PASSTIME

Ćwiczenie stanowi przygotowanie do budowy serwera raportującego upływ czasu.
Jest to praktycznie logika "biznesowa" - będzie przydatna w zadaniu serwerowym.

Zadania

(1) Parsowanie plików YAML

W klasie Tools dostarczyć metody:

static Options createOptionsFromYaml(String fileName) throws Exception

która wczytuje podany plika YAML i na jego podstawie tworzy obiekt klasy Options. Klasa Options jest w projekcie.

Do parsowania użyć biblioeki SnakeYaml (https://mvnrepository.com/artifact/org.yaml/snakeyaml/1.26).

Przykładowa zawartość pliku YAML o nazwie PassTimeOptions.yaml jest następująca (proszę go umieścic w katalogu "user.home"):

```
host: localhost
port: 7777
concurMode: true   # czy klienci działają równolegle?
showSendRes: true  # czy pokazywać zwrócone przez serwer wyniki metody send(...)
clientsMap: # id_klienta -> lista żądań
  Asia:
    - 2016-03-30T12:00 2020-03-30T:10:15
    - 2019-01-10 2020-03-01
    - 2020-03-27T10:00 2020-03-28T10:00
    - 2016-03-30T12:00 2020-03-30T10:15
  Adam:
    - 2018-01-01 2020-03-27
    - 2019-01-01 2020-02-28
    - 2019-01-01 2019-02-29
    - 2020-03-28T10:00 2020-03-29T10:00
```

(2) Raportowanie upływu czasu.

W klasie Time dostarczyć metody:

public static String passed(String from, String to)

zwracającej tekst opisujący upływ czas od daty from do daty to.

Daty są podawane jako napisy w formacie ISO-8601:
- data bez czasu: YYYY-MM-DD
- data z czasem: YYYY-MM-DD**T**GG:MM

Przykłady dat zawiera przykładowy plik YAML.

Opis upływającego czasu ma formę:

```
Od x nazwa_miesiąca_PL (dzień_tygodnia_PL) do y nazwa_miesiąca_PL (dzień_tygodnia_PL)
- mija: d dni, tygodni t
[- godzin: g, minut: m ]
[- kalendarzowo: [ r (lat|lata|rok}, ] [ m (miesięcy|miesiące|miesiąc ),]  [d  (dzień|dni)]  ]
```

gdzie x, y, d, g, m, r, - liczby całkowite
t - jest liczbą całkowitą, gdy liczba tygodni jest całkowita, a rzeczywistą (dwa miejsca po kropce) w przeciwnym razie

Nawiasy kwadratowe oznaczają opcjonalność:
a) część opisująca upływ godzin i minut pojawia się tylko wtedy, gdy w formacie daty użyto czasu (np. 2020-10-10T10:00)
b) część "kalendarzowo:" pojawia się tylko wtedy, gdy minął co najmniej jeden dzień
c) w części kalendarzowej opis upływu lat pojawia się tylko wtedy, gdy minął co najmniej jeden rok, opis upływu miesięcy pojawia się tylko wtedy, gdy minął co najmniej jeden miesiąc, a opis upływu dni pojawia się tylko wtedy, gdy liczba dni nie mieści się w pełnych miesiącack.

Nawiasy okrągłe oznaczają alternatywę - dobór właściwej odmiany słowa dla danej liczby.
Gdy w datach podane są godziny należy uwzględnić zmianę czasu (DST) dla strefy czasowej "Europe/Warsaw".
Jeśli podano błędną datę lub datę-czas metoda ma zwrócić opis wyjątku poprzedzony trzema gwiazdkami.

Przykłady:

```
    println passed( "2000-01-01", "2020-04-01")
    println passed( "2018-01-01", "2020-02-02")
    println passed( "2019-01-01", "2020-04-03")
    println passed( "2020-04-01T10:00", "2020-04-01T13:00")
    println passed( "2020-03-27T10:00", "2020-03-28T10:00") // przed zmianą czasu
    println passed( "2020-03-28T10:00", "2020-03-29T10:00") // po zmianie czasu
    println passed( "2020-03-28T10", "2020-03-29T10:00")
    println passed( "2019-02-29", "2020-04-03")

Na konsoli:
Od 1 stycznia 2000 (sobota) do 1 kwietnia 2020 (środa)
 - mija: 7396 dni, tygodni 1056.57
 - kalendarzowo: 20 lat, 3 miesiące
Od 1 stycznia 2018 (poniedziałek) do 2 lutego 2020 (niedziela)
 - mija: 762 dni, tygodni 108.86
 - kalendarzowo: 2 lata, 1 miesiąc, 1 dzień
Od 1 stycznia 2019 (wtorek) do 3 kwietnia 2020 (piątek)
 - mija: 458 dni, tygodni 65.43
 - kalendarzowo: 1 rok, 3 miesiące, 2 dni
Od 1 kwietnia 2020 (środa) godz. 10:00 do 1 kwietnia 2020 (środa) godz. 13:00
 - mija: 0 dni, tygodni 0
 - godzin: 3, minut: 180
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
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
*** java.time.format.DateTimeParseException: Text '2020-03-28T10' could not be parsed at index 13
*** java.time.format.DateTimeParseException: Text '2019-02-29' could not be parsed: Invalid date 'February 29' as '2019' is not a leap year
```

