Data Analytics at the Edge

Current Topics in Distributed Systems: Internet of Things and Cloud Computing

Narges Mehran, MSc.

SS21

28.04.2021

How to analyze the data?

- Devices and sensors are everywhere
  - and more are coming online every day

- You need a way to analyze all the data coming from your devices

- But it can be expensive to transmit all the data from a sensor to your central analytics engine

Data / Event / Message Streams

“Conceptually, a stream is a (potentially never-ending) flow of data records, and a transformation is an operation that takes one or more streams as input and produces one or more output streams as a result.”

Apache Flink: Dataflow Programming Model

Apache

- The **Apache** HTTP Server Project, colloquially called Apache, is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows.
  - https://www.apache.org/
- One of the most famous web servers (used for hosting Web sites):
  - Apache HTTP Server from Apache Software Foundation

Apache Edgent

- In 2016, *Quarks* was renamed to *Apache Edgent*.
- *Edgent* helps you to shift *from* sending a continuous flow of trivial data to the server, *to* sending only essential and meaningful data as it occurs.
- Edgent communicates with your back-end systems through the following message hubs:
  - MQTT – The messaging standard for IoT
  - IBM Watson IoT Platform – A cloud-based service that provides a device model on top of MQTT
  - Apache Kafka – An enterprise-level message broker

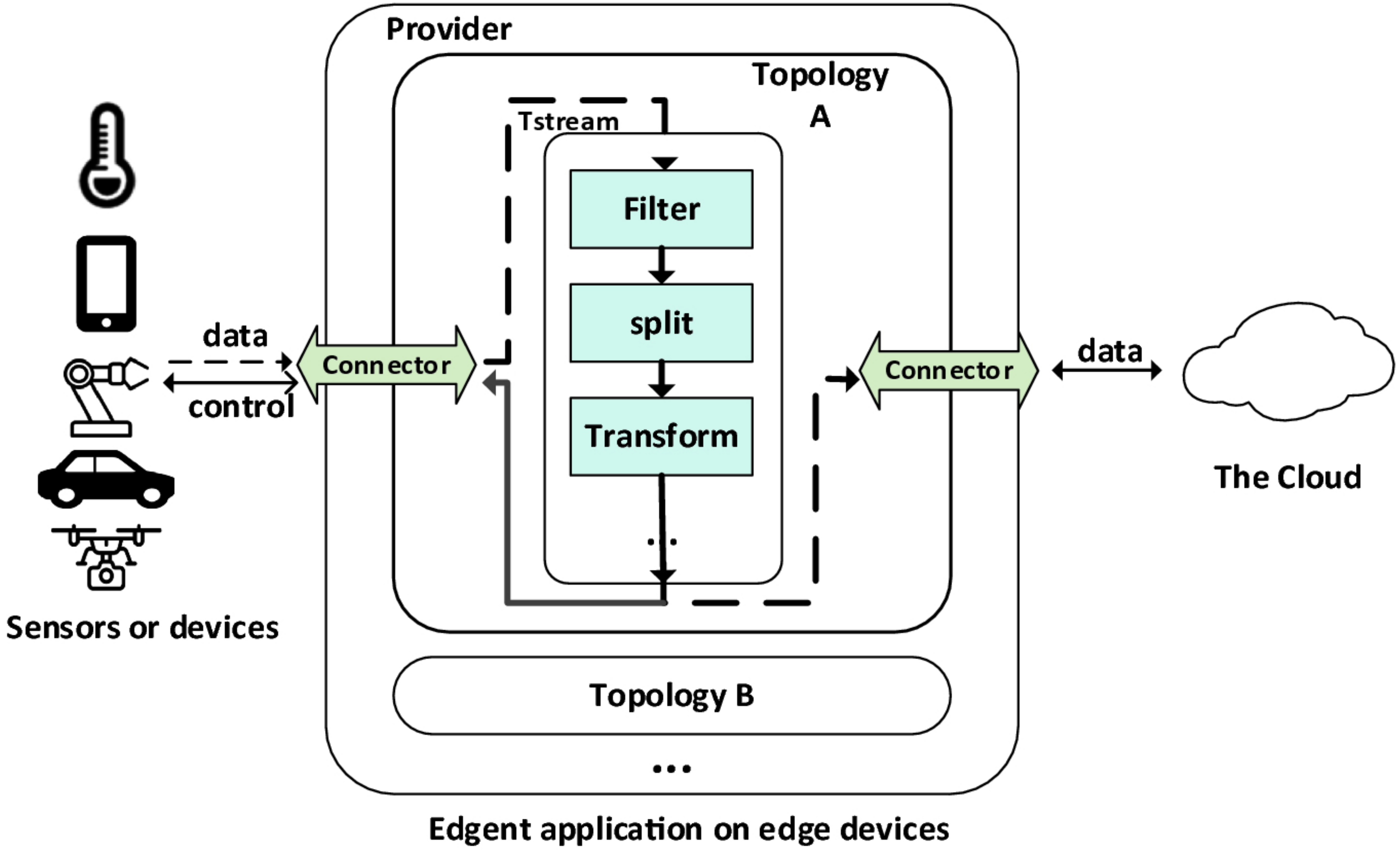
Apache Edgent (cont.)

