**VISOKO UČILIŠTE ALGEBRA**

ZAVRŠNI RAD

**Usporedba performansi različitih implementacija Java platforme na praktičnim primjerima**

Sebastijan Kelemen

Zagreb, prosinac, 2022.

*„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.*

*U Zagrebu, datum predaje rada*

***Ime Prezime***

***(potpisati se kemijskom olovkom)***

**Predgovor**

Zahvaljujem se mentoru Danijelu Kučaku i komentoru Ozrenu Tkalčecu na pomoći i savjetima prilikom izrade i odabira teme ovog završnog rada.

**Prilikom uvezivanja rada, umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada koju ste zaprimili u pozivu na obranu rada.**

**Sažetak**

Od samog svojeg nastanka Java programski jezik usko je povezan sa Java platformom koja je omogućila dostupnost programa napisanih u Javi na različitim uređajima i operativnim sustavima. Tijekom njegovog dugog postojanja pojavilo se mnoštvo različitih implementacija Java platforme različitim idejama, namjenama i ciljevima.

Svrha ovog rada je testiranje i usporedba performansi i ponašanja različitih Java platformi u stvarnim radnim uvjetima putem različitih testnih aplikacija. U tu svrhu također će biti prikazane razne tehnike provedbe mjerenja performansi kao i izrada i bilježenje preciznih rezultata takvih mjerenja.

Praktični doprinos ovog rada su rezultati testova performansi provedenih na svakoj od Java platformi i operativnom sustavu, te metode i postupci izvođenja takvih testova u Java okruženju.

**Ključne riječi:** Java, testovi, rezultati, performansi.

**Abstract**

Since its inception, the Java programming language has been closely associated with the Java platform, which enabled the availability of programs written in Java on various devices and operating systems. During its long existence, there have been many different implementations of the Java platform with different ideas, purposes and goals.

The purpose of this paper is to test and compare the performance and behavior of different Java platforms in real working conditions through different test applications. For this purpose, various performance

measurement techniques will also be presented, as well as the creation and recording of precise results of such measurements.

The practical contribution of this work is the results of the performance tests performed on each of the Java platforms and operating systems, as well as the methods and procedures for performing such tests in the Java environment.

**Key words:** Java, tests, results, performance.

**Sadržaj**

[1. Uvod 1](#__RefHeading___Toc945_2938388945)

[2. Java platforma 3](#__RefHeading___Toc947_2938388945)

[3. Testno okruženje 6](#__RefHeading___Toc949_2938388945)

[3.1. Korišteno testno okruženje 7](#__RefHeading___Toc951_2938388945)

[3.1.1. OpenJDK 11](#__RefHeading___Toc953_2938388945)

[3.1.2. OpenJ9 12](#__RefHeading___Toc955_2938388945)

[3.1.3. GraalVM 13](#__RefHeading___Toc957_2938388945)

[4. Testni alati 14](#__RefHeading___Toc959_2938388945)

[Popis kratica 15](#__RefHeading___Toc961_2938388945)

[Popis slika 16](#__RefHeading___Toc963_2938388945)

[Popis tablica 17](#__RefHeading___Toc965_2938388945)

[Popis kôdova 18](#__RefHeading___Toc967_2938388945)

[Literatura 19](#__RefHeading___Toc969_2938388945)

[Prilog 22](#__RefHeading___Toc971_2938388945)

Sadržaj se kreira automatski pomoću opcija **Reference/References** – **Tablica sadržaja**/**Insert Table of Figures**.

# Uvod

U ovom radu proučiti ćemo tri zasebne Java platforme (HotSpot, OpenJ9 i GraalVM) njihove performanse i posebnosti prilikom izvođenja različitih Java testnih aplikacija na dva različita operativna sustava Windows 11 i Arch Linux. Pod „Java platforma” mislimo na implementaciju Java virtualne mašine (eng. Java virtual machine, JVM). Programa koji je u Java eko sustavu zadužen za izvršavanje programa napisanih u Java programskom jeziku kao i u nekim ostalim programskim jezicima koji su prevedeni u Java „byte-kod”.

Za potrebe točnog i standardiziranog testiranja performansi Java platformi koristi ćemo Phoronix Test Suite, besplatni „benchmark” program otvorenog koda. Navedeni program omogućuje nam jednostavno kreiranje standardiziranih skupova testnih (eng. benchmark) aplikacija kao i njihovo automatizirano izvođenje i bilježenje rezultata. Testne aplikacije su odabrane prema specifičnim kategorijama kako bi se stekao bolji uvid u neko određeno od područje performansi Java platformi. Metodologija kojom ćemo vršiti testiranja Java platformi može se pojednostavljeno opisati na sljedeći način: prvo se odabere operativni sustav i jedna od Java platforma koju ćemo testirati, zatim se u programu Phoronix Test Suite i odabere skup testova te nakon što je testiranje završeno dobijemo rezultate testiranja. Isti postupak je potrebno ponoviti kako bi se omogućila usporedba između različitih Java platformi i operativnih sustava. Svi dobiveni rezultati mjerenja su zatim objedinjeni radi lakše usporedbe. Na kraju su dobiveni rezultati pojašnjeni i prokomentirani te ukoliko određeni test nije uspio opisan i objašnjen razlog neuspjeha.

Na kraju rada sukladno prethodno dobivenim rezultatima opisani su prednosti i nedostaci određenih Java platformi te po potrebi moguće postavke i preinake na Java platformi i Java programima kako bi se povećale performanse ili otklonili problemi prilikom izvođenja nekog testa.

