Programowanie zaawansowane, lab. 11 i 12

Należysz do zespołu programistycznego pracującego nad rozbudowanym projektem. Na serwerze działają usługi generujące cały czas pliki w odpowiednich miejscach na dysku. Jest tam też usługa webowa, która pozwala osobie z zewnątrz pobrać pliki z wybranego przedziału czasowego i odczytać je zdalnie przy użyciu specjalnej aplikacji. Ponieważ plików tych może być dużo, są one stosunkowo małe, a operacja odczytu danych z dysku jest kosztowna, masz za zadanie stworzenie algorytmu, który pozwoli działanie tego serwisu webowego zoptymalizować. Idea jest taka, aby nie czytać wielokrotnie tych samych plików, tylko trzymać je w pamięci RAM i ewentualnie uzupełniać o pliki z innych przedziałów czasowych.

Pomińmy dziś kwestię samych plików i skupmy się na algorytmie przeliczania przedziałów czasowych. Szef dał Ci wolną rękę, jeśli chodzi o sposób realizacji zadania, natomiast oczekuje on, że:

1. Będzie to aplikacja konsolowa.
2. Aplikacja będzie czekała na żądanie od klienta, gdzie żądanie te będzie przedziałem czasowym, w obrębie którego wygenerowane pliki klient chce odczytać. Format wpisywanego przedziału może być dowolny, zaproponowany przez Ciebie. Ma być jednak data od (rok, miesiąc, dzień, godzina, minuta, sekunda) i data do.
3. Aplikacja po otrzymaniu żądania powinna wypisać w konsoli oczekiwany przez klienta przedział czasowy, a następnie przeanalizować, czy i z jakich przedziałów pliki muszą być odczytane (być może wystarczą te, które już są w pamięci RAM) i wypisać te przedziały w konsoli. Kolejny krok to wypisanie przedziałów, które były już wczytane do pamięci wcześniej i które zostaną teraz wykorzystane. Na koniec powinna uzupełnić przetrzymywaną lokalnie informację o odczytanych już przedziałach o nowy zakres i wypisać wszystkie te przedziały. Przykładowo:

```
Otrzymano zadanie odczytu z przedzialu: 2019.04.26 06:30:00 – 2019.04.26 10:00:00
Czytanie z dysku plikow z przedzialu: 2019.04.26 07:05:31 – 2019.04.26 08:00:00
Czytanie z dysku plikow z przedzialu: 2019.04.26 08:20:10 – 2019.04.26 08:20:30
Czytanie z dysku plikow z przedzialu: 2019.04.26 09:50:00 – 2019.04.26 10:00:00
Wykorzystanie wczytanego przedzialu: 2019.04.26 06:30:00 – 2019.04.26 07:05:30
Wykorzystanie wczytanego przedzialu: 2019.04.26 08:00:01 – 2019.04.26 08:20:09
Wykorzystanie wczytanego przedzialu: 2019.04.26 08:20:31 – 2019.04.26 09:49:59
W pamieci trzymane sa teraz przedzialy:
2019.04.25 02:00:00 – 2019.04.25 04:00:00
2019.04.25 12:50:33 – 2019.04.25 14:13:59
2019.04.26 05:20:00 – 2019.04.26 10:00:00
```

4. Aplikacja będzie posiadała mechanizm oczyszczania pamięci. Np. odczytane przedziały mogłyby się usuwać raz na 15 żądań (powinno to być zależne od czasu, ale na potrzeby tego zadania możemy uprościć problem).
5. Wszystko powinno działać w nieskończonej pętli. Po przetworzeniu jednego żądania aplikacja ma czekać na kolejne.
6. Aplikacja powinna być zaprojektowana zgodnie z paradygmatem programowania obiektowego. Kod nie może wykorzystywać fragmentów kodu konkurencyjnych firm, ponieważ szef nie ma zamiaru kupować dodatkowych licencji. Stać go tylko na utrzymanie jednego programisty na 2. roku studiów.

Wskazówka: Pomocne mogą okazać się klasy **java.time.Instant** oraz **java.text.SimpleDateFormat**. W konstruktorze **SimpleDateFormat** mamy możliwość określenia formatu daty wczytywanej

z konsoli. Następnie poprzez metodę **parse()** możemy otrzymać obiekt klasy **Date** i przekonwertować go do obiektu klasy **Instant** poprzez metodę **toInstant()**. Klasa **Instant** natomiast może ułatwić porównywanie przedziałów czasowych przy użyciu metod **isAfter()**, **isBefore()** oraz **equals()**.

Przykład:

```
TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
DateFormat     dateFormat = new SimpleDateFormat("HH:mm:ss");
String         time = "23:15:30";
Instant        instant;

try {
        instant = dateFormat.parse(time).toInstant();
        System.out.println(instant);
        System.out.println(instant.atZone(ZoneOffset.UTC).getHour());
        System.out.println(instant.atZone(ZoneOffset.UTC).getMinute());
        System.out.println(instant.atZone(ZoneOffset.UTC).getSecond());
} catch (ParseException e) {
        System.out.println(e);
}
```