- Apache Edgent:
  - ✓ an open-source programming model and runtime,
  - ✓ that can be embedded in gateways and edge devices,
  - ✓ to analyze streaming data on your edge devices

- Edgent applications process data locally - such as, in a car, or on a Raspberry Pi
  - ✓ before it sends data over a network

- E.g., if your device takes temperature readings from a sensor, 1,000 times per second,
  - ✓ it is more efficient to process the data locally and send only interesting or unexpected results over the network
  - ✓ If 99% of readings are normal, Edgent detects the 1% anomalies and just sends those as device events for further analysis.

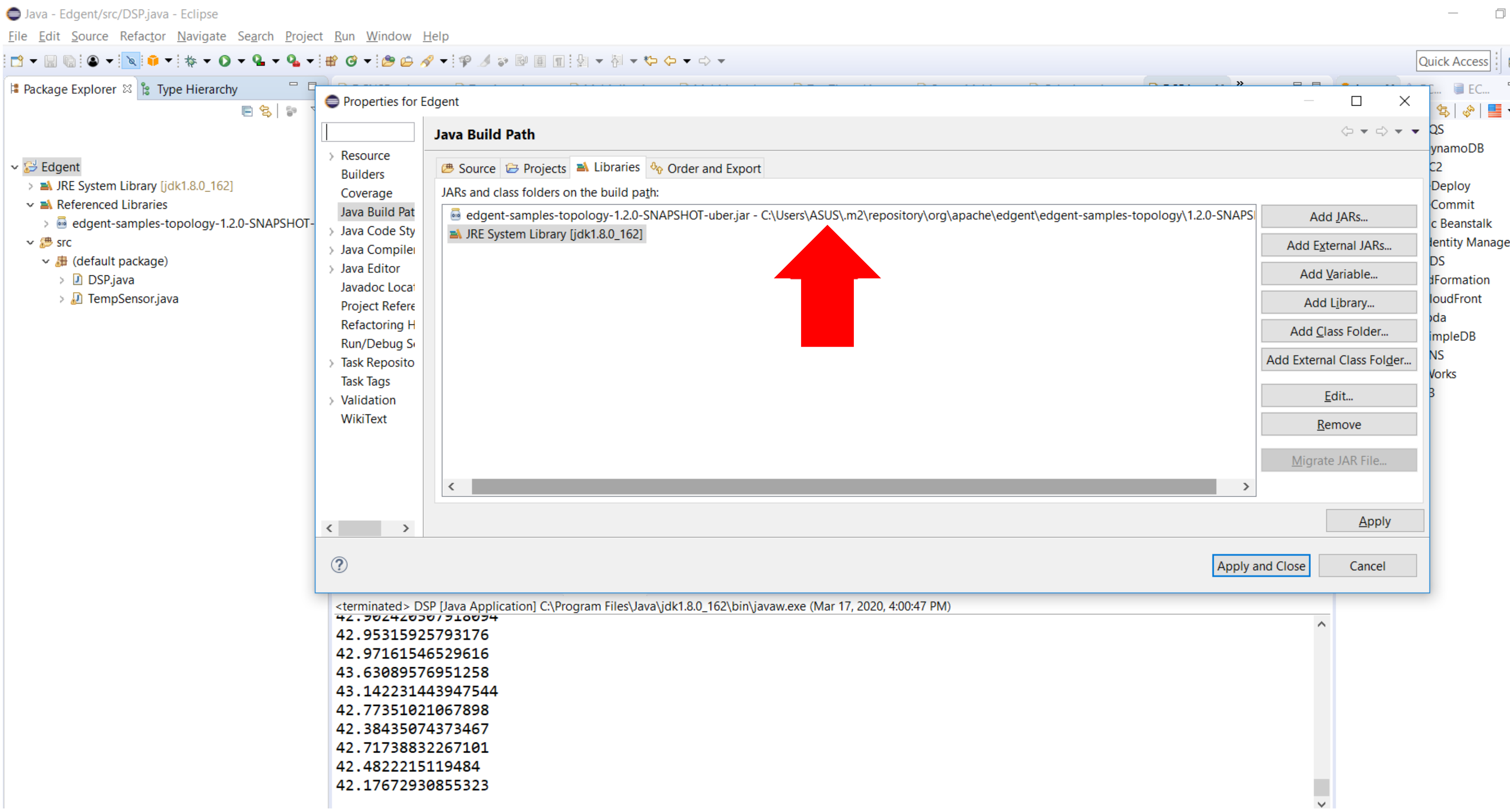
Apache Edgent (cont.)



