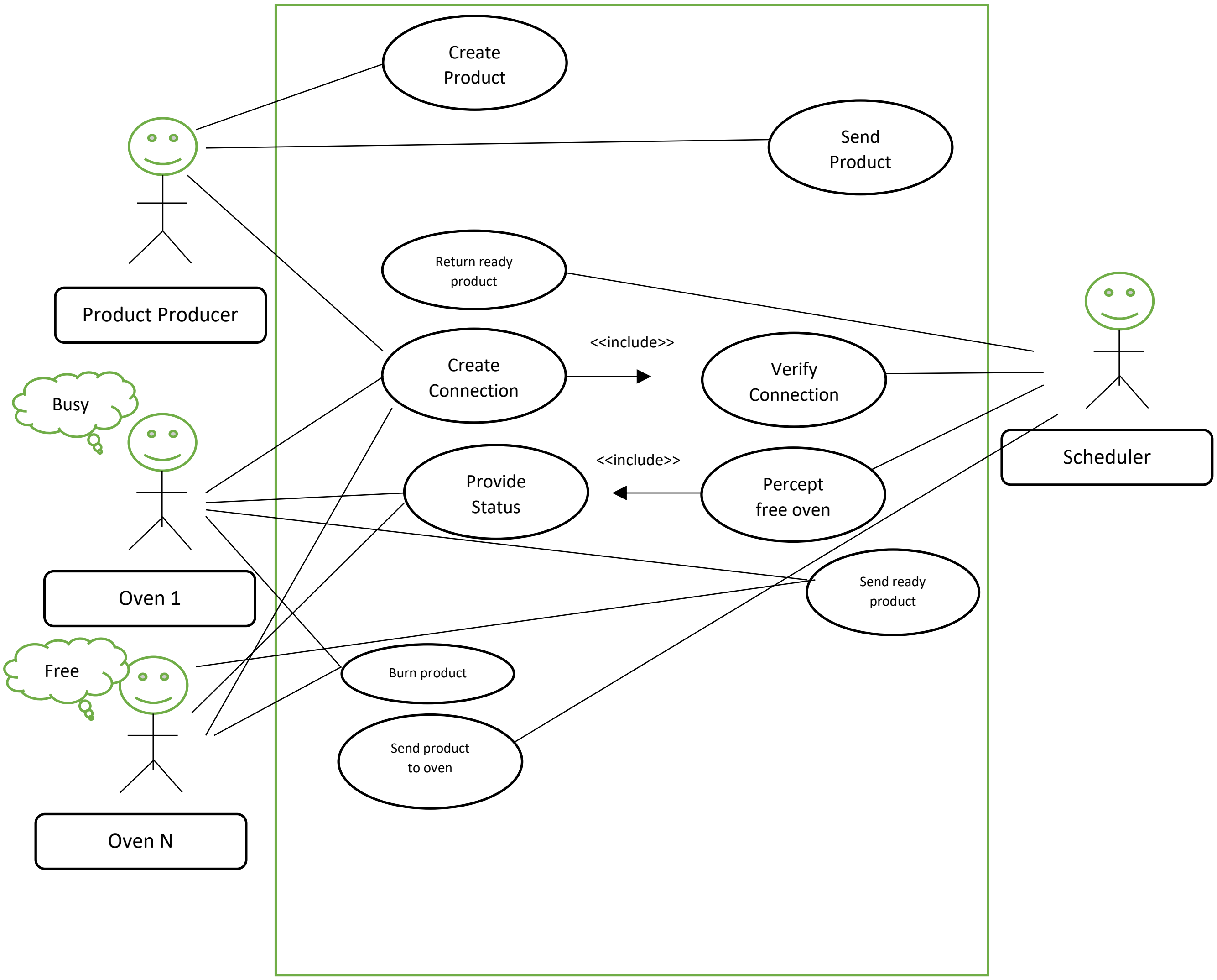
Use case diagram



Use case description

The product producer creates a connection with the scheduler (RabbitMQ). Arbitrary number of ovens also creates a connection to the scheduler (RabbitMQ). Producer produces arbitrary number of products and send the products to schedulter. Scheduler perceives oven's status and schedule/ send the product to free ovens to burn. Ovens return the product to the scheduler after the burning is done and then the scheduler returns the burned product to the product producer. When an oven becomes free, it lets the scheduler know so that it can get the next product (work) (Competing worker pattern)

