CptS 464 Final Project Report

By Hai (Harry) Sun

My project3 code is under hsun1/HWFall2010/CPTS464/project3.

The project is useful in two aspects. On one hand, we're given a fabulous chance to dig into distributed service and learn much about publisher/subscriber working mode. On the other hand, the project drives me to review some key programming issues such as multi-thread, properties parser, random generator, etc.

The project is designed well and so I cannot provide any concrete suggestion about improvement. But if we have a chance to understand why there is a memory exclusion error in our running context, that'll be fine for deeper understanding.

Run-time Print Out

1. PubLauncher

Start the PubLauncher!
Thread #0 started.
Thread #1 started.
Thread #2 started.
Thread #3 started.
Thread #4 started.
Thread #5 started.
All buses have started. Waiting for them to terminate...
Bus11 has published a position message at stop #1 on route Express1 at 8:18:02 PM
Bus22 has published a position message at stop #1 on route Express2 at 8:18:03 PM
Bus21 has published a position message at stop #1 on route Express2 at 8:18:03 PM
Bus12 has published a position message at stop #1 on route Express1 at 8:18:03 PM
Bus13 has published a position message at stop #1 on route Express1 at 8:18:03 PM
Bus23 has published a position message at stop #1 on route Express2 at 8:18:03 PM
Bus22 has published an accident message at stop #1 on route Express2 at 8:18:07 PM
Bus11 has published a position message at stop #2 on route Express1 at 8:18:09 PM
Bus21 has published a position message at stop #2 on route Express2 at 8:18:09 PM
Bus12 has published a position message at stop #2 on route Express1 at 8:18:10 PM
Bus13 has published a position message at stop #2 on route Express1 at 8:18:10 PM
Bus23 has published a position message at stop #2 on route Express2 at 8:18:10 PM
Bus13 has published a position message at stop #3 on route Express1 at 8:18:17 PM
Bus11 has published a position message at stop #3 on route Express1 at 8:18:17 PM
Bus23 has published a position message at stop #3 on route Express2 at 8:18:18 PM
Bus21 has published a position message at stop #3 on route Express2 at 8:18:18 PM
Bus23 has published an accident message at stop #3 on route Express2 at 8:18:23 PM
Bus13 has published a position message at stop #4 on route Express1 at 8:18:24 PM
Bus22 has published a position message at stop #2 on route Express2 at 8:18:25 PM
Bus12 has published a position message at stop #4 on route Express1 at 8:18:25 PM
Bus11 has published a position message at stop #4 on route Express1 at 8:18:25 PM
Bus21 has published a position message at stop #4 on route Express2 at 8:18:26 PM
Bus13 has published a position message at stop #1 on route Express1 at 8:18:30 PM
MemoryMutex_delete:OS semctl() failure, error 0X1
Bus11 has published a position message at stop #1 on route Express1 at 8:18:32 PM
Bus22 has published a position message at stop #3 on route Express2 at 8:18:33 PM
Bus21 has published a position message at stop #5 on route Express2 at 8:18:33 PM
Bus12 has published an accident message at stop #1 on route Express1 at 8:18:37 PM
Bus13 has published a position message at stop #2 on route Express1 at 8:18:38 PM
Bus22 has published an accident message at stop #3 on route Express2 at 8:18:38 PM
Bus11 has published a position message at stop #2 on route Express1 at 8:18:40 PM
Bus21 has published a position message at stop #6 on route Express2 at 8:18:42 PM
Bus23 has published a position message at stop #4 on route Express2 at 8:18:42 PM
Bus13 has published a position message at stop #3 on route Express1 at 8:18:45 PM
Bus23 has published an accident message at stop #4 on route Express2 at 8:18:47 PM

Bus11 has published a position message at stop #3 on route Express1 at 8:18:47 PM
Bus21 has published a position message at stop #1 on route Express2 at 8:18:50 PM
Bus13 has published a position message at stop #4 on route Express1 at 8:18:52 PM
Bus11 has published an accident message at stop #3 on route Express1 at 8:18:53 PM
Bus12 has published a position message at stop #2 on route Express1 at 8:18:53 PM
Bus22 has published a position message at stop #4 on route Express2 at 8:18:56 PM
gmail.com For non-production use only.
Bus13 has published a position message at stop #1 on route Express1 at 8:18:59 PM
Bus12 has published a position message at stop #3 on route Express1 at 8:19:01 PM
Bus22 has published a position message at stop #5 on route Express2 at 8:19:05 PM
Bus23 has published a position message at stop #5 on route Express2 at 8:19:05 PM
Bus21 has published a position message at stop #3 on route Express2 at 8:19:06 PM
Bus13 has published a position message at stop #2 on route Express1 at 8:19:07 PM
Bus12 has published a position message at stop #4 on route Express1 at 8:19:08 PM
Bus11 has published a position message at stop #4 on route Express1 at 8:19:10 PM
Bus22 has published a position message at stop #6 on route Express2 at 8:19:12 PM
Bus23 has published a position message at stop #6 on route Express2 at 8:19:14 PM
Bus13 has published a position message at stop #3 on route Express1 at 8:19:14 PM
Bus21 has published a position message at stop #4 on route Express2 at 8:19:14 PM
Bus12 has published a position message at stop #1 on route Express1 at 8:19:16 PM
Bus11 has published a position message at stop #1 on route Express1 at 8:19:17 PM
n-production use only.
Expires on 07-sep-2011 See wwBus22 has published a position message at stop #1 on route Express2 at 8:19:20 PM
Bus13 has published a position message at stop #4 on route Express1 at 8:19:21 PM
Bus23 has published a position message at stop #1 on route Express2 at 8:19:22 PM
Bus21 has published a position message at stop #5 on route Express2 at 8:19:22 PM
Bus12 has published a position message at stop #2 on route Express1 at 8:19:22 PM
Bus11 has published a position message at stop #2 on route Express1 at 8:19:24 PM
Bus23 has published a position message at stop #2 on route Express2 at 8:19:29 PM
Bus22 has published a position message at stop #2 on route Express2 at 8:19:29 PM
Bus12 has published a position message at stop #3 on route Express1 at 8:19:30 PM
Bus11 has published a position message at stop #3 on route Express1 at 8:19:31 PM
Bus21 has published a position message at stop #6 on route Express2 at 8:19:31 PM
Bus12 has published a position message at stop #4 on route Express1 at 8:19:37 PM
Bus22 has published a position message at stop #3 on route Express2 at 8:19:38 PM
Bus23 has published a position message at stop #3 on route Express2 at 8:19:38 PM
EBus11 has published a position message at stop #4 on route Express1 at 8:19:38 PM
Bus21 has published a position message at stop #1 on route Express2 at 8:19:39 PM
Bus21 has published a position message at stop #2 on route Express2 at 8:19:46 PM
Bus22 has published a position message at stop #4 on route Express2 at 8:19:47 PM
Bus23 has published a position message at stop #4 on route Express2 at 8:19:47 PM
Bus23 has published an accident message at stop #4 on route Express2 at 8:19:53 PM
Bus21 has published a position message at stop #3 on route Express2 at 8:19:55 PM
Bus22 has published a position message at stop #5 on route Express2 at 8:19:56 PM
Bus21 has published a position message at stop #4 on route Express2 at 8:20:03 PM

Bus22 has published a position message at stop #6 on route Express2 at 8:20:04 PM
Bus23 has published a position message at stop #5 on route Express2 at 8:20:11 PM
Bus21 has published a position message at stop #5 on route Express2 at 8:20:12 PM
Bus22 has published a position message at stop #1 on route Express2 at 8:20:12 PM
Bus23 has published a position message at stop #6 on route Express2 at 8:20:19 PM
Bus21 has published a position message at stop #6 on route Express2 at 8:20:21 PM
Bus23 has published a position message at stop #1 on route Express2 at 8:20:28 PM
Bus22 has published a position message at stop #3 on route Express2 at 8:20:28 PM
Bus22 has published an accident message at stop #3 on route Express2 at 8:20:33 PM
Bus23 has published a position message at stop #2 on route Express2 at 8:20:35 PM
Bus23 has published a position message at stop #3 on route Express2 at 8:20:43 PM
Bus22 has published a position message at stop #4 on route Express2 at 8:20:52 PM
Bus23 has published a position message at stop #4 on route Express2 at 8:20:52 PM
Bus22 has published an accident message at stop #4 on route Express2 at 8:20:57 PM
Bus23 has published an accident message at stop #4 on route Express2 at 8:20:58 PM
Bus23 has published a position message at stop #5 on route Express2 at 8:21:15 PM
Bus22 has published a position message at stop #5 on route Express2 at 8:21:15 PM
Bus23 has published a position message at stop #6 on route Express2 at 8:21:23 PM
Bus22 has published a position message at stop #6 on route Express2 at 8:21:23 PM
Finally all done!

