Location transparency

how to avoid accidental distributed connascence

@MilenDyankov, Developer Advocate at AxonIQ

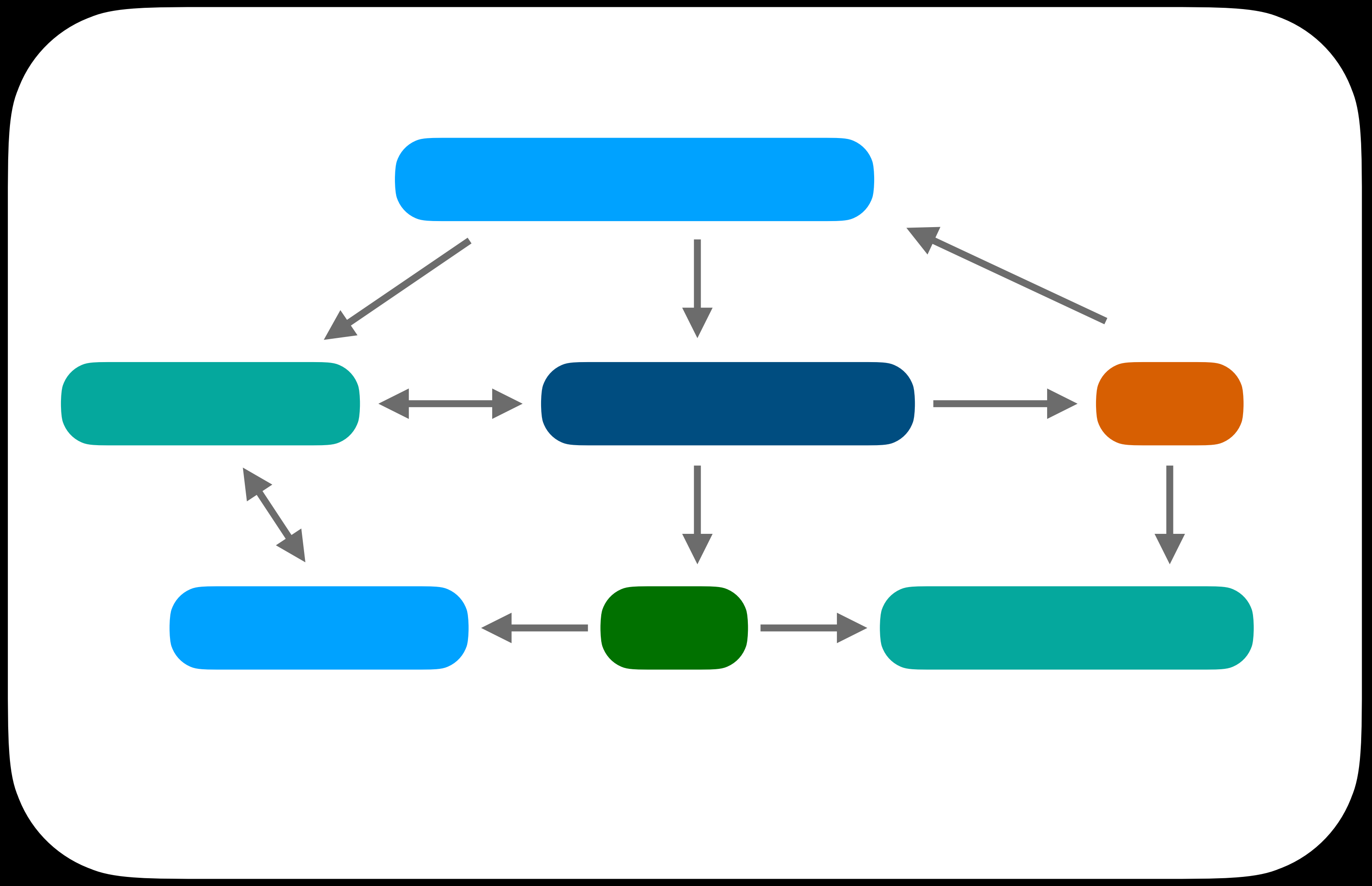
My house

My house