Apache Edgent (cont.)

- What will you find in the following code:
    - ✓ Specifying a provider
    - ✓ Creating a topology
    - ✓ Creating a source TStream
    - ✓ Read from a temperature sensor every second
    - ✓ Convert to JSON
    - ✓ Send data to MQTT / Watson IoT Platform
    - ✓ Printing to output to screen
    - ✓ Submitting your topology
    - ✓ Building and Running

Adding Edgent API



JARs and class folders on the build path:

edgent-samples-topology-1.2.0-SNAPSHOT-uber.jar - C:\Users\ASUS\.m2\repository\org\apache\edgent\edgent-samples-topology\1.2.0-SNAPS

JRE System Library [jdk1.8.0_162]

```
<terminated> DSP [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (Mar 17, 2020, 4:00:47 PM)
42.95315925793176
42.97161546529616
43.63089576951258
43.142231443947544
42.77351021067898
42.38435074373467
42.71738832267101
42.4822215119484
42.17672930855323
```

How to bring data to Edgent?

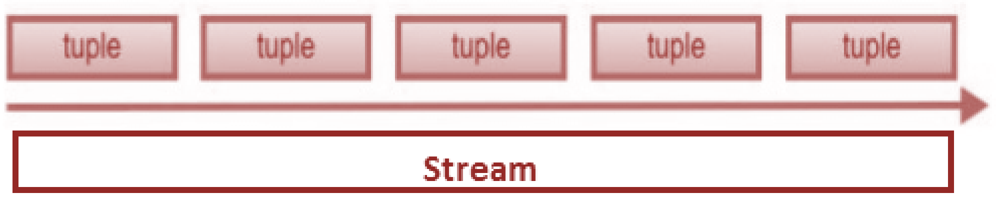
- One of the first things you probably want to do is to bring data into your *Edgent* applications

- The task is to create a source stream that contains the data that is to be processed and analyzed by Edgent

- Edgent provides several connectors providing source streams, such as IoTF, MQTT, Kafka, Files, HTTP-server, etc.
  - ✓ But for a specific device or application you may need to develop your own source streams.

Source Stream

- You need to produce a *source stream (In Edgent, it is shown by* TStream*)*
  - ✓ Data of sensors,
  - ✓ Data polled from an HTTP server
- Each tuple on the stream represents a reading from a sensor



- Three styles of bringing data into Edgent:
  1. Polling - periodically polling of a sensor's value.
  2. Blocking - a request is made to fetch data from a sensor
  3. Events - an event is created (by a non-Edgent framework) when the sensor changes

Window and Sink Stream

- A window represents a continuously updated ordered list of tuples according to the criteria that created it.
- For example, *s.last(10, zero())* declares a window with a single partition that at any time contains the last ten tuples seen on stream *s*.

- Termination point (sink) for a stream.

Topology

- A topology represents an application and describes the structure of your processing graph.

- A topology is a graph of
  - ✓ data streams and
  - ✓ their processing transformations.