2. Operator

MessageType	*Route*	*Vehicle*	*Traffic*	*Stop#*	*#Stop*	*TimeBetweenStops*	*Fill%*	*Timestamp*
position	*Express1*	*Bus12*	*light*	*1*	*4*	*1.500000*	*95*	*8:18:03 PM*
position	*Express2*	*Bus23*	*normal*	*1*	*6*	*3.000000*	*69*	*8:18:03 PM*
position	*Express2*	*Bus22*	*light*	*1*	*6*	*12.250000*	*84*	*8:18:03 PM*
position	*Express1*	*Bus11*	*normal*	*1*	*4*	*2.000000*	*22*	*8:18:02 PM*
position	*Express1*	*Bus13*	*heavy*	*1*	*4*	*2.500000*	*20*	*8:18:03 PM*
position	*Express2*	*Bus21*	*light*	*1*	*6*	*2.250000*	*61*	*8:18:03 PM*
position	*Express1*	*Bus11*	*heavy*	*2*	*4*	*2.500000*	*76*	*8:18:09 PM*
position	*Express2*	*Bus21*	*heavy*	*2*	*6*	*3.750000*	*48*	*8:18:09 PM*
position	*Express1*	*Bus12*	*normal*	*2*	*4*	*2.000000*	*11*	*8:18:10 PM*
position	*Express1*	*Bus13*	*normal*	*2*	*4*	*2.000000*	*3*	*8:18:10 PM*
position	*Express2*	*Bus23*	*light*	*2*	*6*	*2.250000*	*19*	*8:18:10 PM*

position Express1 Bus12 heavy 3 4 2.500000 64 8:18:17 PM
position Express1 Bus11 heavy 3 4 2.500000 81 8:18:17 PM
position Express1 Bus13 light 3 4 1.500000 19 8:18:17 PM
position Express2 Bus23 heavy 3 6 13.750000 24 8:18:18 PM
position Express2 Bus21 light 3 6 2.250000 13 8:18:18 PM
position Express1 Bus13 light 4 4 1.500000 32 8:18:24 PM
position Express2 Bus22 normal 2 6 3.000000 89 8:18:25 PM
position Express1 Bus12 light 4 4 1.500000 41 8:18:25 PM
position Express1 Bus11 heavy 4 4 2.500000 79 8:18:25 PM
position Express2 Bus21 light 4 6 2.250000 25 8:18:26 PM
position Express1 Bus13 normal 1 4 2.000000 58 8:18:30 PM
position Express1 Bus12 light 1 4 11.500000 29 8:18:31 PM
position Express1 Bus11 normal 1 4 2.000000 96 8:18:32 PM
position Express2 Bus22 light 3 6 12.250000 46 8:18:33 PM
position Express2 Bus21 normal 5 6 3.000000 79 8:18:33 PM
position Express1 Bus13 normal 2 4 2.000000 65 8:18:38 PM
accident Express2 Bus22 3 8:18:38 PM
position Express1 Bus11 heavy 2 4 2.500000 79 8:18:40 PM
position Express2 Bus21 normal 6 6 3.000000 73 8:18:42 PM
position Express2 Bus23 normal 4 6 13.000000 94 8:18:42 PM
position Express1 Bus13 light 3 4 1.500000 97 8:18:45 PM
accident Express2 Bus23 4 8:18:47 PM
position Express1 Bus11 normal 3 4 12.000000 49 8:18:47 PM

position Express2 Bus21 normal 1 6 3.000000 89 8:18:50 PM
position Express1 Bus13 normal 4 4 2.000000 81 8:18:52 PM
accident Express1 Bus11 3 8:18:53 PM
position Express1 Bus12 normal 2 4 2.000000 69 8:18:53 PM
position Express2 Bus22 heavy 4 6 3.750000 52 8:18:56 PM
position Express2 Bus21 normal 2 6 3.000000 43 8:18:58 PM
position Express1 Bus13 heavy 1 4 2.500000 22 8:18:59 PM
position Express1 Bus12 heavy 3 4 2.500000 61 8:19:01 PM
position Express2 Bus22 light 5 6 2.250000 41 8:19:05 PM
position Express2 Bus23 normal 5 6 3.000000 21 8:19:05 PM
position Express2 Bus21 light 3 6 2.250000 6 8:19:06 PM
position Express1 Bus13 normal 2 4 2.000000 72 8:19:07 PM
position Express1 Bus12 normal 4 4 2.000000 23 8:19:08 PM
position Express1 Bus11 light 4 4 1.500000 38 8:19:10 PM
position Express2 Bus22 normal 6 6 3.000000 39 8:19:12 PM
position Express1 Bus13 normal 3 4 2.000000 54 8:19:14 PM
position Express2 Bus21 normal 4 6 3.000000 26 8:19:14 PM
position Express1 Bus12 light 1 4 1.500000 95 8:19:16 PM
position Express1 Bus11 normal 1 4 2.000000 61 8:19:17 PM
position Express2 Bus22 heavy 1 6 3.750000 32 8:19:20 PM
position Express1 Bus13 light 4 4 1.500000 20 8:19:21 PM
position Express2 Bus23 light 1 6 2.250000 13 8:19:22 PM
position Express2 Bus21 heavy 5 6 3.750000 36 8:19:22 PM

position	Express1	Bus11	light	2	4	1.500000
31	8:19:24 PM					
position	Express2	Bus23	heavy	2	6	3.750000
79	8:19:29 PM					
position	Express2	Bus22	normal	2	6	3.000000
97	8:19:29 PM					
position	Express1	Bus12	heavy	3	4	2.500000
87	8:19:30 PM					
position	Express1	Bus11	heavy	3	4	2.500000
42	8:19:31 PM					
position	Express2	Bus21	light	6	6	2.250000
31	8:19:31 PM					
position	Express1	Bus12	normal	4	4	2.000000
24	8:19:37 PM					
position	Express2	Bus22	heavy	3	6	3.750000
88	8:19:38 PM					
position	Express2	Bus23	heavy	3	6	3.750000
90	8:19:38 PM					
position	Express1	Bus11	heavy	4	4	2.500000
36	8:19:38 PM					
position	Express2	Bus21	light	1	6	2.250000
61	8:19:39 PM					
position	Express2	Bus21	normal	2	6	3.000000
58	8:19:46 PM					
position	Express2	Bus22	heavy	4	6	3.750000
41	8:19:47 PM					
position	Express2	Bus23	normal	4	6	13.000000
77	8:19:47 PM					
accident		Express2	Bus23			4
8:19:53 PM						
position	Express2	Bus21	normal	3	6	3.000000
97	8:19:55 PM					
position	Express2	Bus22	normal	5	6	3.000000
7	8:19:56 PM					
position	Express2	Bus21	heavy	4	6	3.750000
36	8:20:03 PM					
position	Express2	Bus22	normal	6	6	3.000000
49	8:20:04 PM					
position	Express2	Bus23	normal	5	6	3.000000
25	8:20:11 PM					
position	Express2	Bus21	heavy	5	6	3.750000
49	8:20:12 PM					
position	Express2	Bus22	normal	1	6	3.000000
37	8:20:12 PM					
position	Express2	Bus23	normal	6	6	3.000000
100	8:20:19 PM					

position Express2 Bus22 light 2 6 2.250000 75 8:20:21 PM
position Express2 Bus21 heavy 6 6 3.750000 74 8:20:21 PM
position Express2 Bus23 light 1 6 2.250000 22 8:20:28 PM
position Express2 Bus22 normal 3 6 13.000000 88 8:20:28 PM
accident Express2 Bus22 3 8:20:33 PM
position Express2 Bus23 normal 2 6 3.000000 55 8:20:35 PM
position Express2 Bus22 normal 4 6 13.000000 33 8:20:52 PM
position Express2 Bus23 light 4 6 12.250000 3 8:20:52 PM
accident Express2 Bus22 4 8:20:57 PM
accident Express2 Bus23 4 8:20:58 PM
position Express2 Bus23 light 5 6 2.250000 68 8:21:15 PM
position Express2 Bus22 normal 5 6 3.000000 6 8:21:15 PM
position Express2 Bus23 light 6 6 2.250000 5 8:21:23 PM
position Express2 Bus22 normal 6 6 3.000000 47 8:21:23 PM

3. Passenger1

Getting on Bus12 at stop #1 at 8:25:04 PM, normal, 3 stops left
Arriving at stop #2 at 8:25:11 PM, normal, 2 stops left
Arriving at stop #3 at 8:25:34 PM, light, 1 stops left
Arriving at destination by Bus12 at 8:25:40 PM

4. Passenger2

Getting on Bus22 at stop #3 at 8:26:49 PM, heavy, 5 stops left
Arriving at stop #5 at 8:27:06 PM, normal, 4 stops left
Arriving at stop #6 at 8:27:14 PM, light, 3 stops left
Arriving at stop #1 at 8:27:22 PM, heavy, 2 stops left
Arriving at stop #2 at 8:27:31 PM, heavy, accident, 1 stops left
Arriving at destination by Bus22 at 8:27:55 PM

Source code

1. Labeled code (after diff from original code generated from ddigen tool)
   A. AccidentPublisher.java
   B. AccidentSubscriber.java
   C. PositionPublisher.java
   D. PositionSubscriber.java
   E. makefile

A. AccidentPublisher.java

```
70a71
>       Accident accident;    //the Accident message to be published
95c96
<               publisherMain(domainId, sampleCount);
---
>               //publisherMain(domainId, sampleCount);
106c107
<       private AccidentPublisher() {
---
>       public AccidentPublisher() {
110c111,114
<
---
>       public AccidentPublisher(Accident accident) {
>    super();
>    this.accident = accident;
>       }
113c117
<       private static void publisherMain(int domainId, int sampleCount) {
---
>       public void publisherMain(int domainId, int sampleCount) {
185c189
<                   Accident instance = new Accident();
---
>                   Accident instance = accident;
193c197
<                   final long sendPeriodMillis = 4 * 1000; // 4 seconds
---
>                   final long sendPeriodMillis = 2000;
198c202
<                       System.out.println("Writing Accident, count " + count);
---
>                       //System.out.println("Writing Accident, count " + count);
```

```
204c208
<                         writer.write(instance, instance_handle);
---
>                         //writer.write(instance, instance_handle);
210a215
>          writer.write(instance, instance_handle);
235c240
<
\ No newline at end of file
---
>
```