Actors

1. Product Producer
2. Ovens 1…N

3. Scheduler (In this case RabbitMQ)

Triggers

1. Producer produce a product
2. Produce send the product to scheduler

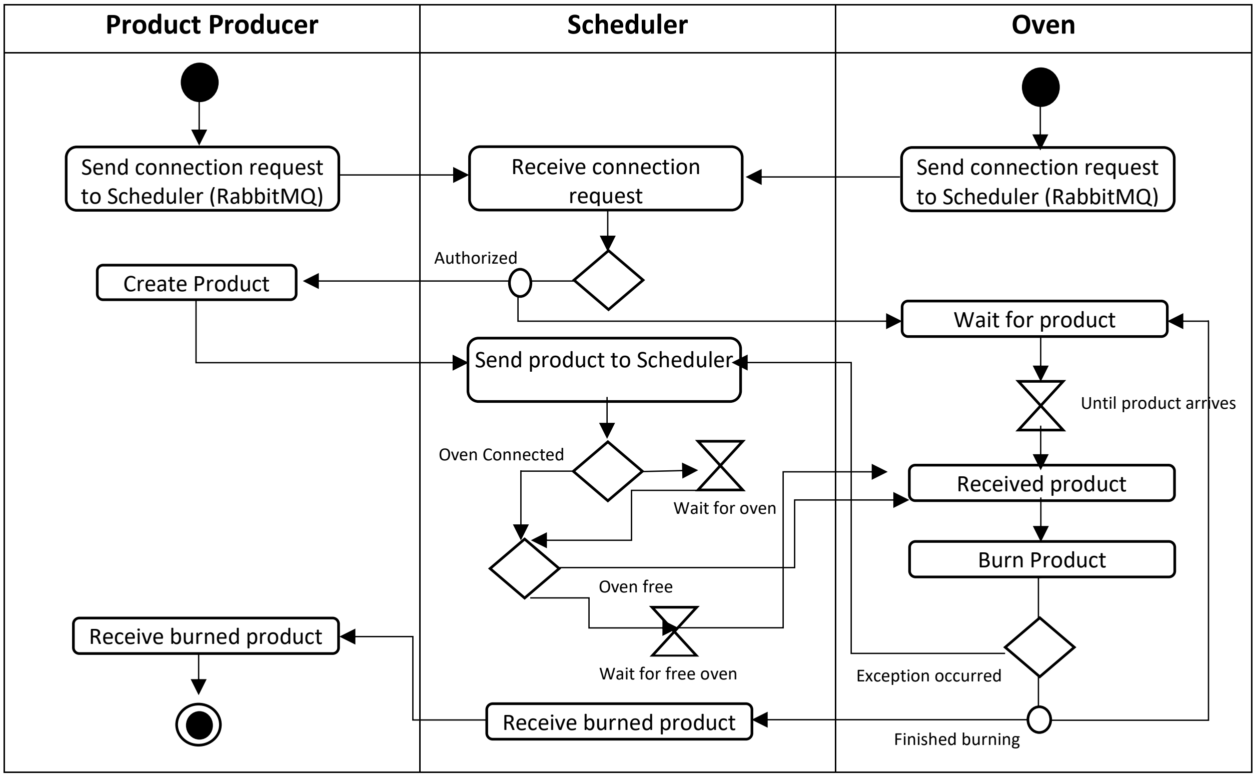
Preconditions

1. Both producer and ovens secure a connection with the scheduler
2. Scheduler has at least 1 oven to burn the product

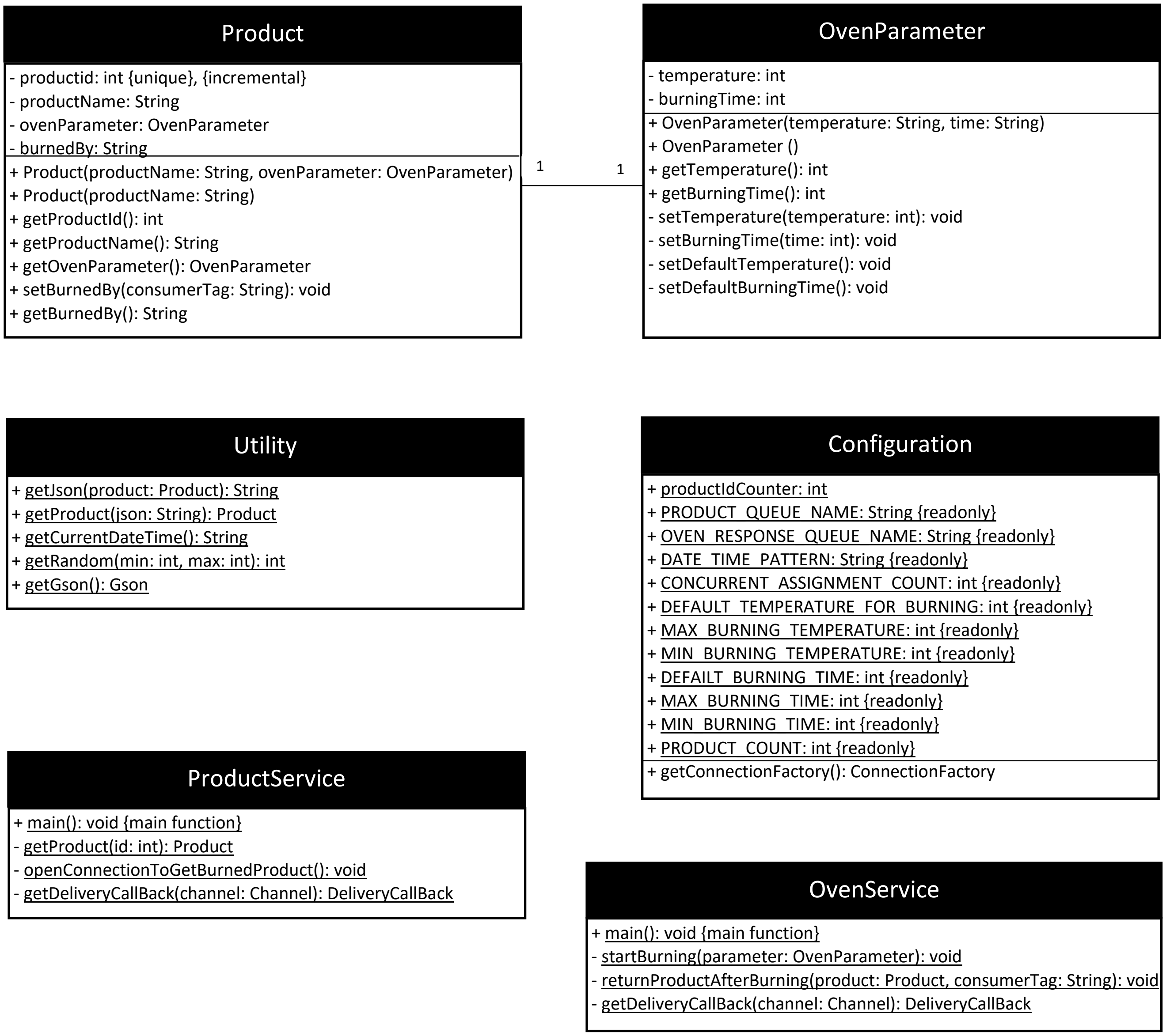
Goals

1. Burn all the products concurrently.
2. Reduce idle time of ovens.

UML Activity Diagram



Class Diagram



```
Product
- productid: int {unique}, {incremental}
- productName: String
- ovenParameter: OvenParameter
- burnedBy: String