- This class provides some fundamental generic methods to create source streams, such as source, poll, strings.

```
DirectProvider dp = new DirectProvider();
Topology topo = dp.newTopology();
```

Provider

- A provider is where the application runs when submit() method is called.

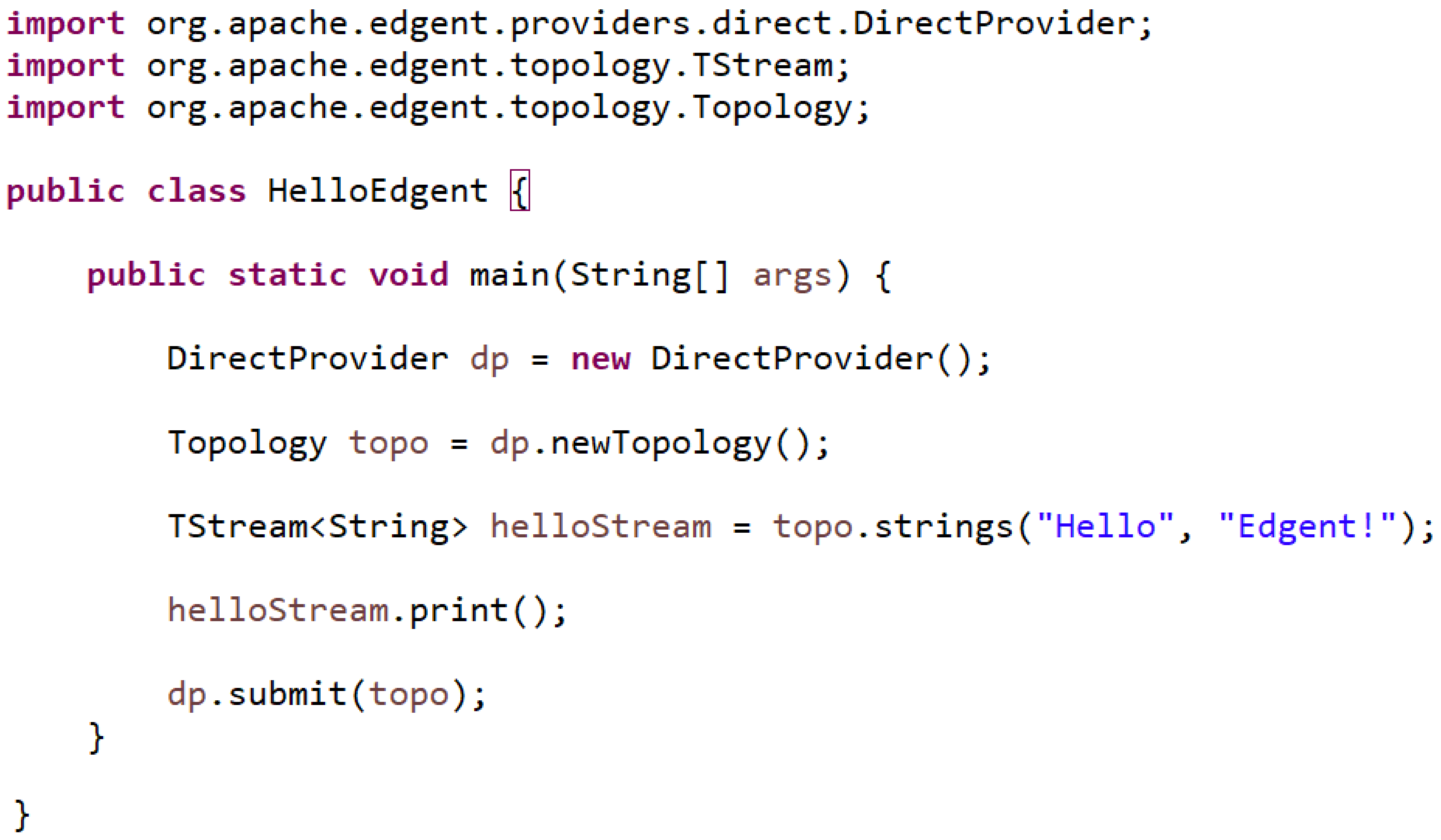
- ***TopologyProvider***
  A provider for creating and executing topologies