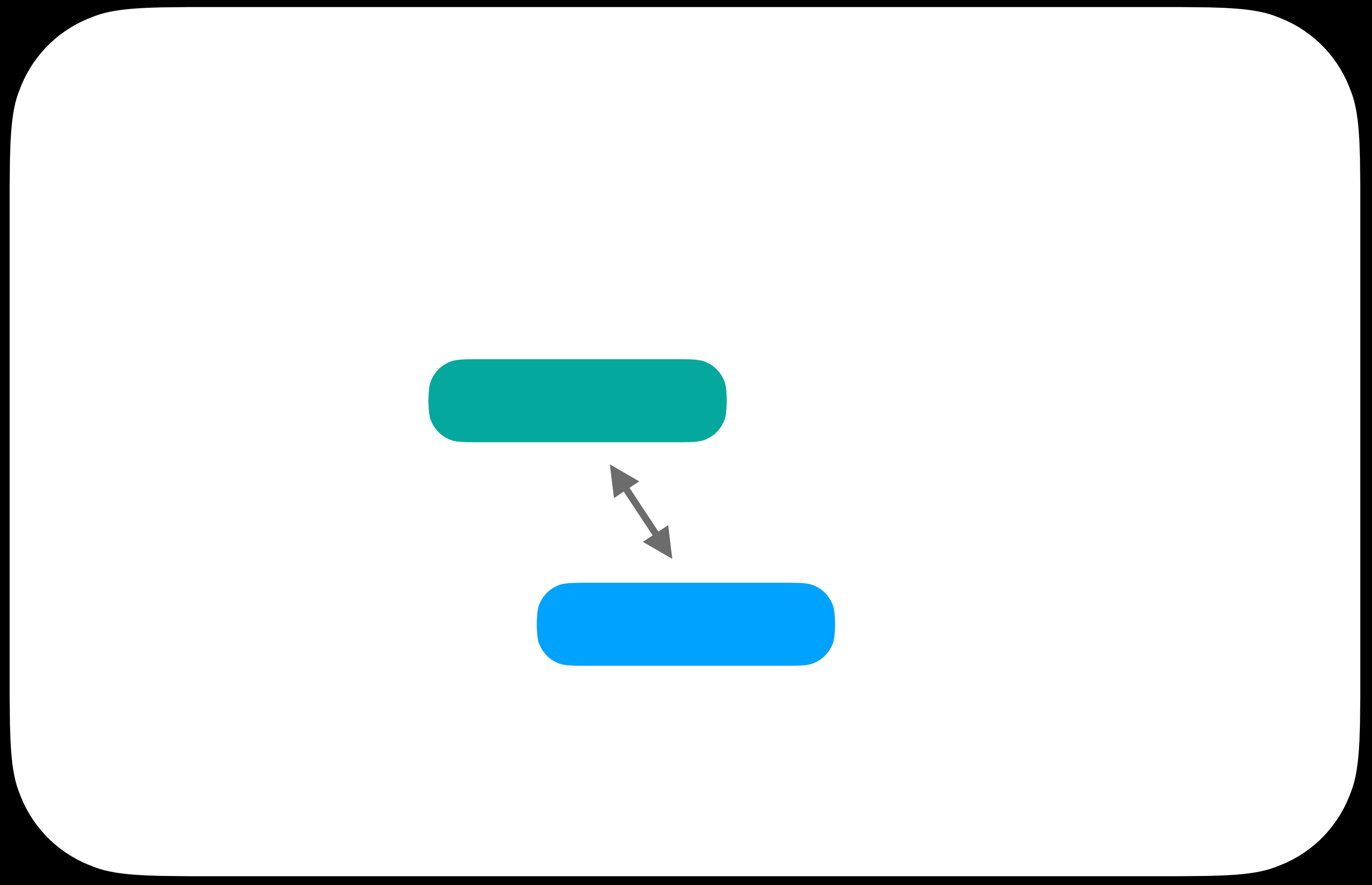
My house

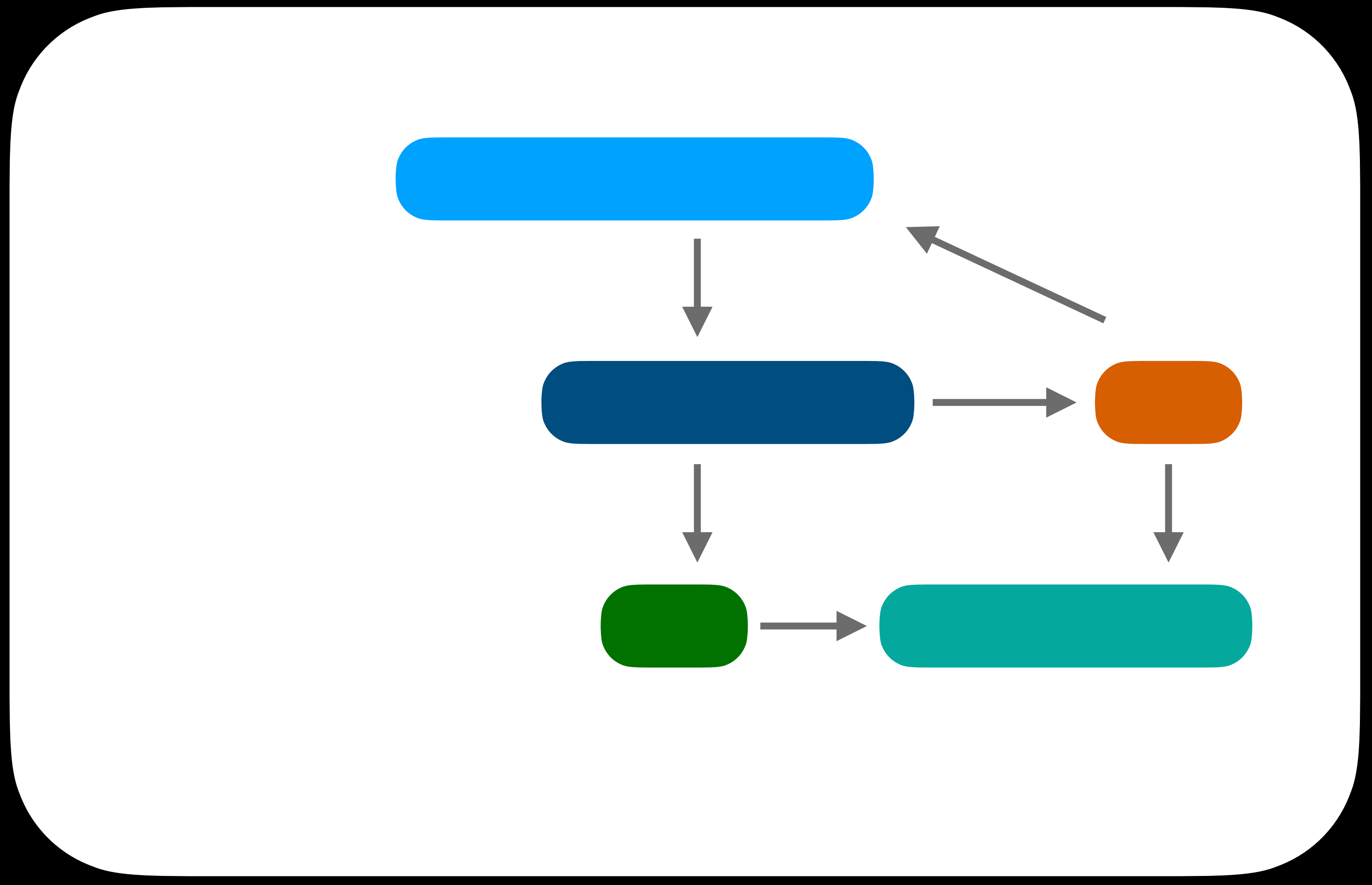
My house

My code

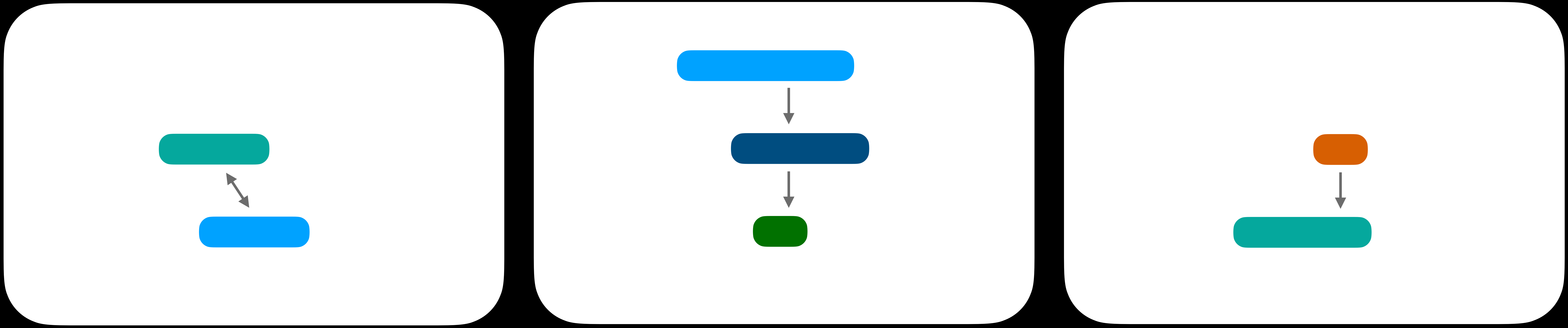


My code

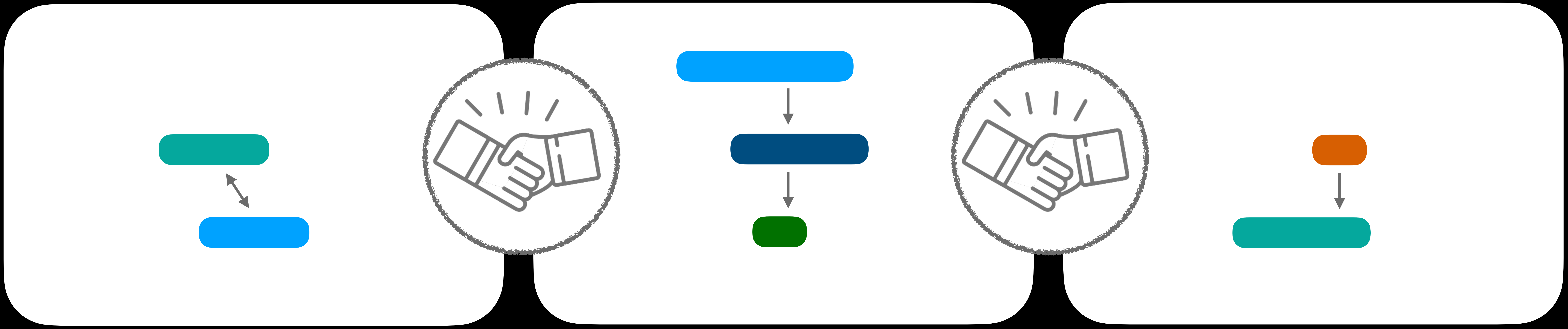




