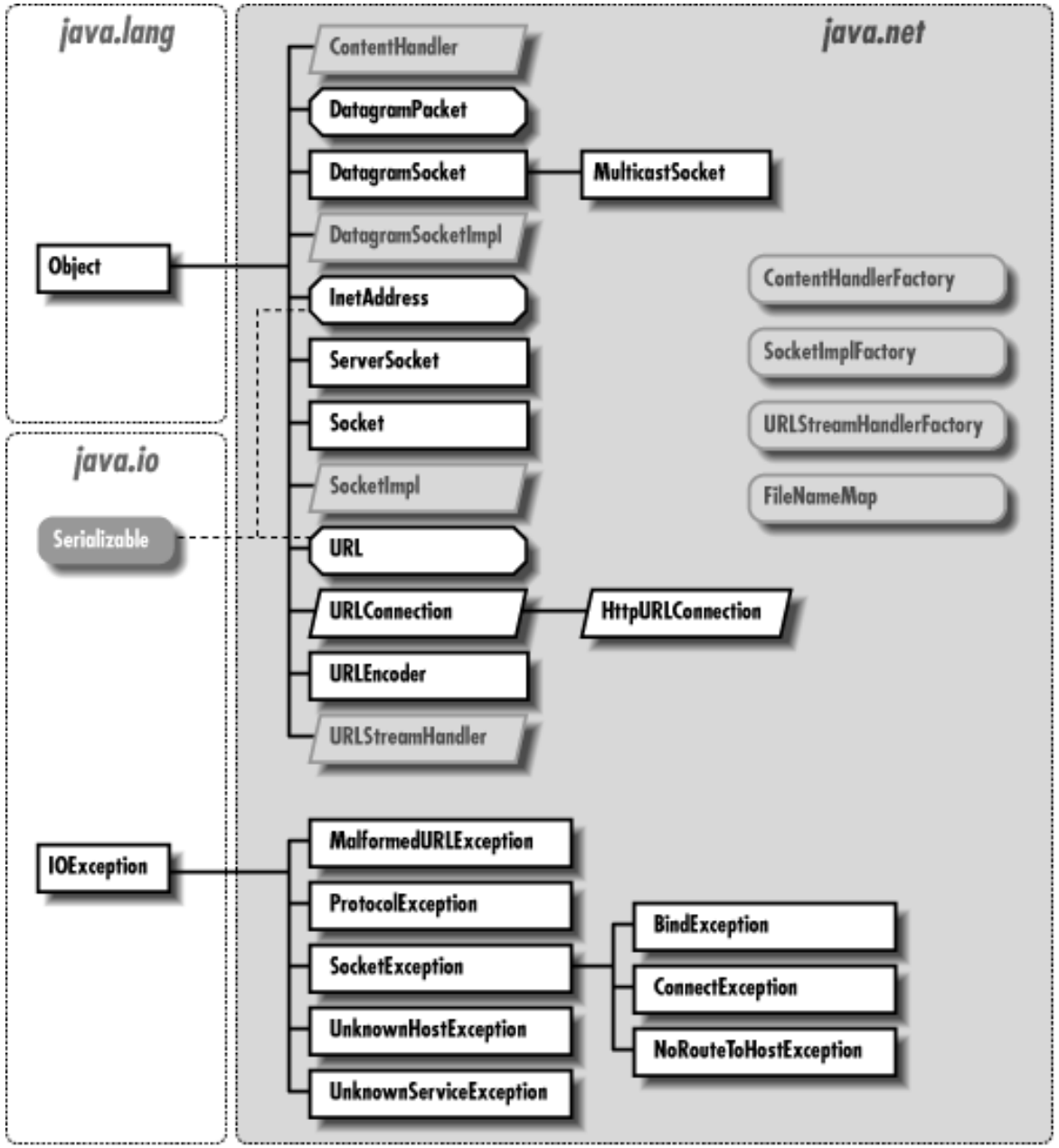
- Sockets – an abstraction of a bi-directional communication channel between hosts, input and output streams are used to send and receive data