# Java platforma

Java platforme su skup program i alata koji omogućavaju izradu i izvršavanje programa napisanih u Java programskom jeziku. Prva Java platforma razvijena je zajedno sa Java programskim jezikom početkom 1996

godine od strane američke korporacije Sun Microsystems. Cilj razvoja Java platforme bili su izrada novog programskog jezika koji će biti jednostavan, robustan i siguran te omogućiti izvršavanje jednom napisanog koda programa na različitim platformama i operativnim sustavima bez prilagodbe koda programa nekom specifičnom sustavu kao kod C i C++ programskih jezika.

Java platforme s kojima ćemo se baviti u ovom radu sastoje se od tri dijela:

* Java virtualni stroj (eng. Java virtual machine, JVM)
* Skup biblioteka
* Skup alata i programa za razvoj Java aplikacija

Ovisno o domeni primjene postoje različita izdanja i specifikacije Java platforme:

* JavaCard - pametne kartice i slični uređaji
* JavaME (eng. Micro edition) - mobilne uređaje i mikro računala
* **JavaSE** (eng. Standard edition) – opća namjena, osobna računala i serverska računala itd.

Sve Java platforme u ovom radu spadaju u **JavaSE** izdanje Java platforme. Bitno je napomenuti da osim gore navedenih platformi postoje i druge platforme koje nisu implementirane prema nekoj od gore navedenih specifikacija, a koriste Java programski jezik. Na primjer Android platforma koristi drugačiju implementaciju virtualnog stroja i skupa biblioteka.

Službena implementacija JavaSE Java platforme je izrađena od strane Oracle korporacije koja je također i njen trenutni vlasnik nakon što je preuzela korporaciju Sun Microsystems koja je bila njen orginalni autor i vlasnik.

Ova implementacija je de facto standard budući da Java platforma nije formalno standardizirana (ISO, ANSI, IEC itd.). Dostupno je za različite operativne sustave i hardverske platforme, poput: Windows, Linux i MacOS operativnih sustava. U svrhu usporedbe ponašanja i performansi Java platforma na različitim operativnim sustavima koristiti ćemo Windows 11 i Arch Linux Linux distribuciju.

Oracle JavaSE implementacija Java platforme dostupno je u dvije pod varijante:

* JRE (eng. Java runtime environment) – sadrži Java virtualni stroj i skup biblioteka, namijenjena izvršavanju java programa
* JDK (eng, Java development kit) – sadrži Java virtualni stroj i skup biblioteka, te dodatne alate i programe koji služe za izradu Java programa.

U ovom radu koristiti ćemo JDK, budući da samo on sadrži potrebne alate koje ćemo koristiti prilikom proučavanja performansi Java platformi, te nam omogućuje izradu testnih aplikacija.

Od svojih početaka Java platforma i Java programski jezik se neprekidno mijenjaju i unapređuju, zbog toga postoje razne verzije i izdanja Java platforme. LTS (eng. Long term support) verzije imaju puno dulju podršku i održavanje (npr. sigurnosne zakrpe, ispravljanje grešaka itd.) od strane razvojnih programera nego obične ne LTS verzije. Nove verzije Java platforme imaju jako brzi razvojni period. Zbog toga profesionalni programi, stručni tekstovi poput knjiga uobičajeno koriste LTS verzije Java platforme.

U trenutku izrade ovoga rada aktualne verzije Java platforme bile su:

* LTS verzije - 8, **11** i 17
* ne LTS verzija – 20

U ovom radu koristili smo LTS verziju **11** jer smo istraživanjem otkrili da ta verzija pruža najbolju kompatibilnosti sa različitim testnim programima i alatima.

# Testno okruženje

U ovom poglavlju opisat ćemo testno okruženje u kojem ćemo provoditi testove performansi Java platformi. Pod „testno okruženje” mislimo na skup hardverskog (hardver nekog računala) i softverskog okruženja (operativni sustavi, izdanja java platformi) u kojem ćemo provoditi testove.

Naravno kako bi osigurali da su rezultati testova ispravni potrebno je osigurati da se svi testovi izvršavaju na istom računalu radi razlike u performansama između različitih računala. Između testova ne smije biti promjena na operativnom sustavu poput različitih nadogradnja (na primjer upravljačkih programa ili jezgra sustava) ili izmjena postavki jer one također mogu utjecati na performanse računala. Potrebno je još osigurati i da je računalo između testova jednako opterećeno, to jest da se osim naših testova u isto vrijeme ne izvršavaju zahtjevniji programi jer oni oduzimaju dostupne resurse računala što opet utječe na rezultate testiranja. U tu svrhu preporučljivo je da se za testiranje koristi na novo instalirani operativni sustav. Komponente računala se znatno više griju prilikom zahtjevnih programa poput testnih programa (eng benchmark). Kako bi ostali unutar svoje dizajnom sigurne temperature potrebno im je osigurati adekvatno hlađenje. Zbog toga komponente računala poput modernih procesora i grafičkih kartica znaju svoju temperaturu te ovisno o njoj prilagođavaju svoju brzinu rada (na primjer u slučaju procesora smanjenjem radnog takta ukoliko je dosegnuta maksimalna sigurna temperatura). To je još jedan čimbenik koji može utjecati na rezultate testova.

Sam proces dobivanja rezultata testiranja je relativno dugotrajan i hardverski zahtjevan, na modernom računalu zna potrajati dva do tri sata za pojedinu Java platformu. Što je testno računalo hardverski jače to će testovi biti brže odrađeni. Zbog takvog dugotrajnog trajanja testiranja potrebno je osigurati da operativni sustav ne postavi računalo u način pripravnosti (eng. „sleep” ili „suspend” ovisno o operativnom sustavu) nakon određenog vremena.

