mapub.org/mapeh

MAP Education and Humanities

Volume 5

MAP Education and Humanities (MAPEH) is a scholarly peer-reviewed international scientific journal published by **MAP** - *Multidisciplinary Academic Publishing*, focusing on empirical and theoretical research in all fields of education and humanities.

E-ISSN: **2744-2373**

ORIGINAL RESEARCH PAPER

SYNCHEDUCA: EDUKACIJSKI ALAT ZA MEHANIZME SINKRONIZACIJE U PROGRAMSKOM JEZIKU JAVA

Miroslav Popović[1]

[1] ***Brightlightness, founder***, *Croatia*

Correspondence concerning this article should be addressed to Miroslav Popović, Brightlightness, founder, Croatia. E-mail: miroslav.popovic@brightlightness.com

ABSTRACT

A strong ICT sector enables economic growth in almost every business sector. Therefore, the quality of education provided to software engineers is an extremely important factor in fostering innovation and having a positive impact on society. Industry applications present challenges of effectively implementing programs comprising interdependent tasks which compete for computational resources. The concepts of building optimal synchronization of program execution are challenging for many students studying software engineering.

This paper provides an overview of projects and guidelines for improving the education of software engineers in respect to learning synchronization mechanisms, particularly focusing on usage of synchronization mechanisms from the Java package java.util. concurrent. The paper introduces the tool SynchEduca, which uses a visual interface to introduce students to synchronization concepts in the Java programming language in a more interactive and enjoyable way, without overshadowing the synchronization concepts with implementation details.

Keywords: software engineer education, concurrent program synchronization, synchronization in Java environment

MAP EDUCATION AND HUMANITIES

Volume 5

ISSN: 2744-2373/ © The Authors. Published by **MAP** - *Multidisciplinary Academic Publishing*.

Article Submitted: **19 June 2024**
Article Accepted: **11 July 2024**
Article Published: **12 July 2024**

https://doi.org/10.53880/2744-2373.2024.5.28

HOW TO CITE THIS ARTICLE

Popović M. (2024). **SynchEduca: Edukacijski alat za mehanizme sinkronizacije u programskom jeziku Java**. MAP Education and Humanities, 5, 28-36. doi: https://doi.org/10.53880/2744-2373.2024.5.28

1. Uvod

U današnjem brzo razvijajućem poslovnom okruženju, oslanjanje na robusna softverska rješenja za vođenje poslovnih procesa postalo je sveprisutno. Od automatizacije svakodnevnih zadataka do omogućavanja složene analize podataka, softver igra ključnu ulogu u osiguravanju operativne učinkovitosti i strateškog donošenja odluka. Tvrtke iz različitih sektora koriste softver za optimizaciju svojih radnih procesa, povećanje produktivnosti i održavanje konkurentske prednosti. Stoga, kvaliteta softvera izravno utječe na kvalitetu poslovnih operacija.

Oslanjanje na softver za kritične poslovne procese naglašava važnost kvalitete softvera. Kvalitetan softver osigurava pouzdanost, sigurnost i učinkovitost, što je ključno za održavanje povjerenja i zadovoljstva korisnika. Jednako je važna i kvaliteta podataka, budući da se poslovne odluke sve više temelje na analizi podataka. Loša kvaliteta podataka može dovesti do pogrešnih zaključaka, negativno utječući na strateško planiranje i operativno izvršenje. Stoga, tvrtke moraju ulagati u robusna softverska rješenja i stroge prakse upravljanja podacima kako bi zaštitile svoj operativni integritet i procese donošenja odluka.

Kvaliteta softvera inherentno je povezana s obrazovanjem i stručnošću inženjera koji ga razvijaju. Softverski inženjeri moraju posjedovati duboko razumijevanje programskih principa, struktura podataka, algoritama i arhitekture sustava. Osim toga, moraju biti vješti u upravljanju složenostima suvremenog razvoja softvera, uključujući sinkronizaciju paralelnih procesa, što je ključno za optimizaciju performansi i iskorištavanje resursa. Sposobnost implementacije učinkovitih mehanizama sinkronizacije posebno je izazovna, ali i ključna, za softverske inženjere koji razvijaju visokoperformansne aplikacije.

Poboljšanje obrazovanja softverskih inženjera stoga je imperativ za poticanje inovacija i održavanje visokih standarda kvalitete softvera koje zahtijevaju suvremene tvrtke. Tradicionalne metode poučavanja koncepta sinkronizacije često su nedostatne, jer mogu biti apstraktne i studentima teške za razumijevanje. Ovaj izazov dodatno se pogoršava nedostatkom interaktivnih i angažirajućih alata koji bi mogli učiniti učenje ovih pojmova pristupačnijim i intuitivnijim.

