ПРОЈЕКАТ ИЗ ПРЕДМЕТА
КОНКУРЕНТНО И ДИСТРИБУИРАНО ПРОГРАМИРАЊЕ
ЗА ШКОЛСКУ 2022/2023
У ЈУНСКОМ И ЈУЛСКОМ ИСПИТНОМ РОКУ

Пројекат из предмета 13Е113КДП и 13С113КДП се у се у јунском и јулском испитном року школске 2022/2023 ради самостално.

Пројекат се ради у програмском језику Јава. Пројекат носи не више од 20 поена.

Предуслови за успешну одбрану пројекта су:

1. Уписана одговарајућа година ЕТФ одговарајућег смера.
2. Благовремена достава писаних материјала и електронске верзије решења (најмање 3 дана пре испита у испитном року или по упутству послатом на листу предмета).
3. Благовремено припремљени услови за неометану проверу рада програма у лабораторији катедре за РТИ ЕТФ-а (уколико је потребно, барем три дана пре одбране потребно је инсталирати одговарајуће програме у договору са дежурним лаборантом).
4. Успешно обављено усмено одбрана рада.

Усмена одбрана рада се састоји из следећег:

1. Кандидат који брани пројекат мора самостално да преведе и инсталира све потребне програме везане за приложено решење.
2. Кандидат мора да поседује потребан ниво знања о задатку.
3. Кандидат мора да буде свестан недостатака предложеног решења и могућности за њихово превазилажење.
4. Кандидат треба да тачно одговори на потребан број питања која се баве тематиком везаном за задатак.

Израда писаних материјала везаних за пројекат подразумева поштовање одговарајуће форме. Према тој форми сваки пројекат треба да има следеће елементе:

1. Насловну страну са јасно израженим обележјима који карактеришу овај факултет и овај предмет. Мора да садржи назив и лого факултета, назив предмета из кога се пројекат брани, назив задатка који се ради, пуно име и презиме аутора, као и број индекса, датум када је начињена прва верзија, датум када је настала текућа верзија и место где је одбрана вршена.
2. На првој страници после наслова рада и имена аутора следи садржај на српском језику писан курзивом - ***Italic*** фонтом *Times New Roman 10 pt* ћириличним писмом.
3. На наредној страни треба да се налази текст задатка који се ради. Уколико текстом задатка нешто није било довољно јасно назначено посебно уоквирити делове који су додати. Уколико предложено решење поседује известан број недостатака, њих назначити на посебан и лако уочљив начин, и предложити алтернативно решење које би отклонило наведене недостатке.
4. У наставку је потребно дати детаљан опис предложеног решења и свих његових карактеристика (овде није потребно стављати имплементационе детаље већ функционалне који су од суштинске важности за разумевање пројекта).
5. Након функционалне спецификације потребно је дати детаљан опис пакета, класа, интерфејса, функција и параметара, користећи **UML** спецификацију за опис интеракције (**дијаграм интеракције**).

6. Упутство за коришћење насталог програмског пакета као целине. Упутство треба да покрива два типа коришћења програмског пакета:

a) коришћење у регуларним ситуацијама од стране особе чији је ниво рачунарског знања минималан, а која нема претходно искуство у раду са сличним пакетима
b) коришћење насталог програмског пакета особе чији је ниво познавања потребних вештина на задовољавајућем нивоу у циљу даљег усавршавања система.

7. Листинг програма са потребном количином коментара није потребно предавати у штампаном већ у електронском облику.
8. Примери рада програма у регуларним и ванредним ситуацијама, са потребним објашњењима.
9. Рад писати на српском језику уз чување оригиналних енглеских термина.