Zawarte w projekcie klasy Main i Options są niemodyfikowalne.
Klasa Options:
package zad1;

```
import java.util.*;

public class Options {

  private String host;
  private int port;
  private boolean concurMode;
  prine boolean showSendRes;
  private Map<String, List<String>> clientsMap = new LinkedHashMap<>();

  public Options(String host, int port, boolean concurMode, boolean showSendRes,
                 Map<String, List<String>> clientsMap) {
    this.host = host;
    this.port = port;
    this.concurMode = concurMode;
    this.showSendRes = showSendRes;
    this.clientsMap = clientsMap;
  }

  public String getHost() {
    return host;
  }

  public int getPort() {
    return port;
  }

  public boolean isConcurMode() {
    return concurMode;
  }

  public boolean isShowSendRes() {
    return showSendRes;
  }

  public Map<String, List<String>> getClientsMap() {
    return clientsMap;
  }

  public String toString() {
    String out = "";  // StringBuilder bardziej efektywny, ale za dużo pisania
    out += host + " " + port + " " + concurMode + " " + showSendRes + "\n";
    for (Map.Entry<String, List<String>> e : clientsMap.entrySet()) {
      out += e.getKey() + ": " + e.getValue() + "\n";
    }
    return out;
  }

}
```

Klasa Main:

```
package zad1;

public class Main {

  public static void main(String[] args) throws Exception {
    String fileName = System.getProperty("user.home") + "/PassTimeOptions.yaml";
    Options opts = Tools.createOptionsFromYaml(fileName);
    System.out.println(opts);
    opts.getClientsMap().forEach( (id, dates) -> {
      System.out.println(id);
      dates.forEach( dpair -> {
        String[] d = dpair.split(" +");
        String info = Time.passed(d[0], d[1]);
        System.out.println(info);
      });
    });
  }

}
```

Uruchomienie metody main z klasy Main (przy założeniu, że plik PassTimeOptions.yaml jest taki jak podano wyżej) spowoduje wyprowadzenie na knsolę następujących informacji:

```
localhost 7777 true true
Asia: [2016-03-30T12:00 2020-03-30T:10:15, 2019-01-10 2020-03-01, 2020-03-27T10:00 2020-03-28T10:00, 2016-03-30T12:00 2020-03-30T10:15]
Adam: [2018-01-01 2020-03-27, 2019-01-01 2020-02-28, 2019-01-01 2019-02-29, 2020-03-28T10:00 2020-03-29T10:00]

Asia
*** java.time.format.DateTimeParseException: Text '2020-03-30T:10:15' could not be parsed at index 11
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
Od 30 marca 2016 (środa) godz. 12:00 do 30 marca 2020 (poniedziałek) godz. 10:15
 - mija: 1461 dni, tygodni 208.71
 - godzin: 35062, minut: 2103735
 - kalendarzowo: 4 lata
Adam
Od 1 stycznia 2018 (poniedziałek) do 27 marca 2020 (piątek)
 - mija: 816 dni, tygodni 116.57
 - kalendarzowo: 2 lata, 2 miesiące, 26 dni
Od 1 stycznia 2019 (wtorek) do 28 lutego 2020 (piątek)
 - mija: 423 dni, tygodni 60.43
 - kalendarzowo: 1 rok, 1 miesiąc, 27 dni
*** java.time.format.DateTimeParseException: Text '2019-02-29' could not be parsed: Invalid date 'February 29' as '2019' is not a leap year
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
```

Format wydruku jest obowiązkowy.

Pomoc:
zastosowac klasy Java Time API (pakiet java.time) m.in.
LocalDate, LocalDateTime, ZonedDateTime, ChronoUnit, Period.