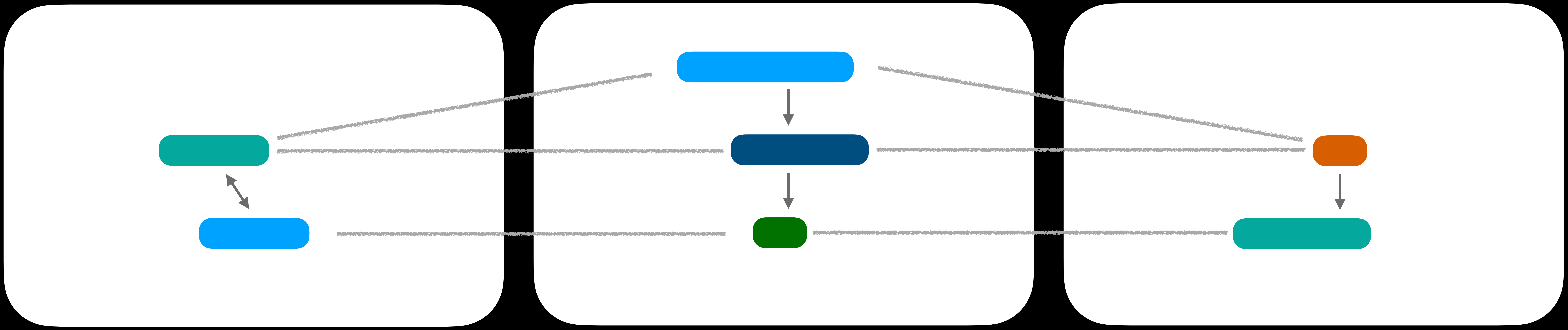
My code



My code



My code

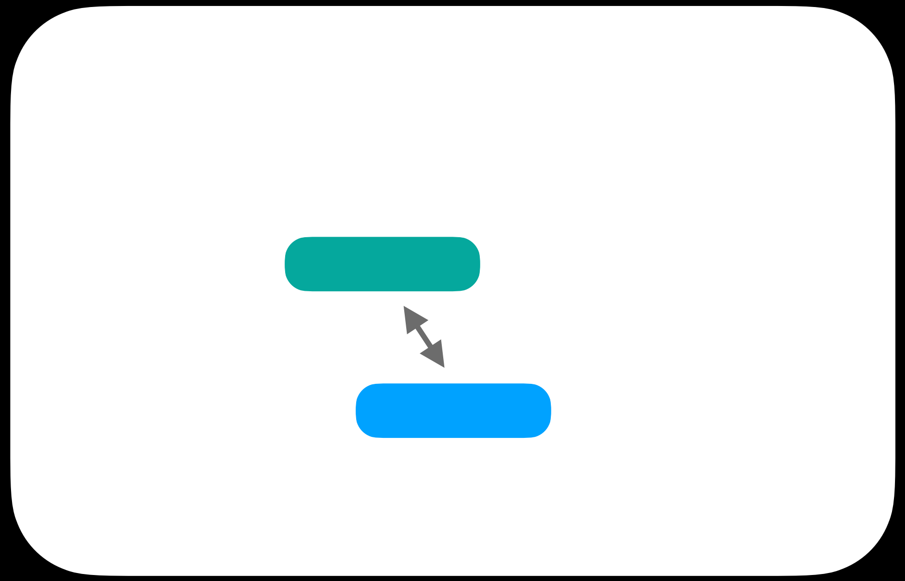
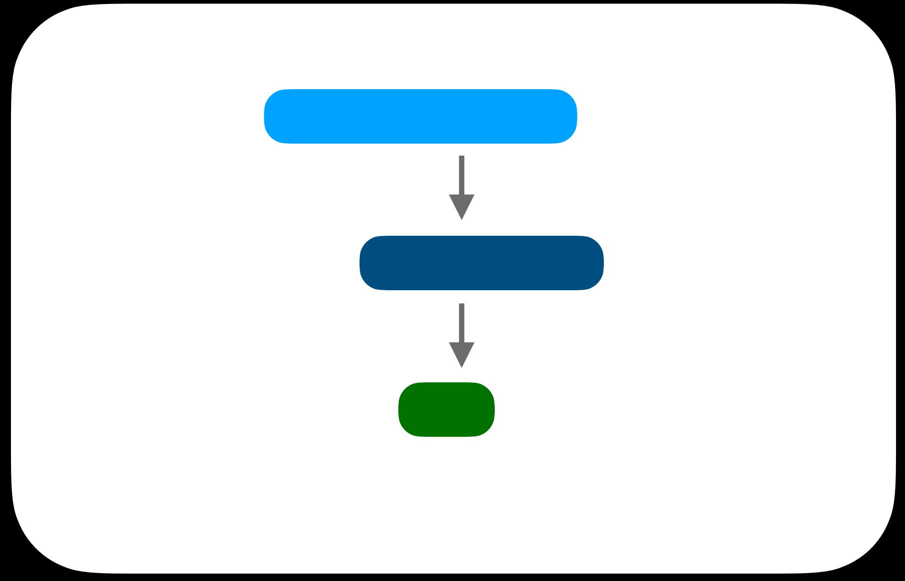
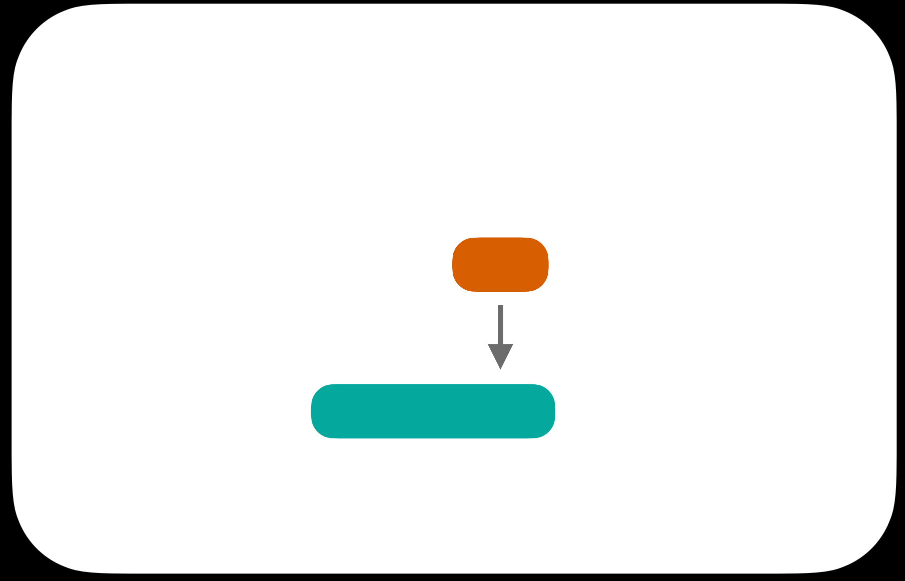


