Dirty Jobs

Instructions and info

QWAN

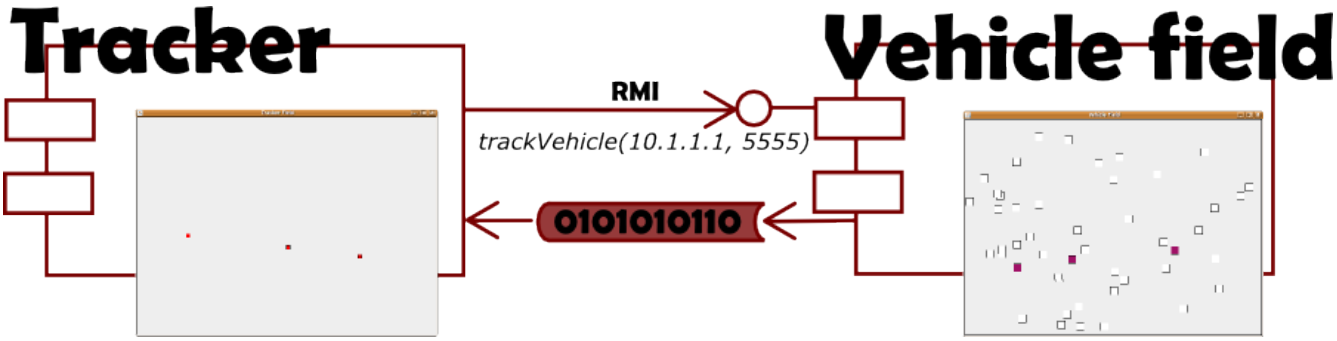
Quality Without A Name

Introduction

We will be working on legacy code, and want to make safe changes using Test Driven Development. The system we are working on was not made that way, and we need to find a way to start with our first tests.

The system

The system is based on actual code we ran into during one of our coaching assignments We simplified it a little and 'depersonalised' it. We are working on a system that tracks vehicles in a network.



The field containing trackable vehicles are somewhere on the network and the tracker we'll be working on can request vehicles to send information on their location, speed, direction and such. Vehicles send that information through a proprietary protocol. And our tracker decodes the messages and puts it in a cache. The current vehicle positions are displayed. See for more information on the protocol: Appendix A

Setting up the exercise

1. Unzip participant-workspace.zip
   This creates the eclipse workspace 'dirtyworkspace'
2. Open eclipse (if you want to use eclipse of course) with the workspace
3. If you prefer to use another text editor and an then
   1. cd dirtyworkspace\dirtyjobs
   2. ant

Dirty Jobs

Instructions and info

QWAN

Quality Without A Name

Running the code

1. Fire up the RMI Registry.

   Some communication is done through RMI (remote method invocation). Therefore before starting anything else, you'll have to fire up the registry.

   ```
   rmiregistry
   ```

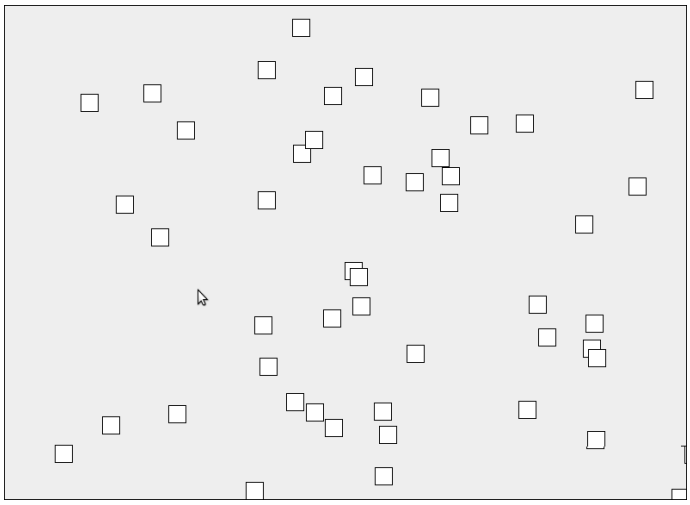
   If it is not in your path, you should be able to find it in your Jdk bin directory.