## Korišteno testno okruženje

Kao hardversko testno okruženje koristili smo stolno računalo sa sljedećim komponentama:

* Matična ploča – Gigabyte B550 AORUS ELITE V2 (rev. 1.0/1.1)
* Procesor – AMD Ryzen 5 5600X
* Radna memorija – Kingston 2 x 8192 MB 3200MHz KHX3200C16D4
* Grafička kartica – Nvidia GeForce GTX 1650
* Pohrana - VPR100 PCI-e GEN3 x4 m.2 SSD
* Hladnjak procesora - Arctic Freezer 34 eSports

Kod softverskog testnog okruženja koristili smo najnovije verzije operativnih sustava te UEFI BIOS -a (verzija F15d) matične ploče, te upravljačkih programa (npr. grafičke kartice) u trenutku pisanja. Svaki operativni sustav je instaliran na potpuno praznu SSD pohranu.

Koristili smo sljedeće operativne sustave:

* Microsoft Windows 11
* Arch Linux

Za razliku od Windows operativnog sustava, operativni sustav Arch Linux nema neku specifičnu verziju izdanja budući da je riječ o operativnom sustavu koji se sustavno nadograđuje sa novim ažuriranjima i izmjenama (eng. rolling release).

Zajedničke karakteristike operativnih sustava poput verzije jezgre (eng. kernel), upravljačkih programa, datotečnih sustava opisani su u Tablici 1.1.

Tablica 1.1 Operativni sustavi

|  |  |  |
| --- | --- | --- |
| Operativni sustav | Microsoft Windows 11 Pro Build 22621 | Arch Linux |
| Jezgra | 10.0.22621.1992 (x86\_64) | 6.4.3-arch1-1 (x86\_64) |
| Upravljački program grafičke kartice | 536.67 (31.0.15.3667) | NVIDIA 535.54.03 |
| Datotečni sustav | NTFS | ext4 |

Detaljne karakteristike za pojedini operativni sustav prikazane su u sljedećim tablicama.

Tablica 1.2 Windows operativni sustav

|  |  |
| --- | --- |
| Operativni sustav | Microsoft Windows 11 Pro Build 22621 |
| Jezgra | 10.0.22621.1992 (x86\_64) |
| Upravljački program grafičke kartice | NVIDIA 536.67 |
| Datotečni sustav | NTFS |

Zbog značajnih razlika u arhitekturi operativnih sustava tablice imaju različita polja ovisno o operativnom sustavu. Na primjer za razliku od Windows operativnog sustava, Arch Linux nam omogućuje promjenu grafičke radne okoline (eng. desktop environment).

Tablica 1.3 Arch Linux operativni sustav

|  |  |
| --- | --- |
| Operativni sustav | Arch Linux |
| Jezgra | 6.4.3-arch1-1 (x86\_64) |
| Upravljački program grafičke kartice | NVIDIA 535.54.03 |
| Radna okolina | KDE Plasma 5.27.6 |
| Program prikaza | X Server 1.21.1.8 |
| Datotečni sustav | ext4 |

**3.2 Korištene Java platforme**

Kako bi nam poslužile za testove Java platforme koje smo odabrali su morale su zadovoljiti određene kriterije. Morale su biti dostupne na operativnim sustavima na kojima smo testirali (Windows i Arch Linux), te biti izrađene prema JavaSE specifikaciji i u trenutku pisanja ovog rada zadnjoj dostupnoj LTS verziji 11 (puna verzija 11.0.19+7). Za sve platforme korištena je JDK varijanta Java platforme. OpenJDK je službena referentna implementacija Java platforme prema JavaSE specifikaciji, stoga je izdanje Java platforme isto kao i verzija OpenJDK Java platforme. Također ostale Java platforme koje su izrađene prema JavaSE specifikaciji su kompatibilne sa OpenJDK Java platformom.

Za Java platforme koje ćemo testirati odabrali smo sljedeće Java platforme sa verzijama kako je prikazano u tablici 1.4.

Tablica 1.4 Java plaforme

|  |  |
| --- | --- |
| Java platforma | verzija |
| OpenJDK | 11.0.19+7 |
| OpenJ9 | 0.38.0 |
| GraalVM | CE 22.3.2 |

### OpenJDK

OpenJDK je službena referentna implementacija JavaSE specifikacije. Besplatna je i otvorenog je koda. Nastala je 2007. godine od strane američke korporacije Sun promjenom licenciranja i objavljivanjem izvornog koda dotadašnje orginalne Java platforme (Sun Java). Budući da je OpenJDK trenutni neslužbeni standard JavaSE specifikacije, mnoge organizacije i korporacije nude različita izdanja izrađena od izvornog koda OpenJDK platforme sa različitim izmjenama, licencama i komercijalnom podrškom. U tablici 1.5 prikazani su samo neka od OpenJDK izdanja i njihove razlike.

* Tablica 1.5 OpenJDK implementacije

|  |  |  |
| --- | --- | --- |
| Ime | Modificiran izvorni kod | Komercijalna podrška |
| Oracle Java SE | DA | DA |
| Oracle OpenJDK | DA | NE |
| SAP SapMachine | DA | DA |
| Red Hat build of OpenJDK | DA | DA |

U ovom radu koristili smo Oracle OpenJDK izdanje u slučaju Windows operativnog sustava i Arch Linux operativnog sustava. Razlika između Oracle OpenJDK izdanja i izvornog OpenJDK koda je u tome da Oracle OpenJDK ne sadrži eksperimentalni „garbage collector” ShenandoahGC.

slika 1.1 ShenandoahGC na Oracle OpenJDK platformi

### OpenJ9

Eclipse OpenJ9 Java platforma (pred hodno poznata pod nazivom IBM J9) je implementacija JavaSE specifikacije prvotno izrađena od strane američke korporacije IBM. Dizajnirana je za kompatibilnost sa JavaSE specifikacijom i da bude prilagodljiva ralzičitim vrstama Java aplikacija od jednostavnih apliakcija u „oblaku” do vrlo kompleksnih aplikacija na „mainframe” računalima. Prvobitno je bila dizajnirana kao virtualni stroj za izvođenje SmallTalk programskog jezika od strane tvrtke ENVY/Smalltalk. Rastom popularnosti Java programskog jezika sredinom devedesetih godina prošlog stoljeća tvrtka ENVY/Smalltalk je 1996. godine kupljena od strane IBM korporacije koja ju je prenamijenila u Java platformu. 2017. godine OpenJ9 platforma je postala besplatna i otvorenog koda promjenom licence i objavom izvornog koda. Te je postala dio neprofitne organizacije Eclipse Foundation koja se trenutno brine o njenom daljnjem razvoju. Trenutno postoje tri izdanja ove Java platforme koja su prikazana u tablici 1.6. U ovom radu koristili smo IBM:Semeru Runtime izdanje kod Windows i Arch Linux operativnog sustava.