Sensing the nonsense

instantly

vs

too late

Coherence

- the quality of being logical and consistent
- the quality of forming a unified whole
- systematic or logical connection or consistency

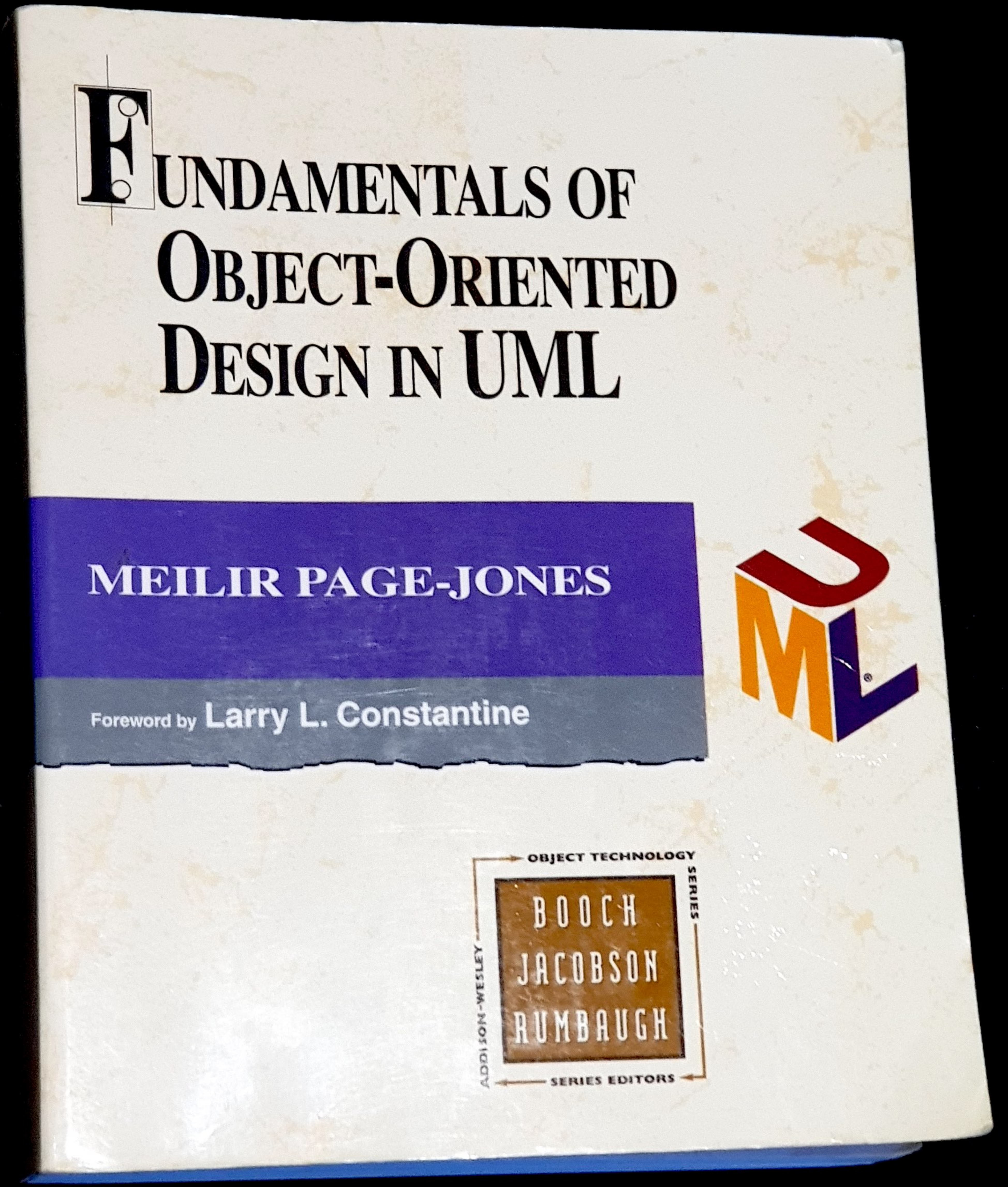
Coupling

| the act of bringing or coming together

| a device that serves to connect the ends of adjacent parts or objects

| the pairing of two items

Connascence



Connascence

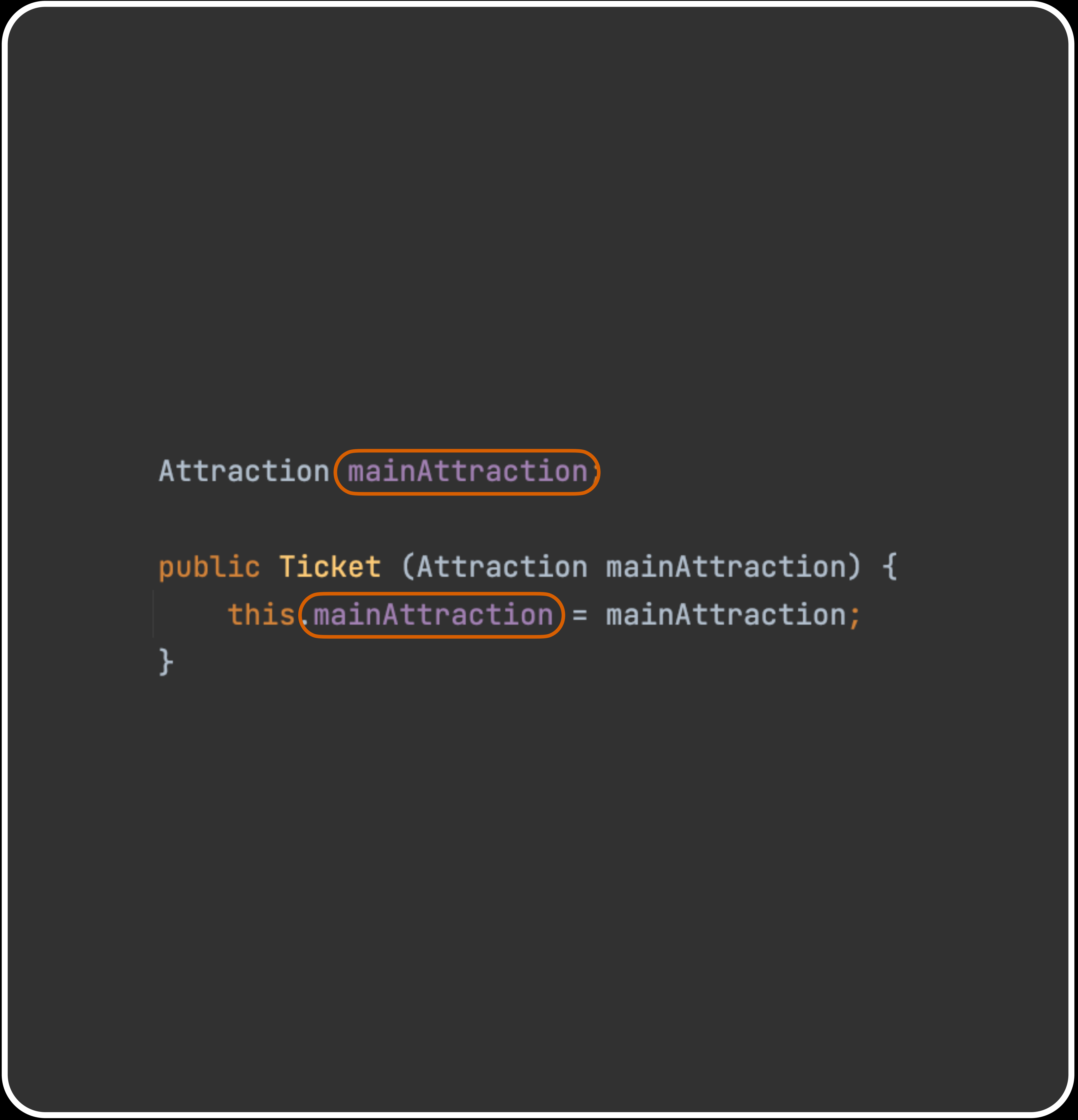