2. Run the vehicle-field

   in the dirtyjobs project directory run:

   ```
   ./vehicle-field
   ```

   You should see something like →

   Keep it running.

3. Run the tracker-field

   If you are using eclipse: there is a predefined runconfiguration “Tracker Gui” in your workspace. Just run that.

   Othewise you can run the script:

   ```
   ./tracker-field 4444
   ```

   (4444 can be any other free port on your machine)

   The tracker tracks three vehicles. You will see another field with three squares and magically on the vehicle-field three squares have a different color.

That's it! Congratulations!

Dirty Jobs

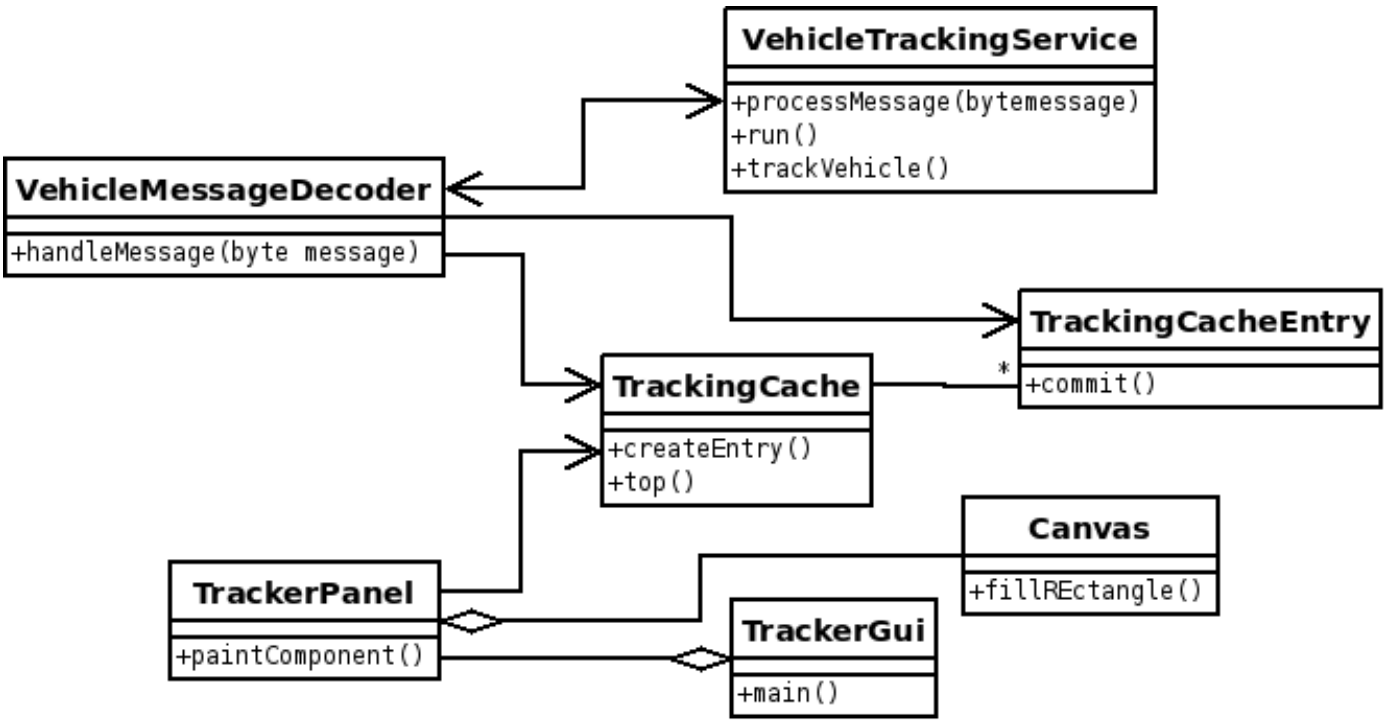
Instructions and info

QWAN

Quality Without A Name

Looking at the code

We will be working on the tracker. We need to make changes and sadly there are no tests.
The picture below shows an overview of the tracker:



First we'll study the code and collect your remarks on the code. Don't be polite ;-).