Tablica 1.6 OpenJ9 izdanja

|  |  |
| --- | --- |
| Ime | Opis |
| IBM:Semeru Runtime | OpenJ9 platforma sa OpenJDK bibliotekom |
| Apache OpenWhisk | OpenJ9 kao dio IBM cloud servisa |
| EclipseIDE | OpenJ9 kao dio Eclipse razvojnog okruženja |

### GraalVM

GraalVM Java platforma temelji se na OpenJDK Java platformi. Za razliku od OpenJDK platforme omogućuje izvođenje programa napisanih u ostalim programskim jezicima (npr. Python, Ruby, R, JavaScript) i sadrži potpuno novi Java prevoditelj (eng. compiler) napisan u Java programskom jeziku. Nasato je kao dio Maxine virtualnog stroja koji je nastao kao pokušaj izrade Java virtualne mašine u Java programskom jeziku umjesto trenutnog C i C++ programskog jezika. Smartalo se da je to preambiciozno te je za početak odlučeno da samo određeni dijelovi virtualnog stroja poput prevoditelja (eng. compiler) budu napisani u Java programskom jeziku. U početku je Graal bio samo eksperimentalni prevoditelj koji je dodan u OpenJDK platformu, a kasnije 2019. godine je izdvojen iz OpenJDK platforme i postao je zasebna Java platforma. Postoje dva izdanja GraalVM platforme koje se razlikuju po licenci i izvornom kodu. Razlike su prikazane u tablici 1.7. U ovom radu koristili smo Community izdanje.

Tablica 1.7 GraalVM izdanja

|  |  |
| --- | --- |
| Izdanje | Licenca |
| Community | GPL2 (besplatna i otvorenog koda) |
| Enterprise | Trialware (Komercijalna) |

# Testni alati

Tema ovog poglavlja su testovi u računarstvu opčenito i testni alati koji se koriste u tu svrhu. Test (eng. benchmark) služi kako bi dobili informacije o relativnim performansama nekog objekta poput neke hardverske komponente (npr. procesor, grafička kartica) ili softverskog programa (programske platforme, baza podataka itd.) putem standardiziranih metoda.

Kako bi testovi bili točni i pravilni potrebno je pridržavati sedam bitnih karakteristika to su:

1. Relevantnost – Testovi trebaju mjeriti bitna svojstva nekog sustava
2. Reprezentacija – Prikaz rezultata treba biti u standardnom obliku općenito prihvaćenom od ostatka akademske i industrijske zajednice
3. Jednakost – Svi sustavi trebaju biti testirani pravedno i jednako
4. Mogućnost ponavljanja – Dobiveni rezultati bi se trebali moći ponovno reproducirati i verificirati
5. Ekonomičnost – Testovi trebaju bit ekonomični
6. Prilagodljivost – Testovi bi se trebali moći izvršavati na različitim sustavima sa različitim resursima
7. Transparentnost – Testovi bi trebali biti jednostavni i lako razumljivi

Kako bi se pojednostavila izrada testova koji su prilagođeni prema gore navedenim pravilima postoje testni alati. Testni alati nam omogućuju automatizirano testiranje uz dobro definirane testove, dobivanje i prikaz rezultata prema prema općenito prihvaćenim standardima u različitim formatima. Na tržištu postoje različite vrste testnih alata od profesionalnih komercijalnih rješenja do testnih alata posebnih namjena. Za testiranje Java platformi odabrali smo testni alat Phoronix Test Suite, besplatni testni alat otvorenog koda dostupan na Windows i Linux operativnim sustavima. Za naše potrebe testiranja koristili smo najnoviju verziju u trenutku pisanja v10.8.4.

## Phoronix Test Suite

Phoronix Test Suite je besplatni testni alat otvorenog koda izrađen od strane autor Michael Larabel -a i Matthew Tippett -a. Prva verzija pojavila se 2008. godine i od tada se neprekidno razvija. Omogućava potpuno automatsko izvođenje svih koraka u nekom testiranju.

1. Preuzimanje i instalacija
2. Izvršavanje
3. Izvještavanje i prikaz rezultata

U ovom alatu, testovi su podijeljeni u dvije skupine to su: testovi („test profile) koji služe za testiranje pojedinog programa i skupovi testova („test suite”) koji sadrže više testova koji su određene tematike poput baza podataka. Sami testovi se sastoje od opisne XML datoteke koja sadrži sve potrebne informacije o nekom testu i obične „shell” skripte koja samo služi za pokretanje programa kojega želimo testirati. Skup testova se također sastoji od opisne XML datoteke koja sadrži informacije o skupu testova (nazivu, kategoriji testiranja itd.) kao i popis testova koje taj skup sadržava. Ovakva testna arhitektura omogućuje jednostavno kreiranje novih kao i modifikaciju postojećih testova. Svaki test ili skup testova koji želimo izvršiti se automatski preuzima sa internet stranice openbenchmarking.org gdje se nalazi više od 650 različitih testova, kao i veliki broj skupova testova različitih kategorija. Navedena stranica također omogućuje slanje rezultata nekog testa kao i usporedbu rezultata sa ostalim korisnicima. U trenutku izvođenja testova moguće je i prikupljanje podataka softverskoj (npr. podaci o operativnom sustavu) i hardverskoj platformi na kojoj se izvode testovi kao i o njenom resursnom opterećenju prilikom izvršavanja testova. Rezultati testova su pohranjeni u obliku XML datoteka. Radi lakše usporedbe više različitih rezultata testova se mogu objediniti u jedinstveni rezultat koji je također pohranjen u XML datoteci. Navedene XML datoteke se zatim mogu pretvoriti u izvješća u obliku PDF ili HTML datoteka. Pregled i uređivanje testova dostupno je putem grafičkog internet sučelja ili putem jednostavne modifikacije XML datoteka.