B. AccidentSubscriber.java

```
96c96
<               subscriberMain(domainId, sampleCount);
---
>               //subscriberMain(domainId, sampleCount);
107c107
<         private AccidentSubscriber() {
---
>         public AccidentSubscriber() {
114c114
<         private static void subscriberMain(int domainId, int sampleCount) {
---
>         public void subscriberMain(int domainId, int sampleCount) {
188,189d187
<                     final long receivePeriodSec = 4;
<
193,194c191,192
<                         System.out.println("Accident subscriber sleeping for "
<                                                 + receivePeriodSec + " sec...");
---
>                         //System.out.println("Accident subscriber sleeping for "
>                                                 //+ receivePeriodSec + " sec...");
196c194
<                               Thread.sleep(receivePeriodSec * 1000);    // in millisec
---
>                               Thread.sleep(2000);    // in millisec
248,249c246,249
<                                     System.out.println(
<
((Accident)_dataSeq.get(i)).toString("Received",0));
---
>               Accident instance = (Accident)_dataSeq.get(i);
>               System.out.format("%s%14s%9s%18d%53s\n",  "accident",  instance.route,
instance.vehicle, instance.stopNumber, instance.timestamp);
```

```
>                                       //System.out.println(
>
//((Accident)_dataSeq.get(i)).toString("Received",0));
264c264
<
\ No newline at end of file
---
>
```

C. PositionPublisher.java

```
70a71,72
>       String message;    //for test of passing Object into this
>       Position position; //the Position message to be published
95c97
<               publisherMain(domainId, sampleCount);
---
>               //this.publisherMain(domainId, sampleCount);
106c108
<       private PositionPublisher() {
---
>       public PositionPublisher() {
108a111,115
>
>       public PositionPublisher(String message){
>    super();
>    this.message = message;
>       }
109a117,119
>       public PositionPublisher(Position position){
>    this.position = position;
>       }
112,113c122,125
<
<       private static void publisherMain(int domainId, int sampleCount) {
---
>       public void test(){
>    System.out.println("Hello!");
>       }
>       public void publisherMain(int domainId, int sampleCount) {
185c197
<                       Position instance = new Position();
---
>                       Position instance = position;
193c205
<                       final long sendPeriodMillis = 4 * 1000; // 4 seconds
---
```

```
>                   final long sendPeriodMillis = 2000;//(long)(instance.timeBetweenStops
* 1000);
198c210
<                       System.out.println("Writing Position, count " + count);
---
>                       //System.out.println("Writing Position, count " + count);
201,202c213,214
<
<
---
>                                       //System.out.println("Position   stopnumber:   "   +
instance.stopNumber);
>                                       //System.out.println("Position   timeInterval:   "   +
instance.timeBetweenStops);
203a216,221
>           try {
>                   Thread.sleep(sendPeriodMillis);
>           } catch (InterruptedException ix) {
>                   System.err.println("Interrupted");
>                   break;
>           }
205c223
<                       try {
---
>                       /*try {
210c228
<                       }
---
>                       }*/
235c253
<
\ No newline at end of file
---
>
```

D. PositionSubscriber.java

```
96c96
<               subscriberMain(domainId, sampleCount);
---
>               //subscriberMain(domainId, sampleCount);
107c107
<       private PositionSubscriber() {
---
>       public PositionSubscriber() {
114c114
<       private static void subscriberMain(int domainId, int sampleCount) {
```

```
---
>       public void subscriberMain(int domainId, int sampleCount) {
168c168,170
<                   }
---
>                   }
>         //Create a content filter.
>         //ContentFilteredTopic cft = participant.create_contentfilteredtopic(topic.get_name() + " (filtered)", topic, "route MATCH 'Express1' and stopNumber = 2", null);
188c190
<                   final long receivePeriodSec = 4;
---
>                   final long receivePeriodSec = 2;
193,194c195,196
<                       System.out.println("Position subscriber sleeping for "
<                                               + receivePeriodSec + " sec...");
---
>                       //System.out.println("Position subscriber sleeping for "
>                                               //+ receivePeriodSec + " sec...");
248,249c250,253
<                                   System.out.println(
<                                       ((Position)_dataSeq.get(i)).toString("Received",0));
---
>               Position instance = (Position)_dataSeq.get(i);
>               System.out.format("%s%14s%9s%10s%8d%8d%18f%10d%17s\n", "position", instance.route, instance.vehicle, instance.trafficConditions, instance.stopNumber, instance.numStops, instance.timeBetweenStops, instance.fillInRatio, instance.timestamp);
>                                   //System.out.println(
>                                                                       //((Position)_dataSeq.get(i)).toString("Received",0));
264c268
<
\ No newline at end of file
---
>
```

E. makefile

```
25c25
<                                                   JAVA_SOURCES = ./Position.java ./PositionSeq.java ./PositionTypeSupport.java ./PositionTypeCode.java ./PositionDataReader.java ./PositionDataWriter.java ./PositionSubscriber.java ./PositionPublisher.java
---
>                                                   JAVA_SOURCES
```

= ./Position.java ./PositionSeq.java ./PositionTypeSupport.java ./PositionTypeCode.java ./PositionDataReader.java ./PositionDataWriter.java ./PositionSubscriber.java ./PositionPublisher.java ./TestPositionRun.java ./PubThread.java ./PubLauncher.java ./OperatorSubscriber.java ./PositionSubscriberFirst.java ./P.java ./PassengerSubscriber.java ./PositionSubscriberSecond.java ./InputTest.java ./InputParameter.java
47a48,64
> TestPositionRun: ./TestPositionRun.class
> $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" TestPositionRun $(ARGS)
>
> PubThread: ./PubThread.class
> $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" PubThread $(ARGS)
>
> PubLauncher: ./PubLauncher.class
> $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" PubLauncher $(ARGS)
>
> OperatorSubscriber: ./OperatorSubscriber.class
> $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" OperatorSubscriber $(ARGS)
>
> PassengerSubscriber: ./PassengerSubscriber.class
> $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" PassengerSubscriber $(ARGS)
>
> InputTest: ./InputTest.class
> $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" InputTest $(ARGS)

2. Complete code

A. Accident related

A1: AccidentPublisher.java
A2: AccidentSubscriber.java

B. Position related

B1: PositionPublisher.java
B2: PositionSubscriber.java
B3: PositionSubscriberFirst.java
B4: PositionSubscriberSecond.java

C. PubLauncher, PubThread and Passenger related

C1: PubLauncher.java
C2: PubThread.java
C3: PassengerSubscriber.java

D. Other help files

D1: InputParameter.java
D2: makefile_position_i86Linux2.6gcc4.1
D3: OperatorSubscriber.java
D4: P.java
D5: pub.properties

A. Accident related

A1: AccidentPublisher.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;

import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.publication.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;

//
============================================================
======================

public class AccidentPublisher {
	Accident accident;    //the Accident message to be published
	// ----------------------------------------------------------------------
	// Public Methods
	// ----------------------------------------------------------------------

	public static void main(String[] args) {
		// --- Get domain ID --- //
		int domainId = 0;
```

```
        if (args.length >= 1) {
            domainId = Integer.valueOf(args[0]).intValue();
        }

        // -- Get max loop count; 0 means infinite loop --- //
        int sampleCount = 0;
        if (args.length >= 2) {
            sampleCount = Integer.valueOf(args[1]).intValue();
        }

        /* Uncomment this to turn on additional logging
        Logger.get_instance().set_verbosity_by_category(
            LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
            LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
        */

        // --- Run --- //
        //publisherMain(domainId, sampleCount);
    }



    // --------------------------------------------------------------------
    // Private Methods
    // --------------------------------------------------------------------

    // --- Constructors: -----------------------------------------------

    public AccidentPublisher() {
        super();
    }

    public AccidentPublisher(Accident accident) {
   super();
   this.accident = accident;
    }
    // --------------------------------------------------------------------

    public void publisherMain(int domainId, int sampleCount) {

        DomainParticipant participant = null;
        Publisher publisher = null;
        Topic topic = null;
        AccidentDataWriter writer = null;

        try {
```

```
        // --- Create participant --- //

        /* To customize participant QoS, use
           the configuration file
           USER_QOS_PROFILES.xml */

        participant = DomainParticipantFactory.TheParticipantFactory.
              create_participant(
                    domainId,
DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
        if (participant == null) {
              System.err.println("create_participant error\n");
              return;
        }

        // --- Create publisher --- //

        /* To customize publisher QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        publisher = participant.create_publisher(
              DomainParticipant.PUBLISHER_QOS_DEFAULT, null /* listener
*/,
              StatusKind.STATUS_MASK_NONE);
        if (publisher == null) {
              System.err.println("create_publisher error\n");
              return;
        }


        // --- Create topic --- //

        /* Register type before creating topic */
        String typeName = AccidentTypeSupport.get_type_name();
        AccidentTypeSupport.register_type(participant, typeName);

        /* To customize topic QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        topic = participant.create_topic(
              "Example Accident",
              typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
              null /* listener */, StatusKind.STATUS_MASK_NONE);
        if (topic == null) {
              System.err.println("create_topic error\n");
```

```
        return;
}

// --- Create writer --- //

/* To customize data writer QoS, use
     the configuration file USER_QOS_PROFILES.xml */

writer = (AccidentDataWriter)
        publisher.create_datawriter(
                topic, Publisher.DATAWRITER_QOS_DEFAULT,
                null /* listener */, StatusKind.STATUS_MASK_NONE);
if (writer == null) {
        System.err.println("create_datawriter error\n");
        return;
}

// --- Write --- //

/* Create data sample for writing */
Accident instance = accident;

InstanceHandle_t instance_handle = InstanceHandle_t.HANDLE_NIL;
/* For a data type that has a key, if the same instance is going to be
     written multiple times, initialize the key here
     and register the keyed instance prior to writing */
//instance_handle = writer.register_instance(instance);

final long sendPeriodMillis = 2000;

for (int count = 0;
        (sampleCount == 0) || (count < sampleCount);
        ++count) {
        //System.out.println("Writing Accident, count " + count);

        /* Modify the instance to be written here */


        /* Write data */
        //writer.write(instance, instance_handle);
        try {
                Thread.sleep(sendPeriodMillis);
        } catch (InterruptedException ix) {
                System.err.println("INTERRUPTED");
                break;
        }
```

