Apache Flink hands-on

instructions

Vasiliki Kalavri
kalavriv@inf.ethz.ch
@vkalavri

Resources

- **Training**: http://training.data-artisans.com/
- **Setup instructions** (own laptop): http://training.data-artisans.com/devEnvSetup.html
- **Data**: http://training.data-artisans.com/exercises/taxiData.html
- **DataStream API documentation**: https://ci.apache.org/projects/flink/flink-docs-release-1.6/dev/datastream_api.html

Taxi Rides

```
rideId        : Long      // a unique id for each ride
taxiId        : Long      // a unique id for each taxi
driverId      : Long      // a unique id for each driver
isStart       : Boolean   // TRUE for start events, FALSE for end
startTime     : DateTime  // the start time of a ride
endTime       : DateTime  // the end time of a ride,
                          //   "1970-01-01 00:00:00" for start
startLon      : Float     // the longitude of the start location
startLat      : Float     // the latitude of the start location
endLon        : Float     // the longitude of the end location
endLat        : Float     // the latitude of the end location
passengerCnt  : Short     // number of passengers on the ride
```

Taxi Fares

```
rideId         : Long      // a unique id for each ride
taxiId         : Long      // a unique id for each taxi
driverId       : Long      // a unique id for each driver
startTime      : DateTime  // the start time of a ride
paymentType    : String    // CSH or CRD
tip            : Float     // tip for this ride
tolls          : Float     // tolls for this ride
totalFare      : Float     // total fare collected
```

Generate Ride Events

```
// get an ExecutionEnvironment
StreamExecutionEnvironment env =
  StreamExecutionEnvironment.getExecutionEnvironment();

// configure event-time processing
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);

// get the taxi ride data stream
DataStream<TaxiRide> rides = env.addSource(
  new TaxiRideSource("/path/to/nycTaxiRides.gz", maxDelay,
servingSpeed));
```

max event delay

event speedup factor

Generate Fare Events

```
// get an ExecutionEnvironment
StreamExecutionEnvironment env =
  StreamExecutionEnvironment.getExecutionEnvironment();

// configure event-time processing
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);

// get the taxi fare data stream
DataStream<TaxiFare> rides = env.addSource(
  new TaxiFareSource("/path/to/nycTaxiFares.gz", maxDelay,
servingSpeed));
```

Test Setup

1. Open **com.dataartisans.flinktraining.exercises.datastream_java.utils.ExerciseBase** in your IDE

2. Update **pathToRideData** and **pathToFareData**

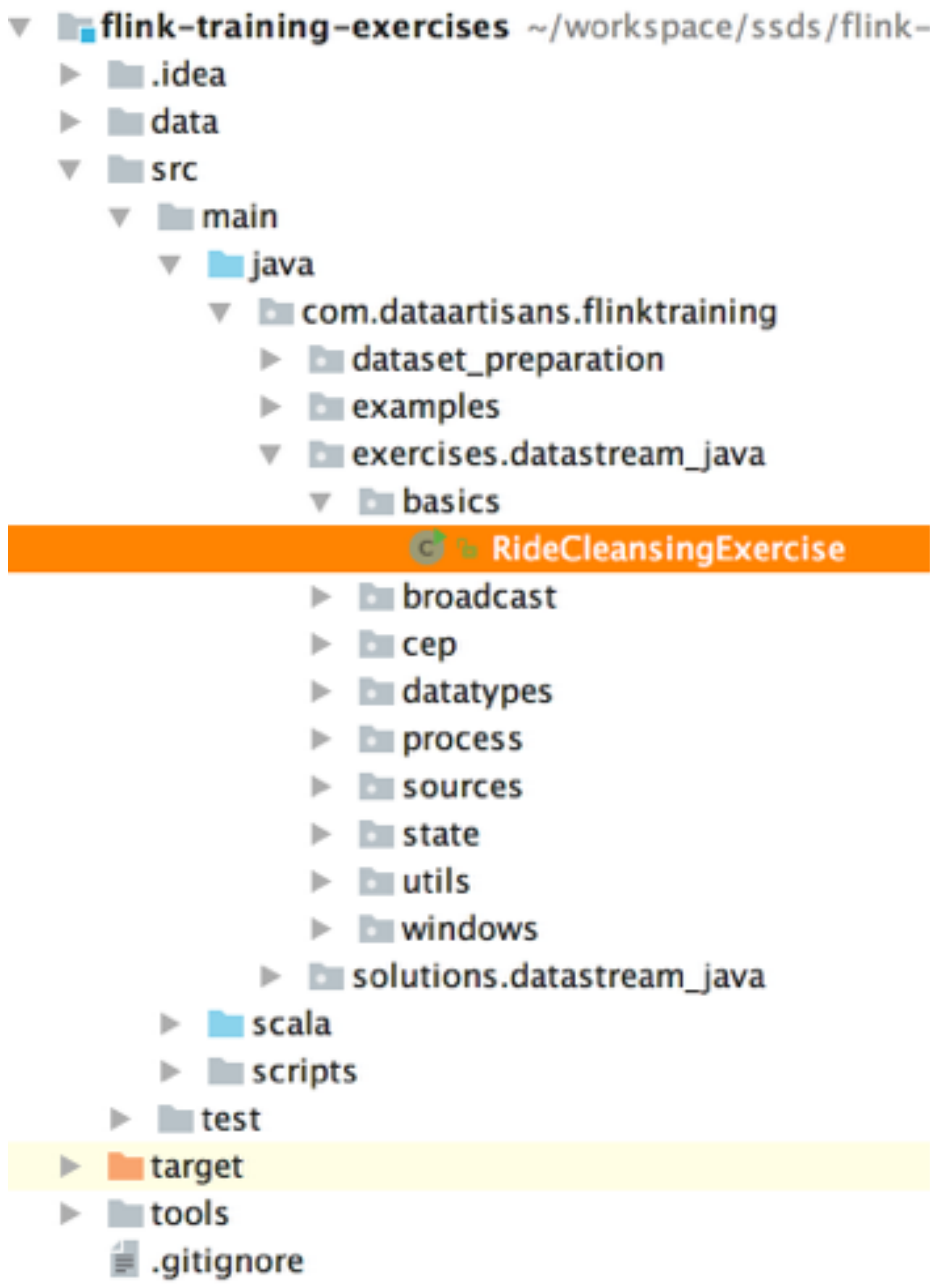
3. Open **com.dataartisans.flinktraining.examples.datastream_java.basics.RideCount** in your IDE