## Instalacija i upotreba Phoronix Test Suite -a

Instalacija ovog alata se razlikuje ovisno o operativnom sustavu. Zbog različitog uobičajenog načina distribucije softvera kod Windows i Linux operativnih sustava.

### Arch Linux instalacija

Aplikacije u Arch Linux operativnom sustav podijeljene su u zasebne pakete koji su pohranjeni u internet repozitorijima. Prilikom instalacije dovoljno je jednostavno upisati putem grafičkog ili sučelja u komandnoj liniji naziv aplikacije koju želimo instalirat i program zadužen za upravljanje paketima (eng. package manager) će preuzeti i instalirati traženu aplikaciju.

Kako bi instalirali Phoronix Test Suite potrebo je poduzeti sljedeće korake:

1. Budući da Phoronix Test Suite se ne nalazi u službenim Arch Linux repozitorijima, potrebno je koristiti neslužbeni repozitorij AUR (Arch Linux User Repository ) koji sadrži pakete koje su izradili sami korisnici Arch Linux operativnog sustava. U u kojem se nalazi Phoronix Test Suite.
2. Prije same instalacije potrebno je instalirati pakete o kojima ovisi Phoronix Test Suite paket (eng. dependencies) koristeći sljedeću naredbu u naredbenom redku:

pacman -S base-devel php wget

1. Sami paket možemo preuzeti i instalirati uz pomoć sljedećih naredbi:

wget https://aur.archlinux.org/cgit/aur.git/snapshot/phoronix-test-suite.tar.gz

tar -xvf phoronix-test-suite.tar.gz

cd phoronix-test-suite

makepkg -c -i -s

1. Nakon toga aplikacija je dostupna u Arch Linux operativnom sustavu, možemo je pokrenuti iz naredbenog retka putem naredbe:

phoronix-test-suite

### Windows instalacija

U Windows 11 operativnom sustavu Phoronix Test Suite instalira se preuzimanjem aplikacije putem službene stranice [www.phoronix-test-suite.com](http://www.phoronix-test-suite.com/), te otpakirati arhivu koja sadrži aplikaciju i pokrenuti instalaciju na način da pokrenemo install.bat datoteku. Te nakon toga uz pomoć naredbenog redka (eng. command prompt) pokrenemo aplikaciju uz naredbu:

phoronix-test-suite

### Upotreba Phoronix Test Suite -a

Phoronix Test Suite izvršava se u naredbenom retku naredbom **phoronix-test-suite**. Aplikacijom upravljamo uz pomoć korisničkih opcija. Korisničke opcije su podijeljene po različitim kategorijama.

Naredbom:

phoronix-test-suite help

možemo dobiti popis svih dostupnih opcija podijeljenih po kategorijama.

Kategorije su:

* System – Za dobivanje podataka o hardverskoj platformi (Sustavu) na kojoj se aplikacija nalazi.
* Test Installation – Instalacija i priprema testova ili skupova testova, te za uklanjanje već instaliranih.
* Testing – Izvršavanje testova i skupova testova
* Batch Testing – Automatizirano izvršavanje više testova i skupova testova odjednom (eng. batch-mode)
* OpenBenchmarking.org – Za slanje i pregled različitih rezultata (lokalnih i od ostalih korisnika), preuzimanje testova i skupova testova.
* Information – Pruža informacije o testovima, skupovima testova, intaliranim testovima i skupovima testova.
* Result Management – Upravljane rezultatatima testiranja, uređivanje rezultata, sortiranje rezultata i tako dalje.
* Other – Ostalo opcije poput pomoći, verzije aplikacije i tako dalje.
* Result Analysis – Analiza dobivenih rezultata
* Modules – Upravljane modulima, dijelovi aplikacije su podijeljeni u module poput na primjer kreiranja izvješča rezultata testova u html formatu (modul html\_results\_export)
* Debugging – Za otklanjanje grešaka (eng. debug), na primjer prilikom izrade novog testa.
* User Configuration – Korisničke postavke poput na primjer mrežnih postavki.
* Result Export – Izvoz rezultata u različitim formatima poput html i pdf formata.
* Phoromatic – Za Phoromatic internet servis koji omogućuje udaljenu kontrolu računala koja izvršavaju testove.
* Result Viewer – Internet preglednik rezultata testiranja, omogučava pregled i uređivanje rezultata.

Najbrži način da bi smo pokrenuli test ili skup testova je uz sljedeću naredbu:

phoronix-test-suite benchmark [ime testa ili skupa testova], na primjer

phoronix-test-suite benchmark c-ray

Nakon toga trebamo odabrati hoćemo li spremiti rezultate testiranja i ako smo odgovorili potvrdno onda unesemo ime rezultata. Nakon toga počinje izvršavanje testa ili skupa testova te se po završetku testiranja rezultati spremaju u XML datoteku pod nazivom koji smo prethodno odabrali. Na kraju možemo odabrati hočemo li rezultate testiranja poslati na

OpenBenchmarking.org repozitorij, te trebali se pokrenuti internet preglednik rezultata kako bi smo proučili rezultate.

