Zadanie

PASSTIME_SERVER

Serwer (klasa Server) podaje informacje o mijającym czasie. Klienci (klasa Client) :
a) łączą się z serwerem
b) wysyłają żądania

Dla każdego klienta serwer prowadzi log jego zapytań i ich wyników oraz ogólny log wszystkich żądań wszystkich klientów. Logi są realizowane w pamięci wewnętrznej serwera (poza systemem plikowym).

Protokół

Żądanie	Odpowiedź	Przykład
login id	logged in	login Adam
dataOd dataDo	opis upływu czasu wg secyfikacji z S_PASSTIME	2019-01-20 2020-04-01
bye	logged out	
bye and log transfer	zawartośc logu klienta	w przykładowym wydruku z działania klasy Main

Budowa klasy Server

konstrukto: :
public Server(String host, int port)

Wymagane metody:

- metoda: **public void startServer()** - uruchamia server w odrębnym wątku,
- metoda: **public void stopServer()** - zatrzymuje działanie serwera i wątku w którym działa
- metoda: **String getServerLog()** - zwraca ogólny log serwera

Wymagania konstrukcyjne dla klasy Server

- multipleksowania kanałów gniazd (użycie selektora).
- serwer może obsługiwać równolegle wielu klientów, ale obsługa żądań klientów odbywa się w jednym wątku

Budowa klasy Client

konstruktor:
public Client(String host, int port, String id), gdzie id - identyfikator klienta

Wymagane metody:

- metoda: **public void connect()** - łączy z serwerem
- metoda: **public String send(String req)** - wysyła żądanie req i zwraca odpowiedź serwera

Wymagania konstrukcyjne dla klasy Client

- nieblokujące wejście - wyjście

Dodatkowo stworzyć klasę **ClientTask**, umożliwiającą uruchamianie klientów w odrębnych wątkach poprzez ExecutorService.
Obiekty tej klasy tworzy statyczna metoda:

public static ClientTask create(Client c, List<String> reqs, boolean showSendRes)

gdzie:
c - klient (obiekt klasy Client)
reqs - lista zapytań o uplyw czasu

Kod działający w wątku ma wykonywać następując działania:

- łączy się z serwerem,
- wysyła żądanie "login" z identyfikatorem klienta
- wysyła kolejne żądania z listy reqs
- wysyła "bye and log transfer" i odbiera od serwera log zapytań i ich wyników dla danego klienta

Jeżeki parametr ***showSendRes*** jest true to po każdym send odpowiedź serwera jest wypisywana na konsoli. Niezależnie od wartości parametru należy zapewnić, by log klienta był dostępny jak tylko klient zakończy działanie.

Dodatkowo dostarczyć klasy **Time** (logika obliczania czasu) oraz **Tools** (wczytywanie opcji i żadań klientów, potrzebnych do dzialania klasy Main).

Przygotowana klasa Main ilustruje przypadki interakcji klient-serwer:

```
package zad1;

import java.util.*;
import java.util.concurrent.*;

public class Main {

  public static void main(String[] args) throws Exception {
    String fileName = System.getProperty("user.home") + "/PassTimeServerOptions.yaml";
    Options opts = Tools.createOptionsFromYaml(fileName);
    String host = opts.getHost();
    int port = opts.getPort();
    boolean concur =  opts.isConcurMode();
    boolean showRes = opts.isShowSendRes();
    Map<String, List<String>> clRequests = opts.getClientsMap();
    ExecutorService es = Executors.newCachedThreadPool();
    List<ClientTask> ctasks = new ArrayList<>();
    List<String> clogs = new ArrayList<>();

    Server s = new Server(host, port);
    s.startServer();

    // start clients
    clRequests.forEach( (id, reqList) -> {
      Client c = new Client(host, port, id);
      if (concur) {
        ClientTask ctask = ClientTask.create(c, reqList, showRes);
        ctasks.add(ctask);
        es.execute(ctask);
      } else {
        c.connect();
        c.send("login " + id);
        for(String req : reqList) {
          String res = c.send(req);
          if (showRes) System.out.println(res);
        }
        String clog = c.send("bye and log transfer");
```

```
        System.out.println(clog);
      }
    });

    if (concur) {
      ctasks.forEach( task -> {
        try {
          String log = task.get();
          clogs.add(log);
        } catch (InterruptedException | ExecutionException exc) {
          System.out.println(exc);
        }
      });
      clogs.forEach( System.out::println);
      es.shutdown();
    }
    s.stopServer();
    System.out.println("\n=== Server log ===");
    System.out.println(s.getServerLog());
  }

}
```