4. Run the **main()** method

5. Watch the result stream!

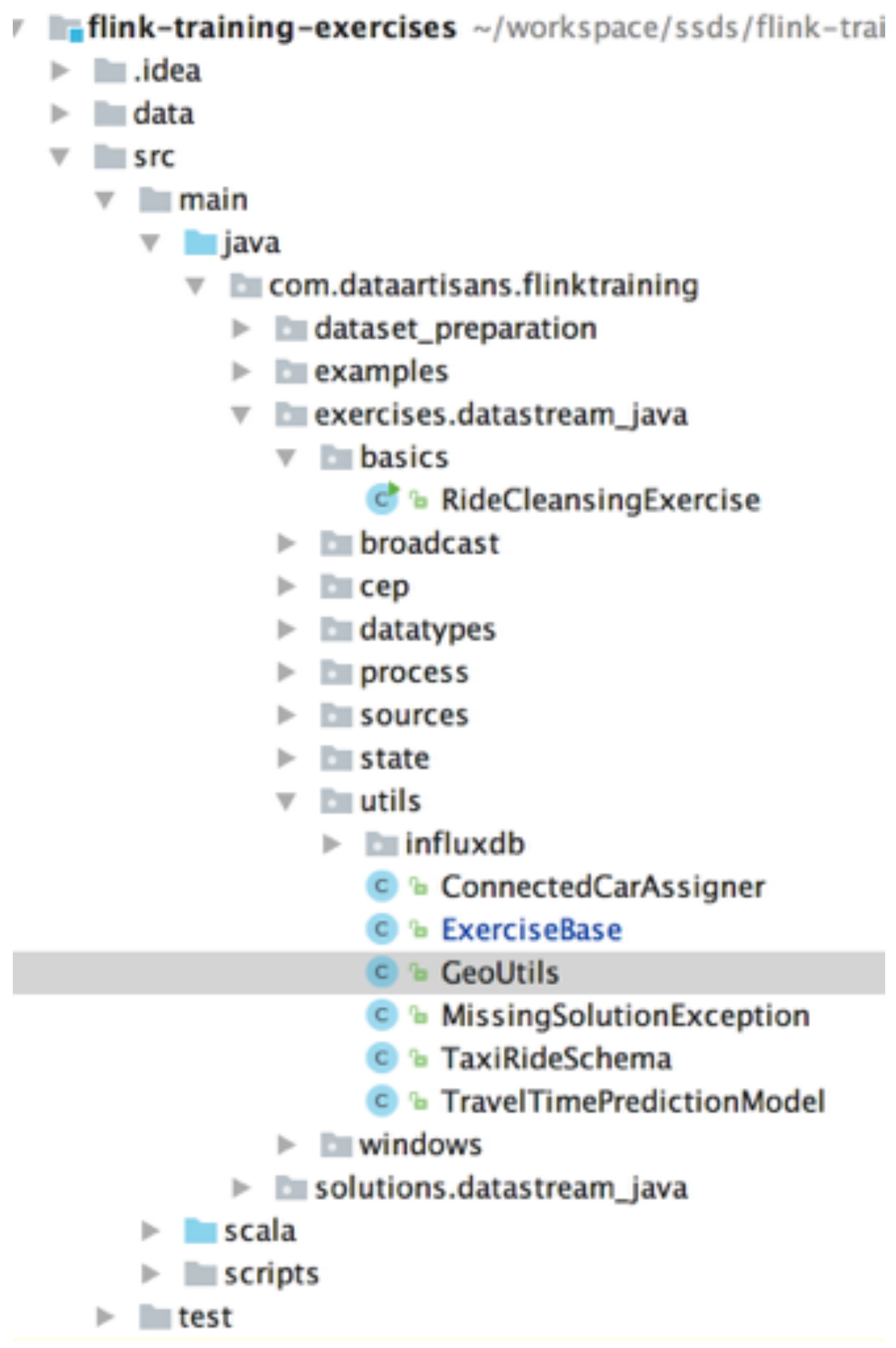
Exercise #1: RideCleansing

Filter out rides that do not start/end in NYC



```
▼ flink-training-exercises ~/workspace/ssds/flink-
  ▶ .idea
  ▶ data
  ▼ src
    ▼ main
      ▼ java
        ▼ com.dataartisans.flinktraining
          ▶ dataset_preparation
          ▶ examples
          ▼ exercises.datastream_java
            ▼ basics
                RideCleansingExercise
            ▶ broadcast
            ▶ cep
            ▶ datatypes
            ▶ process
            ▶ sources
            ▶ state
            ▶ utils
            ▶ windows
          ▶ solutions.datastream_java
      ▶ scala
      ▶ scripts
    ▶ test
  ▶ target
  ▶ tools
    .gitignore
```