## Testiranje Java platformi

Kod testiranja Java platformi prvo će biti potrebno odabrati vrste i kategorije testova. Uz pomoć OpenBenchmarking.org internet repozitorija testova odabrali smo sljedeće testove koje smo zatim spojili u skup testova (eng. benchmark suite). Testove smo odabrali prema sljedećim kategorijama, ako je test dio skupa testova naziv toga skupa i njegova verzija je u zagradama:

* Strojno učenje, analitika i statistika
  + als - ALS algoritam iz Spark ML biblioteke (renaissance-1.3.0)
  + Chi-square – statistični test „chi-squre” iz Spark ML biblioteke (renaissance-1.3.0)
  + dec-tree – „Random forest” algoritam koji se primjenjuje kod strojnog učena, iz Spark ML biblioteke (renaissance-1.3.0)
  + gauss-mix - izračunava model Gaussove mješavine koristeći maksimiziranje očekivanja. (renaissance-1.3.0)
  + Log-regression – algoritam logističke regresije, iz Spark ML biblioteke (renaissance-1.3.0)
  + movie-lens - preporučuje film temeljem ALS algoritma (renaissance-1.3.0)
  + naive-bayes - „Multinomial Naive Bayes” algoritam (renaissance-1.3.0)
  + page-rank – „Page Rank algoritam koji se koristi kod određivanja pozicije stranice kod Google internet pretraživanja (eng. rank) (renaissance-1.3.0)
* Paralelizam i konkurentno izvođenje
  + reactors – mikro testovi za testiranje performansi Reactors.io biblioteke za reaktivno programiranje (renaissance-1.3.0)
  + akka-uct - „ Unbalanced Cobwebbed Tree” test unutar biblioteke za internet programe Akka (renaissance-1.3.0)
* Baze podataka
  + db-shootout – „shootout” test koristeći nekoliko baza podataka u radnoj memoriji (renaissance-1.3.0)
  + h2 – test transakcija u bazi podataka, koristeći bazu u radnoj memoriji H2 (dacapobench-1.0.1)
* Grafika
  + sunflow – test performansi sustava za iscrtavanje (rendering) otvorenog koda Sunflow
  + j2dbench – različiti grafički testovi za mjerenje java 2d grafičkih performansi (performanse iscrtavanje teksta, slika i grafike)
* Znanstveni i numerički testovi
  + monte-carlo – Monte Carlo algoritam
  + sparse-matrix-multiply – množenje rijetko popunjene matrice
  + lu-dense-matrix – factorization – faktorizacija gusto popunjenih matrica
  + fast-fourier-transformation - brza Fourierova transformacija
  + jacobi-successive-over-relaxation – algoritam „Jacobi Successive Over-Relaxation”
* Funkcionalni
  + future-genetic – test genetskog algoritma uz pomoć Jenetics biblioteke (renaissance-1.3.0)
  + gradle-perf – testovi performansi Gradle alata
  + eclipse – testovi performansi Eclipse razvojnog okruženja (dacapobench-1.0.1)
  + jython – brzina izvođenja Java implementacije Python programskog jezika (dacapobench-1.0.1)
  + jmh – biblioteka za Java mikro testove
  + bork – enkripcija datoteka, mjeri brzinu potrebnu za enkripciju neke datoteke
* Internet
  + finagle-http – performanse http zahtjeva (renaissance-1.3.0)
  + tradesoap – simulira trgovanje dionicama putem day trader platforme (dacapobench-1.0.1)
  + tradebeans - simulira trgovanje dionicama putem day trader platforme (dacapobench-1.0.1)
  + vertx – testovi performansi vertx biblioteke, ovaj test smo izradili kao kao primjer dodavanja testa u Phoronix Test Suite

Skup testova (eng. test suite) u Phoronix Test Suite u smo kreirali na sljedeći način:

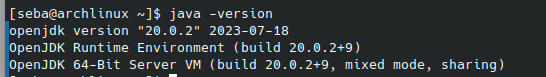
* Kreiramo skup testova uz naredbu:
  + phoronix-test-suite build-suite
* Ispunimo tražene podatke o nazivu, vrsti i autoru (eng. maintainer)
  + Naziv: java-jvm
  + Vrsta: Ostalo (eng. other)
  + Autor: Sebastijan Kelemen
* U izborniku dostupnih opcija sada možemo dodavati testove ili druge skupove testova (opcije 1 i 2). Dodali smo sve testove iz gore navedenih kategorija. Na primjer kako bi smo dodali test ALS algortima potrebno je dodati „pts/renaissance-1.3.0” (pod brojem 385) skup testova koji sadrži taj test.
* Kada smo dodali sve testove, odaberemo opciju za izlaz i spremanje (eng. Save and Exit). Naš skup testova je sada spremljen i možemo ga pokrenuti sa
  + phoronix-test-suite benchmark java-jvm
  + Phoronix Test Suite je pohranio naš skup testova kao XML datoteku koja sadrži informacije koje smo prethodno ispunili (ime, vrsta, autor, popis testova u skupu testa itd.)
  + Naš skup testova također možemo objaviti na repozitoriju testova OpenBenchmark.org

Sada možemo početi sa pokretanjem našeg skupa testova „java-jvm”. Prvo će biti potrebno odabrati jednu Java platformu koju želimo testirati. Zbog same arhitekture Java platformi prema JavaSE specifikaciji biti će potrebno testirati jednu po jednu platformu. Java programi su skup binarnih datoteka koje sadrže Java bit kod koji se učitava u Java virtualni stroj koji ih zatim izvršava. Zbog toga na primjer kako bi pokrenuli neki program napisan u Java programskom jeziku potrebno je pozvati java izvršni program (koji je dio prethodno instalirane Java platforme) u naredbenom retku koji će pokrenuti java virtualni stroj i započeti izvršavanje našeg programa. Testovi koje ćemo koristiti očekuju da je putanja do lokacije gdje smo instalirali određenu Java platformu bude dostupna na varijabli okruženja (eng. PATH variable). Stoga promjenom varijable okruženja prema lokaciji određene Java platforme možemo odabrati trenutno aktivnu Java platformu.