```
            writer.write(instance, instance_handle);
                }

                //writer.unregister_instance(instance, instance_handle);

            } finally {

                // --- Shutdown --- //

                if(participant != null) {
                    participant.delete_contained_entities();

                    DomainParticipantFactory.TheParticipantFactory.
                        delete_participant(participant);
                }
                /* RTI Data Distribution Service provides finalize_instance()
                    method for people who want to release memory used by the
                    participant factory singleton. Uncomment the following block of
                    code for clean destruction of the participant factory
                    singleton. */
                //DomainParticipantFactory.finalize_instance();
            }
        }
}
```

A2: AccidentSubscriber.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;

import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.subscription.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;

//
==========================================================
======================

public class AccidentSubscriber {
        // ---------------------------------------------------------------------
        // Public Methods
        // ---------------------------------------------------------------------

        public static void main(String[] args) {
```

```
        // --- Get domain ID --- //
        int domainId = 0;
        if (args.length >= 1) {
            domainId = Integer.valueOf(args[0]).intValue();
        }

        // -- Get max loop count; 0 means infinite loop --- //
        int sampleCount = 0;
        if (args.length >= 2) {
            sampleCount = Integer.valueOf(args[1]).intValue();
        }


        /* Uncomment this to turn on additional logging
        Logger.get_instance().set_verbosity_by_category(
            LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
            LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
        */

        // --- Run --- //
        //subscriberMain(domainId, sampleCount);
    }



    // -------------------------------------------------------------------------
    // Private Methods
    // -------------------------------------------------------------------------

    // --- Constructors: -----------------------------------------------------

    public AccidentSubscriber() {
        super();
    }


    // -------------------------------------------------------------------------

    public void subscriberMain(int domainId, int sampleCount) {

        DomainParticipant participant = null;
        Subscriber subscriber = null;
        Topic topic = null;
        DataReaderListener listener = null;
        AccidentDataReader reader = null;
```

```
        try {

            // --- Create participant --- //

            /* To customize participant QoS, use
                the configuration file
                USER_QOS_PROFILES.xml */

            participant = DomainParticipantFactory.TheParticipantFactory.
                create_participant(
                    domainId,
DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
            if (participant == null) {
                System.err.println("create_participant error\n");
                return;
            }

            // --- Create subscriber --- //

            /* To customize subscriber QoS, use
                the configuration file USER_QOS_PROFILES.xml */

            subscriber = participant.create_subscriber(
                DomainParticipant.SUBSCRIBER_QOS_DEFAULT, null /* listener
*/,
                StatusKind.STATUS_MASK_NONE);
            if (subscriber == null) {
                System.err.println("create_subscriber error\n");
                return;
            }

            // --- Create topic --- //

            /* Register type before creating topic */
            String typeName = AccidentTypeSupport.get_type_name();
            AccidentTypeSupport.register_type(participant, typeName);

            /* To customize topic QoS, use
                the configuration file USER_QOS_PROFILES.xml */

            topic = participant.create_topic(
                "Example Accident",
                typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
                null /* listener */, StatusKind.STATUS_MASK_NONE);
            if (topic == null) {
```

```
            System.err.println("create_topic error\n");
            return;
        }

        // --- Create reader --- //

        listener = new AccidentListener();

        /* To customize data reader QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        reader = (AccidentDataReader)
            subscriber.create_datareader(
                topic, Subscriber.DATAREADER_QOS_DEFAULT, listener,
                StatusKind.STATUS_MASK_ALL);
        if (reader == null) {
            System.err.println("create_datareader error\n");
            return;
        }

        // --- Wait for data --- //

        for (int count = 0;
              (sampleCount == 0) || (count < sampleCount);
              ++count) {
            //System.out.println("Accident subscriber sleeping for "
                                        //+ receivePeriodSec + " sec...");
            try {
                Thread.sleep(2000);    // in millisec
            } catch (InterruptedException ix) {
                System.err.println("INTERRUPTED");
                break;
            }
        }
    } finally {

        // --- Shutdown --- //

        if(participant != null) {
            participant.delete_contained_entities();

            DomainParticipantFactory.TheParticipantFactory.
                delete_participant(participant);
        }
        /* RTI Data Distribution Service provides the finalize_instance()
           method for users who want to release memory used by the
```

```
                participant factory singleton. Uncomment the following block of
                code for clean destruction of the participant factory
                singleton. */
            //DomainParticipantFactory.finalize_instance();
        }
    }

    // --------------------------------------------------------------------
    // Private Types
    // --------------------------------------------------------------------

    //
==========================================================
==================

    private static class AccidentListener extends DataReaderAdapter {

        AccidentSeq _dataSeq = new AccidentSeq();
        SampleInfoSeq _infoSeq = new SampleInfoSeq();

        public void on_data_available(DataReader reader) {
            AccidentDataReader AccidentReader =
                (AccidentDataReader)reader;

            try {
                AccidentReader.take(
                    _dataSeq, _infoSeq,
                    ResourceLimitsQosPolicy.LENGTH_UNLIMITED,
                    SampleStateKind.ANY_SAMPLE_STATE,
                    ViewStateKind.ANY_VIEW_STATE,
                    InstanceStateKind.ANY_INSTANCE_STATE);

                for(int i = 0; i < _dataSeq.size(); ++i) {
                    SampleInfo info = (SampleInfo)_infoSeq.get(i);

                    if (info.valid_data) {
            Accident instance = (Accident)_dataSeq.get(i);
            System.out.format("%s%14s%9s%18d%53s\n",  "accident",  instance.route,
instance.vehicle, instance.stopNumber, instance.timestamp);
                        //System.out.println(
                            //((Accident)_dataSeq.get(i)).toString("Received",0));


                    }
                }
            } catch (RETCODE_NO_DATA noData) {
```

```
                    // No data to process
                } finally {
                    AccidentReader.return_loan(_dataSeq, _infoSeq);
                }
            }
        }
}
```

B. Position related

B1: PositionPublisher.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;

import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.publication.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;

// ===========================================================================

public class PositionPublisher {
    String message;    //for test of passing Object into this
    Position position; //the Position message to be published
    // ---------------------------------------------------------------------
    // Public Methods
    // ---------------------------------------------------------------------

    public static void main(String[] args) {
        // --- Get domain ID --- //
        int domainId = 0;
        if (args.length >= 1) {
            domainId = Integer.valueOf(args[0]).intValue();
        }

        // -- Get max loop count; 0 means infinite loop --- //
        int sampleCount = 0;
        if (args.length >= 2) {
            sampleCount = Integer.valueOf(args[1]).intValue();
        }

        /* Uncomment this to turn on additional logging
        Logger.get_instance().set_verbosity_by_category(
```

```
            LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
            LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
        */

        // --- Run --- //
        //this.publisherMain(domainId, sampleCount);
    }



    // ----------------------------------------------------------------------
    // Private Methods
    // ----------------------------------------------------------------------

    // --- Constructors: -----------------------------------------------------

    public PositionPublisher() {
        super();
    }

    public PositionPublisher(String message){
    super();
    this.message = message;
    }

    public PositionPublisher(Position position){
    this.position = position;
    }

    // ----------------------------------------------------------------------
    public void test(){
    System.out.println("Hello!");
    }
    public void publisherMain(int domainId, int sampleCount) {

        DomainParticipant participant = null;
        Publisher publisher = null;
        Topic topic = null;
        PositionDataWriter writer = null;

        try {
            // --- Create participant --- //

            /* To customize participant QoS, use
                the configuration file
                USER_QOS_PROFILES.xml */
```

```
        participant = DomainParticipantFactory.TheParticipantFactory.
            create_participant(
                domainId,
DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                null /* listener */, StatusKind.STATUS_MASK_NONE);
        if (participant == null) {
            System.err.println("create_participant error\n");
            return;
        }

        // --- Create publisher --- //

        /* To customize publisher QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        publisher = participant.create_publisher(
            DomainParticipant.PUBLISHER_QOS_DEFAULT, null /* listener
*/,
            StatusKind.STATUS_MASK_NONE);
        if (publisher == null) {
            System.err.println("create_publisher error\n");
            return;
        }


        // --- Create topic --- //

        /* Register type before creating topic */
        String typeName = PositionTypeSupport.get_type_name();
        PositionTypeSupport.register_type(participant, typeName);

        /* To customize topic QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        topic = participant.create_topic(
            "Example Position",
            typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
            null /* listener */, StatusKind.STATUS_MASK_NONE);
        if (topic == null) {
            System.err.println("create_topic error\n");
            return;
        }

        // --- Create writer --- //
```

```
                /* To customize data writer QoS, use
                    the configuration file USER_QOS_PROFILES.xml */

                writer = (PositionDataWriter)
                      publisher.create_datawriter(
                            topic, Publisher.DATAWRITER_QOS_DEFAULT,
                            null /* listener */, StatusKind.STATUS_MASK_NONE);
                if (writer == null) {
                      System.err.println("create_datawriter error\n");
                      return;
                }

                // --- Write --- //

                /* Create data sample for writing */
                Position instance = position;

                InstanceHandle_t instance_handle = InstanceHandle_t.HANDLE_NIL;
                /* For a data type that has a key, if the same instance is going to be
                    written multiple times, initialize the key here
                    and register the keyed instance prior to writing */
                //instance_handle = writer.register_instance(instance);

                final long sendPeriodMillis = 2000;//(long)(instance.timeBetweenStops *
1000);

                for (int count = 0;
                       (sampleCount == 0) || (count < sampleCount);
                       ++count) {
                      //System.out.println("Writing Position, count " + count);