Kako bi se riješio ovaj obrazovni izazov, potrebni su inovativni alati i metodologije poučavanja. Jedno takvo rješenje je SynchEduca, alat osmišljen da studentima predstavi koncepte sinkronizacije putem vizualnog sučelja. SynchEduca ima za cilj učiniti učenje o sinkronizaciji interaktivnijim i zanimljivijim. Olakšanjem usvajanja znanja SynchEduca pomaže pripremiti sljedeću generaciju softverskih inženjera da zadovolje zahtjeve industrije. U nastavku rada opisan je alat SynchEduca, nakon čega slijedi pregled

2. Alat SynchEduca

SynchEduca je javna web aplikacija otvorenog koda (Popovic, 2018), stvorena za podučavanje novih generacija softverskih inženjera upotrebi mehanizama sinkronizacije. Budući da je programski jezik Java široko prisutan u industrijskoj praksi, te se koristi u mnogim poslovnim domenama za razvoj sustava koji obuhvaćaju skupove sinkroniziranih zadataka, SynchEduca se fokusira na mehanizme sinkronizacije dostupne u programskom jeziku Java prisutne unutar paketa java.util.concurrent. Mehanizmi sinkronizacije koje podržava alat SynchEduca uključuje CyclicBarrier, CountDownLatch, Semaphore i Phaser (Oracle Inc., 2024). Glavni cilj alata SynchEduca je pružiti okruženje za isprobavanje i igranje gdje studenti mogu eksperimentirati s mehanizmima sinkronizacije bez da budu preopterećeni složenošću implementacijske tehnologije. Alat SynchEduca je stvoren kao nastavak istraživanja opisanog u Popović i dr. (2018) i Popović (2011).

U jeziku Java, ali i u drugim jezicima, mnoge linije koda potrebne za izražavanje konstrukata kojima se postiže rješenje sinkronizacije mogu biti obeshrabrujuće za osobu koja tek počinje učiti o konkurentnom izvođenju zadataka. Čak i sam početak stvaranja novog programskog zadatka u obliku dretve u Java programskom jeziku predstavlja sasvim novi koncept izvođenja programske logike koji mnogi studenti smatraju izazovnim. Iz višegodišnjeg iskustva rada sa studentima, poteškoća nastaje pri mentalnom mapiranju tijeka izvršavanja koda. Shvaćanje da se stvara nova neovisna instanca izvršavanja koda s vlastitim programskim brojačem koji izvodi logiku definiranog segmenta koda dretve zna biti zbunjujuća. Potrebno je razumjeti temeljne mehanizme stvaranja procesa i dretvi, koji se oslanjaju na interakciju s jezgrom operacijskog sustava, da bi se razumjelo kako iz glavnog tijeka izvršavanja programa nastaje više tijekova izvrša-

vanja (Jakobović i dr., 2018; Prasad i dr., 2011; Hoare, 1974). Naredbe koje omogućuju takvo ponašanje koda nazivaju se sistemskim pozivima. Naredbe sistemskih poziva stupaju u interakciju s jezgrom operacijskog sustava i nisu intuitivne po svojoj sintaksi bez poznavanja gore navedenih složenosti.

Alat SynchEduca je stvoren u skladu sa smjernicama opisanim u Joint Task Force on Computing Curricula (2013) i savjetima u Scott (2009) kako se poduku ne bi započinjalo s teškim temama konkurentnosti tijeka izvođenja više-zadačnih programa. SynchEduca ublažava spomenute izazove oslanjajući se na vizualnu reprezentaciju tijeka naredbi unutar grafičkih prikaznika. Uz uklanjanje ove prve prepreke stvaranja više zadataka, SynchEduca otvara mogućnost za učenje sinkronizacije zadataka koji se konkurentno izvode pomoću specifičnih mehanizmima paketa java.util.concurrent programskog jezika Java. S vizualnom reprezentacijom mehanizama iz Java programskog jezika, SynchEduca može privući pažnju čak i onih studenata koji nisu razumjeli sistemske pozive za stvaranje procesa ili niti.

SynchEduca koristi grafičko sučelje za ilustraciju sinkronizacijskih koncepata kao što su CyclicBarrier, CountDownLatch, Phaser i Semaphore. Pristup koji je SynchEduca koristi pomaže shvaćanju navedenih mehanizama sinkronizacije te omogućuje studentima da se fokusiraju na razumijevanje temeljnih principa rada mehanizama umjesto da se opterećuju tehničkim detaljima. Omogućujući studentima da vizualno konstruiraju i manipuliraju sinkronizacijskim scenarijima, SynchEduca potiče dublje razumijevanje kako ti mehanizmi funkcioniraju i koja je njihova važnost u razvoju softvera.