- ***DirectProvider*** is a TopologyProvider that runs a submitted topology as a Job in threads in the current machine

  - ✓ A job (execution of a topology) continues to execute while any of its elements have remaining work, such as any of the topology's source streams can generate tuples

  - ✓ "Endless" source streams never terminate - e.g., a stream created by generate(), poll(), or events().
    Hence, a job with such sources runs until either it or some other entity terminates it

Hello Edgent!



```java
import org.apache.edgent.providers.direct.DirectProvider;
import org.apache.edgent.topology.TStream;
import org.apache.edgent.topology.Topology;

public class HelloEdgent {

    public static void main(String[] args) {

        DirectProvider dp = new DirectProvider();

        Topology topo = dp.newTopology();

        TStream<String> helloStream = topo.strings("Hello", "Edgent!");

        helloStream.print();

        dp.submit(topo);
    }

}
```

Java APIs

APIs helpful to download, stream and process a **.csv* file:

```
java.io.BufferedInputStream;

java.io.InputStream;

java.net.URL;

org.apache.commons.csv;
```

Getting started with Apache Edgent Samples

- Open the *Clone or download* pulldown at the Edgent Samples GitHub repository
- Click on *Download ZIP*
- Unpack the downloaded ZIP
  - ➢ cd <the cloned or unpacked downloaded samples folder>
  - ➢ ./mvnw clean install
  - ➢ cd topology ./run-sample.sh HelloEdgent *# prints a hello message and terminates*
  - ➢ Hello
  - ➢ Edgent!
  - ➢ ...

Assignment 6

- The programs written in the ***Edgent*** framework will be accepted.

Please write a *`GpsAnalytics`* application and consider the following steps:

1. Each group member just pick a user from the following dataset:
   - ❑ This is a GPS trajectory dataset collected in (Microsoft Research Asia) GeoLife project by 182 users in a period of over two years (from April 2007 to August 2012). https://www.microsoft.com/en-us/research/publication/mining-interesting-locations-and-travel-sequences-from-gps-trajectories/

Assignment 6

2. Your code should detect the anomalous data in the dataset. You can assume that it is a criminal GPS tracking device. Therefore, you need to utilize the mean and variance measurements. Then try to calculate the Gaussian distribution (next slide provides the helps). Afterward, compare the probability of each data. Lowest and highest probability should be reported.

3. Please just consider the latitude and longitude of each row in the dataset:
   Field 1: Latitude in decimal degrees.
   Field 2: Longitude in decimal degrees
   (OPTIONAL).

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$p(x_i) >>> p(x_{i-1})$ or $p(x_i) <<< p(x_{i-1}) \rightarrow$ Anomalous data