                      /* Modify the instance to be written here */
                      //System.out.println("Position          stopnumber:          "          +
instance.stopNumber);
                      //System.out.println("Position          timeInterval:          "          +
instance.timeBetweenStops);
                      /* Write data */
          try {
                Thread.sleep(sendPeriodMillis);
          } catch (InterruptedException ix) {
                System.err.println("Interrupted");
                break;
          }
                      writer.write(instance, instance_handle);
                      /*try {
                            Thread.sleep(sendPeriodMillis);
```

```
                } catch (InterruptedException ix) {
                    System.err.println("INTERRUPTED");
                    break;
                }*/
            }

            //writer.unregister_instance(instance, instance_handle);

        } finally {

            // --- Shutdown --- //

            if(participant != null) {
                participant.delete_contained_entities();

                DomainParticipantFactory.TheParticipantFactory.
                    delete_participant(participant);
            }
            /* RTI Data Distribution Service provides finalize_instance()
                method for people who want to release memory used by the
                participant factory singleton. Uncomment the following block of
                code for clean destruction of the participant factory
                singleton. */
            //DomainParticipantFactory.finalize_instance();
        }
    }
}
```

B2: PositionSubscriber.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;

import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.subscription.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;

// ===========================================================================

public class PositionSubscriber {
    // -----------------------------------------------------------------
    // Public Methods
```

```
// ---------------------------------------------------------------------

public static void main(String[] args) {
    // --- Get domain ID --- //
    int domainId = 0;
    if (args.length >= 1) {
        domainId = Integer.valueOf(args[0]).intValue();
    }

    // -- Get max loop count; 0 means infinite loop --- //
    int sampleCount = 0;
    if (args.length >= 2) {
        sampleCount = Integer.valueOf(args[1]).intValue();
    }


    /* Uncomment this to turn on additional logging
    Logger.get_instance().set_verbosity_by_category(
        LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
        LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    // --- Run --- //
    //subscriberMain(domainId, sampleCount);
}



// ---------------------------------------------------------------------
// Private Methods
// ---------------------------------------------------------------------

// --- Constructors: -----------------------------------------------------

public PositionSubscriber() {
    super();
}


// ---------------------------------------------------------------------

public void subscriberMain(int domainId, int sampleCount) {

    DomainParticipant participant = null;
    Subscriber subscriber = null;
    Topic topic = null;
```

```
        DataReaderListener listener = null;
        PositionDataReader reader = null;

        try {

            // --- Create participant --- //

            /* To customize participant QoS, use
                the configuration file
                USER_QOS_PROFILES.xml */

            participant = DomainParticipantFactory.TheParticipantFactory.
                create_participant(
                    domainId,
DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
            if (participant == null) {
                System.err.println("create_participant error\n");
                return;
            }

            // --- Create subscriber --- //

            /* To customize subscriber QoS, use
                the configuration file USER_QOS_PROFILES.xml */

            subscriber = participant.create_subscriber(
                DomainParticipant.SUBSCRIBER_QOS_DEFAULT, null /* listener
*/,
                StatusKind.STATUS_MASK_NONE);
            if (subscriber == null) {
                System.err.println("create_subscriber error\n");
                return;
            }

            // --- Create topic --- //

            /* Register type before creating topic */
            String typeName = PositionTypeSupport.get_type_name();
            PositionTypeSupport.register_type(participant, typeName);

            /* To customize topic QoS, use
                the configuration file USER_QOS_PROFILES.xml */

            topic = participant.create_topic(
                "Example Position",
```

```
                typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
                null /* listener */, StatusKind.STATUS_MASK_NONE);
        if (topic == null) {
            System.err.println("create_topic error\n");
            return;
        }
    //Create a content filter.
    //ContentFilteredTopic cft = participant.create_contentfilteredtopic(topic.get_name() + " (filtered)", topic, "route MATCH 'Express1' and stopNumber = 2", null);

        // --- Create reader --- //

        listener = new PositionListener();

        /* To customize data reader QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        reader = (PositionDataReader)
            subscriber.create_datareader(
                topic, Subscriber.DATAREADER_QOS_DEFAULT, listener,
                StatusKind.STATUS_MASK_ALL);
        if (reader == null) {
            System.err.println("create_datareader error\n");
            return;
        }

        // --- Wait for data --- //

        final long receivePeriodSec = 2;

        for (int count = 0;
              (sampleCount == 0) || (count < sampleCount);
              ++count) {
            //System.out.println("Position subscriber sleeping for "
                                //+ receivePeriodSec + " sec...");
            try {
                Thread.sleep(receivePeriodSec * 1000);    // in millisec
            } catch (InterruptedException ix) {
                System.err.println("INTERRUPTED");
                break;
            }
        }
    } finally {

        // --- Shutdown --- //
```

```
            if(participant != null) {
                participant.delete_contained_entities();

                DomainParticipantFactory.TheParticipantFactory.
                    delete_participant(participant);
            }
            /* RTI Data Distribution Service provides the finalize_instance()
               method for users who want to release memory used by the
               participant factory singleton. Uncomment the following block of
               code for clean destruction of the participant factory
               singleton. */
            //DomainParticipantFactory.finalize_instance();
        }
    }

    // --------------------------------------------------------------------
    // Private Types
    // --------------------------------------------------------------------

    //
==========================================================
==================

    private static class PositionListener extends DataReaderAdapter {

        PositionSeq _dataSeq = new PositionSeq();
        SampleInfoSeq _infoSeq = new SampleInfoSeq();

        public void on_data_available(DataReader reader) {
            PositionDataReader PositionReader =
                (PositionDataReader)reader;

            try {
                PositionReader.take(
                    _dataSeq, _infoSeq,
                    ResourceLimitsQosPolicy.LENGTH_UNLIMITED,
                    SampleStateKind.ANY_SAMPLE_STATE,
                    ViewStateKind.ANY_VIEW_STATE,
                    InstanceStateKind.ANY_INSTANCE_STATE);

                for(int i = 0; i < _dataSeq.size(); ++i) {
                    SampleInfo info = (SampleInfo)_infoSeq.get(i);

                    if (info.valid_data) {
                Position instance = (Position)_dataSeq.get(i);
```

```
                System.out.format("%s%14s%9s%10s%8d%8d%18f%10d%17s\n",
"position", instance.route, instance.vehicle, instance.trafficConditions,
instance.stopNumber, instance.numStops, instance.timeBetweenStops, instance.fillInRatio,
instance.timestamp);
                                    //System.out.println(
                                        // ((Position)_dataSeq.get(i)).toString("Received",0));


                                }
                            }
                        } catch (RETCODE_NO_DATA noData) {
                            // No data to process
                        } finally {
                            PositionReader.return_loan(_dataSeq, _infoSeq);
                        }
                    }
                }
            }
```

B3: PositionSubscriberFirst.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;

import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.subscription.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;

//
==========================================================
======================

public class PositionSubscriberFirst {
    public static P po;
    public static boolean flag = false;
    public static String timestamp;
    public static String route;
    public static String vehicle;
    public static int stopNumber;
    public static int numStops;
    public static float timeBetweenStops;
    public static String trafficConditions;
    public static int fillInRatio;
    public static int remainingStops;
```

```
public static int routeNumber;    //the route, Express 1 or 2
public static int sourceStop;    //the source stop
public static int destinationStop;    //the destination stop

public static boolean getOn = false;
// ----------------------------------------------------------------------
// Public Methods
// ----------------------------------------------------------------------

public static void main(String[] args) {
        // --- Get domain ID --- //
        int domainId = 0;
        if (args.length >= 1) {
                domainId = Integer.valueOf(args[0]).intValue();
        }

        // -- Get max loop count; 0 means infinite loop --- //
        int sampleCount = 0;
        if (args.length >= 2) {
                sampleCount = Integer.valueOf(args[1]).intValue();
        }


        /* Uncomment this to turn on additional logging
        Logger.get_instance().set_verbosity_by_category(
                LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
                LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
        */

        // --- Run --- //
        //subscriberMain(domainId, sampleCount);
}



// ----------------------------------------------------------------------
// Private Methods
// ----------------------------------------------------------------------

// --- Constructors: -----------------------------------------------------

public PositionSubscriberFirst() {
        super();
}
public PositionSubscriberFirst(P po) {
```

```
super();
this.po = po;
 }
 public PositionSubscriberFirst(P po, InputParameter ip) {
super();
this.po = po;
routeNumber = ip.routeNumber;
sourceStop = ip.sourceStop;
destinationStop = ip.destinationStop;
 }

 // ---------------------------------------------------------------------

 public void subscriberMain(int domainId, int sampleCount) {

        DomainParticipant participant = null;
        Subscriber subscriber = null;
        Topic topic = null;
        DataReaderListener listener = null;
        PositionDataReader reader = null;

        try {

            // --- Create participant --- //

            /* To customize participant QoS, use
                the configuration file
                USER_QOS_PROFILES.xml */

            participant = DomainParticipantFactory.TheParticipantFactory.
                  create_participant(
                        domainId,
DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                        null /* listener */, StatusKind.STATUS_MASK_NONE);
            if (participant == null) {
                  System.err.println("create_participant error\n");
                  return;
            }

            // --- Create subscriber --- //

            /* To customize subscriber QoS, use
                the configuration file USER_QOS_PROFILES.xml */

            subscriber = participant.create_subscriber(
                  DomainParticipant.SUBSCRIBER_QOS_DEFAULT,  null  /*  listener
```

```
*/,
                    StatusKind.STATUS_MASK_NONE);
                if (subscriber == null) {
                    System.err.println("create_subscriber error\n");
                    return;
                }

                // --- Create topic --- //

                /* Register type before creating topic */
                String typeName = PositionTypeSupport.get_type_name();
                PositionTypeSupport.register_type(participant, typeName);

                /* To customize topic QoS, use
                   the configuration file USER_QOS_PROFILES.xml */

                topic = participant.create_topic(
                    "Example Position",
                    typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
                if (topic == null) {
                    System.err.println("create_topic error\n");
                    return;
                }
            //Create a content filter.
            String filterExpression = "route MATCH 'Express" + routeNumber + "' and
stopNumber = " + sourceStop;
            ContentFilteredTopic cft =
participant.create_contentfilteredtopic(topic.get_name() + " (filtered)", topic,
filterExpression, null);