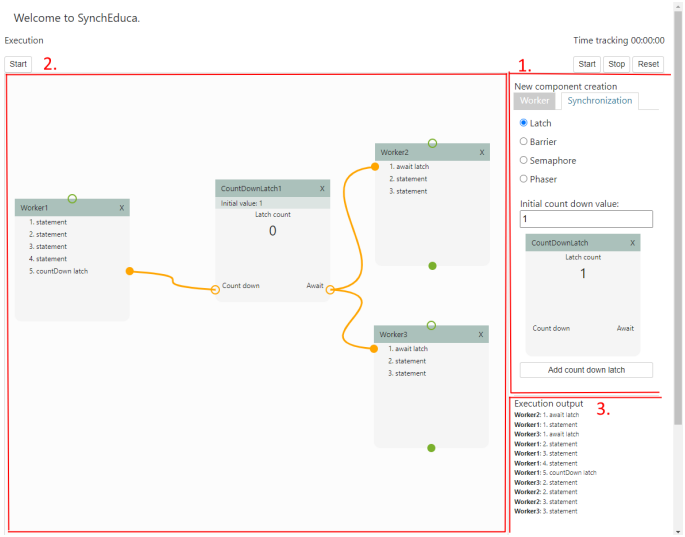
Interaktivna priroda alata također potiče aktivno učenje, za koje je dokazano da poboljšava zadržavanje pažnje i razumijevanje. Studenti mogu eksperimentirati s različitim sinkronizacijskim strategijama, promatrati njihove učinke u stvarnom vremenu i iterativno usavršavati svoje pristupe. Ovo praktično iskustvo neprocjenjivo je u pomaganju studentima da razviju praktične vještine potrebne za implementaciju učinkovite sinkronizacije u stvarnim aplikacijama.

2.1 Interaktivna sinkronizacijska okolina

Razvojno okruženje alata SynchEduca zahtijeva malo ili nikakvo znanje o sinkronizaciji u Java programskom okruženju. Stvaranje konkurentnih tijekova izvođenja u alatu SynchEduca temelji se na kreiranju i povezivanju bilo kojeg broja grafičkih prikaznika. Prikaznici (eng. widgets) se stvaraju pomoću sučelja desne bočne trake (1) i mogu se pozicionirati povlačenjem (eng. Drag-and-drop) radi preuređenja njihovog položaja u radnom prostoru (2). Slikom 1. prikazan je izgled razvojnog okruženja SynchEduca.

Grafička povezanost prikaznika izražena je putem točaka sučelja za interakciju među prikaznicima. Da li je povezivanje moguće ovisi o specifičnim tipovima sučelja prikaznika. Moguće je povezati samo odgovarajuća sučelja dvaju prikaznika ako su označena istom bojom. Prikaznici su podijeljeni u dvije skupine prema njihovoj ulozi u izvođenju tijeka rada, a izvođenje progamske logike obuhvaćene prikaznicima pokreće se klikom na gumb Start. Ishod izvođenja naredbi moguće je pratiti pomoću sučelje koje se nalazi na dnu desne bočne trake (3)

Slika 1.
Razvojno okruženje alata SynchEduca.



Izvršni prikaznici sadrže popis naredbi koje se izvršavaju jedna za drugom (na imperativan način). Naredbe mogu biti obične naredbe ili bilo koja od podržanih sinkronizacijskih primitiva koje komuniciraju sa sinkronizacijskim prikaznicima. Sinkronizacijski primitivi komuniciraju sa specifičnim mehanizmom sinkronizacijskog prikaznika i, ovisno o stanju sinkronizacijskog mehanizma, mogu blokirati daljnje izvršavanje naredbi u radnom prikazniku. Složeniji kontrolni tokovi, poput petlji i uvjetnog grananja, još nisu podržani za izvršni prikaznik.

Sinkronizacijski prikaznici pružaju grafičko pojednostavljenje mehanizama sinkronizacije u programskom okruženju Java. Sinkronizacijski prikaznici ne sadrže naredbe. Umjesto toga, sadrže stanje koje se konfigurira tijekom stvaranja prikaznika. Prikaznik također sadrži sučelja (grafička) koja omogućuju promjenu stanja na temelju specifične logike mehanizma ugrađene u prikaznik. Na izloženo sučelje moguće je spojiti samo odgovarajući primitiv iz izvršnog prikaznika, te će izvođenje primitiva u izvršnom prikazniku izazvati promjenu stanja mehanizma u sinkronizacijskom prikazniku.

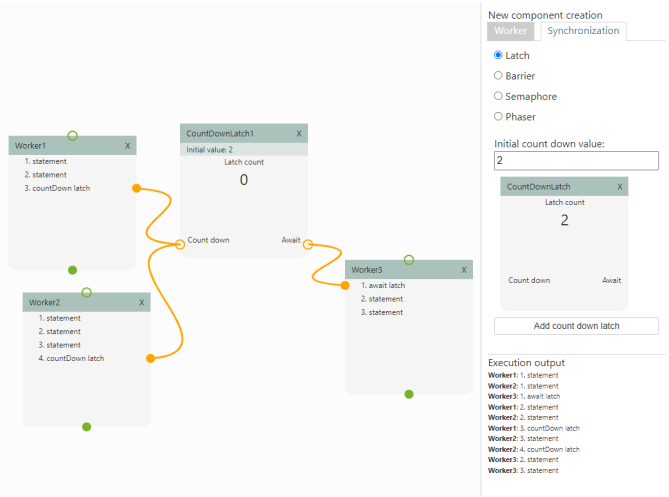
2.2 Sinkronizacija pomoću CountDownLatch mehanizma

SynchEduca nudi mehanizam koji pomaže interaktivnom učenju sinkronizacijskog mehanizma CountDownLatch iz java.util.concurrent paketa programskog jezika Java. U alatu SynchEduca mehanizam CountDownLatch je vizualna reprezentacija koja oponaša semantiku CountDownLatch mehanizma iz java.util.concurrent paketa dizajniranog za upravljanje ovisnostima između izvršitelja zadataka (dretvi). Mehanizam funkcionira definiranjem početnog broja koji predstavlja broj prethodećih zadataka koji trebaju dovršiti svoje izvođenje prije nego što započne izvođenje drugih zadataka. U takvom scenariju izvršitelji prethodećih zadataka po završetku izvođenja zadanih naredbi smanjuju navedeni broj CountDownLatch mechanizma pozivanjem metode *countDown*. Ostali izvršitelji mogu pozvati metodu *await* kako bi obustavili svoje izvođenje dok broj u CountDownLatch mehanizmu ne dosegne nulu, time osiguravajući da su svi potrebni zadaci dovršeni prije nego što se nastavi izvođenje njihove logike.

