# Esper – podstawy – zadanie własne

## ScoreEvents

1. Pobierz wersję początkową naszego projektu

```
git clone https://github.com/BigDataStreamProcessing/esper-ScoreEvents.git
```

2. Przejdź do katalogu projektu, a następnie usuń powiązanie z projektem zewnętrznym

```
cd esper-ScoreEvents
git remote rm origin
```

3. Otwórz katalog `esper-ScoreEvents` w *IntelliJ IDEA*.

4. Zapoznaj się z jego zawartością.

    a. Odnajdź fragment, który zawiera polecenia EPL dostarczane do kompilacji.

```
// Compile the EPL statement
EPCompiler compiler = EPCompilerProvider.getCompiler();
EPCompiled epCompiled;
try {
    epCompiled = compiler.compile( s: """
            @public @buseventtype create json schema ScoreEvent(house string, character string, score int, ets string, its string);
            @name('answer') SELECT house, score, character, ets, its
                  from ScoreEvent#ext_timed(java.sql.Timestamp.valueOf(its).getTime(), 3 sec)""", compilerArgs);
}
catch (EPCompileException ex) {
    // handle exception here
    throw new RuntimeException(ex);
}
return epCompiled;
```

        i. Zapoznaj się ze schematem, który został zarejestrowany.

        ii. Zapoznaj się z poleceniem, którego wyniki będą pobierane jako rezultat przetwarzania

    b. Odnajdź fragment, który dostarcza efekt kompilacji do obiektu `EPRuntime`.

```
EPCompiled epCompiled = getEPCompiled(config);

// Connect to the EPRuntime server and deploy the statement
EPRuntime runtime = EPRuntimeProvider.getRuntime( uri: "http://localhost:port", config);
EPDeployment deployment;
try {
    deployment = runtime.getDeploymentService().deploy(epCompiled);
}
catch (EPDeployException ex) {
    // handle exception here
    throw new RuntimeException(ex);
}
```

    c. W której linii tworzony jest obiekt `EPStatement` i z którym zapytaniem jest powiązany? Na podstawie czego następuje to powiązanie?

d. Odnajdź fragment, który przypisuje obiekt Listenera do zapytania

```
// Add a listener to the statement to handle incoming events
resultStatement.addListener( (newData, oldData, stmt, runTime) -> {
    for (EventBean eventBean : newData) {
        System.out.printf("R: %s%n", eventBean.getUnderlying());
    }
});
```

e. Odnajdź fragment, który wykorzystując bibliotekę *DataFaker* (`https://www.datafaker.net`) zasila nasz silnik przetwarzania strumieni danych

```
Faker faker = new Faker();

long startTime = System.currentTimeMillis();
while (System.currentTimeMillis() < startTime + (1000L * howLongInSec)) {
    waitToEpoch();
    for (int i = 0; i < noOfRecordsPerSec; i++) {
        String house = faker.harryPotter().house();
        Timestamp eTimestamp = faker.date().past( atMost: 30, TimeUnit.SECONDS);
        eTimestamp.setNanos(0);
        Timestamp iTimestamp = Timestamp.valueOf(LocalDateTime.now().withNano( nanoOfSecond: 0));

        Schema<Object, ?> schema = Schema.of(
                field( name: "house", () -> house),
                field( name: "character", () -> faker.harryPotter().character()),
                field( name: "score", () -> String.valueOf(faker.number().randomDigitNotZero())),
                field( name: "ets", eTimestamp::toString),
                field( name: "its", iTimestamp::toString)
        );

        JsonTransformer<Object> transformer = JsonTransformer.builder().build();
        String record = transformer.generate(schema, limit: 1);
        runtime.getEventService().sendEventJson(record, s1: "ScoreEvent");
    }
}
```

f. W jaki sposób wykorzystywane są parametry programu `noOfRecordsPerSec` oraz `howLongInSec`?

g. Do czego służy funkcja `waitToEpoch`?