Provjeru trenutno aktivne Java platforme možemo napraviti na sljedeći način u naredbenom retku:

java -version

Ova naredba ispisuje verziju i vrstu Java platforme koja je dostupna na varijabli okruženja.

slika 2.1 Primjer za OpenJDK Java platformu

Kod završetka testiranja pojedine Java platforme Phoronix Test Suite kreira rezultate u obliku XML datoteke. Da bi smo dobili jedinstveni rezultat za sve tri Java platforme potrebno je objediniti rezultate koristeći opciju „merge-results” i popis trenutno spremljenih rezultata, na primjer:

phoronix-test-suite merge-results openjdk, openj9 graalVM

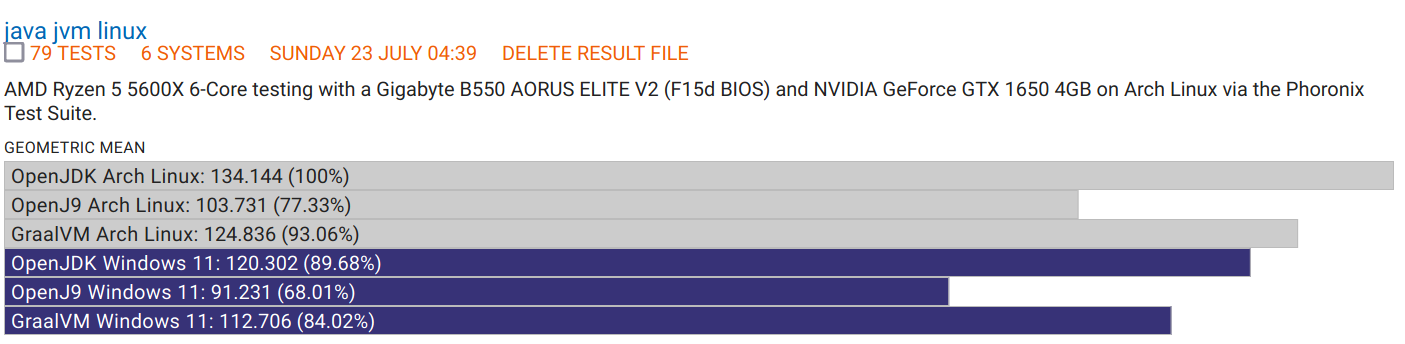
Gdje su openJDK, openJ9 i graalVM nazivi rezultata testiranja prethodnih testiranja Java platformi. Nakon što odaberemo ime objedinjenih rezultata, isti će biti spremljeni u obliku XML datoteka i mogu se pregledavati i uređivati poput svih ostalih rezultata.

Kada smo završili testiranja svih Java platformi na oba dva operativna sustava (Arch Linux i Windows) na isti ih gore opisani način objedinjujemo. Na ovaj način konačna XML datoteka sadrži informacije o svim Java platformama na svim operativnim sustavima.

## Rezultati testiranja

Rezultati testiranja se mogu vidjeti uz pomoć Phoronix Test Suite ugrađenog internet preglednika rezultata ili ih možemo izvesti u PDF ili HTML formatu.

Iz sumarnih rezultata, kada se uračunaju svi testovi vidljivo je da su sve tri testne Java platforme bile sveukupno brže na operativnom sustavu Arch Linux nego na Windows 11 operativnom sustavu. Također je vidljiva konzistentna razlika u brzini Java platformi neovisno o operativnom sustavu na kojem su instalirane. OpenJDK Java platforma je najbrža, zatim GraalVM, a najsporija je OpenJ9 Java platforma.

slika 3.1 Sumarni rezultati testiranja

Određeni testovi nisu mogli bit izvršeni zbog raznih programskih grešaka prilikom izvođenja testova. OpenJ9 je Java platforma sa najvećim brojem neuspjelih testova.

U tablici 1.7 prikazani su neuspjeli testovi te Java platforme na kojima test nije uspio. Ustvrdili smo da kod Java platforme OpenJ9 postoje razlike u implementaciji osnovnih Java klasa koje su dio Java platforme. Na primjer drugačija implementacija klase „ObjectInputStream” uzrukuje grešku kod svih dolje navedenih testova. U slučaju „movie-lens” testa koji nije uspio ni na jednoj Java platformi problem je u samom testu koji nije dobro napisan.

Tablica 1.7 Neuspjeli testovi

|  |  |
| --- | --- |
| Test | Java platforma |
| als | OpenJ9 |
| random-forest | OpenJ9 |
| movie-lens | OpenJ9, OpenJDK, GraalVM |
| naive-bayes | OpenJ9 |
| page-rank | OpenJ9 |
| tradebeans | OpenJ9 |
| tradesoap | OpenJ9 |

# Zaključak

Podaci dobiveni putem ovog rada mogu biti korisni prilikom odabira Java platforme u različitim poslovnim uvjetima. OpenJDK se pokazao kao trenutno najbrža Java platforma, što nije iznenađenje budući da se radi o neslužbenom standardu koji samim time ima najveći broj korisnika i razvojnih programera. Također se Arch Linux pokazao sveukupno bržim od Windows 11 operativnog sustava ma da razlika nije vrlo značajna. Testiranja izrađena u sklopu ovog rada tek su mali dio informacija o performansama različitih Java platformi. Java platforme kao i operativni sustavi imaju mnoštvo postavki koje bi značajnije mogle utjecati na rezultate testova. Metode i načini testiranja performansi koje smo pokazali u ovom radu mogu se lako primijeniti na ostala područja u računarstvu, (na primjer performanse hardvera poput na primjer procesora i grafičkih kartica.