                // --- Create reader --- //

                listener = new PositionListener();

                /* To customize data reader QoS, use
                   the configuration file USER_QOS_PROFILES.xml */

                reader = (PositionDataReader)
                    subscriber.create_datareader(
                        cft, Subscriber.DATAREADER_QOS_DEFAULT, listener,
                        StatusKind.STATUS_MASK_ALL);
                if (reader == null) {
                    System.err.println("create_datareader error\n");
                    return;
                }
```

```
            // --- Wait for data --- //

            for (int count = 0;
                    (sampleCount == 0) || (count < sampleCount);
                    ++count) {
                //System.out.println("Position subscriber sleeping for "
                                            //+ receivePeriodSec + " sec...");
                try {
                        Thread.sleep(2000);    // in millisec
                po.timestamp = timestamp;
                po.route = route;
                po.vehicle = vehicle;
                po.stopNumber = stopNumber;
                po.numStops = numStops;
                po.timeBetweenStops = timeBetweenStops;
                po.trafficConditions = trafficConditions;
                po.fillInRatio = fillInRatio;
                po.remainingStops = remainingStops;
                //this.wait();
                if(flag)
                    Thread.currentThread().interrupt();
                //Thread.sleep(receivePeriodSec * 1000);
                    } catch (InterruptedException ix) {
                        //System.err.println("INTERRUPTED");
                        return;
                    }
                }
            } finally {

                // --- Shutdown --- //

                if(participant != null) {
                    participant.delete_contained_entities();

                    DomainParticipantFactory.TheParticipantFactory.
                        delete_participant(participant);
                }
                /* RTI Data Distribution Service provides the finalize_instance()
                    method for users who want to release memory used by the
                    participant factory singleton. Uncomment the following block of
                    code for clean destruction of the participant factory
                    singleton. */
                //DomainParticipantFactory.finalize_instance();
            }
        }
```

```
    // ----------------------------------------------------------------
    // Private Types
    // ----------------------------------------------------------------

    //
================================================================
==================

    private static class PositionListener extends DataReaderAdapter {

        PositionSeq _dataSeq = new PositionSeq();
        SampleInfoSeq _infoSeq = new SampleInfoSeq();



        public void on_data_available(DataReader reader) {
            PositionDataReader PositionReader =
                (PositionDataReader)reader;

            try {
                PositionReader.take(
                    _dataSeq, _infoSeq,
                    ResourceLimitsQosPolicy.LENGTH_UNLIMITED,
                    SampleStateKind.ANY_SAMPLE_STATE,
                    ViewStateKind.ANY_VIEW_STATE,
                    InstanceStateKind.ANY_INSTANCE_STATE);

                for(int i = 0; i < _dataSeq.size(); ++i) {
                    SampleInfo info = (SampleInfo)_infoSeq.get(i);

                    if (info.valid_data) {
                 if(!getOn) {
                Position instance = (Position)_dataSeq.get(i);
                String accidentOrNot = "";

                timestamp = instance.timestamp;
                route = instance.route;
                vehicle = instance.vehicle;
                stopNumber = instance.stopNumber;
                numStops = instance.numStops;
                timeBetweenStops = instance.timeBetweenStops;
                trafficConditions = instance.trafficConditions;
                fillInRatio = instance.fillInRatio;
                //if accident happens, add string into printout
                if(timeBetweenStops > 12)
```

```
                    accidentOrNot = "accident, ";
                //calculate the number of remaining stops next
                if(destinationStop < sourceStop)
                    remainingStops = destinationStop + numStops - sourceStop;
                else
                    remainingStops = destinationStop - sourceStop;
                System.out.println("Getting on " + vehicle + " at stop #" + stopNumber
+ " at " + timestamp + ", " + trafficConditions + ", " + accidentOrNot + remainingStops
+ " stops left");
                getOn = true;
                //System.out.println(instance.vehicle + " of route " + instance.route + "
stops in " + instance.stopNumber + " at " + instance.timestamp + " with traffic " +
instance.trafficConditions + " and stopinterval " + instance.timeBetweenStops);

                flag = true;
                            //this.interrupt();
                //System.out.println(
                            // ((Position)_dataSeq.get(i)).toString("Received",0));

            }
                    }
                }
              } catch (RETCODE_NO_DATA noData) {
                    // No data to process
              } finally {
                    PositionReader.return_loan(_dataSeq, _infoSeq);
              }
        }
    }
}
```

B4: PositionSubscriberSecond.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;

import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.subscription.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;

//
=============================================================
======================
```

```
public class PositionSubscriberSecond {
	P po = new P();
	public static boolean flag = false;
	public static int counter;
	// ---------------------------------------------------------------------
	// Public Methods
	// ---------------------------------------------------------------------

	public static void main(String[] args) {
		// --- Get domain ID --- //
		int domainId = 0;
		if (args.length >= 1) {
			domainId = Integer.valueOf(args[0]).intValue();
		}

		// -- Get max loop count; 0 means infinite loop --- //
		int sampleCount = 0;
		if (args.length >= 2) {
			sampleCount = Integer.valueOf(args[1]).intValue();
		}


		/* Uncomment this to turn on additional logging
		Logger.get_instance().set_verbosity_by_category(
			LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
			LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
		*/

		// --- Run --- //
		//subscriberMain(domainId, sampleCount);
	}



	// ---------------------------------------------------------------------
	// Private Methods
	// ---------------------------------------------------------------------

	// --- Constructors: -----------------------------------------------------

	public PositionSubscriberSecond() {
		super();
	}
	public PositionSubscriberSecond(P po) {
	super();
	this.po = po;
```

```
    }

    // ----------------------------------------------------------------------

    public void subscriberMain(int domainId, int sampleCount) {

        DomainParticipant participant = null;
        Subscriber subscriber = null;
        Topic topic = null;
        DataReaderListener listener = null;
        PositionDataReader reader = null;

        try {

            // --- Create participant --- //

            /* To customize participant QoS, use
               the configuration file
               USER_QOS_PROFILES.xml */

            participant = DomainParticipantFactory.TheParticipantFactory.
                create_participant(
                    domainId,
DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
            if (participant == null) {
                System.err.println("create_participant error\n");
                return;
            }

            // --- Create subscriber --- //

            /* To customize subscriber QoS, use
               the configuration file USER_QOS_PROFILES.xml */

            subscriber = participant.create_subscriber(
                DomainParticipant.SUBSCRIBER_QOS_DEFAULT, null /* listener
*/,
                StatusKind.STATUS_MASK_NONE);
            if (subscriber == null) {
                System.err.println("create_subscriber error\n");
                return;
            }

            // --- Create topic --- //
```

```
        /* Register type before creating topic */
        String typeName = PositionTypeSupport.get_type_name();
        PositionTypeSupport.register_type(participant, typeName);

        /* To customize topic QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        topic = participant.create_topic(
            "Example Position",
            typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
            null /* listener */, StatusKind.STATUS_MASK_NONE);
        if (topic == null) {
            System.err.println("create_topic error\n");
            return;
        }
    //Create a content filter.
    /*String cft_param_list[] = {po.vehicle};
    StringSeq cft_parameters = new
StringSeq(java.util.Arrays.asList(cft_param_list));
    System.out.println(cft_parameters);*/
    //cft_parameters.set(0,po.vehicle);
    counter = po.remainingStops;
    //System.out.println("Now counter is: " + counter);
    String filtered = "vehicle MATCH '" + po.vehicle +"'";
    ContentFilteredTopic cft =
participant.create_contentfilteredtopic(topic.get_name() + " (filtered)", topic, filtered,
null);
    if(cft == null) {
    System.err.println("create cft error!\n");
    return;
    }

        // --- Create reader --- //

        listener = new PositionListener();

        /* To customize data reader QoS, use
           the configuration file USER_QOS_PROFILES.xml */

        reader = (PositionDataReader)
            subscriber.create_datareader(
                cft, Subscriber.DATAREADER_QOS_DEFAULT, listener,
                StatusKind.STATUS_MASK_ALL);
        if (reader == null) {
            System.err.println("create_datareader error\n");
            return;
```

```
            }

            // --- Wait for data --- //



            for (int count = 0;
                  (sampleCount == 0) || (count < sampleCount);
                  ++count) {
                //System.out.println("Position subscriber sleeping for "
                                          //+ receivePeriodSec + " sec...");
                try {
                      Thread.sleep(2000);    // in millisec
               if(flag)
              Thread.currentThread().interrupt();
                } catch (InterruptedException ix) {
                      //System.err.println("INTERRUPTED");
                      return;
                }
            }
        } finally {

            // --- Shutdown --- //

            if(participant != null) {
                participant.delete_contained_entities();

                DomainParticipantFactory.TheParticipantFactory.
                      delete_participant(participant);
            }
            /* RTI Data Distribution Service provides the finalize_instance()
                method for users who want to release memory used by the
                participant factory singleton. Uncomment the following block of
                code for clean destruction of the participant factory
                singleton. */
            //DomainParticipantFactory.finalize_instance();
        }
    }

    // --------------------------------------------------------------------
    // Private Types
    // --------------------------------------------------------------------

    //
==========================================================
==================
```

```
    private static class PositionListener extends DataReaderAdapter {

        PositionSeq _dataSeq = new PositionSeq();
        SampleInfoSeq _infoSeq = new SampleInfoSeq();

        public void on_data_available(DataReader reader) {
            PositionDataReader PositionReader =
                (PositionDataReader)reader;

            try {
                PositionReader.take(
                    _dataSeq, _infoSeq,
                    ResourceLimitsQosPolicy.LENGTH_UNLIMITED,
                    SampleStateKind.ANY_SAMPLE_STATE,
                    ViewStateKind.ANY_VIEW_STATE,
                    InstanceStateKind.ANY_INSTANCE_STATE);

                for(int i = 0; i < _dataSeq.size(); ++i) {
                    SampleInfo info = (SampleInfo)_infoSeq.get(i);

                    if (info.valid_data) {
                Position instance = (Position) _dataSeq.get(i);
            if(counter > 1) {
                counter --;
                        //Position instance = (Position)_dataSeq.get(i);
                String accidentOrNot = "";
                if(instance.timeBetweenStops > 12)
                accidentOrNot = "accident, ";
                System.out.println("Arriving at stop #" + instance.stopNumber + " at "
+ instance.timestamp + ", " + instance.trafficConditions + ", " + accidentOrNot +
counter + " stops left");
                //System.out.println(instance.vehicle + " of route " + instance.route + "
stops in " + instance.stopNumber + " at " + instance.timestamp + " with traffic " +
instance.trafficConditions + " and stopinterval " + instance.timeBetweenStops);
                //counter --;
            }
            else {
                System.out.println("Arriving at destination by " + instance.vehicle + "
at " + instance.timestamp);
                flag = true;
            }
                        //System.out.println(
                            // ((Position)_dataSeq.get(i)).toString("Received",0));
```

```
                    }
                }
            } catch (RETCODE_NO_DATA noData) {
                // No data to process
            } finally {
                PositionReader.return_loan(_dataSeq, _infoSeq);
            }
        }
    }
}
```

