Praca domowa 08 – control

Termin zwrotu : 16 maja godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Należy zaprojektować i wykonać komponent EJB o nazwie *ControlImpl*, którego interfejs udostępniać winien następujące metody :

```
@Remote
public interface IControlRemote {
	public void start();
	public void stop();
	public void increment(int i);
	public int  counter();
	public int  errors();
}
```

Komponent może znajdować się w jednym z dwóch stanów : *zliczania* lub *wstrzymania*. Początkowo komponent znajduje się w stanie wstrzymania a licznik komponentu jest wyzerowany. Przejście ze stanu wstrzymania do stanu zliczania następuje w efekcie wywołania metody *start()*. Przejście ze stanu zliczania do stanu wstrzymania następuje w efekcie wywołania metody *stop()*. Wykonanie metody *increment(n)* w stanie zliczania powoduje zmianę wartości licznika o wartość *n*. Wykonanie metody *increment(n)* w stanie wstrzymania nie zmienia wartości licznika, powoduje natomiast zarejestrowanie faktu niepoprawnego żądania poprzez zwiększenie licznika błędów (error). Podobnie wywołanie metody *start()* w stanie zliczania lub metody *stop()* w stanie wstrzymania jest operacją niepoprawną, skutkującą wyłącznie zwiększeniem licznika błędów (error). Metoda *counter()* zwraca aktualny stan licznika, metoda *errors()* zwraca informację o łącznej ilości nieprawidłowych wywołań (stan licznika błędów error).

Sterowanie procesu obliczeń winno być zaimplementowane w postaci servletu o nazwie *Control,* który zobowiązany jest do zarejestrowania zadania (metoda *register()*). Servlet otrzymuje przekazywaną w żądaniu (url) jako parametr informację o zleconej do wykonania przez komponent operacji. W efekcie zdekodowania parametru winien wywołać odpowiednią dla żądania metodę komponentu *ControlImpl*. Servlet otrzymać może w żądaniu parametry *login* lub *logout* które skutkować winny rozpoczęciem oraz zakończeniem procesu zliczania. Parametr żądania *state* przyjmujący wartość całkowitą nakazuje zmniejszyć bądź zwiększyć wartość licznika. Jeżeli dla parametru *state* nie podano w żądaniu url wartości należy przyjąć, że licznik zwiększany jest o wartość 1. Parametr żądania *result* winien skutkować odesłaniem przez servlet liczby całkowitej będącej różnicą aktualnego stanu oraz ilości błędnych żądań zarejestrowanych przez komponent.

A więc w przykładowym ciągu żądań :	Odpowiedź servletu:	Uwagi :
`http://...../Control/?state=2`	`pusta strona`	`żądanie błędne`
`http://...../Control/?login`	`pusta strona`	
`http://...../Control/?state=2`	`pusta strona`	`licznik += 2`

```
http://......./Control/?state                pusta strona              licznik += 1
http://......./Control/?result               2
http://......./Control/?login                pusta strona              żądanie błędne
http://......./Control/?state=-1             pusta strona              licznik -= 1
http://......./Control/?logout               pusta strona
http://......./Control/?state                pusta strona              żądanie błędne
http://......./Control/?logout               pusta strona              żądanie błędne
http://......./Control/?result               -2
```

Program ma być zapisany w trzech plikach : `IControlRemote.java` zawierającym definicję interfejsu komponentu zdefiniowanego w pliku `ControlImpl.java` , oraz kod servletu `Control.java`. Poszczególne elementy rozwiązania nie mogą korzystać z bibliotek zewnętrznych innych niż niezbędne moduły serwera (jak np. gf-client.jar, javaee.jar itp.).

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –cp <app-server-modules> –Xlint Control.java IControlRemote.java ControlImpl.java
```

Wymagania :

- Klasa implementująca komponent winna zostać zdefiniowane w pliku `ControlImpl.java`.
- Interfejs udostępniający wymagane metody oraz zwracające wartości liczników obsługiwanych przez komponent `ControlImpl.java` winien zostać zdefiniowany w pliku `IControlRemote.java`.
- Servlet nadzorujący proces obliczeń zapisać należy w pliku `Control.java`
- W pliku README.pdf winien być zawarty opis architektury proponowanego rozwiązania.

Sposób oceny **:**

- 1 pkt – **Weryfikacja** : czy program jest skompletowany i spakowany zgodnie z ogólnymi zasadami przesyłania zadań.
- 1 pkt – **Kompilacja** : każdy z plików winien być kompilowany bez jakichkolwiek błędów lub ostrzeżeń (w sposób omówiony wyżej)
- 1 pkt – **Wykonanie** : program powinien wykonywać się bez jakichkolwiek błędów i ostrzeżeń (dla pliku danych wejściowych zgodnych z wyżej zamieszczoną specyfikacją) z wykorzystaniem omówionych wyżej parametrów linii komend
- 2 pkt – **README** : plik README.pdf dokumentuje w sposób kompletny i właściwy sposób zestawiania połaczenia
- 1 pkt – **Styl kodowania** : czy funkcji i zmienne posiadają samo-wyjaśniające nazwy ? Czy podział na funkcje ułatwia czytelność i zrozumiałość kodu ? Czy funkcje eliminują (redukują) powtarzające się bloki kodu ? Czy wcięcia, odstępy, wykorzystanie nawiasów itp. (formatowanie kodu) są spójne i sensowne ?
- 4 pkt – **Poprawność algorytmu** : czy algorytm został zaimplementowany poprawnie a wynik odpowiada prawidłowej (określonej zbiorem danych testowej) wartości.