Kestirimle Hedefi Bul

Recep Furkan Koçyiğit

Programın Çalıştırılması:

1. https://www.apache.org/dyn/closer.cgi?path=/kafka/3.4.0/kafka_2.13-3.4.0.tgz adresinde Apache Kafka indirilir.
2. İndirilen klasörde konsol açılır.
3. .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties komutu çalıştırılır.
4. Aynı klasörde başka bir konsol açılır.
5. .\bin\windows\kafka-server-start.bat .\config\server.properties komutu çalıştırılır.
6. Bin\windows dosya yolunda bir adet daha konsol açılır.
7. kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic sensor1 komutu çalıştırılır.
8. kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic sensor2 komutu çalıştırılır.
9. kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic center komutu çalıştırılır.
10. World uygulaması bir IDE yardımıyla ayağa kaldırılır.
11. Sensor1 uygulaması bir IDE yardımıyla ayağa kaldırılır.
12. Sensor2 uygulaması bir IDE yardımıyla ayağa kaldırılır.
13. Center uygulaması bir IDE yardımıyla ayağa kaldırılır.

Verilen vaka Spring Boot kullanılarak Java 17 ile yapılmıştır. Vaka kapsamında 4 adet uygulama yapılmıştır ve haberleşmeleri için Apache Kafka kullanılmıştır.

World Uygulaması:

2 adet sensör ve bir adet hedefin rastgele yerleştirilmesi için yapılmıştır. 8080 portu ile ayağa kalkar. “localhost:8080/world/simulate” adresi ile rastgele yerleştirilen Sensör1, Sensö2 ve Hedef1 için Sensör1 ve Sensör2 uygulamalarına kendi ve hedef konum bilgilerini gönderir.

İş Parçacıkları:

- KafkaProducerConfig.java → 9092 portunda çalışan Apache Kafka üzerinden Sensör1 ve Sensör2 uygulamalarına mesaj gönderilmesi için yapılan konfigürasyondur.
- WorldController.java → Simülasyonu başlatmak için yazılmış REST web servisidir.

- LocationDto.java → Sensörlere düzlemde nerede olduğu bilgisinin gönderilmesi için yazılmıştır.
- GetSensorWithTargetDto.java → İçerisinde 2 adet LocationDto nesnesi bulunmaktadır. Bunlar sensör ve hedef için lokasyon bilgileridir.
- WorldService.java → Sensörleri ve hedefi düzlemde rastgele yerleştirir ve ilgili sensörlere Kafka ile bulunduğu konum ve hedefin konumunu iletir.

Sensör1 Uygulaması

Apache Kafka üzerinde "sensor1" başlığını okuyarak kendi ve hedefin konum bilgilerini elde eder. Bu bilgileri kullanarak kerteriz bilgisini Kafka üzerinde "center" başlığına yazar.

İş Parçacıkları:

- KafkaConsumerConfig.java → Kafka üzerindeki "sensor1" başlığının okunması için yapılan konfigürasyondur.
- KafkaProducerConfig.java → Kafka üzerine "center" başlığına kerteriz bilgisinin gönderilmesi için yapılan konfigürasyondur.
- GetSensorWithTargetDto.java → Sensörün ve hedefin konum bilgilerini tutar.
- LocationDto.java → Konum bilgilerini x ve y olarak tutar.
- LocationWithAngleDto.java → Sensörün konum bilgisi ve kerteriz bilgisini tutar.
- KafkaListeners.java → Sensör ve hedefin konum bilgisinden kerteriz bilgisini elde eder. Sensörün lokasyonu ve kerteriz bilgisini "center" başlığına gönderir.
- Kerteriz bilgisinin bulunması: Düzlemdeki iki noktadan oluşan doğrunun eğimi bulunur. Eğimin arctan() alınarak açı bilgisi elde edilir.

Sensör2 Uygulaması

Apache Kafka üzerinde "sensor2" başlığını okuyarak kendi ve hedefin konum bilgilerini elde eder. Bu bilgileri kullanarak kerteriz bilgisini Kafka üzerinde "center" başlığına yazar.