C. PubLauncher, PubThread and Passenger related

C1: PubLauncher.java

```
/*
 * The PubLauncher reads all of its initialization parameters from a properties
 *   file called pubsub.properties and starts a thread for each vehicle on each
 *   route. A thread (PubThread) publishes all the messages, alerts (accidents)
 *   and positions for this vehicle at each bus stop along the route.
 */
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Properties;

public class PubLauncher {

    Properties props = new Properties();
    int numRoutes;                    //number of routes
    int numVehicles;                  //number of vehicles for each route

    String route1;                    //name of route1
    String route2;                    //name of route2

    int route1numStops;               //number of bus stops on route1
    int route1TimeBetweenStops;       //base time interval between stops on route1
    String route1Vehicle1;            //name of Bus#1 on route1
    String route1Vehicle2;            //name of Bus#2 on route1
    String route1Vehicle3;            //name of Bus#3 on route1

    int route2numStops;               //number of bus stops on route2
    int route2TimeBetweenStops;       //base time interval between stops on route2
    String route2Vehicle1;            //name of Bus#1 on route2
    String route2Vehicle2;            //name of Bus#2 on route2
    String route2Vehicle3;            //name of Bus#3 on route2
```

```
    /*
      * Step 1: parse pub.properties file and read all info inside for PubThread.
      * Notice difference between Windows and Linux directory for file path.
      * Method parsePubProperties is used for parsing.
      */
    /*
      * Step 1 start----------------------------------------------------
      */
    private void parsePubProperties(){
    //parse pub.properties and read corresponding values
        try{
            props.load(new FileInputStream("pub.properties"));

            numRoutes = Integer.parseInt(props.getProperty("numRoutes"));
            numVehicles= Integer.parseInt(props.getProperty("numVehicles"));
            //printout("No of Routes is: ",numRoutes);
            //printout("No of Buses is: ",numVehicles);

            route1 = props.getProperty("route1");
            route2 = props.getProperty("route2");
            //printout("Name of Route1 is: ",route1);
            //printout("Name of Route2 is: ",route2);

            route1numStops = Integer.parseInt(props.getProperty("route1numStops"));
            route1TimeBetweenStops =
Integer.parseInt(props.getProperty("route1TimeBetweenStops"));
            route1Vehicle1 = props.getProperty("route1Vehicle1");
            route1Vehicle2 = props.getProperty("route1Vehicle2");
            route1Vehicle3 = props.getProperty("route1Vehicle3");
            //printout("No of Stops in Route1 is: ",route1numStops);
            //printout("Interval stops for Route1(seconds) is:
",route1TimeBetweenStops);
            //printout("Name of Bus1 in route1 is: ",route1Vehicle1);
            //printout("Name of Bus2 in route1 is: ",route1Vehicle2);
            //printout("Name of Bus3 in route1 is: ",route1Vehicle3);

            route2numStops = Integer.parseInt(props.getProperty("route2numStops"));
            route2TimeBetweenStops =
Integer.parseInt(props.getProperty("route2TimeBetweenStops"));
            route2Vehicle1 = props.getProperty("route2Vehicle1");
            route2Vehicle2 = props.getProperty("route2Vehicle2");
            route2Vehicle3 = props.getProperty("route2Vehicle3");
            //printout("No of Stops in Route2 is: ",route2numStops);
            //printout("Interval stops for Route2(seconds) is:
",route2TimeBetweenStops);
            //printout("Name of Bus1 in route2 is: ",route2Vehicle1);
```

```
            //printout("Name of Bus2 in route2 is: ",route2Vehicle2);
            //printout("Name of Bus3 in route2 is: ",route2Vehicle3);

        }//catch exception in case properties file does not exist
        catch(IOException e){
            e.printStackTrace();
        }
}

/*private static void printout(String message, Object o){
//test print out method for parsePubProperties
      System.out.format("%s %s%n", message, o.toString());
}*/
/*
  * Step 1 end----------------------------------------------------
  */
/*
  * Step 2: start() runs all PubThreads according to parsed parameters
  */
/*
  * Step 2 start----------------------------------------------------
  */

private void start() throws InterruptedException{

      String[][] busNames = new String[numRoutes][numVehicles];
      String[] routeGroup = {route1, route2};
      int[] stopGroup = {route1numStops, route2numStops};
      int[] intervalGroup = {route1TimeBetweenStops, route2TimeBetweenStops};
      busNames[0][0] = route1Vehicle1;
      busNames[0][1] = route1Vehicle2;
      busNames[0][2] = route1Vehicle3;
      busNames[1][0] = route2Vehicle1;
      busNames[1][1] = route2Vehicle2;
      busNames[1][2] = route2Vehicle3;
      ArrayList<Thread> busGroup = new ArrayList<Thread>();
      for(int i = 0; i < numRoutes; i ++)
            for(int j = 0; j < numVehicles; j ++)
                  busGroup.add(new Thread(new PubThread(routeGroup[i],
                              stopGroup[i], intervalGroup[i], busNames[i][j])));
      System.out.println("Start the PubLauncher!");
      Iterator<Thread> it = busGroup.iterator();
      int threadOrder = 0;
      while(it.hasNext()) {
            System.out.println("Thread #" + threadOrder + " started.");
            threadOrder ++;
```

```
            it.next().start();
        }
        System.out.println("All buses have started. Waiting for them to terminate...");
        busGroup.get(0).join();
        busGroup.get(1).join();
        busGroup.get(2).join();
        busGroup.get(3).join();
        busGroup.get(4).join();
        busGroup.get(5).join();
        System.out.println("Finally all done!");
    }

    /*
     * Step 2 end----------------------------------------------------
     */

    public static void main(String[] args) throws InterruptedException {
        PubLauncher pl = new PubLauncher();
        pl.parsePubProperties();
        pl.start();
    }

}
```

C2: PubThread.java

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Random;

/*
 * This is the Bus class with implemented thread interface.
 * Each PubThread represents a vehicle on a route. It receives the following
 * information before starting:
 *      The route and the vehicle it represents.
 *      The number of stops along the route.
 *      The time spent by the vehicle between two stops.
 * Once all PubThreads are created, they are started by the PubLauncher. At
 * each stop, the thread publishes a position message and an accident message
 * depending on the situation.
 */

public class PubThread implements Runnable{

    private final static int looptimes = 3;            //the running loop times for a bus
    private final static int accidentcost = 10;      //the seconds for a bus to cost in case of
an accident
```

```
    String route;                          //route's name on which the bus runs
    int numOfStops;                        //number of stops the bus should go
    int baseInterval;                      //the time interval between stops for a
bus to run
    String busName;                        //the unique identifier for the bus
    String traffic;                        //the traffic condition in each stop
    int fillInRatio;                       //the number of passengers in the bus
    boolean accidentFlag;                  //the accident indicator
    //Position position = new Position();//the position message
    //PositionPublisher pp;                //the position publisher

    public PubThread(){
    }

    public PubThread(String route, int numOfStops, int interval, String busName){
    //constructor with parsed route info from PubLauncher
        this.route = route;
        this.numOfStops = numOfStops;
        this.baseInterval = interval;
        this.busName = busName;
    }

    /*
     * Step 1: three random number generators for traffic, fillinratio and accident
     * And one interval calculator
     */
    /*
     * Step 1 start----------------------------------------------------
     */
    private float intervalCal(int baseInterval, String traffic, boolean accidentFlag){
        float interval = baseInterval;

        if(traffic.equals("heavy"))
            interval *= 1.25;
        else if(traffic.equals("light"))
            interval *= 0.75;
        if(accidentFlag)
            interval += accidentcost;

        return interval;
    }
    private String trafficGenerator(){
    //three chances: normal with 50%, heavy and light with each 25%
    //so use r.nextInt(4) we have 25% for 0,1,2,3. 0->heavy, 1->light
    //and 2,3 together ->normal.
```

```
        String condition;
        Random r = new Random();
        int ran = r.nextInt(4);
        if(ran == 0)
            condition = "heavy";
        else if(ran == 1)
            condition = "light";
        else
            condition = "normal";
        return condition;
    }

    private int fillInRatioGenerator(){
    //return a number of passengers in the bus, within [1, 100].
        Random r = new Random();
        return r.nextInt(100) + 1;
    }

    private boolean accidentGenerator(){
    //return the Boolean value of accident occurence. from [0,9] there is 10
    //percent chance to encounter an accident.
        Random r = new Random();
        if(r.nextInt(10) == 9) //9 or any single digit in [0,9]
            return true;
        else
            return false;
    }