5. Uruchom program

```
R: {score=3, its=2024-02-21 09:18:07.0, character=Nagini, house=Hufflepuff, ets=2024-02-21 09:17:54.0}
R: {score=1, its=2024-02-21 09:18:07.0, character=Antioch Peverell, house=Gryffindor, ets=2024-02-21 09:18:02.0}
R: {score=9, its=2024-02-21 09:18:08.0, character=Vincent Crabbe, Sr., house=Wampus, ets=2024-02-21 09:17:55.0}
R: {score=9, its=2024-02-21 09:18:08.0, character=Aragog, house=Gryffindor, ets=2024-02-21 09:17:49.0}
R: {score=3, its=2024-02-21 09:18:09.0, character=Goyle Sr., house=Gryffindor, ets=2024-02-21 09:18:07.0}
R: {score=9, its=2024-02-21 09:18:09.0, character=James Potter, house=Pukwudgie, ets=2024-02-21 09:17:49.0}
R: {score=8, its=2024-02-21 09:18:10.0, character=Alecto Carrow, house=Horned Serpent, ets=2024-02-21 09:17:44.0}
```

6. Zmień zapytanie na takie, które będzie wydobywało te wyniki, które przekraczają średni rezultat uzyskany dla danego domu w ciągu ostatnich 10 sekund. Skoryguj parametry tak, aby liczba rekordów generowanych w sekundzie była równa 20, a czas generowania danych był równy 30 sekund.
   Uruchom tak skorygowany program.

```
R: {score=8, character=Ronan, house=Wampus, avgscore=6.666666666666667}
R: {score=6, character=Madam Rosmerta, house=Thunderbird, avgscore=5.333333333333333}
R: {score=9, character=Horace Slughorn, house=Thunderbird, avgscore=6.25}
R: {score=8, character=Seamus Finnigan, house=Ravenclaw, avgscore=6.333333333333333}
```

. . .

```
R: {score=6, character=Mr. Roberts, house=Pukwudgie, avgscore=4.928571428571429}
R: {score=9, character=Luna Lovegood, house=Hufflepuff, avgscore=5.346153846153846}
R: {score=6, character=Horace Slughorn, house=Ravenclaw, avgscore=4.916666666666667}
R: {score=6, character=Poppy Pomfrey, house=Gryffindor, avgscore=3.84}
```

## Twój własny zestaw danych

7. Zapoznaj się z możliwościami biblioteki *DataFaker*
   `https://www.datafaker.net/documentation/getting-started/`
   `https://www.datafaker.net/documentation/providers/`

8. Opracuj swój własny generator strumienia danych oparty na swoim własnym schemacie zdarzeń.
   Kilka elementów jest obowiązkowych:
   a. Znacznik czasowy zdarzeń (inny niż czas systemowy opóźniony w stosunkud o czasu systemowego najwyżej o 40 sekund) – `ets` (*event timestamp*)
   b. Znacznik czasu systemowego – `its` (*internal timestamp*)
   c. Atrybut, lub atrybuty, po których można dane grupować (zawierający/e ograniczoną liczbę możliwych wartości (5-10)
   d. Atrybut lub atrybuty, których wartości można agregować (wartości numeryczne)
   e. Atrybut lub atrybuty dodatkowe, opisujące zdarzenie

   *DataFaker* ma ponad 200 tzw. *providerów* danych. To naprawdę oznacza duży wybór.
   Użyj wyobraźni. Niech to nie będzie „kalka” *ScoreEvents*. Zawsze możesz opracować generator strumienia danych swojej własnej rzeczywistości (wykraczającej poza wbudowanych *providerów*) wykorzystując *providerów* ogólnych (generujących wartości będące np. datami, liczbami, czy wartościami z podanego przez Ciebie zbioru). Niech to nie będzie rzeczywistość „magiczna”. Postaraj się opracować generator danych „z naszego świata”. Przykładowo:

   - oglądnięcia filmów na platformach VOD
   - wykorzystanie współdzielonych pojazdów miejskich
   - podróże taksówkami
   - przeloty samolotów
   - żądania HTTP generowane na portalach przez ich użytkowników
   - itp.

   Jeśli masz jakieś wątpliwości przedyskutuj swój pomysł z prowadzącym.

   Zwróć uwagę, że kurs dotyczy przetwarzania strumieni danych. Twój generator ma być generatorem zdarzeń… nie każda dana jest zdarzeniem. Nie każdy zbiór krotek można traktować jak strumień.