Wyniki na konsoli

A. Zawartośc pliku PassTimeServerOptions.yaml

```
host: localhost
port: 7777
concurMode: false   # czy klienci działają równolegle?
showSendRes: false  # czy pokazywać zwrócone przez serwer wyniki metody send(...)
clientsMap: # id_klienta -> lista żądań
  Asia:
    - 2019-01-10 2020-03-01
    - 2020-03-27T10:00 2020-03-28T10:00
  Adam:
    - 2018-01-01 2020-03-27
    - 2020-03-28T10:00 2020-03-29T10:00
```

A. Wynik:

```
=== Asia log start ===
logged in
Request: 2019-01-10 2020-03-01
Result:
Od 10 stycznia 2019 (czwartek) do 1 marca 2020 (niedziela)
 - mija: 416 dni, tygodni 59.43
 - kalendarzowo: 1 rok, 1 miesiąc, 20 dni
Request: 2020-03-27T10:00 2020-03-28T10:00
Result:
Od 27 marca 2020 (piątek) godz. 10:00 do 28 marca 2020 (sobota) godz. 10:00
 - mija: 1 dzień, tygodni 0.14
 - godzin: 24, minut: 1440
 - kalendarzowo: 1 dzień
logged out
=== Asia log end ===

=== Adam log start ===
logged in
Request: 2018-01-01 2020-03-27
Result:
Od 1 stycznia 2018 (poniedziałek) do 27 marca 2020 (piątek)
 - mija: 816 dni, tygodni 116.57
 - kalendarzowo: 2 lata, 2 miesiące, 26 dni
Request: 2020-03-28T10:00 2020-03-29T10:00
Result:
Od 28 marca 2020 (sobota) godz. 10:00 do 29 marca 2020 (niedziela) godz. 10:00
 - mija: 1 dzień, tygodni 0.14
 - godzin: 23, minut: 1380
 - kalendarzowo: 1 dzień
```

```
logged out
=== Adam log end ===


=== Server log ===
Asia logged in at 16:54:09.507
Asia request at 16:54:09.554: "2019-01-10 2020-03-01"
Asia request at 16:54:10.193: "2020-03-27T10:00 2020-03-28T10:00"
Asia logged out at 16:54:10.256
Adam logged in at 16:54:10.318
Adam request at 16:54:10.382: "2018-01-01 2020-03-27"
Adam request at 16:54:10.444: "2020-03-28T10:00 2020-03-29T10:00"
Adam logged out at 16:54:10.506
```

B, W pliku PassTimeServerOptions.yaml ustawiono:

```
concurMode: true   # czy klienci działają równolegle?
showSendRes: false  # czy pokazywać zwrócone przez serwer wyniki metody send(...)
```

B. Wynik - zmienia się log serwera:

```
=== Server log ===
Asia logged in at 16:59:23.494
Adam logged in at 16:59:23.494
Asia request at 16:59:23.541: "2019-01-10 2020-03-01"
Adam request at 16:59:24.071: "2018-01-01 2020-03-27"
Asia request at 16:59:24.102: "2020-03-27T10:00 2020-03-28T10:00"
Adam request at 16:59:24.118: "2020-03-28T10:00 2020-03-29T10:00"
Asia logged out at 16:59:24.165
Adam logged out at 16:59:24.165
```

