2016-2017

Examenproject – 2e zittijd

Boete berekening bij verkeersovertredingen

Inleiding

Dit document beschrijft het examenproject van het vak Software engineering 3 voor de **tweede zittijd**. Dit project wordt **individueel** uitgewerkt. Het wordt afgegeven, gedemonstreerd en toegelicht op de dag van het examen (zie verder, 'Verloop van het examen').

Opdracht

Ontwerp en implementeer een systeem voor het bepalen van het boetebedrag bij verkeersovertredingen. Dit omvat een aantal aspecten die in wat volgt worden omschreven.
Opmerking. Met ***instelbaar*** en ***instellen*** wordt in wat volgt steeds bedoelt: instelbaar vanuit de setup code, daar waar de objecten worden geassembleerd ('wiring') en de toepassing wordt gestart. Met *in-memory lijst* wordt bedoelt dat gegevens hard-coded in de setup code worden aangemaakt en meegegeven aan de applicatie.

A. Generator

De *generator* is een standalone Java applicatie die wordt gebruikt om test overtredingen aan te maken en door te sturen, zodat de component voor boete berekening (*Penalty calculator*, zie verder) hiermee kan uitgetest worden.
Er zijn twee types van overtredingen: *snelheid* en *emissie*.

Snelheid

Het overtreden van de maximaal toegestane snelheid op een bepaalde locatie. Dit type overtreding bevat volgende informatie:

- **Timestamp** – het tijdstip van de overtreding
- **Nummerplaat** – String – de nummerplaat van het voertuig dat de overtreding heeft begaan
- **Straat –** String – de straat waarin de overtreding plaatsvond (bv. 'Kerkstraat')
- **Gemeente –** String – de stad of gemeente waar de overtreding plaatsvond (bv. 'Antwerpen')
- **Maximum snelheid** – Integer – de maximaal toegelaten snelheid in km/u
- **Snelheid** – Integer – de gemeten snelheid in km/u.

Emissie

Het niet voldoen aan de minimale euronorm[1] op een bepaalde locatie. Dit type overtreding bevat volgende informatie:

- **Timestamp** – het tijdstip van de overtreding
- **Nummerplaat** – String – de nummerplaat van het voertuig dat de overtreding heeft begaan

[1] Elke wagen heeft een bepaalde euronorm die aangeeft hoe mileuvriendelijk deze is. In bepaalde steden mogen enkel wagens met een voldoende hoge euronorm de stad binnenrijden.

- **Straat –** String – de straat waarin de overtreding plaatsvond (bv. 'Kerkstraat')
- **Gemeente –** String – de stad of gemeente waar de overtreding plaatsvond (bv. 'Antwerpen')
- **Minimum euronorm** – Integer – de minimale euronorm
- **Euronorm** – Integer – de euronorm van het voertuig

De verschillende velden worden als volgt random gegenereerd

- *Timestamp* het tijdstip van generatie van het bericht
- *Nummerplaat* random geselecteerd uit een instelbare in-memory lijst van nummerplaten in het geldige Europese formaat (1 cijfer + 3 letters + 3 cijfers gescheiden door een '-')
- *Straat* een random string (max. 100 char)
- *Gemeente* random geselecteerd uit een instelbare in-memory lijst van gemeenten
- *Maximum snelheid* random geselecteerd uit een instelbare in-memory lijst (bv. *30, 50, 70, 120*)
- *Snelheid van het voertuig* random tussen [maximum snelheid + 1] en een instelbare bovengrens.
- Minimale euronorm random 3,4,5 of 6.
- Euronorm van het voertuig random tussen 1 en [minimale euronorm – 1]

De generator kan gestart en gestopt worden en heeft een random frequentie met instelbare grenzen. Stel bv. dat de grenzen op 1000 en 3000 worden ingesteld, dan dient er telkens een random aantal milliseconden dat ligt tussen 1000 en 3000 gewacht te worden alvorens een nieuwe overtreding wordt gegenereerd. Als de grenzen samenvallen heeft de generator een vaste frequentie. Ook de verhouding tussen de verschillende types overtredingen moet instelbaar zijn, en dit als een percentage waarbij de som van de instellingen 100% is. Bv. indien 70% snelheid en 30% emissie, dient elke gegenereerde overtreding een kans van 1 op 7 te hebben dat het een snelheidsovertreding is en een kans van 1 op 3 dat het een emissie overtreding is. Dit hoeft niet uitgemiddeld te worden over de tijd, de kans mag gewoon bepaald worden op het moment van generatie van de overtreding.