CountDownLatch mehanizam je posebno koristan kada je potrebno osigurati da više pripremnih zadataka dovrši izvođenje prije nego što glavni zadatak nastavi s izvođenjem daljnjih operacija. Na primjer, moguće je imati više zadataka koji učitavaju podatke, a glavni zadatak treba obraditi sve učitane podatke. Korištenjem CountDownLatch mehanizma, glavni zadatak može čekati dok svi zadaci učitavanja podataka ne završe izvođenje, osiguravajući dostupnost podataka prije nego što obrada započne.

Slika 2. prikazuje izvođenje primjera sinkronizacijskog scenarija koristeći mehanizam CountDownLatch u alatu SynchEduca. U scenariju tri izvršitelja sinkroniziraju svoje tijekove rada. Izvršavanje naredbi izvršitelja Worker3 ne smije započeti prije nego izvršitelji Worker1 i Worker2 izvrše svoje naredbe. Worker1 mora izvršiti dvije naredbe, a Worker2 mora izvršiti tri naredbe kako bi pripremili imaginarne resurse za izvršitelja Worker3. Nakon što Worker1 i Worker2 izvrše naredbe koje pripremaju resurse za Worker3, svaki od njih izvršava naredbu *countDown*. *CountDownLatch1* mehanizam koji je inicijalno postavljen na vrijednost dva, bit će smanjen ovim naredbama i njegova će vrijednost dosegnuti nulu. Nakon što dosegne vrijednost nula, *CountDownLatch1* će odblokirati sve izvršitelje koji su bili blokirani u svom izvršavanju. U ovom slučaju, samo je Worker3 bio blokiran u svom izvršavanju, budući da je izvršio naredbu *await* prije nego što je vrijednost odbrojavanja smanjena na nulu. Kada je vrijednost odbrojavanja smanjena na nulu, Worker3 je nastavio izvršavanje svojih naredbi. Primijetite da vrijednost odbrojavanja ostaje nula nakon završetka ovog scenarija.

Slika 2.
Primjer korištenja mehanizma CoutDownLatch u alatu SynchEduca.

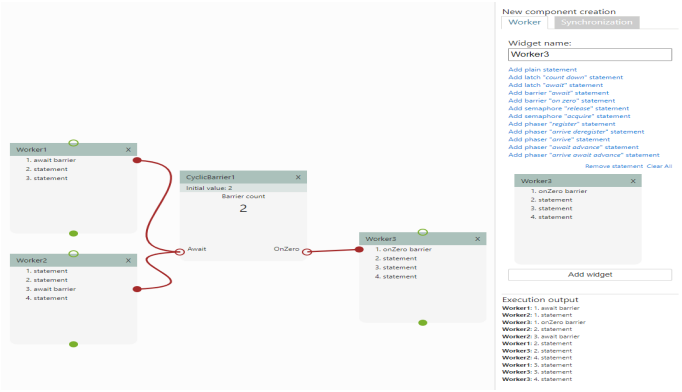


2.3 Sinkronizacija pomoću CyclicBarrier mehanizma

Alat SynchEduca nudi mehanizam koji pomaže u interaktivnom učenju sinkronizacijskog mehanizma CyclicBarrier iz paketa java.util.concurrent programskog jezika Java. U alatu SynchEduca mehanizam CyclicBarrier je vizualna reprezentacija koja nalikuje semantici mehanizma CyclicBarrier iz paketa java.util.concurrent. CyclicBarrier je dizajniran kako bi omogućio skupini izvršitelja da čekaju jedni druge kako bi svi zajedno stigli do zajedničke barijere. Kada svi sudionici dostignu ovu barijeru, mogu nastaviti sa svojim izvršavanjem. Za razliku od CountDownLatch, koji se koristi samo jednom,

CyclicBarrier se može ponovno koristiti, što ga čini pogodnim za scenarije u kojima je potrebno više puta čekati da grupa izvršitelja stigne do zajedničke faze izvođenja.

Slika 3.
Primjer korištenja mehanizma CyclicBarrier u alatu SynchEduca.