- production of two or more together
- having been born together
- the act of growing together

Connascence

```java
Attraction mainAttraction;

public Ticket (Attraction mainAttraction) {
    this.mainAttraction = mainAttraction;
}
```

Connascence

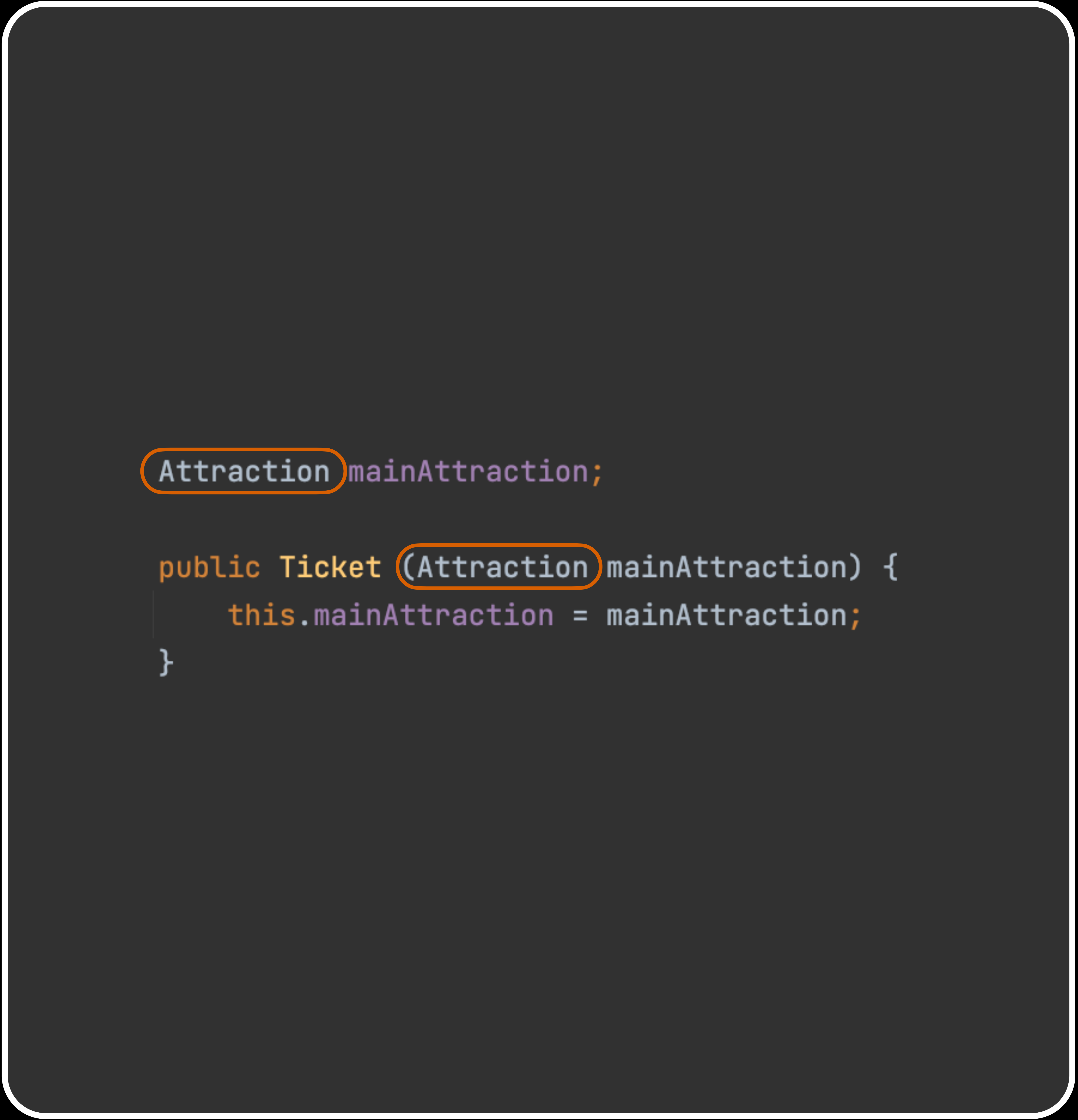
- of name



```
Attraction mainAttraction;

public Ticket (Attraction mainAttraction) {
    this.mainAttraction = mainAttraction;
}
```