+ Product(productName: String, ovenParameter: OvenParameter)
+ Product(productName: String)
+ getProductId(): int
+ getProductName(): String
+ getOvenParameter(): OvenParameter
+ setBurnedBy(consumerTag: String): void
+ getBurnedBy(): String
```

```
OvenParameter
- temperature: int
- burningTime: int
+ OvenParameter(temperature: String, time: String)
+ OvenParameter ()
+ getTemperature(): int
+ getBurningTime(): int
- setTemperature(temperature: int): void
- setBurningTime(time: int): void
- setDefaultTemperature(): void
- setDefaultBurningTime(): void
```

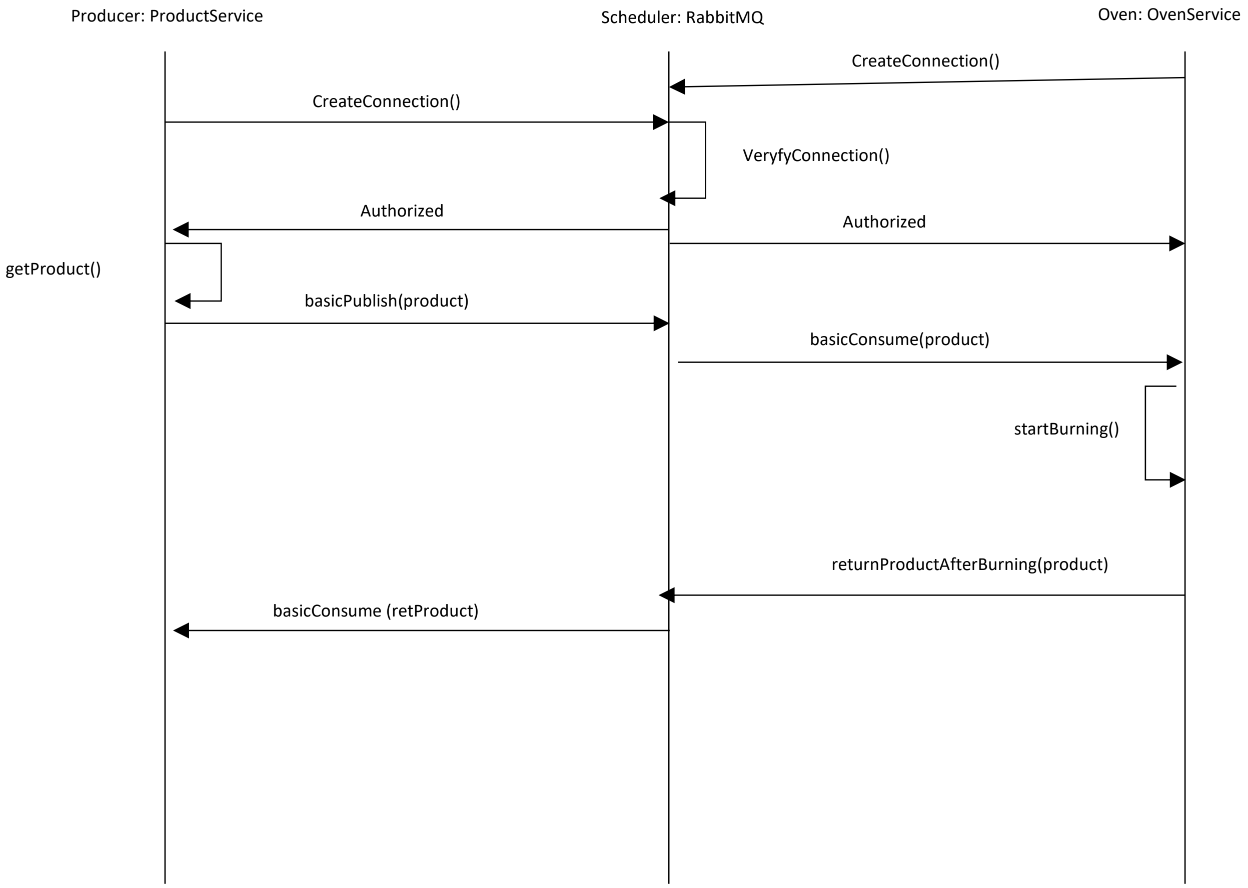
```
Utility
+ getJson(product: Product): String
+ getProduct(json: String): Product
+ getCurrentDateTime(): String
+ getRandom(min: int, max: int): int
+ getGson(): Gson
```

```
Configuration
+ productIdCounter: int
+ PRODUCT_QUEUE_NAME: String {readonly}
+ OVEN_RESPONSE_QUEUE_NAME: String {readonly}
+ DATE_TIME_PATTERN: String {readonly}
+ CONCURRENT_ASSIGNMENT_COUNT: int {readonly}
+ DEFAULT_TEMPERATURE_FOR_BURNING: int {readonly}
+ MAX_BURNING_TEMPERATURE: int {readonly}
+ MIN_BURNING_TEMPERATURE: int {readonly}
+ DEFAILT_BURNING_TIME: int {readonly}
+ MAX_BURNING_TIME: int {readonly}
+ MIN_BURNING_TIME: int {readonly}
+ PRODUCT_COUNT: int {readonly}
+ getConnectionFactory(): ConnectionFactory
```