Exercise #1: GeoUtils

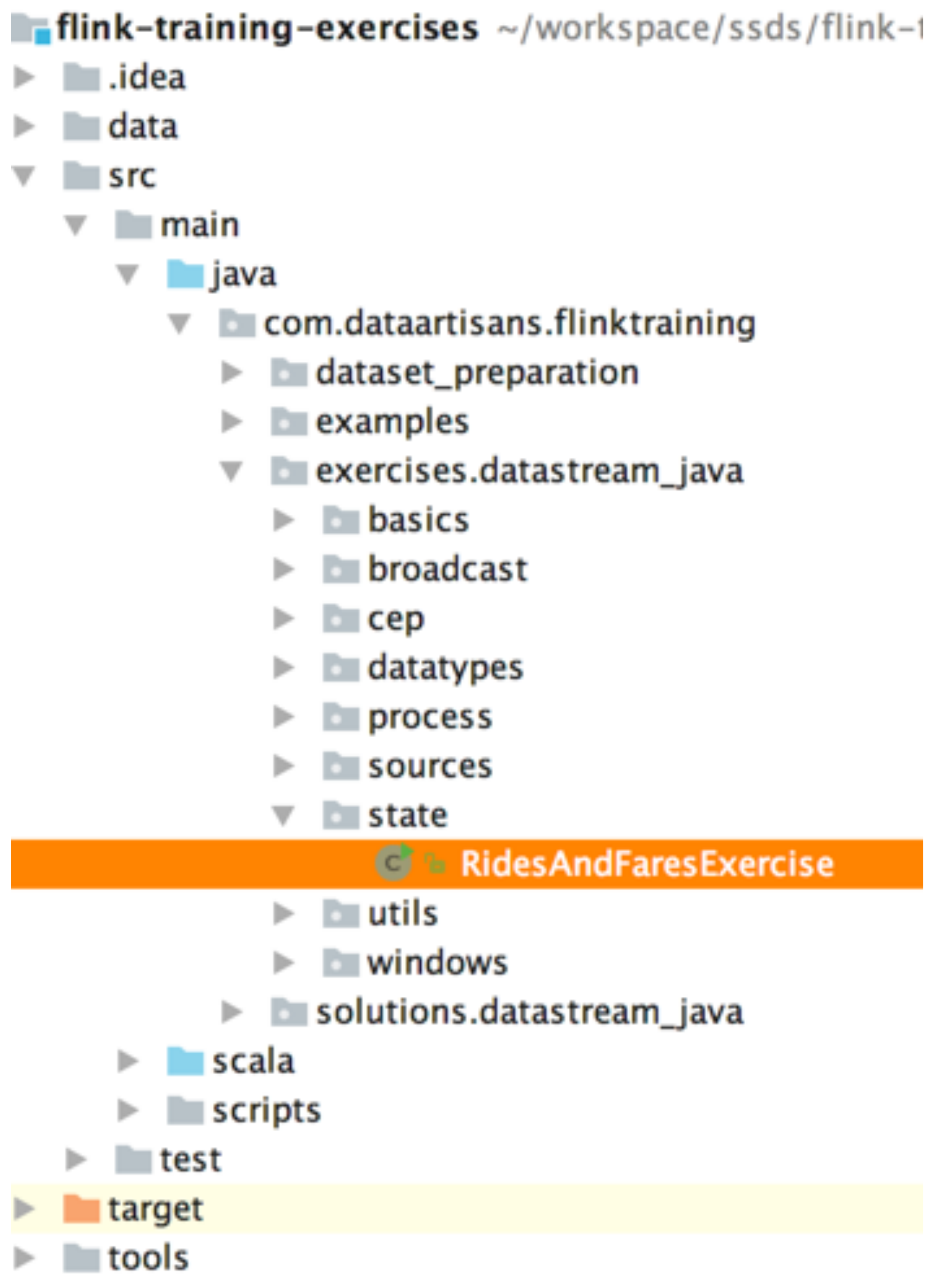


```
▼ flink-training-exercises ~/workspace/ssds/flink-trai
  ▶ .idea
  ▶ data
  ▼ src
    ▼ main
      ▼ java
        ▼ com.dataartisans.flinktraining
          ▶ dataset_preparation
          ▶ examples
          ▼ exercises.datastream_java
            ▼ basics
                RideCleansingExercise
            ▶ broadcast
            ▶ cep
            ▶ datatypes
            ▶ process
            ▶ sources
            ▶ state
            ▼ utils
              ▶ influxdb
                ConnectedCarAssigner
                ExerciseBase
                GeoUtils
                MissingSolutionException
                TaxiRideSchema
                TravelTimePredictionModel
            ▶ windows
          ▶ solutions.datastream_java
      ▶ scala
      ▶ scripts
    ▶ test
```

```java
/**
 * Checks if a location specified by longitude and latitude values is
 * within the geo boundaries of New York City.
 *
 * @param lon longitude of the location to check
 * @param lat latitude of the location to check
 *
 * @return true if the location is within NYC boundaries, otherwise false.
 */
public static boolean isInNYC(float lon, float lat){}
```