    /*
     * Step 1 end----------------------------------------------------
     */
    /*
     * Step 2: the core run() method in which each bus will run 3 rounds.
     * Its interval between buses is dynamically determined by both traffic
     * condition and accident occurrence. Each interval passes in a sleep mode.
     * Before the interval (bus stop) publish current message, including bus
     * name, stop #, route name, timestamp, traffic, accident (if any), passenger
     * fill-in-ratio.
     * An assistant method printout to print messages out for test.
     */
    /*
     * Step 2 start----------------------------------------------------
     */
    public void run() {
        //here put all the messages to be output by the bus thread in place
        for(int i = 0; i < looptimes; i++){
```

```
int currentStop = 1;    //the beginning stop #
while (currentStop <= numOfStops){
    //printout all messages here
    traffic = trafficGenerator();
    accidentFlag = accidentGenerator();
    //get the interval here from accident and traffic conditions
    float interval = intervalCal(baseInterval,traffic,accidentFlag);       //the
actual interval
    //fill out position message
    Position position = new Position();
    position.timestamp      =      DateFormat.getTimeInstance().format(new
Date());
    position.route = route;
    position.vehicle = busName;
    position.stopNumber = currentStop;
    position.numStops = numOfStops;
    position.timeBetweenStops = interval;
    position.trafficConditions = traffic;
    position.fillInRatio = fillInRatioGenerator();
    System.out.println(position.vehicle  +  "  has  published  a  position
message at stop #" + position.stopNumber + " on route " + position.route + " at " +
position.timestamp);

    //position message ready!
    new PositionPublisher(position).publisherMain(0,1);
    //PositionPublisher ready
    //fill out accident message if accident flag is set
    if(accidentFlag) {
        Accident accident = new Accident();
        accident.timestamp   =   DateFormat.getTimeInstance().format(new
Date());
        accident.route = route;
        accident.vehicle = busName;
        accident.stopNumber = currentStop;
        System.out.println(accident.vehicle + " has published an accident
message at stop #" + accident.stopNumber + " on route " + accident.route + " at " +
accident.timestamp);
        new AccidentPublisher(accident).publisherMain(0,1);
    }
    //printout(busName, currentStop, route, traffic, fillInRatioGenerator(),
accidentFlag, interval);
    try {
        long sleepInterval = (long)(interval * 1000);
        Thread.sleep(sleepInterval);
    } catch (InterruptedException e) {
        e.printStackTrace();
```

```
                    }
                    currentStop ++;
                }
            }

        }

    private static void printout(String busName, int currentStop, String route,
            String traffic, int passenger, boolean accident, double interval){
        String accidentInd = "No";
        if(accident)
            accidentInd = "Yes";
        System.out.format("%s at stop: %d on route: %s at time: %s with traffic: %s " +
                "& %d persons by %s accident & interval: %f.%n", busName,
currentStop,
                route, DateFormat.getTimeInstance().format(new Date()), traffic,
                passenger, accidentInd, interval);
    }
    /*
     * Step 2 end----------------------------------------------------
     */

    public static void main(String[] args) {
        //System.out.println(new PubThread().trafficGenerator());
        //System.out.println(intervalCal(2, "heavy", false));
        new Thread(new PubThread("Express1", 4, 2, "Bus11")).start();
    }

}
```

C3: PassengerSubscriber.java

```
public class PassengerSubscriber implements Runnable {
    P po = new P();
    InputParameter ip = new InputParameter();
    public static String route;
    public static String source;
    public static String destination;
    /*po.routeNumber = Integer.parseInt(args[0]);
    po.sourceStop = Integer.parseInt(args[1]);
    po.destinationStop = Integer.parseInt(args[2]);*/
    public PassengerSubscriber() {
    }

    public void run() {
    ip.routeNumber = Integer.parseInt(route);
    ip.sourceStop = Integer.parseInt(source);
```

```
        ip.destinationStop = Integer.parseInt(destination);
        //System.out.println("Hi: " + ip.routeNumber);
        new PositionSubscriberFirst(po, ip).subscriberMain(0,0);
        //System.out.println("Current busname: " + po.vehicle);
        new PositionSubscriberSecond(po).subscriberMain(0,0);
         }

         public static void main(String[] args) {
        if(args.length > 0) {
              route = args[0];
              source = args[1];
              destination = args[2];
        }
        new Thread(new PassengerSubscriber()).start();
         }
}
```

D. Other help files

D1: InputParameter.java

```
public class InputParameter {
       public int routeNumber;
       public int sourceStop;
       public int destinationStop;

       public InputParameter() {
       }
       public InputParameter(int routeNumber, int sourceStop, int destinationStop) {
       this.routeNumber = routeNumber;
       this.sourceStop = sourceStop;
       this.destinationStop = destinationStop;
       }
}
```

D2: makefile_position_i86Linux2.6gcc4.1

```
############################################################################
# makefile_position_i86Linux2.6gcc4.1.1jdk
#
# (c) Copyright, Real-Time Innovations, 2010.  All rights reserved.
# No duplications, whole or partial, manual or electronic, may be made
# without express written permission.  Any such copies, or
# revisions thereof, must display this notice unaltered.
# This code contains trade secrets of Real-Time Innovations, Inc.
#
#
# This makefile was automatically generated by rtiddsgen.
#
# To compile, type:
#   gmake -f makefile_position_i86Linux2.6gcc4.1.1jdk
#
```

```
# Note: This makefile is only meant to build our example applications and
#       may require alterations to build on your system.
#
# Make sure that javac and java are in your path.
############################################################################

JAVA_PATH = java
JAVAC_PATH = javac

JAVA_SOURCES
=  ./Position.java ./PositionSeq.java ./PositionTypeSupport.java ./Posit
ionTypeCode.java ./PositionDataReader.java ./PositionDataWriter.java ./
PositionSubscriber.java ./PositionPublisher.java ./TestPositionRun.java
 ./PubThread.java ./PubLauncher.java ./OperatorSubscriber.java ./Positio
nSubscriberFirst.java ./P.java ./PassengerSubscriber.java ./PositionSub
scriberSecond.java ./InputTest.java ./InputParameter.java

CLASS_FILES  = $(JAVA_SOURCES:%.java=%.class)

RTI_CLASSPATH := $(NDDSHOME)/class/nddsjava.jar

%.class : %.java
    $(JAVAC_PATH)  -classpath .:$(RTI_CLASSPATH) $<

all: $(CLASS_FILES)

#
# Convenient way to run the java programs
#

export LD_LIBRARY_PATH :=
$(NDDSHOME)/lib/i86Linux2.6gcc4.1.1jdk:/usr/lib/lwp:$(LD_LIBRARY_PATH)

PositionPublisher: ./PositionPublisher.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)"  PositionPublisher
$(ARGS)

PositionSubscriber: ./PositionSubscriber.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)"  PositionSubscriber
$(ARGS)

TestPositionRun: ./TestPositionRun.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)"  TestPositionRun $(ARGS)

PubThread: ./PubThread.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)"  PubThread $(ARGS)

PubLauncher: ./PubLauncher.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" PubLauncher $(ARGS)

OperatorSubscriber: ./OperatorSubscriber.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" OperatorSubscriber
$(ARGS)

PassengerSubscriber: ./PassengerSubscriber.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" PassengerSubscriber
$(ARGS)
```

```
InputTest: ./InputTest.class
    $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" InputTest $(ARGS)
```

D3: OperatorSubscriber.java

```
public class OperatorSubscriber implements Runnable{
      boolean flag;

      public OperatorSubscriber(boolean flag) {
     this.flag = flag;
      }

      public void run() {
     //new PositionSubscriber().subscriberMain(0,0);
     if(flag)
           new AccidentSubscriber().subscriberMain(0,0);
     else
           new PositionSubscriber().subscriberMain(0,0);
      }

      public static void main(String[] args) {
     System.out.format("%s%8s%14s%10s%8s%8s%20s%8s%12s",
"MessageType","Route","Vehicle","Traffic","Stop#","#Stop","TimeBetweenStops","Fill%"
,"Timestamp");
     new Thread(new OperatorSubscriber(true)).start();
     new Thread(new OperatorSubscriber(false)).start();
      }
}
```

D4: P.java

```
public class P {
      public String timestamp;
      public String route;
      public String vehicle;
      public int stopNumber;
      public int numStops;
      public float timeBetweenStops;
      public String trafficConditions;
      public int fillInRatio;
      public int remainingStops;
      /*public int routeNumber;
      public int sourceStop;
      public int destinationStop;*/

      public P() {
      }
      public P(String timestamp, String route, String vehicle, int stopNumber, int numStops,
```

```
float timeBetweenStops, String trafficConditions, int fillInRatio, int remainingStops) {
      this.timestamp = timestamp;
      this.route = route;
      this.vehicle = vehicle;
      this.stopNumber = stopNumber;
      this.numStops = numStops;
      this.timeBetweenStops = timeBetweenStops;
      this.trafficConditions = trafficConditions;
      this.fillInRatio = fillInRatio;
      this.remainingStops = remainingStops;
      /*this.routeNumber = routeNumber;
      this.sourceStop = sourceStop;
      this.destinationStop = destinationStop;*/
       }
}
```

D5: pub.properties

```
numRoutes=2
numVehicles=3

route1=Express1
route2=Express2

#route1
route1numStops=4
#seconds for the interval
route1TimeBetweenStops=2
route1Vehicle1=Bus11
route1Vehicle2=Bus12
route1Vehicle3=Bus13

#route2
route2numStops=6
#seconds for the interval
route2TimeBetweenStops=3
route2Vehicle1=Bus21
route2Vehicle2=Bus22
route2Vehicle3=Bus23
```