Оригинал рада треба да буде откуцан само са једне стране листова А4 формата (210 x 297 mm). Користити маргине: **2.5 cm** горња, **2 cm** доња, лева и десна. Рад треба буде писан ћириличним писмом уз коришћење фонта *Times New Roman*, величина писма: 10 типографских тачака *(10 pt)* у две колоне размакнуте **0,5 cm** уз поравнање типа *justify*. Рад куцати обичним проредом и двоструким проредом између пасуса. Почетак пасуса куцати од почетка колоне. Поднаслове у раду писати масним словима (**Bold)** великим словима величина писма: 12 типографских тачака. Сва слова у раду треба да буду црна, а позадина бела. Све табеле и слике треба да имају одговарајући наслов и да буду нумерисане. Бројеви и наслови табела налазе се увек изнад табела. Бројеви и наслови слика налазе се увек испод слика. Нумерација страна се пише у доњем десном углу. Насловна страна се не нумерише. Насловна страна треба буде писан ћириличним писмом уз коришћење фонта *Arial* у једној колони уз поравнање типа *center*. Насловна страна садржи: Назив Универзитета (величина писма: 16 типографских тачака); Назив факултета (величина писма: 16 типографских тачака); Име, средње слово, презиме и број индекса студента (величина писма: 16 типографских тачака); Наслов пројекта (величина писма: 22 типографске тачке); Предмет из кога се пројекат ради (величина писма: 16 типографских тачака); Место, година (величина писма: 14 типографских тачака).

НАПОМЕНА: Непоштовање горе наведених правила вуче умањење освојеног броја поена на усменој одбрани пројекта или у потпуности забрањује исту.

Пројекат јун 2022/2023
Дистрибуирана симулација

Пројектовати дистрибуирани рачунарски систем који треба да омогући да већи број рачунара у мрежи извршава дистрибуирану симулацију засновану на симулатору дискретних догађаја. Програм треба да ради у систему који се састоји од више рачунара повезаних у LAN (Local Area Network) или WAN (Wide Area Network).

У систему постоји три типа програма:

1. Централни сервер који служи за пријем и праћење рада извршавања симулације, чување информација о доступним чворовима у мрежи и могућност поновног стартовања појединих симулација.
2. Радна станица која служи за извршавање симулација које добија од централног сервера као листу логичких компонента и веза између њих. Радна станица, током рада, размењује поруке које су јој упутиле друге радне станице или централни сервер.
3. Кориснички програм, који задаје листу логичких компонената које је потребно симулирати као, везе између тих логичких компонената, као и потребних параметара везаних за симулацију.

Процес започиње тако што кориснички програм задаје аргументе логичке шеме које је потребно симулирати. Шема се састоји из листе логичких компонената, веза која описују како су повезани излази једних компоненти са улазима дугих компоненти, као и типа симулације коју жели да покрене, као и крајњег, логичког, времена које симулација треба да достигне. Након тога кориснички програм контактира централни сервер коме прослеђује, као један посао, тај скуп компоненти, листу веза и тип симулације како би их симулирао. Када централни сервер прими посао и његове параметре, он дели тај посао на већи број мањих послова које ће проследити радним станицама. Сервер на основу расположивог броја радних станица дели примљени посао на известан број мањих послове тако да сваки од њих буде приближно исте сложености. Сложеност посла се процењује на основу броја компоненти, просечног времена потребног да се изврши симулација поједине компоненте, као и броја веза које један посао има са другим пословима.

Када радна станица прими посао, креира нит намењену извршавању тог посла, и у тој нити започиње са његовим извршавањем на основу примљених параметара. Посао се извршава на радној станици тако што покренута нит радне станице прво креира одговарајући тип симулатора, а онда позива методу **`simulate()`** инстанце класе за симулацију дискретних догађаја (инстанце класе изведене из апстрактне класе **`Simulator`**). Да би се обављало паралелно процесирање потребно је прво иницијализовати ову инстанцу одговарајућим бафером (потребно је имплементирати интерфејс **`SimBuffer`**) преко кога се обавља комуникација са осталим рачунарима који обављају симулацију. Поред тога, потребно је иницијализовати и симулатор компонентама и визама које ће обављати симулацију позивањем методе, метода **`setNetlist(Netlist<V> netlist)`**. Компоненте и везе се иницијализују користећи посао који је сервер креирао на основу аргумента које је корисник задао.

