Ime i prezime:_____

Raspodijeljeni sustavi 4. rok –2015/16.

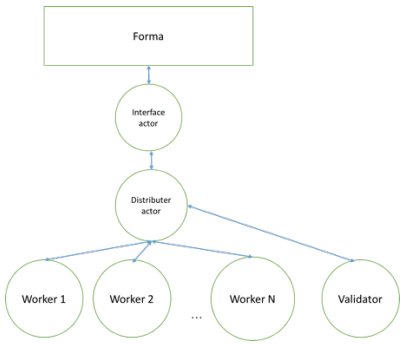
Napomena: U svim zadatcima kada odgovarate sa servera koristite Context.Sender.Tell…

Prvi dio

Zadatak 1 . (2 boda)

Kreirajte Akka.NET aplikaciju koja će učitati novosti iz mape na disku te provjeriti valjanost njihovog kontrolnog koda i ispisati samo one valjane. U mapi na disku može biti proizvoljan broj datoteka, a sama datoteka može sadržavati proizvoljan broj novosti. U sadržaju datoteke, svaka novost je u svome retku te se svaka sastoji od ID-a, naslova, sadržaja i kontrolnog koda. Dijelovi novosti su odvojeni jednim znakom postotka (%). Za kontrolni kod uvijek pretpostavite da će biti string od 16 znakova gdje su svaka 4 odvojena razmakom. Kontrolni kod je validan ako i samo ako je zbroj svake četvorke dijeljiv s brojem 4.

Akka.NET sustav se mora sastojati od 4 vrste actora pri čemu *Interface actor* je taj koji komunicira s formom, *worker actor* je taj koji izvlači novosti iz datoteke, kreira od njih objekte i šalje distributoru, *distrubter actor* kreira **potreban broj** *worker-a*, prikuplja rezultate, od kojih sve rezultate šalje *validtor actoru* koji validira kontrolne kodove svake dobivene novosti te distributeru vraća samo one koje su validne. Nakon primitka samo validnih novosti, distributer šalje interface aktoru novosti za ispis. Na slici 1 je vizualno prikazano kako su *actor*-i posloženi.



Slika 1 Actori u aplikaciji

Kada pritisnete botun za pokretanje:

1. Interface actoru se šalje poruka Start
2. nakon čega on prosljeđuje poruku Distributer actoru.
3. Distributer actor šalje poruku ReadFile svim svojim Workerima, a njih će biti ovisno o broju datoteka u direktorijua. ReadFile u sebi sadrži putanju do datoteke nad kojom će worker odraditi svoje operacije.
4. Workeri kada završe, odgovaraju porukom FileData distributoru,
5. koji nakon što primi odgovor od svih Workera sve primljene novosti šalje validator actoru u poruci FileData.
6. Distributer skuplja odgovore svih workera u varijablu storage koja je tipa FileData.
7. Po primitku svih artikala od workera, šalje svoj storage validator actoru. Validator actor potom provjerava kontrolne kodove
8. i u poruci ***ValidData*** šalje samo novosti koje su validne distributer actoru.
9. Distributer actor primitkom ValidData poruke njen sadržaj prosljeđuje interface actoru (nije važno je li u FileData ili ValidData poruci) i postavlja svoj storage na null.
10. Interface actor ispisuje poruku.

Kako bi vam se pomoglo, pripremljen je predložak za zadatak u kojem imate napravljene sve potrebne klase za *actor-e,* klasu za novosti te je napravljen direktorij s datotekama na kojem ćete raditi. Ujedno su vam dane implementirane metode za čitanje svih imena iz direktorija te metoda za čitanje sadržaja jedne datoteke, metoda za spajanje primljenih novosti u postojeći storage.

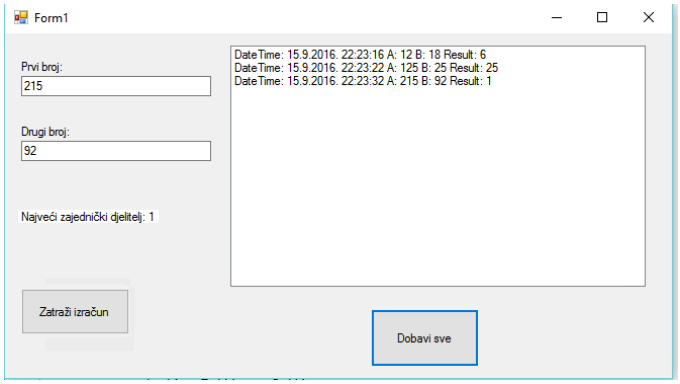
Zadatak 2 (3 boda)

Napravite klijent – server Akka.NET (koristeći Akka.Remote) aplikaciju u kojoj server podržava dvije različite poruke: CalculateGcd i GetHistory.