CyclicBarrier osigurava da svi izvršitelji dostignu određenu fazu izvođenja prije nego što bilo koji od njih nastavi dalje. Ovo je posebno korisno u iterativnim algoritmima gdje se izvršitelji trebaju sinkronizirati na više faza izvršavanja. Na primjer, u simulaciji ili računskom algoritmu gdje više radnika paralelno obrađuje podatke, može se zahtijevati da svaki radnik trebati doći do sinkronizacijske točke prije nego što započne sljedeću iteraciju. Korištenjem CyclicBarrier mehanizma osigurava se da svi radnici čekaju jedni druge dok ne završe trenutnu iteraciju prije nego što krenu na sljedeću. CyclicBarrier također može imati registriranu akciju barijere, a ona je poseban zadatak koji se izvršava kada svi radnici dosegnu točku barijere. Akcija barijere je korisna za izvođenje zadatka koja se mora izvršiti prije nego radnici nastave u sljedeću iteraciju izvršenja. Primjer akcije barijere može biti spajanje rezultata ili ažuriranje zajedničkih resursa.

Slika 3. prikazuje izvršavanje primjera sinkronizacijskog scenarija korištenjem CyclicBarrier mehanizma u SynchEduca. U scenariju, Worker1 i Worker2 sinkroniziraju svoje izvođenje izvršavanjem *await* naredbe. Nakon izvršenja naredbe, početna vrijednost brojača barijere se smanjuje i izvršitelji ostaju blokirani na toj naredbi dok brojač barijere ne dostigne nulu. Worker1 ostaje blokiran dok Worker2 ne izvrši *await* naredbu. Ovo rezultira time da brojač barijere dostigne vrijednost nula, koja resetira brojač na početnu vrijednost, oslobađa sve izvršitelje koji su bili blokirani *await* naredbom i izvršava zadatak koji je bio registriran za izvršenje *onZero* barijernom naredbom. U ovom slučaju, vrijednost nula brojača barijere omogućuje izvršiteljima Worker1, Worker2 i Worker3 da nastave s izvršavanjem naredbi od točke gdje su bili blokirani.

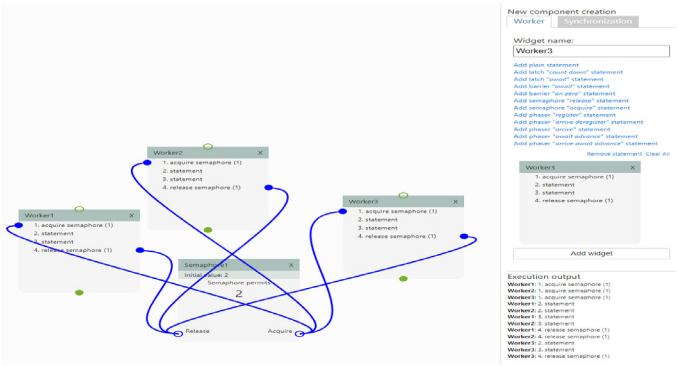
2.4 Sinkronizacija pomoću Semaphore mehanizma

SynchEduca nudi mehanizam koji pomaže interaktivnom učenju sinkronizacijskog mehanizma Semaphore iz paketa java.util.concurrent programskog jezika Java. U alatu SynchEduca mechanizam Semaphore je vizualni prikaz koji oponaša semantiku mehanizma Semaphore iz paketa java.util.concurrent. Mehanizam funkcionira na način da održava skup dozvola koje radnici moraju pribaviti prije nego što mogu nastaviti s izvršavanjem zadataka. Kada radnik pribavi dozvolu, broj dostupnih dozvola se smanjuje. Kada radnik oslobodi dozvolu, broj dostupnih dozvola se povećava. Ako nema dostupnih dozvola, radnici koji pokušavaju pribaviti dozvolu bit će blokirani dok dozvola ne postane dostupna. Scenarij sinkronizacije primjenom mehanizma Semaphore moguće je zamišljati kao ulaz na popularan događaj pomoću skupu ulaznica. Postoji samo ograničen broj dostupnih ulaznica, i kada su sve zauzete, novi posjetitelji ne mogu ući dok netko ne napusti događaj i vrati ulaznicu.

U okruženju s više dretvi, Semaphore ograničava broj dretvi koje mogu istovremeno pristupiti određenom resursu, osiguravajući kontrolirani pristup i sprječavajući sukobe oko resursa. Primjer je server koji rukuje s više klijentskih veza. Kako bi se spriječilo preopterećenje i osigurala stabilna izvedba, server može koristiti Semaphore za ograničavanje broja istovremenih veza. Pri uspostavi novih veza pribavlja se dozvolu i dozvolu se oslobađa nakon zatvaranja veze, omogućujući uspostavljanje drugih veza. Drugi uobičajeni slučaj korištenja je upravljanje pristupom skupu resursa, poput veza s bazom podataka ili dretvi u dretvenom skupu.

Slika 4. prikazuje izvođenje primjera sinkronizacije koristeći mehanizam Semaphore u alatu SynchEduca. U scenariju, semafor je inicijalno postavljen da sadrži dvije dozvole. Sva tri radnika izvode isti niz naredbi. Radnici pokušavaju dobiti jednu dozvolu od semafora *Semaphore1*. Budući da postoje samo dvije dozvole, samo će dvojica radnika uspjeti, dok će jedan radnik ostati blokiran u pokušaju dobivanja dozvole. U ovom slučaju Worker3 nije dobio dozvolu za nastavak. Radnik ostaje blokiran dok jedan od radnika ne vrati dozvolu u semafor Semaphore1.