B. Message broker

Om hoge frequenties van overtredingen te kunnen bufferen voor verwerking, wordt gebruik gemaakt van een **message broker** (RabbitMQ,..) tussen de generator en de penalty-calculator. Het transport formaat dat wordt gebruikt is XML. Elke overtreding wordt apart verstuurd. Het is vrij te kiezen of er wordt gewerkt met 1 queue voor alle types van overtredingen of met een aparte queue per type.

C. Penalty calculator

Deze standalone Java applicatie berekent het bedrag van de boete voor een binnenkomende overtreding en stuurt dit bedrag samen met de overtreding uit via een uitgaande queue. Hiervoor moeten **boete factoren** worden opgevraagd via de proxy van een *penalty-service* (zie zip file op BB). De factoren dienen in memory bijgehouden te worden om onnodige netwerk trafiek te

vermijden. Het moet instelbaar zijn na hoeveel seconden deze opnieuw opgevraagd worden[2]. Bv. indien ingesteld op 50000 seconden worden de boetefactoren pas opnieuw opgevraagd indien het bij de berekening van een boete meer dan 50000 seconden geleden is dat deze nog eens werden opgevraagd.
Voor het bepalen van het boetebedrag zijn er verschillende berekeningsmethoden. Het moet instelbaar zijn van welke methode de verwerkingseenheid gebruik maakt.

1. **Zonder historiek**.
    a. Een snelheidsboete wordt als volgt berekent:
    *bedrag = (snelheid – maximaal toegelaten snelheid in de zone) * snelheid boetefactor*
    b. Een emissieboete wordt als volgt berekent (er wordt geen rekening gehouden met hoever men onder de euronorm zit):
    bedrag = emissie boetefactor

2. **Met historiek.** Het bedrag is in dit geval afhankelijk van het aantal begane overtredingen in het verleden:

    *bedrag = bedrag zonder historiek + (# overtredingen door nummerplaat * historiek factor)*

    Het aantal overtredingen in het verleden kan voor een gegeven nummerplaat opgevraagd worden via de *penalty-service*. Dit wordt nooit gecached (in tegenstelling tot de factoren, zie hoger)

Aangezien een lage emissiezone typisch bewaakt wordt door meerdere camera's, komt het vaak voor dat er kort achter elkaar overtredingen binnenkomen voor dezelfde nummerplaat in dezelfde gemeente/stad. Het systeem dient deze eruit te filteren door elke emissie overtreding te bufferen (in memory bij te houden) gedurende een instelbaar aantal seconden (bv. 3600). Overtredingen met eenzelfde nummerplaat en gemeente/stad als een gebufferde overtreding worden genegeerd (wel gelogd).

De overtreding + het berekende boetebedrag wordt op een uitgaande queue gezet voor verdere verwerking door de facturatie afdeling. Hier hoeft verder niets meer mee te gebeuren.

Indien de *penalty-service* een IOException gooit voor de settings, wordt teruggevallen op default waarden (deze zijn instelbaar vanuit de setup code), tenzij er al waarden gecached zijn van een vorige aanroep, dan worden deze gebruikt. Indien de *penalty-service* een IOException gooit voor een nummerplaat, wordt een instelbaar aantal keer geprobeerd deze opnieuw aan te roepen en dit na een instelbare tijd. Indien het uiteindelijk niet lukt, wordt dit gelogd en wordt er voor deze overtreding geen boete berekend en niets uitgestuurd via de queue. Bij een onbekende of ongeldige nummerplaat wordt dit gelogd en wordt er geen boete berekent en niets uitgestuurd. Ook indien er iets anders misloopt (bv. op uitgaande queue zetten), wordt de error gelogd en geen boete berekend en uitgestuurd.
De verwerking van volgende overtredingen gaat in elk geval door.

