

```

/*
 * Copyright 2017 data Artisans GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */

```

```

package
com.dataartisans.flinktraining.exercises.datastream_java.process;

import
com.dataartisans.flinktraining.exercises.datastream_java.datatypes.C
onectedCarEvent;
import
com.dataartisans.flinktraining.exercises.datastream_java.utils.Compa
reByTimestampAscending;
import
com.dataartisans.flinktraining.exercises.datastream_java.utils.Conne
ctedReader;
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.common.state.ValueState;
import org.apache.flink.api.common.state.ValueStateDescriptor;
import org.apache.flink.api.common.typeinfo.TypeHint;
import org.apache.flink.api.common.typeinfo.TypeInformation;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.configuration.Configuration;
import org.apache.flink.streaming.api.TimeCharacteristic;
import org.apache.flink.streaming.api.TimerService;
import org.apache.flink.streaming.api.datastream.DataStream;
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironmen
t;
import org.apache.flink.streaming.api.functions.ProcessFunction;
import org.apache.flink.streaming.api.windowing.time.Time;
import org.apache.flink.util.Collector;

import java.util.PriorityQueue;

public class CarEventSort {
    public static void main(String[] args) throws Exception {

        // read parameters
    }
}

```

```

        ParameterTool params =
ParameterTool.fromArgs(args);
        String input = params.getRequired("input");

        // set up streaming execution environment
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);

        // connect to the data file
        DataStream<String> carData =
env.readTextFile(input);

        // map to events
        DataStream<ConnectedCarEvent> events = carData
            .map(new MapFunction<String,
ConnectedCarEvent>() {
                @Override
                public ConnectedCarEvent
map(String line) throws Exception {
                    return
ConnectedCarEvent.fromString(line);
                }
            })
            .assignTimestampsAndWatermarks(new
ConnectedCarAssigner());

        // sort events
        events
            .keyBy("carId")
            .timeWindow(Time.seconds(10),
Time.seconds(5))
            .sum("speed")
            .print();

        env.execute("Sort Connected Car Events");
    }
}

```