C, W pliku PassTimeServerOptions.yaml ustawiono:

```
concurMode: false  # czy klienci działają równolegle?
showSendRes: true  # czy pokazywać zwrócone przez serwer wyniki metody send(...)
```

C. Wyniki:

```
Od 10 stycznia 2019 (czwartek) do 1 marca 2020 (niedziela)
 - mija: 416 dni, tygodni 59.43
 - kalendarzowo: 1 rok, 1 miesiąc, 20 dni
Od 27 marca 2020 (piątek) godz. 10:00 do 28 marca 2020 (sobota) godz. 10:00
 - mija: 1 dzień, tygodni 0.14
 - godzin: 24, minut: 1440
 - kalendarzowo: 1 dzień
=== Asia log start ===
logged in
Request: 2019-01-10 2020-03-01
Result:
Od 10 stycznia 2019 (czwartek) do 1 marca 2020 (niedziela)
 - mija: 416 dni, tygodni 59.43
 - kalendarzowo: 1 rok, 1 miesiąc, 20 dni
Request: 2020-03-27T10:00 2020-03-28T10:00
Result:
Od 27 marca 2020 (piątek) godz. 10:00 do 28 marca 2020 (sobota) godz. 10:00
 - mija: 1 dzień, tygodni 0.14
 - godzin: 24, minut: 1440
 - kalendarzowo: 1 dzień
logged out
=== Asia log end ===

Od 1 stycznia 2018 (poniedziałek) do 27 marca 2020 (piątek)
 - mija: 816 dni, tygodni 116.57
 - kalendarzowo: 2 lata, 2 miesiące, 26 dni
Od 28 marca 2020 (sobota) godz. 10:00 do 29 marca 2020 (niedziela) godz. 10:00
 - mija: 1 dzień, tygodni 0.14
 - godzin: 23, minut: 1380
 - kalendarzowo: 1 dzień
=== Adam log start ===
```

```
logged in
Request: 2018-01-01 2020-03-27
Result:
Od 1 stycznia 2018 (poniedziałek) do 27 marca 2020 (piątek)
 - mija: 816 dni, tygodni 116.57
 - kalendarzowo: 2 lata, 2 miesiące, 26 dni
Request: 2020-03-28T10:00 2020-03-29T10:00
Result:
Od 28 marca 2020 (sobota) godz. 10:00 do 29 marca 2020 (niedziela) godz. 10:00
 - mija: 1 dzień, tygodni 0.14
 - godzin: 23, minut: 1380
 - kalendarzowo: 1 dzień
logged out
=== Adam log end ===


=== Server log ===
Asia logged in at 17:02:34.537
Asia request at 17:02:34.583: "2019-01-10 2020-03-01"
Asia request at 17:02:35.145: "2020-03-27T10:00 2020-03-28T10:00"
Asia logged out at 17:02:35.207
Adam logged in at 17:02:35.207
Adam request at 17:02:35.207: "2018-01-01 2020-03-27"
Adam request at 17:02:35.270: "2020-03-28T10:00 2020-03-29T10:00"
Adam logged out at 17:02:35.335
```

Podsumowanie - klasy w projekcie i co należy zrobić.

Klasa	Uwagi
Main	jest w projekcie - niemodyfikowalna
Options	jest w projekcie - niemodyfikowalna
Time	do zrobienia wg specyfikacji S_PASSTIME ew. już gotowa z zad. 3
Tools	do zrobienia wg specyfikacji S_PASSTIME ew. już gotowa z zad. 3
Server	do zrobienia
Client	do zrobienia
ClientTask	do zrobienia

Uwaga: dobrą praktyką byłoby rozdzielenie klas na różne pakiety (server, client, tools itp.), ale NIE ROBIMY TEGO, by nie zwiększac już i tak trochę skomplikowanej struktury zadania. Wszystkie klasy są więc w pakiecie zad1.