[2] Boetetarieven worden af en toe aangepast door de overheid en men wil vermijden dat de software hiervoor moet gestart en gestopt worden.

D. Verwachte uitbreidingen

Volgende wijzigingen/uitbreidingen kunnen verwacht worden in de toekomst

- Nieuwe types van overtredingen (bv. parkeren)
- Meer realistische berekeningen van de boetes (andere formules, complexere factoren,…)
- Wijzigingen aan de API's van alle services (zowel protocol als formaat)
- Gebruik van een ander type broker en wijzigingen aan het formaat van berichten op de broker
- Andere mechanismen voor het genereren van overtredingen (bv. uit een bestand)

E. Niet-functionele vereisten

- Er wordt kwaliteitsvolle **code** opgeleverd voor zowel de generator als de calculator. UML kan hierbij een hulpmiddel zijn.
- De 2 applicaties zijn onafhankelijk van elkaar, ze delen geen code en communiceren enkel via de message broker. Alle verwerking van gegevens gebeurt verder volledig in memory, er wordt geen externe databank gebruikt. Als broker wordt een open source messaging systeem zoals RabbitMQ of ActiveMQ gebruikt (cloud of lokaal geïnstalleerd naar keuze). Het berichtenformaat op de broker is XML.
- De code wordt geschreven in Java - in het Engels[3]. Je gebruikt enkel de standaard JDK (versie 8)[4]. Je gebruikt geen externe code/library's/… behalve voor het benaderen van de message broker, formaat conversie tussen JSON/XML/Java en voor logging (zie verder). Voor JSON en XML conversie mag gewerkt worden met een library naar keuze (org.json, javax.json, castor,…).
- Logging mag gebeuren
  - ofwel op System.out en dit in een static method van een zelf te schrijven Logger class waaraan info (message, level, exception) kan doorgegeven worden vanuit code die de log wil schrijven.
  - ofwel via slf4j-log4j, log4J of een andere Java logging library
- Het gebruik van gradle of maven is toegelaten maar niet verplicht.
- Het uitwerken van unit testen is optioneel.

Oplevering

Het project wordt individueel uitgewerkt en afgegeven door dit voor de aanvang van het examen door te sturen naar het e-mail adres be.kdg.inf.se3@gmail.com. Dit e-mail adres mag enkel gebruikt worden voor het doorgeven van de oplevering. Je stuurt een (drive, dropbox,…) **link** door naar een **zip** (geen rar, niet als attachment). De naam van de zip is als volgt: achternaam_voornaam.zip.

Verloop van het examen

Het examen duurt ongeveer 40 minuten (waarvan 20' voorbereiding)

- Voor binnenkomst zet je op je laptop de 2 projecten open (in IntelliJ of andere IDE).
- Je komt binnen zodra een vorige student het lokaal verlaat, tekent de lijst en trekt 2 vragen die je uitschrijft op papier, gesloten boek. De vragen handelen over concepten uit de cursus. Zie BB voor een overzicht van de mogelijke vragen.

[3] Je wordt in dit vak niet beoordeeld op de kwaliteit van het Engels op zich, wel op de kwaliteit van de code/commentaar
[4] Spring (boot) mag indien gewenst wel gebruikt worden

- Wanneer de vorige student klaar is met zijn verdediging, koppel je je laptop aan de beamer. Het project wordt toegelicht, bevraagd en beoordeeld (70%). De 2 vragen worden overlopen en besproken met eventuele bijvragen (30%).

Beoordelingscriteria

- Een goed begrip van de tijdens de les aangebrachte concepten (zie slides)
- Een correcte toepassing hiervan in het project
- Cleane en onderhoudbare code (met inbegrip van logging en error handling) die voldoet aan de standaard Java code conventies en zinvol gedocumenteerd is.
- De beheersing van het project: het ontwerp, de code, de te verwachten uitbreidingen en hoe de code hierop voorzien is.

Veel succes!
Bart Vochten