ДОМАЋИ ИЗ КОНКУРЕНТНОГ И ДИСТРИБУИРАНОГ ПРОГРАМИРАЊА ЗА ШКОЛСКУ 2016/2017 У ЈУНСКОМ ИСПИТНОМ РОКУ

Домаћи задатак из предмета 13Е113КДП и 13С113КДП се школске 2016/2017 ради самостално.

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

Предуслови за успешну одбрану домаћег су:

1: Уписана одговарајућа година ЕТФ-а одговарајућег смера.

2: Благовремена достава писаних материјала и електронске верзије решења.

3: Благовремено припремљени услови за неометану проверу рада програма у лабораторији катедре за РТИ ЕТФ-а (барем три дана пре одбране домаћег 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 домаћи 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 домаћи задатак.

Израда писаних материјала везаних за домаћи подразумева поштовање одговарајуће форме. Према тој форми сваки домаћи 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 домаћи 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 који 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 потребној 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:

а) коришћење у регуларним ситуацијама од стране особе чији је ниво рачунарског знања минималан, а која нема претходно искуство у раду са сличним пакетима.

б) 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 *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 величине као у тексту (не мање од 10 pt).

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

Домаћи задатак Јун 2017
ЈАВА ЛИНДА - RMI

Пројектовати дистрибуирани рачунарски систем који треба да омогући дистрибуирану обраду и међусобну удаљену синхронизацију временски захтевних послова. Ова синхронизација треба да се обави по угледу на виртуелни заједнички простор торки заједнички за све послове унутар скупа послова који постоји код библиотеке C-Linda.
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 праћење рада извршавања дистрибуиране обраде, чување информација о доступним чворовима у мрежи и могућност поновног стартовања појединих послова.
2. Радна станица која од централног сервера добија послове које треба одрадити.
3. Кориснички програм, који задаје посао који треба урадити и његове параметре.

Процес започиње тако што кориснички програм задаје посао који је потребно обрадити. Овај посао представља програм писан у програмском језику Јава запакован у *jar* архиву. Након тога кориснички програм контактира централни сервер коме прослеђује тај посао и параметре потребне за његову обраду. Када централни сервер прими посао и његове параметре, он га прослеђује једној радној станици, чека резултат обраде и враћа комплетан резултат клијенту. Када радна станица прими посао започиње са његовим извршавањем, на основу примљених параметара. Посао се извршава на радној станици тако што радна станица покреће посао који је дат као *jar* архива којој прослеђује параметре које је клијент задао. Као посебан параметар поставља путању до *jar* архиве у којој се налази библиотека за удаљену синхронизацију. Та библиотека је реализована по угледу на библиотеку *C-Linda* која се користи за синхронизацију. Интерфејс који је потребно за задовољи је дат у наставку овог документа. Пошто се у датом корисничком послу може покренути више послова (метода eval(...)), они треба да буду распоређени на слободне радне станице. Када се заврши извршавање програма, радна станица прикупљене излазне токове података прослеђује серверу, као и резултате извршавања. 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 није исправна, обавештава корисника који је посао престао да се извршава. Након тога корисник треба да одлучи да ли посао прекинути или да се прекинути део проследи некој слободној радној станици. Уколико корисник није доступан прекида се извршавање целог посла. 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.
Параметри посла које клијент задаје су: команда која се задаје Јава виртуелној машини да би покренуо архиву, датотеке које са клијентског рачунара треба пребацити на радну станицу да би команде могле да се изврше (не више од 6), датотеке које са радне станице треба пребацити на клијентски рачунар да које представљају резултате (не више од 6). Ови параметри могу да се задају или путем корисничког интерфејса или путем текстуалне датотеке.

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, *Aborted* – корисник је одустао од извршавања посла.
Проблем речити искључиво користећи **Јава RMI** мрежну комуникацију. 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 треба да буде развијен користећи Java **SWING** компоненте). Радна станица треба да има могућност покретања и без корисничког интерфејса.

Интерфејс за рад у дистрибуираном окружењу:

```
package rs.ac.bg.etf.kdp;

import java.io.*;

public interface Linda extends Serializable {
	/**
	 * Ubacuje torku u prostor torki. Nije dozvoljeno da bilo slati bilo koji
	 * string koji je null
	 */
	public void out(String[] tuple);

	/**
	 * Dohvatanje torke iz prostora torki. Ovo je blokirajuca operacija.
	 * Ukoliko je neko od polja unutar ovog niza postavljeno na vrednost
	 * null onda to polje takodje treba popuniti. Ukoliko ima vise torki
	 * uzima se bilo koja.
	 * Nakon ove operacije torka se vise ne nalazi u prostoru torki.
	 */
	public void in(String[] tuple);

	/**
	 * Dohvatanje torke iz prostora torki. Ovo je neblokirajuca operacija.
	 * Ukoliko je neko od polja unutar ovog niza postavljeno na vrednost null
	 * onda to polje takodje treba popuniti. Ukoliko torka postoji onda se kao
	 * rezultat vraca vrednost true, u suprotnom se vraca vrednost false.
	 * Ukoliko ima vise torki uzima se bilo koja. Nakon ove operacije torka
	 * se vise ne nalazi u prostoru torki.
	 */
	public boolean inp(String[] tuple);

	/**
	 * Cita torku iz prostora torki. Ovo je blokirajuca operacija. Ukoliko je
	 * neko od polja unutar ovog niza postavljeno na vrednost null onda to
	 * polje takodje treba popuniti. Ukoliko ima vise torki uzima se bilo
	 * koja. Nakon ove metode torka se i dalje nalazi u prostoru torki.
	 */
	public void rd(String[] tuple);

	/**
	 * Cita torku iz prostora torki. Ovo je neblokirajuca operacija. Ukoliko
	 * je neko od polja unutar ovog niza postavljeno na vrednost null onda to
	 * polje takodje treba popuniti. Ukoliko torka postoji onda se kao
	 * rezultat vraca vrednost true, u suprotnom se vraca vrednost false.
	 * Ukoliko ima vise torki uzima se bilo koja. Nakon ove operacije torka
	 * se i dalje nalazi u prostoru torki.
	 */
	public boolean rdp(String[] tuple);
```

```
        /** Pokretanje nove niti na datom racunaru. */
        public void eval(String name, Runnable thread);

        /**
         * Pokretanje nove niti na datom racunaru. Pokrece se izvrsavanje metode
         * zadate parametrom methodName koja prima argimente arguments.
         * Izvrsavanje se pokrece na instanci klase zadate nazivom klase className
         * i argumentima konstruktora initargs.
         */
        public void eval(String className, Object[] construct, String methodName,
                    Object[] arguments);
}
```