Exercise #2: State

Join each TaxiRide
with its TaxiFare



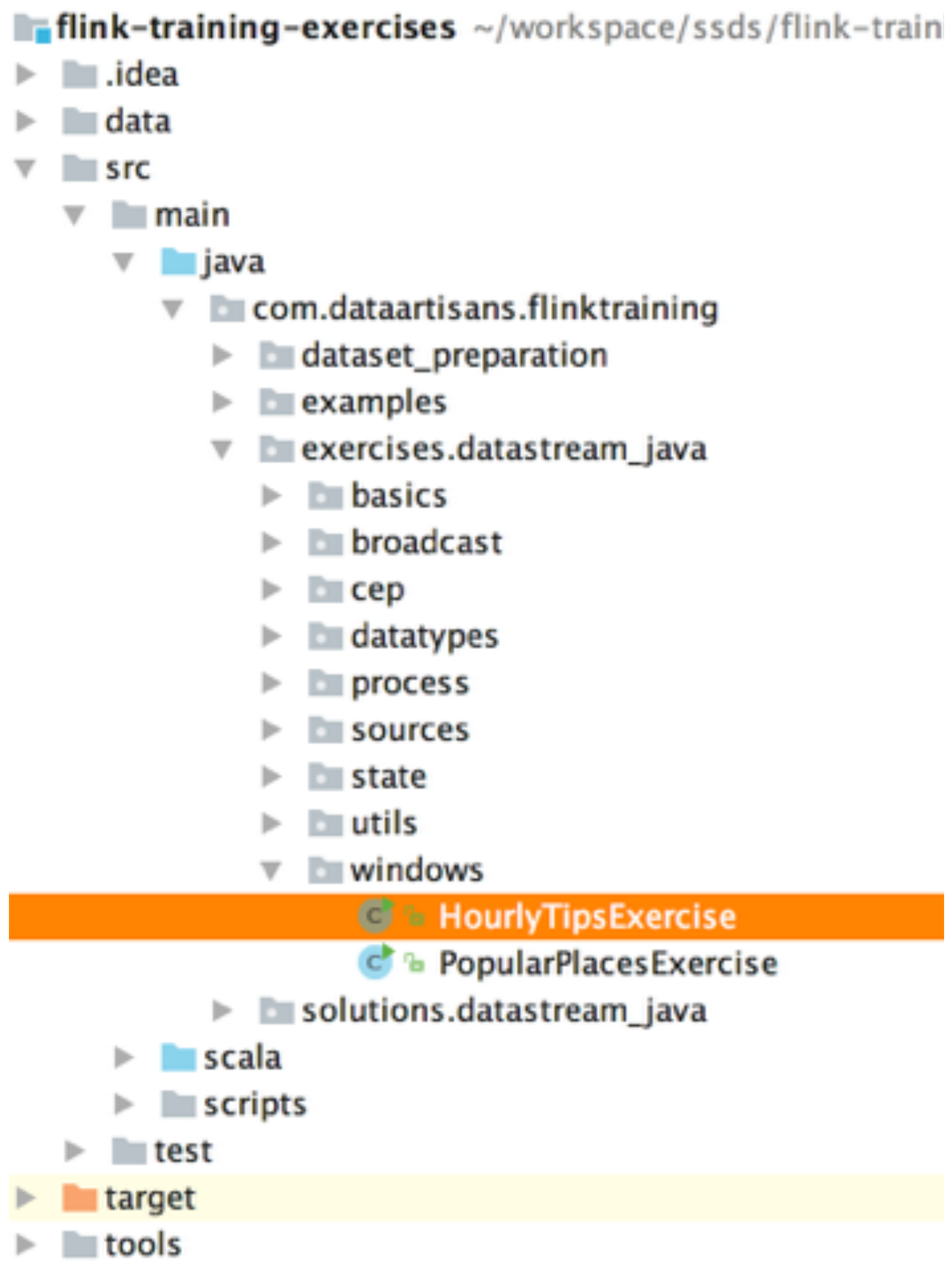
```
flink-training-exercises ~/workspace/ssds/flink-
▶ .idea
▶ data
▼ src
  ▼ main
    ▼ java
      ▼ com.dataartisans.flinktraining
        ▶ dataset_preparation
        ▶ examples
        ▼ exercises.datastream_java
          ▶ basics
          ▶ broadcast
          ▶ cep
          ▶ datatypes
          ▶ process
          ▶ sources
          ▼ state
              RidesAndFaresExercise
          ▶ utils
          ▶ windows
        ▶ solutions.datastream_java
    ▶ scala
    ▶ scripts
  ▶ test
▶ target
▶ tools
```

Exercise #2: Hints

- The order of arrival of the ride and fare events is not guaranteed.
- If you receive a fare event, store it until you receive the matching ride and vice versa. Once you have a match, you can emit the result.
- Clear the state once it is no longer needed!
- Check: https://ci.apache.org/projects/flink/flink-docs-release-1.6/dev/stream/state/state.html#using-managed-keyed-state

Exercise #3: Windows

Compute which driver is earning the most tips every hour



```
flink-training-exercises ~/workspace/ssds/flink-train
▶ .idea
▶ data
▼ src
  ▼ main
    ▼ java
      ▼ com.dataartisans.flinktraining
        ▶ dataset_preparation
        ▶ examples
        ▼ exercises.datastream_java
          ▶ basics
          ▶ broadcast
          ▶ cep
          ▶ datatypes
          ▶ process
          ▶ sources
          ▶ state
          ▶ utils
          ▼ windows
              HourlyTipsExercise
              PopularPlacesExercise
        ▶ solutions.datastream_java
    ▶ scala
    ▶ scripts
  ▶ test
▶ target
▶ tools
```

Exercise #3: Hints

1. Compute in 2 steps:
   1.1. Total tips per hour per driver
   1.2. Driver with max tip
2. Use an incremental `AggregateFunction` with a `ProcessWindowFunction` for 1.1
   2.1. https://ci.apache.org/projects/flink/flink-docs-release-1.6/dev/stream/operators/windows.html#processwindowfunction-with-incremental-aggregation
3. Use a global window for 1.2