Forma klijenta izgleda kao na slici 2. Aplikacija koju radite jest računanje najveće zajedničke vrijednosti na serveru i spremanje povijesti izračuna. Na klijentu se unose 2 velika broja (tipa long) koja se pritiskom na tipku "Zatraži izračun" šalju serveru porukom CalculateGcd. Server za dobivena 2 broja računa najvećeg zajedničkog djelitelja vrijednosti A i B te sprema u datoteku trenutni datum i vrijeme, vrijednosti A i B te rezultat, od čega su svi ovi podatci odvojeni znakom "$".

Pritiskom na dugme "Dobavi sve", klijent šalje poruku serveru GetHistory. Primitkom te poruke, server iz datoteke pročita sve prijašnje izračune koji su spremljeni te ih porukom History pošalje klijentu. Klijent ispiše dobivenu povijest u svome listboxu.

Tipka "Zatraži izračun" treba biti isključena u periodu od kad je tipka pritisnuta pa sve dok klijent ne dobije rezultat.



Slika 2 Izgled klijenta

Interface actor je taj koji ima pristup formi, a *communication actor* šalje i prima poruke od servera. Vodite računa o tome da se tipka za izračun ne može pritisnuti dok oba broja nisu unesena. Vodite računa o tome da oba samo jednom kreirate.

Interface i communication actor mogu primiti sljedeće poruke:

- CalculateGcd
- GetHistory
- History
- GcdResult

Provider actor na serveru može primiti CalculateGcd i GetHistory.

Kako bi vam se pomoglo pripremljeni su predlošci za klijent i server. Predlošci uključuju:

- Stvaranje actor sustava
- Konfiguracije za remote komunikaciju
- Čitanje i pisanje iz datoteke na strani servera (bez razdvajanja po posebnom znaku #)
- Poseban projekt s porukama za komunikaciju servera i klijenta (već uključen u solution)
- Prazne klase za potrebne actore

Ime i prezime:_____

Drugi dio

Zadatak 3 (2 boda)

Napravite klijent – server Akka.NET aplikaciju u kojoj će server u datoteci spremati stanje bankovnog računa. Klijent šalje zahtjev za povlačenjem novca, a server sprema novo stanje računa u datoteku. Klijent u TextBox unosi iznos koji želi podignuti.

Klijent šalje zahtjev serveru i očekuje odgovor o uspješnosti. Zahtjev se šalje porukom Withdraw i to je jedna od dvije poruke koje server prepoznaje. Klijent prepoznaje poruke WithdrawAck, Start i Retry. Start – pokreće sustav i prenosi unesenu vrijednost iz textboxa, Retry – posebna poruka za ponovno slanje, a WithdrawAck – potvrda da je odrađen posao na serveru skupa s rezultatom. Na lblPoruka ispišite je li uspješno podignut novac

Server sam sebi šalje poruku Deposit koja sadrži slučajan broj između 100 i 500. Depositom se povećava stanje bankovnog računa.

Svaki put kada se stanje računa uveća ili umanji, potrebno je novo stanje spremiti u datoteku.

Pouzdanost prijenosa poruka osigurajte *exactly-once-processing* semantikom. Napravite da server u 50% slučajeva "zaboravi" poslati potvrdu o izvršenoj operaciji klijentu. Klijent ima 1 ICancelable. Dakle, klijent svako 2 sekunde pokušava poslati ponovno **sve** poruke odjednom za koje nije dobio odgovor od servera.

Informacije o ponovnim slanjima – id i amount se ispisuju u RichTextBox.

Pripazite da se u TextBox na klijentu mora unijeti pozitivan broj.

Kako bi vam se pomoglo pripremljeni su predlošci za klijent i server. Predlošci uključuju:

- Stvaranje actor sustava
- Konfiguracije za remote komunikaciju
- Poseban projekt s porukama za komunikaciju servera i klijenta (već uključen u solution)

Zadatak 4. (3 boda)

Za ovaj zadatak ste dobili gotov server koji prima poruku za spremanje sadržaja u datoteku i slanje odgovora klijentu. Na serveru ne morate ništa mijenjati.

Prilagodite implementaciju Maekawinog algoritma za međusobno isključivanje na način da kritični odsječak predstavljaju konzola za unos i dobiveni server. Kada dobije pristup kritičnom odsječku, klijent čeka unos s konzole nakon čega šalje poruku serveru. Klijent oslobađa kritični odjeljak tek kada primi odgovor od servera da je posao obavljen.

Poruka koja se šalje je Save i jednostavno prenosi uneseni string.

Klijenata može biti proizvoljan broj.

Modificirajte program i tako da svaki klijent čeka X milisekundi prije nego pošalje zahtjev za kritičnom sekcijom.

Objasnite razliku između Lamportovog algoritma za međusobno isključivanje i Maekawas algoritma za međusobno isključivanje. Kako funkcionira drugi?

NAPOMENA: Programi koji u bilo kojem trenutku puknu ili se ne budu mogli uopće pokrenuti će biti ocijenjeni s 0 bodova.