Universiteti i Prishtinës
Fakulteti i Shkencave Matematike-Natyrore
Departamenti i Matematikës
Shkenca Kompjuterike

Punim Seminarik

Lënda: **Programimi i Distribuar**

Tema: **Java Remote Desktop**

Punoi:
Imran Berisha

Profesor:
Eliot Bytyçi

Prishtinë, 2017

Hyrje

Në ditët e sotme nevoja për aplikacione që kryejnë funksione të caktuara në dobi të njerëzve është gjithmonë në rritje. Java është një nga gjuhët më të popullarizuara për zhvillim të aplikacioneve në botë. Aplikacioni që do të shqyrtohet është një Remote Desktop (Desktop në distancë).
Ky aplikacion është zhvilluar në Java dhe funksioni i tij kryesor është interaksioni në mes dy kompjuterëve. Ky aplikacion lejon që një kompjuter të kontrollojë apo shikojë një apo më shumë kompjuterë të tjerë në distancë.

Java Remote Desktop

Ky aplikacion mundëson shikimin e desktop-it në distancë, kontrollim të lëvizjeve të mausit, kontrollim të butonave të mausit dhe kontrollim të tastierës. JRD është aplikacion i tipit client-server dhe komunikon me anë të TCP. Lidhja në mes serverit dhe klientit/ëve është bërë me anë të socket-ëve.
Një socket është një pikë fundore e një lidhjeje dy anësore komunikimi në mes të dy programeve që punojnë në rrjetë. Klasat e socket përdoren për të prezentuar lidhjen në mes të një programi klient dhe një programi server. [1]

RmServer

Kjo është pjesa e serverit që pret për lidhje të klientit/ëve, dhe për çdo klient të kyçur, paraqitet një dritare grafike (frame) e re që shfaq ekranin e klientit aktual. Kur lëvizet mausi në kornizë, kjo ndodh edhe në anën e klientit. E njëjta ndodh nëse shfrytëzojnë butonët e mausit apo tastierën.

RmClient

Kjo është pjesa e klientit funksioni kryesor i së cilës është dërgimi i i një imazhi të ekranit (screenshot) të desktop-it të klientit në çdo periudhë të caktuar kohore (çdo 50ms në rastin tonë).

Ekzekutimi

Supozojmë që kërkohet të bëhet lidhja në mes të dy kompjuterëve. Njëri kompjuter duhet të ekzekutojë klasën *ServerMain.java* që gjendet në RmServer. Pas ekzekutimit, shfrytëzuesit i shfaqet një input dialog box që kërkon të jepet një numër i portit. Ky numër duhet të jetë më i madh se 1024. Arsyeja është sepse portet më të vogla se 1024 janë të rezervuara.
Në anën tjetër, kompjuteri tjetër duhet të ekzekutojë klasën *ClientMain.java* që gjendet në RmClient. Pas ekzekutimit, shfrytëzuesit i shfaqet një input dialog box që i kërkon të japë IP adresën e serverit (kompjuterit të parë) dhe portin e po atij kompjuteri.

Libraritë e përdorura

Kryesisht janë përdorur libraritë e java.awt, java.net dhe javax.swing. Disa nga libraritë më të rëndësishme janë përmendur në vijim:

java.awt.Robot – gjeneron input evente për qëllime të test automatizimit, vetë-demonstrime dhe aplikacione të tjera që ku nevojitet kontrolli i mausit dhe tastierës

java.awt.Rectangle – specifikon një zonë në një rrafsh koordinativ që është e rrethuar nga pika e lartë e majtë (x,y) e objektit të Rectangle në rrafshin koordinativ, gjerësia dhe gjatësia e saj

java.awt.GraphicsDevice – përshkruan pajisjet grafike që mund të jenë të gatshme në një ambient të veçantë grafik. Këtu përfshihen ekranet dhe printerët. Këtu mund të jenë shumë ekrane dhe printerë në një instancë të *GraphicsEnvironment*

java.awt.Dimension – enkapsulon (rrethon) gjatësine dhe gjerësinë e një komponenti në një objekt të vetëm

java.net.ServerSocket – implementon socket-a të serverit. Një server socket pret që që të vijnë kërkesa nga rrjeti. Bën diçka bazuar në atë kërkesë, dhe pastaj me shumë gjasë i kthen një rezultat atij që ka dërguar kërkesën

javax.swing.JFrame – shërben për vizatimin e dritares grafike

javax.swing.JOptionPane – shërben për shfaqjen e dialog box-eve që i kërkojnë shfrytëzuesit ndonjë vlerë apo i shfaqin diqka atij

javax.swing.ImageIcon – implementim i interfejsit Icon që vizaton ikona nga imazhet

java.awt.*event.KeyEvent, KeyListener, MouseEvent, MouseListener, MouseMotionListener* – shërbejnë për kontrollim të ngjarjeve të mausit dhe tastierës

java.util.Scanner – një skaner i thjeshtë i tekstit që mund të analizojë tipe primitive dhe stringje duke përdorur shprehje të rregullta

java.io.ObjectOutputStream – shkruan llojë të të dhënave primitive dhe grafikon objekte të Java-s në OutputStream. [2]

Klasat

Në këtë aplikacion janë përdorur dy pako (folderë): *RmServer* dhe *RmClient* që përmbajnë klasët e serverit dhe klientit respektivisht.

Klasat e *RmServer* janë:

ManageClient.java – menaxhon klientin duke vizatuar një GUI për secilin klient të lidhur me anë të metodës *drawGUI()*, lexon dimensionet e ekranit të klientit, pranon screenshot-a me anë të klasës *ReceiveScreen* dhe i dërgon ngjarje klientit me anë të klasës *SendCommands*

MouseCommands.java – përcakton konstante që përdoren për të reprezentuar komandat e serverit

ReceiveScreen.java – pranon screenshot-a nga klienti dhe i shfaq ato në server si dhe përmban klasën *SendCommands* që implementon ngjarjet e mausit dhe tastierës

ServerMain.java – klasa kryesore që përmban metodën *drawGUI()* për vizatim të kornizës së serverit, metodën *init()* për inicializim ku bëhet lidhja me anë të socket dhe metodën *main()* për testim.
Klasat e *RmClient* janë:
SendScreen.java – dërgon screenshot-a për çdo periudhë të caktuar kohore (50 ms) dhe i dërgon ato te serveri
ReceiveComms.java – pranon komandat e serverit dhe i ekzekuton ato te klienti
MouseCommands.java – përcakton konstante që përdoren për të reprezentuar komandat e serverit
ClientMain.java – klasa që lidhet me serverin, krijon GUI-n e klientit, dërgon screenshot-e te klienti dhe pranon komanda nga serveri duke shfrytëzuar klasat *SendScreen* dhe *ReceiveComms* respektivisht si dhe përmban klasën *main()* për testim.

Referencat:

[1] http://docs.oracle.com/javase/tutorial/networking/sockets/

[2] https://docs.oracle.com/javase/7/docs/api/