Connascence

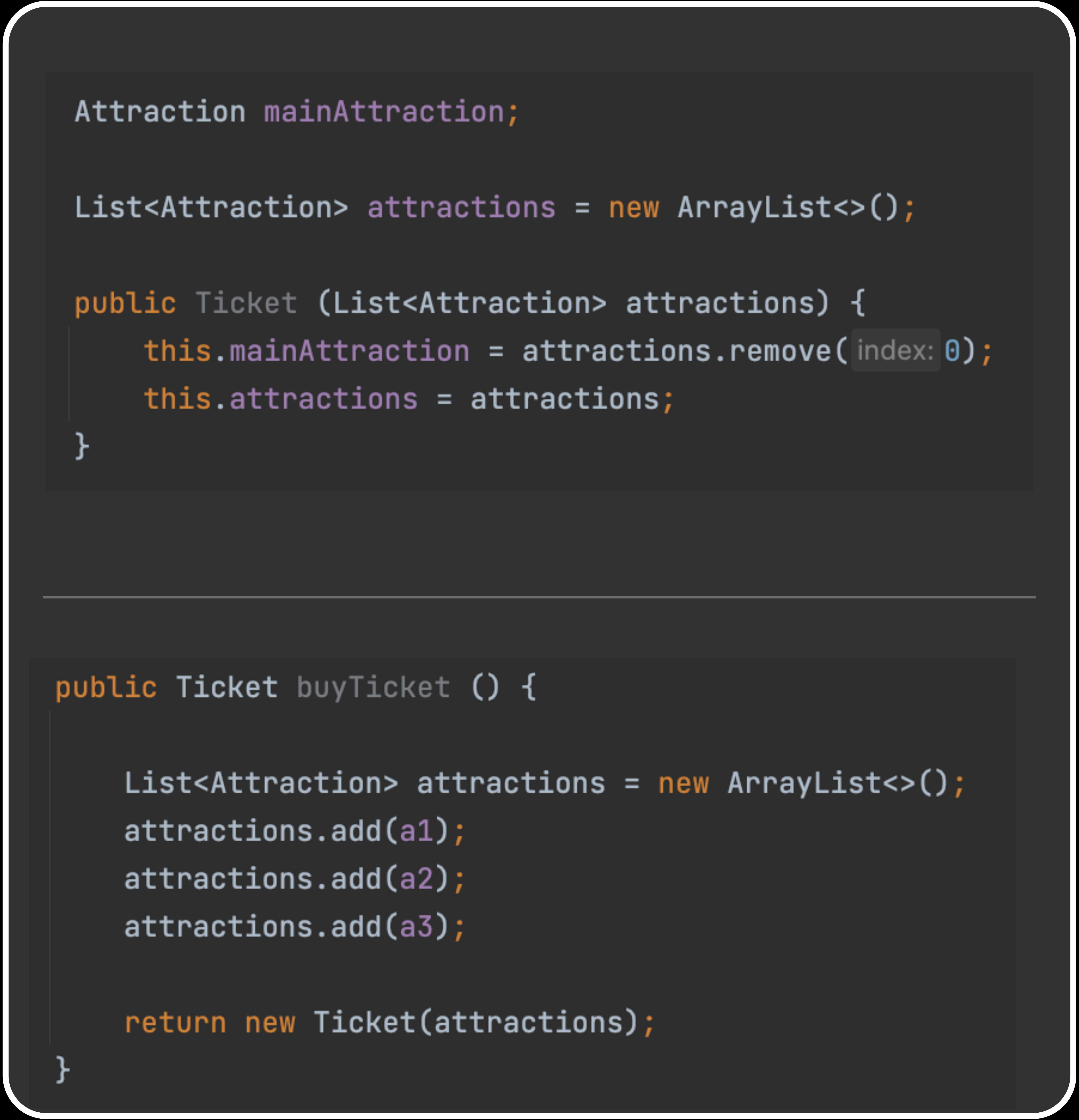
- of name
- **of type**



```
Attraction mainAttraction;

public Ticket (Attraction mainAttraction) {
    this.mainAttraction = mainAttraction;
}
```

Connascence

- of name
- of type

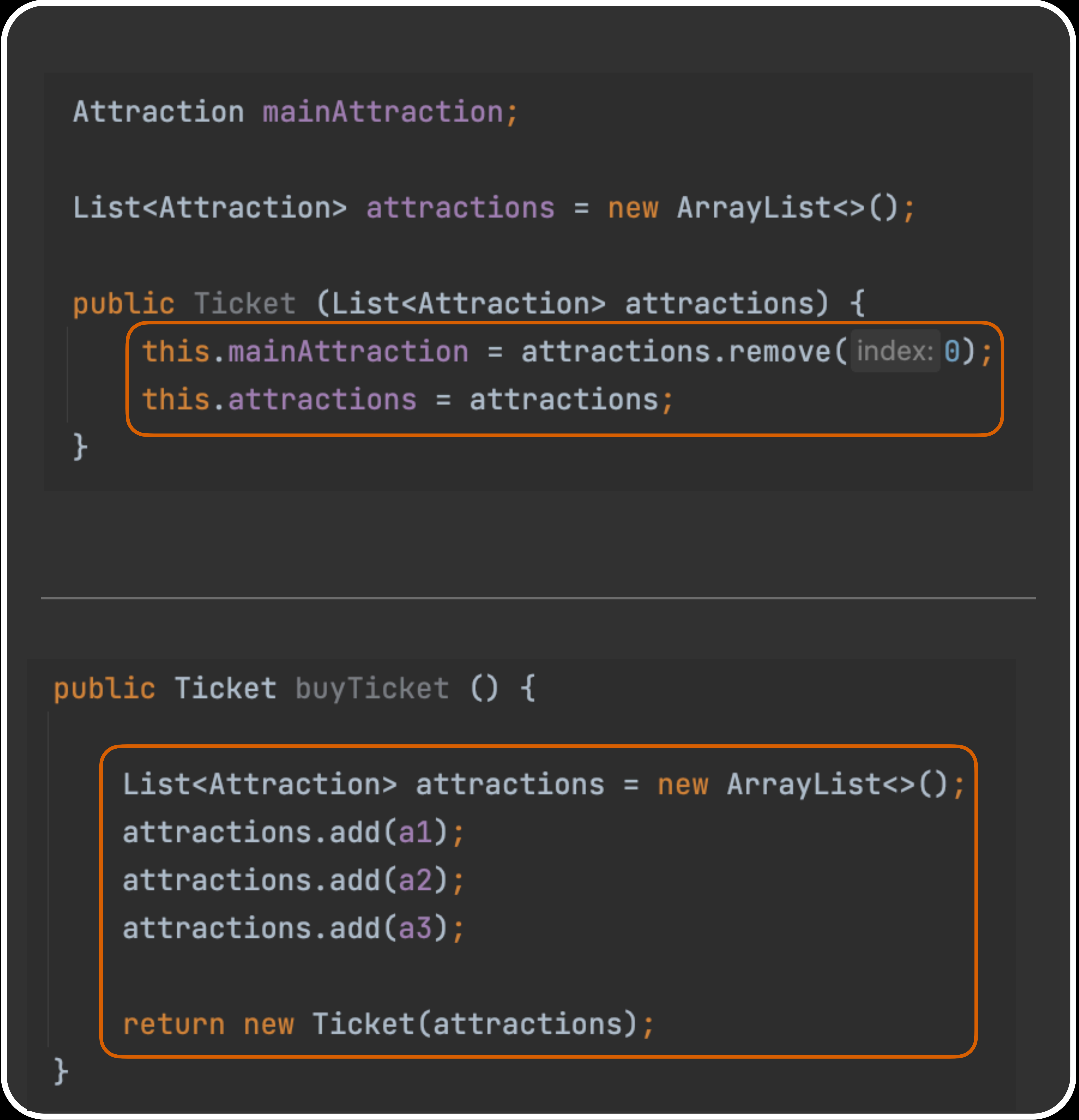


```java
Attraction mainAttraction;

List<Attraction> attractions = new ArrayList<>();

public Ticket (List<Attraction> attractions) {
    this.mainAttraction = attractions.remove( index: 0);
    this.attractions = attractions;
}
```

```java
public Ticket buyTicket () {

    List<Attraction> attractions = new ArrayList<>();
    attractions.add(a1);
    attractions.add(a2);
    attractions.add(a3);

    return new Ticket(attractions);
}
```