Slika 4.
Primjer korištenja mehanizma Semaphore u alatu SynchEduca.



2.5 Sinkronizacija pomoću Phaser mehanizma

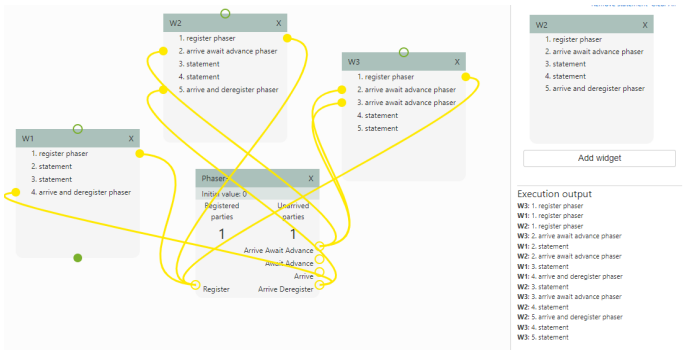
Alat SynchEduca nudi mehanizam koji pomaže interaktivnom učenju mehanizma sinkronizacije Phaser iz paketa java.util.concurrent Java programskog jezika. U alatu SynchEduca mehanizam Phaser je vizualni prikaz koji nalikuje semantici mehanizma Phaser iz paketa java.util.concurrent. Phaser je napredna verzija tradicionalnih mehanizama sinkronizacije poput CyclicBarrier i CountDownLatch, nudeći veću fleksibilnost i kontrolu nad procesom sinkronizacije.

Osnovna svrha Phaser-a je omogućiti skupini radnika da se više puta međusobno čekaju kako bi došli do zajedničke točke sinkronizacije, poznate kao faza. Svaka faza predstavlja točku izvođenja u koju svi sudjelujući radnici moraju stići u izvršavanju svojih zadataka prije nego što ijedan od njih može nastaviti na sljedeću fazu. Ovo svojstvo čini mehanizam Phaser posebno korisnim za scenarije gdje grupa radnika treba zajedno raditi u više koraka ili etapa, s potrebom sinkronizacije na svakom koraku.

Rad mehanizma Phaser zasniva se na registriranju radnika koji sudjeluju u procesu sinkronizacije. Radnici se oslanjaju na operaciju *arrive* mehanizma Phaser pomoću koje označavaju da su došli do točke sinkronizacije, a zatim čekaju nastavak izvođenja kad nastupi sljedeća faza. Jednom kada svi registrirani sudionici stignu, Phaser prelazi na sljedeću fazu, omogućujući svim registriranim sudionicima da nastave svoje izvršavanje.

Jedna od ključnih značajki mehanizma Phaser je njegova sposobnost upravljanja dinamičkim promjenama u broju radnika koji sudjeluju u fazi sinkronizacije. Za razliku od mehanizma CyclicBarrier, koji zahtijeva fiksni broj stranaka, Phaser dopušta sudionicima sinkronizacije da se dinamički registriraju i odjavljuju tijekom izvršavanja zadataka. To ga čini vrlo prilagodljivim za različita radne opterećenja i složene zahtjeve sinkronizacije.

Slika 5.
Primjer korištenja mehanizma Phaser u alatu SynchEduca.



Slika 5 prikazuje izvođenje primjera sinkronizacije koristeći mehanizam Phaser u alatu SynchEduca. Phaser vizualno prikazuje broj registriranih i nepristiglih stranaka. Primjer pokazuje kako se Phaser koristi za sinkronizaciju broja stranaka koje se dinamički uključuju i odjavljuju iz sinkronizacije. Scenarij počinje s Phaser-om koji je inicijalno postavljen na nula registriranih i nepristiglih stranaka. Radnici W1, W2 i W3 registriraju se za prvu fazu izvršavanja. Prva faza završava nakon što radnik W1 primjenom naredbe *arrive and deregister* pristiže i odjavljuje se iz Phaser-a. Radnici W2 i W3 nastavljaju na drugu fazu. Druga faza završava nakon što radnik W2 primjenom naredbe *arrive and deregister* pristiže i odjavljuje se iz Phaser-a. Radnik W3 sam nastavlja na treću fazu izvršavanja. U ovom slučaju, Phaser je primijenjen da međusobno isključi istovremeno izvršavanje zadataka W1, W2 i W3. Kada se iz redoslijeda izvršavanja zanemare sinkronizacijski primitivi, uočava se da je Phaser-om u ovom primjeru nametnuto sekvencijalno izvršavanje naredbi iz zadataka W1 → W2 → W3.

3. Pozicioniranje alata SynchEduca u edukacijskoj primjeni

Kvaliteta obrazovanja softverskih inženjera ključni je čimbenik inovacija u ICT sektoru, što zauzvrat potiče gospodarski rast u različitim industrijama. Prepoznajući važnost obrazovanja softverskih inženjera, provedena su brojna istraživanja i razvijeni alati i metodologije usmjereni na poboljšanje obrazovnog iskustva studenata softverskog inžen-

jerstva, posebno u području konkurentnog programiranja i sinkronizacije.