Please focus on the classes

- VehicleTrackingService
- VehicleMessageDecoder

We'll be working on those later.

Dirty Jobs

Instructions and info

QWAN

Quality Without A Name

First story

As a tracker I want the same color as on the big screen to be displayed when tracking vehicle

Vehicles take on another color when they are tracked. The story is about taking on the same color in the tracker field. Oh – eh the color is random.

A hint: The color is returned by the trackVehicle message:

```
public interface TrackableField extends Remote {
 /**
  * ask a trackable gently to track a vehicle.
  *
  * If the request is accepted the method returns a color that
  * vehicle takes on and the vehicle will start sending info
  * using the protocol defined in Protocol and
  * CodecUtils to the specified host and port.
  *
  * All messages will contain the identification byte passed here.
  *
  * @param host the host to send protocol messages to
  * @param port the port to send protocol messages to
  * @param identification
  * @return the rgb value of the color the sending vehicle takes on.
  * @throws RemoteException
  */
  public abstract Integer trackVehicle(String host, int port,
                          byte identification) throws RemoteException;

}
```

The rule of the game: *NO CODE WITHOUT TESTS!*

Dirty Jobs

Instructions and info

QWAN

Quality Without A Name

Second story

The as a tracker I want the tracked vehicles position extrapolated when a vehicle discontinues sending information.

Vehicles buggy in in this version. Sometimes they stop sending messages. (you can simulate that with the field-control script. The tracker should extrapolate positions at that point.

Dirty Jobs

Instructions and info

QWAN
Quality Without A Name

Appendix A

The protocol:

Protocol contains data in a compressed data stream containing Full and Partial frames
More specifically; repeatedly one full frame is send and then 4 parital frames:

```
|F|P|P|P|P

It is up to the receiver to assemble full frames based on partials

Full Frame
Fixed length 15 bytes
|F|CRC|I|T|P|D|S|E|A|C|
F Frametype
     [1 bytes 0x00 = Full Frame]
CRC                     [1 bytes crc32]
T TimeStamp             [4 bytes unsigned long int Little Endian]
I Identification        [1 byte identification]
P Position              [4 bytes |X|Y| small int x, small int y]]
D Direction             [2 bytes [X|Y]
                             0|1 = south,
                             0|FF = north,
                             1|0 = east,
                             FF|0 = west]
S Speed                 [1 bytes byte value]
E EngineOn              [1 bytes byte value]
A Dangerouslyclose      [1 bytes boolean (1 = true, 0 = false)]
C Collision             [1 bytes boolean (1 = true, 0 = false)]
```

Dirty Jobs

Instructions and info

QWAN

Quality Without A Name

Partial Frame

Variable length 5 bytes + data (length depends on Frametype - see above)

|F|CRC|I|T|DDDD

F Frametype [1 bytes 0x01 = Position]
[0x02 = Direction]
[0x03 = Speed]
[0x04 = EngineOn]
[0x05 = Dangerouslyclose]
[0x06 = Collision]

T Timestamp

I Identification

D data (max 4 bytes)

Appendix B

Protocol utilities:

See also docs/javadocs/index.html in the project.

CodecUtils.createFullFrame

```
public static byte[] createFullFrame(byte ident,
                                     int tstamp,
                                     int xpos,
                                     int ypos,
                                     Direction dir,
                                     Speed spd,
                                     Engine eon,
                                     CloseToOther close,
                                     Collision coll)
```

Creates a full frame to be send to the tracker. Frame is a byte array with the format as specified in appendix A