Connascence

- of name
- of type
- of algorithm / convention

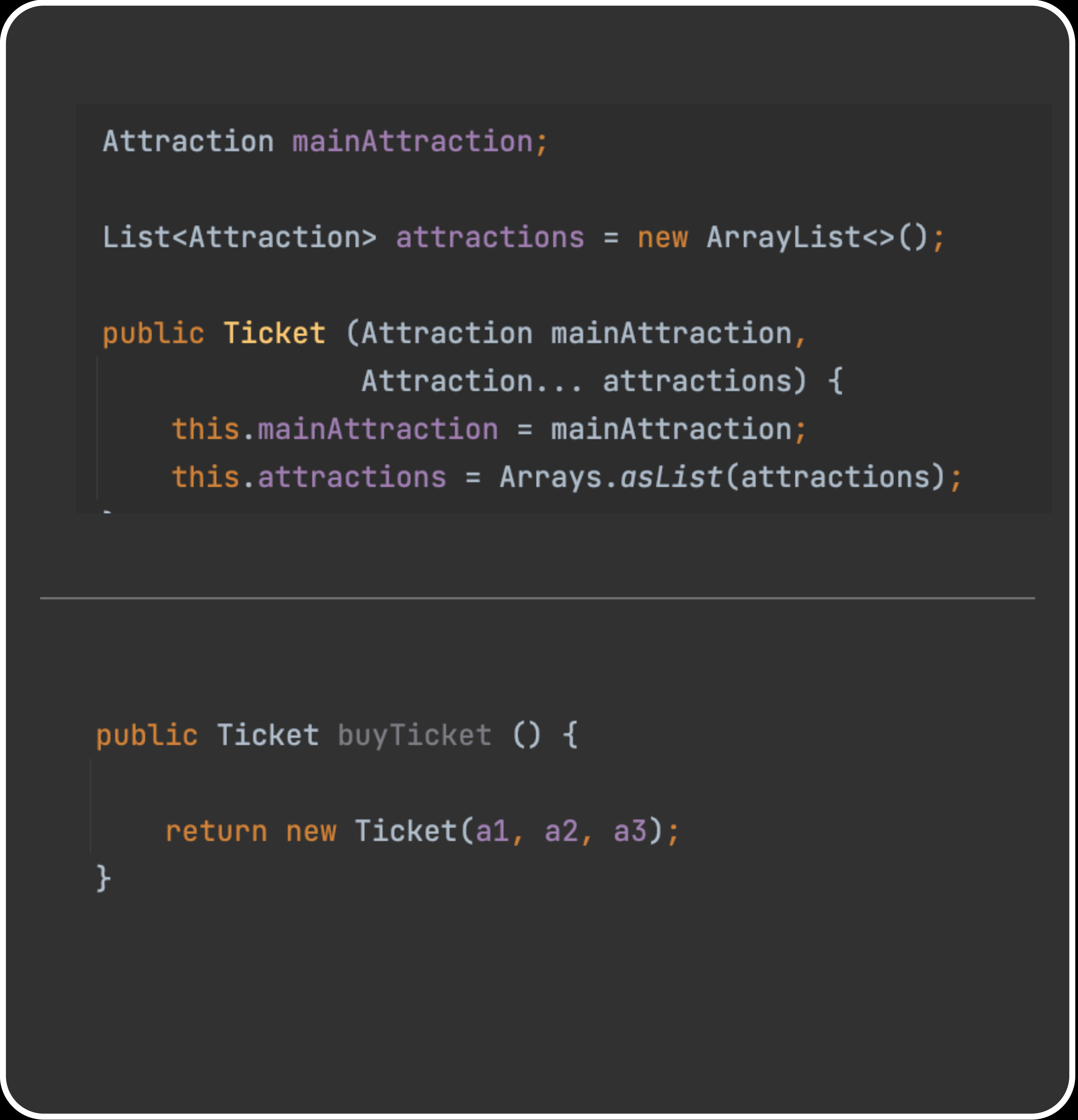


```
Attraction mainAttraction;

List<Attraction> attractions = new ArrayList<>();

public Ticket (List<Attraction> attractions) {
    this.mainAttraction = attractions.remove( index: 0);
    this.attractions = attractions;
}
```

```
public Ticket buyTicket () {

    List<Attraction> attractions = new ArrayList<>();
    attractions.add(a1);
    attractions.add(a2);
    attractions.add(a3);

    return new Ticket(attractions);
}
```