3.1 Obrazovni alati i platforme

Razvijeno je nekoliko obrazovnih alata koji pomažu u podučavanju složenih koncepta softverskog inženjerstva. Na primjer, obrazovna okruženja poput Greenfoot (Henriksen & Kölling, 2004) i BlueJ (Kölling i dr., 2003) široko se koriste za podučavanje objektno orijentiranog programiranja putem vizualnih sučelja i interaktivnih simulacija. Alati poput Jeliot (Ben-Ari, 2002) i Alice (Cooper, 2000) nude slične vizualne i interaktivne pristupe, uglavnom usmjerene na uvodne tečajeve programiranja. Ti alati pomažu studentima da razumiju koncepte programiranja pružajući opipljiviju i vizualniju reprezentaciju apstraktnih ideja. Međutim, njihov fokus je prvenstveno na osnovnim vještinama programiranja. VisuAlgo (Halim, 2015) je više usmjeren na napredne programske algoritme i strukture podataka. Dok ove platforme uspješno angažiraju studente i poboljšavaju njihovo znanje o programskim algoritmima i strukturama podataka, one su nedostatne kada je riječ o podučavanju tema poput sinkronizacije i konkurentnog programiranja.

3.2 Podučavanje koncepata sinkronizacije

Podučavanje sinkronizacije i konkurentnog programiranja tradicionalno se oslanja na teorijske tečajeve i vježbe programiranja zasnovane na principu objašnjavanja primjenom ploče i krede. Klasični udžbenici, poput Gagne i dr. (2018), pružaju dubinska teorijska objašnjenja i primjere. Iako su klasični udžbenici kao resursi neprocjenjivi, mogu biti izazovni za studente koji ih usvajaju bez praktičnog iskustva.

Kroz literaturu, vizualizacija i interaktivno učenje su također primijenjeni u podučavanju sinkronizacije i konkurentnog programiranja. ThreadMentor (Carr i dr., 2003) je višeplatformski pedagoški alat dizajniran da pojednostavi podučavanje i učenje višedretvenog programiranja. Pružajući sustav vizualizacije koji prikazuje interakcije dretvi i sinkronizacijske primitive, omogućuje studentima da razumiju ponašanje programskih dretvi. Jedan od pristupa viđenih u literaturi za pomoć studentima u savladavanju konkurentnog programiranja je vizualizacijski alat Convit (Järvinen i dr., 2005), Java aplet koji simulira konkurentne programe, omogućujući studentima da istražuju i mijenjaju sinkronizacijske naredbe u pseudokodnom okruženju.

3.3 Doprinosi alata SynchEduca

Alat SynchEduca je stvoren u skladu sa smjernicama opisanim u Joint Task Force on Computing Curricula (2013). Ističe se od ostalih edukacijskih alata posebnim fokusom na obrazovne izazove povezane s podučavanjem sinkronizacijskih koncepta iz Java paketa java.util.concurrent. Iako u smjernicama nije eksplicitno zadano programsko okruženje, često se u industriji primjenjuje Java programsko okruženje za izvođenje sinkronizacije više-zadačnih programa. U literaturi nije zabilježen rad koji stvaranjem interaktivnog web simulacijskog okruženja popularizira primjenu mehanizama iz Java paketa java.util.concurrent. Alat SynchEduca je prilagođen da pruži usredotočeno i interaktivno učenje sinkronizacije koristeći interaktivna vizualna sučelja koja omogućuju studentima eksperimentiranje s sinkronizacijskim primitivcima poput CyclicBarrier, CountDownLatch, Phaser i Semaphore.

Pružajući angažirano, praktično iskustvo, SynchEduca pomaže studentima da shvate složene koncepte sinkronizacije bez da budu preplavljeni detaljima tehničke implementacije. Ovaj pristup ne samo da poboljšava razumijevanje, već i priprema studente da učinkovito primjenjuju te koncepte u stvarnim scenarijima razvoja softvera.

U usporedbi s tradicionalnim teorijskim pristupima i općim programskim alatima, SynchEduca nudi nekoliko prednosti:

1. **Interaktivno učenje**: Studenti mogu manipulirati i promatrati učinke sinkronizacijskih mehanizama u stvarnom vremenu, potičući aktivno učenje i bolje zadržavanje.

2. **Integracija usmjerena na kurikulum**: Alat je dizajniran da se besprijekorno integrira u tečajeve softverskog inženjerstva.

3. **Dostupnost putem interneta**: SynchEduca je alat za edukacija koji nema zahtjeva za instalacijom, te je dostupan iz preglednika, što ga čini lakšim za korištenje od strane studenata.

4. **Opipljivo znanje uz apstrakciju implementacijske tehnologije**: SynchEduca smanjuje početnu barijeru znanja potrebnu za sinkronizaciju konkurentnih zadataka tako što ne zasjenjuje koncepte sinkronizacije s implementacijom u specifičnom programskom jeziku, a pruža konkretna

znanja primjene sinkronizacijskih mehanizama iz Java paketa java.util.concurrent.