# Popis slika

[slika 1.1 ShenandoahGC na Oracle OpenJDK platformi 11](#Slika!0|sequence)

[slika 2.1 Primjer za OpenJDK Java platformu 26](#Slika!1|sequence)

[slika 3.1 Sumarni rezultati testiranja 28](#Slika!2|sequence)

# Popis tablica

Napomena: na naslov **Popis tablica** primijenite stil Heading 1, a zatim ručno maknite brojčanu oznaku. Tablicu slika umećete na način da odaberete **Reference** - **References**u glavnom izbornikui zatim opciju **Umetni tablicu slika – Insert** **Table Caption**. Odaberite opciju da umećete Tablicu. Prihvatite sve zadane opcije.

# Popis kôdova

Napomena: na naslov **Popis kôdova** primijenite stil Heading 1, a zatim ručno maknite brojčanu oznaku. Tablicu slika umećete na način da odaberete **Reference** - **References**u glavnom izbornikui zatim opciju **Umetni tablicu slika – Insert** **Table Caption**. Odaberite opciju da umećete kôd. Prihvatite sve zadane opcije.

# Literatura

Svaki autor piše popis literature na kraju rada. Svaka stavka koja je navedena u popisu literature mora se nalaziti u radu. U popisu literature ne mogu biti stavke koje nisu korištene u radu.

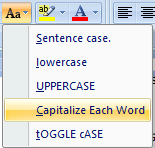
Popis literature se piše stilom literatura.

1. Stallings, W. *Data and Computer Networks*. London: John Wiley, 2006.
2. Stallings, W. *Local Computer Networks*. London: John Wiley, 2006a.
3. ATM Forum, User-Network Interface (UNI) Specification, http://www.atmforum.com, travanj. 2010.
4. Brady, P.T. A statistical Analysis of On-off Patterns in 16 Conversation, *Bell System Technical Journal*, 47,1 (1998), 55-62.
5. Brady, N. A statistical Analysis of Use Case. *Proceedings of the 7th International Conference on Telecommunications ConTEL,* Zagreb, (2003), 45-52.
6. Lilys, M. Final data structures. *Doktorski rad*. Sveučilište u Zagrebu, 2010.

Na naslov **Literatura** primijenite stil Heading 1, a zatim ručno maknite brojčanu oznaku (to je važno kako bi i skraćenice ušle u sadržaj na početku rada, prije uvoda). Pri kreiranju popisa literature koristite stil literatura.

Ako u popis literature upisujete Internet izvor važno je navesti puni link do korištenog dokumenta i datum kada je dokument preuzet.

Imenima autora i standardizacijskih organizacija potrebno je promijeniti ručno font u opciji **Home – Velika mala slova - Change Case**



i zatim odabrati opciju **Veliko početno slovo svake riječi** - **Capitalize Each World.**

Pozivanje na literaturu u tekstu se provodi na način:

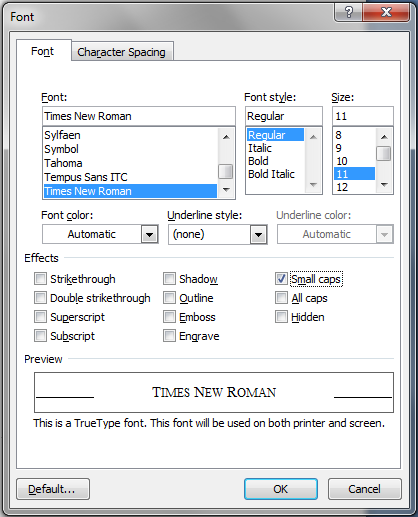
* Povezivanje lokalnih mreža s javnom mrežom provodi se uporabom usmjerivača (Martinis, 1998).

Ako za istu referencu postoji više autora, onda je referenciranje potrebno provesti kako slijedi:

* Povezivanje lokalnih mreža s javnom mrežom provodi se uporabom usmjerivača (Martinis et al., 1990),

Konačno, ako se na istu tvrdnju odnosi više od jedne reference, onda je zapis sljedeći:

* Povezivanje lokalnih mreža s javnom mrežom provodi se uporabom usmjerivača (Martinis, 1998; Alba, 1997).



Svaku stavku u literaturi treba u tekstu barem jednom referencirati.

# Prilog

Završni rad može imati priloge, ali se oni ne prilažu uz pisanu verziju završnog rada već se mogu priložiti na završnom ispitu ukoliko povjerenstvo na završnom ispitu tako odluči. Važno je čuvati svu poratnu dokumentaciju koja je nastala pri izradi završnog rada.

S unutarnje strane na zadnjim koricama originala, kao i svake kopije završnog rada, pričvršćuje se CD s kompletnim završnim radom u izvornom formatu (npr. .doc) i .pdf formatu sa svom popratnom dokumentacijom i programima. Pri čemu je obvezno da na tom CD- u postoji i dokument koji opisuje kako se rezultat njegova diplomskog rada (softver ili hardver) koristi (ili kako se npr. izvode mjerenja koja je opisao u radu). Ako se radi o softveru nužno je opisati i kako se programska podrška instalira.

Ova stranica predstavlja omot CD-a na kojem se treba nalaziti finalna verzija rada u digitalnom obliku i prilozi radu, ukoliko ih ima (objašnjenje u prethodnom poglavlju).

Ova stranica se ne uvezuje u ukoričenu verziju rada već se ispisuje, sastavlja se omot CD-a koji se pričvršćuje na posljednju stranicu korica.

**NASLOV ZAVRŠNOG RADA**



Pristupnik: Hrvoje Horvat, JMBAG

Mentor: Dobar Voditelj

Datum: 02. 02. 2006.