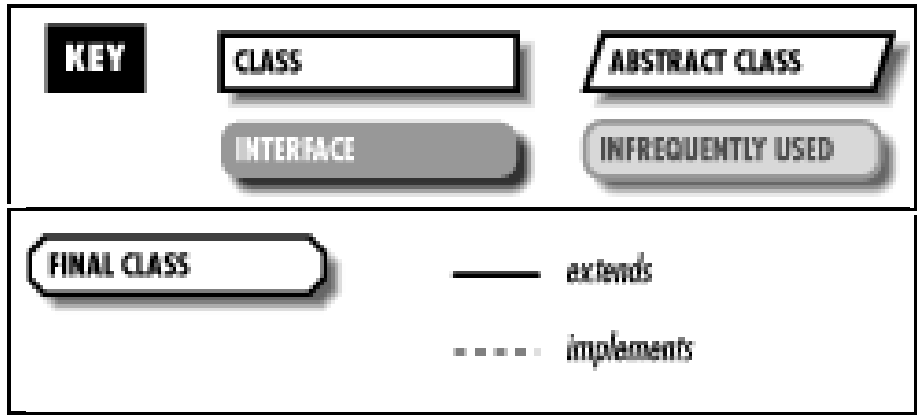


- Interfaces – a point of interconnection between a computer and a network
  - Hardware Interface – Network Interface Card
  - Software Interface – Loopback Interface (lo0)



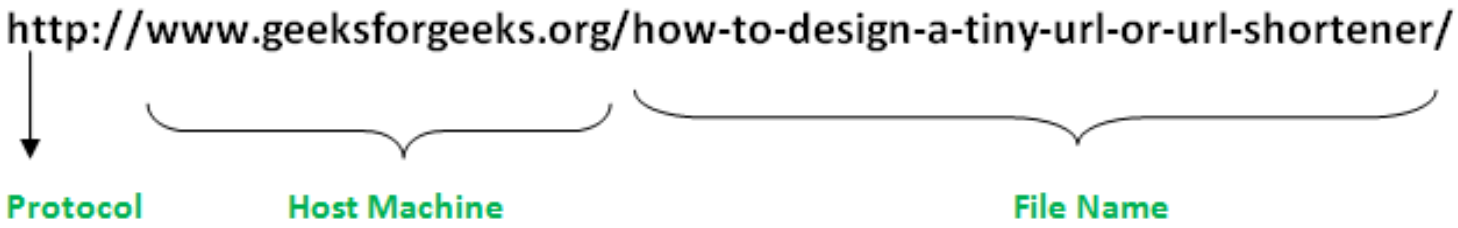
Package Class Hierarchy





URL Class Overview

- Class URL represents a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object



- Components of a URL: protocol, hostname, file name, port number(optional)
- Include 6 constructor, 22 methods (1 static)
- Throw 4 exceptions:
  - MalformedURLException
  - SecurityException
  - URISyntaxException
  - IOException

Example Constructors

- public URL(String protocol, String host, int port, String file) throws MalformedURLException: creates a URL by putting together the given parts
- public URL(String protocol, String host, String file) throws MalformedURLException: Identical to the previous constructor, except that the default port for the given protocol is used
- public URL(String url) throws MalformedURLException: creates a URL from the given String
- public URL(URL context, String url) throws MalformedURLException: creates a URL by parsing together the URL and String arguments.

Example Methods

- public final InputStream openStream() throws IOException:
  - Opens a connection to this URL and returns an InputStream for reading from that connection
  - Returns an input stream for reading from the URL connection
- public String toString():
  - Constructs a string representation of this URL. The string is created by calling the toExternalForm method of the stream protocol handler for this object
  - Returns a string representation of this object
- public int getPort():
  - Gets the port number of this URL
  - Returns the port number, or -1 if the port is not set
- public String getProtocol():
  - Get the protocol name of this URL
  - Returns the protocol of this URL

Example Program

- JavaNetURLExample:
  - Get the URL authority, Default Port, File Name, Host Name, Path and Protocol Name of given webpage http://www.gnu.org/licenses/gpl.txt
- JavaNetURLMoreExample:
  - Display the content of the webpage http://www.gnu.org:80/licenses/gpl.txt in text format