Симулатор обавља обраду све док логичко време симулације не достигне задату вредност **на свим** симулаторима који обављају паралелно израчунавање. Компоненте креирају листу порука (догађаја), за себе или за друге симулаторе (рачунаре), позивајући методу **`List<Event<V>> execute(Event<V> msg)`**. Поруке се транспортују до симулатора користећи класу која имплементира интерфејс **`SimBuffer`**. Радна станица одлучује коме да проследи поруку на основу листе доступних радих станица и листе које каже који порт које логичке компоненте је повезан са којим портом које друге логичке

компоненте коју је задао корисник. Када се заврши симулацију радна станица прикупља стања свих логичких компонената позивом методе **`getState()`** класе **`Netlist`** и прослеђује их централном серверу који та стања касније треба да обједини и да их проследи клијенту.

Да би се обезбедило прикупљање информација о активним радним станицама, централни сервер на сваких **`x`** секунди проверава да ли је нека радна станица исправна. Уколико сервер утврди да нека радна станица која је до тог тренутка обављала симулацију није више исправна, прекида целокупно извршавање дате симулације и покреће је поново на преосталим радним станицама. Након слања захтева за обраду кориснички програм може да раскине везу са централним сервером. Веза може бити раскинута гашењем програма или затварањем комуникационог канала. Када се следећи пут повеже, кориснички програм може да тражи резултате претходно задате обраде. Треба обезбедити да централни сервер може да у паралели да прима већи број послова које је потребно обрадити. Кориснички програм може од централног сервера да тражи информације о статусу посла, а може да тражи и резултате. Одмах по стартовању радне станице шаљу централном серверу информацију о томе да су стартоване и број послова које могу у паралели да обрађују. Број послова које радне станице могу да приме је аргумент који им се поставља приликом покретања. Треба обезбедити да симулација може да ради чак и ако постоји само једна радна станица која има на располагању само једну активну нит.

Параметри посла које клијент задаје су: листа логичких компонената које је потребно симулирати, листу веза која каже који излазни порт које компоненте је повезан са којим улазним портом које компоненте. Ови параметри се налазе унутар две датотеке чија се имена уносе, као и име датотеке у коју је потребно сместити резултате симулације коју формира сервер на основу резултата свих компонента. На радним станицама је потребно проследити делове ових датотека у зависности од расподеле како компонената по радним станицама. Иницијализација компонената се постиже позивом методе **`public void addComponent(String[] data)`**, док се додавање веза постиже позивом методе **`public void addConnection(String[][] data)`**. Поред овога клијент задаје тип симулације коју жели да покрене, као и крајње логичко време које симулација треба да достигне.

Централни сервер у лог уписује време када је пристигао сваки посао, број под којим је посао сачуван, име рачунара коме је посао прослеђен, време када је посао завршен и његов тренутни статус. Статус посла може да буде: **`Ready`** – приспео на сервер, али није никоме прослеђен, **`Scheduled`** – тренутно се прослеђује радној станици, **`Running`** – извршавање је у току, **`Done`** – посао се успешно извршио, **`Failed`** – посао није могао да се изврши (емитовао је неки изузетак), **`Aborted`** – корисник је одустао од извршавања посла.

Проблем решити користећи мрежну комуникацију у програмском језику Јава. Решење треба да буде независно од посла који се обавља. За сваки од ова три типа рачунара треба да постоји одговарајући графички кориснички интерфејс (GUI). Радна станица треба да има могућност покретања и без корисничког интерфејса.

Поред напоменутог, извештај на својој последњој страни треба да садржи и резултате мерења перформанси. Ради мерења перформанси потребно је извршити подешавање система према следећим карактеристикама:

1) Одабрати посао које је потребно извршавати.
2) Извршити мерења колико је времена потребног за задато израчунавање користећи 1, 2, 4, 5, 9, 12 и 16 рачунара.