5. **Korištenje bez podrazumijevanog znanja**: S vizualnom reprezentacijom mehanizama iz Java programskog jezika, SynchEduca može privući pažnju čak i onih studenata koji nisu vješti u programskom jeziku Java, kao i onih studenata koji nisu razumjeli sistemske pozive za stvaranje procesa ili niti.

4. Zaključak

Kvaliteta softvera koji pokreće moderne poslovne procese uvelike ovisi o obrazovanju i vještinama softverskih inženjera. Poboljšanje ovog obrazovanja, posebno u složenim područjima poput sinkronizacije, ključno je za osiguranje da softverski inženjeri mogu razvijati kvalitetan i pouzdan softver. SynchEduca predstavlja značajan korak naprijed u tom pogledu, nudeći inovativan i učinkovit alat za podučavanje koncepta sinkronizacije. Čineći ove koncepte pristupačnijima i zanimljivijima, SynchEduca pomaže pripremiti sljedeće generacije softverskih inženjera da zadovolje zahtjeve industrije i potaknu buduće tehnološke napretke.

U ovom radu prikazan je pregled radova koji su posvećeni edukaciji softverskih inženjera. Poseban naglasak stavljen je na podučavanje upotrebe sinkronizacijskih mehanizama, a još specifičnije na upotrebu mehanizama sinkronizacije. Doprinos alata SynchEduca je pozicioniran u postojećoj edukacijskoj praksi s obzirom na specifičnost korištenja sinkronizacijskih mehanizama iz Java programskog paketa java.util.concurrent. Iako postoji mnogo alata i platformi dostupnih za podučavanje programiranja i osnovnih koncepta konkurencije, SynchEduca adresira specifičnu potrebu za interaktivnim pristupom igre pokušaja i pogrešaka u podučavanju sinkronizacije. Iskorištavajući vizualne i praktične metode učenja, SynchEduca poboljšava obrazovno iskustvo, osiguravajući da studenti softverskog inženjerstva budu dobro pripremljeni za suočavanje sa složenošću konkurentnog programiranja u svojim profesionalnim karijerama koristeći mehanizme iz Java paketa java.util.concurrent.

5. Literatura

Ben-Ari, M., Myller, N., Sutinen, E., & Tarhio, J. (2002). Perspectives on program animation with Jeliot. In S. Diehl (Eds.), *Lecture Notes in Computer Science,* vol 2269. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45875-1_3

Carr, S., Mayo, J., & Shene, C. (2003). Thread-Mentor: a pedagogical tool for multithreaded programming. ACM J. Educ. Resour. Comput., 3, 1.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, 15(5), 107-116.

Gagne, G., Silberschatz, A., & Galvin, P. B., (2018). *Operating System Concepts* (10th ed.). Wiley

Halim, S. (2015). VisuAlgo - Visualising Data Structures and Algorithms Through Animation. Retrieved June 19, 2024 from https://visualgo.net/en

Henriksen, P., & Kölling, M. (2004). Greenfoot: combining object visualisation with interaction. Conference on Object-oriented programming systems, languages, and applications (OOPSLA '04), 73–82. https://doi.org/10.1145/1028664.1028701

Hoare, C. A. R. (1974). Monitors: An operating system structuring concept. *Communications of the ACM*, 17(10), 549-557.

Jakobovic, D., Budin, L., Jelenkovic, L., & Golub, M. *Operacijski sustavi* (4. izdanje). (2018). Element: Zagreb

Järvinen, H., Tiusanen, M., & Virtanen, A. (2005). Convit , a Tool for Learning Concurrent Programming.

Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science; 2013; ACM; New York (NY), USA.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ System and its Pedagogy. Computer Science Education, 13(4), 249–268. https://doi.org/10.1076/csed.13.4.249.17496

Oracle Inc. () CountDownLatch documentation, a part of java.util.concurrent package. Retrieved June 17, 2024 from https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CountDownLatch.html

Oracle Inc. CyclicBarrier documentation, a part of java.util.concurrent package. Retrieved June 17, 2024 from https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CyclicBarrier.html

Oracle Inc. Semaphore documentation, a part of java.util.concurrent package. Retrieved June 17, 2024 from https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Semaphore.html

Oracle Inc. Phaser documentation, a part of java.util.concurrent package. Retrieved June 17, 2024 from https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Phaser.html

Popovic, M. (2018). SynchEduca. Retreived June 17, 2024 from https://gitlab.com/popmir/synch_educa_plumb

Popović, M., Vladimir, K., & Šilić, M. (2018). Application of social game context to teaching mutual exclusion. *Automatika*, *59*(2), 208–219. https://doi.org/10.1080/00051144.2018.1522462

Popovic, M. (2011). *Sinkronizacija potrošačkih programa* [Doktorska disertacija, Fakultet elektrotehnike i računarstva: Sveučilište u Zagrebu]. 540821. https://www.bib.irb.hr:8443/540821

Prasad, S. K., Chtchelkanova, A., Das, S., Dehne, F., Gouda, M., Gupta, A., ... & Wu, J. (2011, March). NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: core topics for undergraduates. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 617-618).

Scott, M. L. (2009, March). Don't start with Dekker's algorithm: Top-down introduction of concurrency. *In Workshop on Multicore Programming Education*.