İş Parçacıkları:

- KafkaConsumerConfig.java → Kafka üzerindeki "sensor2" başlığının okunması için yapılan konfigürasyondur.
- KafkaProducerConfig.java → Kafka üzerine "center" başlığına kerteriz bilgisinin gönderilmesi için yapılan konfigürasyondur.
- GetSensorWithTargetDto.java → Sensörün ve hedefin konum bilgilerini tutar.
- LocationDto.java → Konum bilgilerini x ve y olarak tutar.
- LocationWithAngleDto.java → Sensörün konum bilgisi ve kerteriz bilgisini tutar.
- KafkaListeners.java → Sensör ve hedefin konum bilgisinden kerteriz bilgisini elde eder. Sensörün lokasyonu ve kerteriz bilgisini "center" başlığına gönderir.
- Kerteriz bilgisinin bulunması: Düzlemdeki iki noktadan oluşan doğrunun eğimi bulunur. Eğimin arctan() alınarak açı bilgisi elde edilir.

Center Uygulaması

Verilen vakada merkezi temsil etmektedir. Sensörlerin lokasyon ve kerteriz bilgisinden hedefin yerini tespit eder.

İş Parçacıkları:

- KafkaConsumerConfig.java → Kafka üzerindeki "center" başlığının okunması için yapılan konfigürasyondur.
- FunctionDto.java → y = mx + c olarak verilen doğru denkleminde m ve c değerlerini tutar.
- LocationDto.java → Konum bilgilerini x ve y olarak tutar.
- LocationWithAngleDto.java → Sensörün konum bilgisi ve kerteriz bilgisini tutar.
- KafkaListeners.java → Sensörlerden aldığı konum ve kerteriz bilgilerinden hedefin yerini tayin eder. Kerteriz bilgisinden doğru denklemi oluşturulur. Daha sonrasında iki doğrunun kesiştiği noktanın bulunması için aşağıdaki gibi bir eşitlik kurulur:
    - y = m1x + c1 ve y = m2x + c2
    - m1x + c1 = m2x + c2
    - x = (c2 - c1) / (m1 – m2)
    - x değeri bulunduktan sonra doğru denkleminde yerine koyulur ve y değeri de elde edilir.

Projede kullanılan pom.xml deki bağımlılıklar ve kullanım nedenleri

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.kafka</groupId>
   <artifactId>spring-kafka</artifactId>
</dependency>

<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-devtools</artifactId>
   <scope>runtime</scope>
   <optional>true</optional>
</dependency>
<dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <optional>true</optional>
</dependency>
```

spring-boot-starter-web → Rest servis için kullanılmıştır.

spring-kafka → Apache Kafka için kullanılmıştır.

spring-boot-devtools → Hızlı uygulama restartı için kullanılmıştır.

Lombok → @Getter @Setter gibi sağladığı anotasyonlar için kullanımıştır.

Uygulama Görüntüleri

World uygulamasında düzleme yerleştirilen sensör ve hedefleri konum bilgileri:

```
Target : LocationDto(x=322, y=-337)
Sensor1 : LocationDto(x=-180, y=-161)
Sensor2 : LocationDto(x=433, y=163)
```

Center uygulamasında Sensor1 ve Sensor2 uygulamasından gönderilen kerteriz bilgileri ve merkezin tahmin ettiği hedefin yeri:

```
s1: LocationWithAngleDto(locationDto=LocationDto(x=-180, y=-161), angle=-19.320544324636806)
s2: LocationWithAngleDto(locationDto=LocationDto(x=433, y=163), angle=77.48332608556609)
Target is found(322, -337) by Center
```

Örnek2:

World uygulamasında düzleme yerleştirilen sensör ve hedefleri konum bilgileri:

```
Target : LocationDto(x=-1, y=5)
Sensor1 : LocationDto(x=-5, y=1)
Sensor2 : LocationDto(x=5, y=-1)
```

Center uygulamasında Sensor1 ve Sensor2 uygulamasından gönderilen kerteriz bilgileri ve merkezin tahmin ettiği hedefin yeri:

```
s1: LocationWithAngleDto(locationDto=LocationDto(x=-5, y=1), angle=45.0)
s2: LocationWithAngleDto(locationDto=LocationDto(x=5, y=-1), angle=-45.0)
Target is found(-1, 5) by Center
```