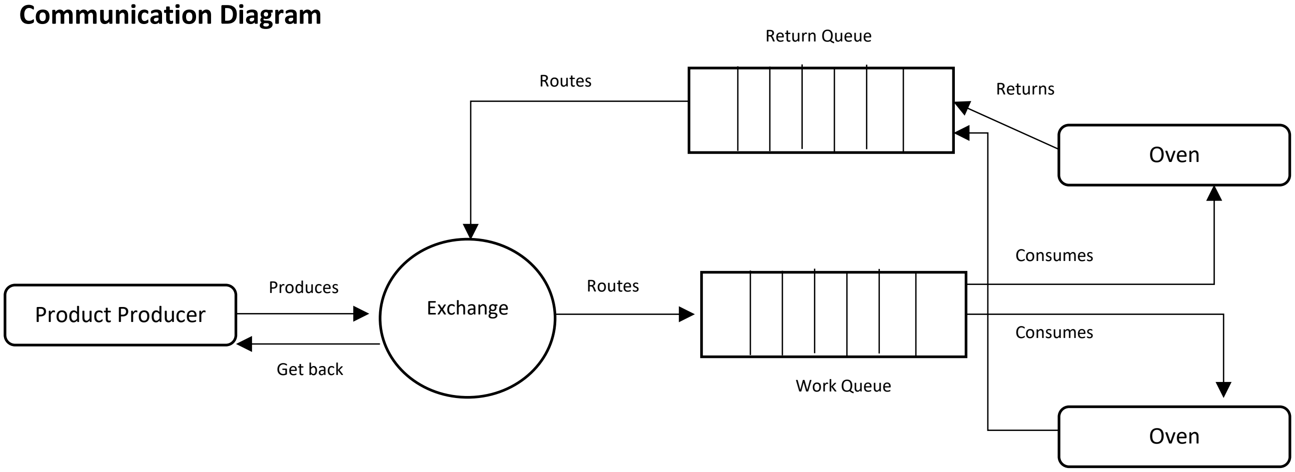
```
ProductService
+ main(): void {main function}
- getProduct(id: int): Product
- openConnectionToGetBurnedProduct(): void
- getDeliveryCallBack(channel: Channel): DeliveryCallBack
```

```
OvenService
+ main(): void {main function}
- startBurning(parameter: OvenParameter): void
- returnProductAfterBurning(product: Product, consumerTag: String): void
- getDeliveryCallBack(channel: Channel): DeliveryCallBack
```

Sequence Diagram



Communication Diagram


Return Queue
Routes
Returns
Oven
Consumes
Product Producer
Produces
Exchange
Routes
Get back
Consumes
Work Queue
Oven