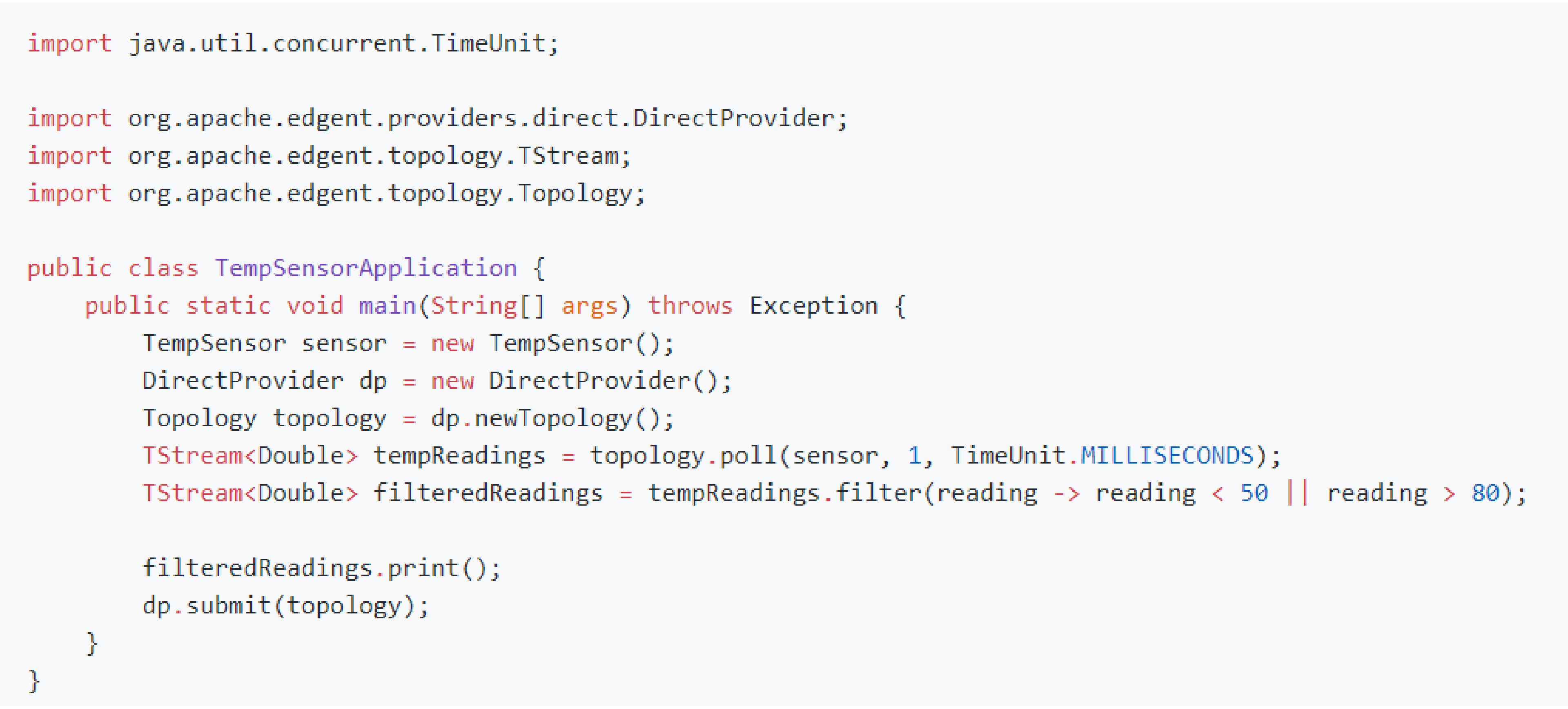
```java
import java.util.Random;

import org.apache.edgent.function.Supplier;

/**
 * Every time get() is called, TempSensor generates a temperature reading.
 */
public class TempSensor implements Supplier<Double> {
    double currentTemp = 65.0;
    Random rand;

    TempSensor(){
        rand = new Random();
    }

    @Override
    public Double get() {
        // Change the current temperature some random amount
        double newTemp = rand.nextGaussian() + currentTemp;
        currentTemp = newTemp;
        return currentTemp;
    }
}
```



```java
import java.util.concurrent.TimeUnit;

import org.apache.edgent.providers.direct.DirectProvider;
import org.apache.edgent.topology.TStream;
import org.apache.edgent.topology.Topology;

public class TempSensorApplication {
    public static void main(String[] args) throws Exception {
        TempSensor sensor = new TempSensor();
        DirectProvider dp = new DirectProvider();
        Topology topology = dp.newTopology();
        TStream<Double> tempReadings = topology.poll(sensor, 1, TimeUnit.MILLISECONDS);
        TStream<Double> filteredReadings = tempReadings.filter(reading -> reading < 50 || reading > 80);

        filteredReadings.print();
        dp.submit(topology);
    }
}
```

https://github.com/apache/incubator-retired-edgent-website/blob/master/site/docs/old-edgent-getting-started.md

References

- https://www.itec.aau.at/~narges/edgent/content/docs/home.html
- https://quarks-edge.github.io/quarks/docs/javadoc/index.html
- https://github.com/apache/incubator-retired-edgent
- https://github.com/apache/incubator-retired-edgent-samples
- https://quarks-edge.github.io/quarks/docs/javadoc/index.html
- https://repo1.maven.org/maven2/org/apache/edgent/
- https://www.researchgate.net/profile/Peter-Panfilov/publication/329456632_Building_Predictive_Maintenance_Framework_for_Smart_Environment_Application_Systems/links/5c2ba5ff92851c22a353636c/Building-Predictive-Maintenance-Framework-for-Smart-Environment-Application-Systems.pdf
- https://www.youtube.com/channel/UC_uXzbJmQzPkODsEE2E7PgA/videos