Connascence

- of name
- of type
- of algorithm / convention

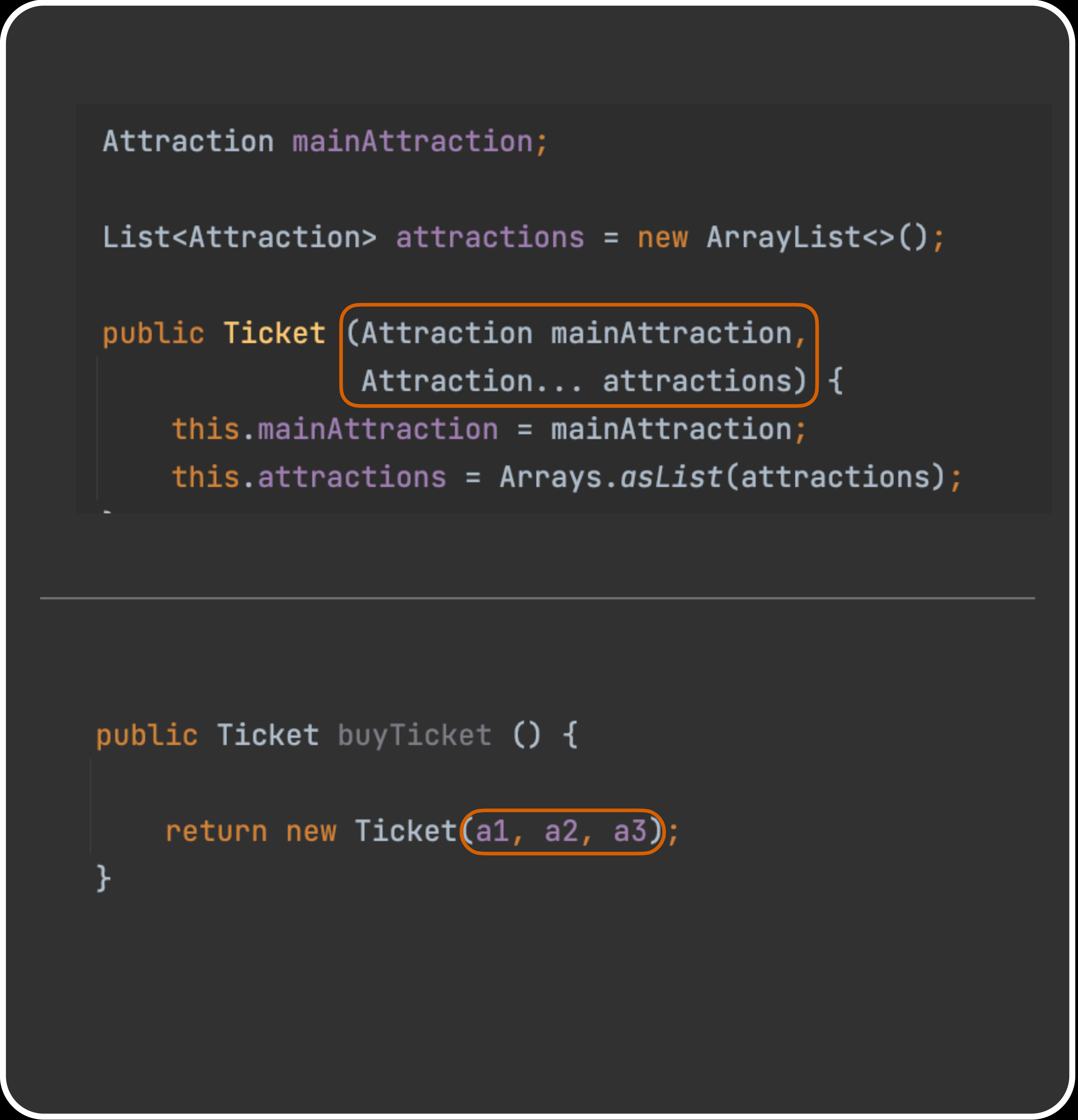


```
Attraction mainAttraction;

List<Attraction> attractions = new ArrayList<>();

public Ticket (Attraction mainAttraction,
                    Attraction... attractions) {
    this.mainAttraction = mainAttraction;
    this.attractions = Arrays.asList(attractions);
```

```
public Ticket buyTicket () {

    return new Ticket(a1, a2, a3);
}
```

Connascence

- of name
- of type
- of algorithm / convention
- **of position**



```
Attraction mainAttraction;

List<Attraction> attractions = new ArrayList<>();

public Ticket (Attraction mainAttraction,
               Attraction... attractions) {
    this.mainAttraction = mainAttraction;
    this.attractions = Arrays.asList(attractions);
```

```
public Ticket buyTicket () {

    return new Ticket(a1, a2, a3);
}
```

Connascence

- of name
- of type
- of algorithm / convention
- of position

- of execution
- of timing
- of value
- of identity
- of difference (contranascence)

Connascence

- of name
- of type
- of algorithm / convention
- of position

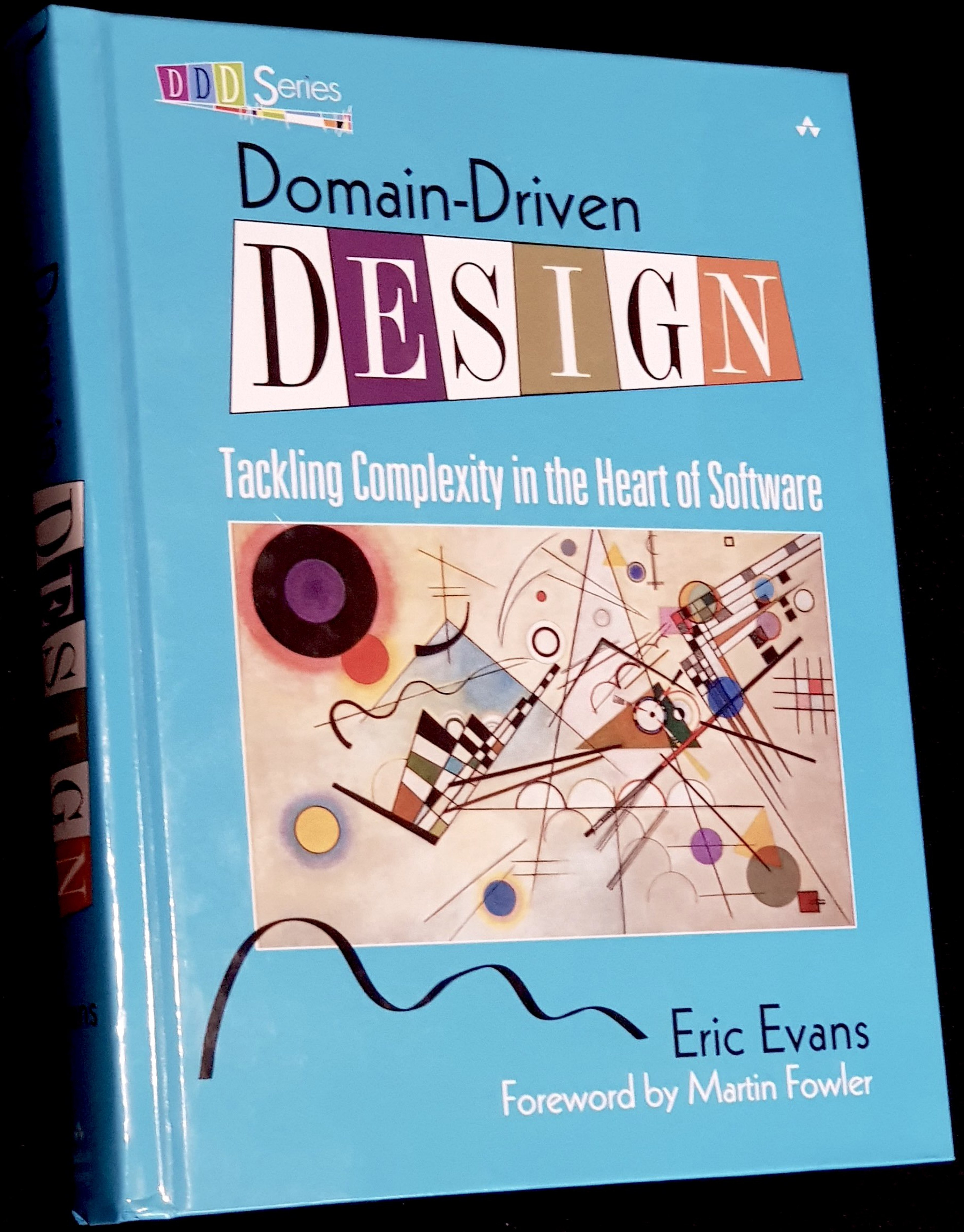
- of execution
- of timing
- of value
- of identity
- of difference (contranascence)

What's behind it? Coherence or coupling?

Aggregate

- cluster of associated objects that we treat as a unit for the purpose of data changes

- has a root and a boundary


DDD Series
Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Aggregate

| cluster of associated objects that we treat as a unit for the purpose of data changes

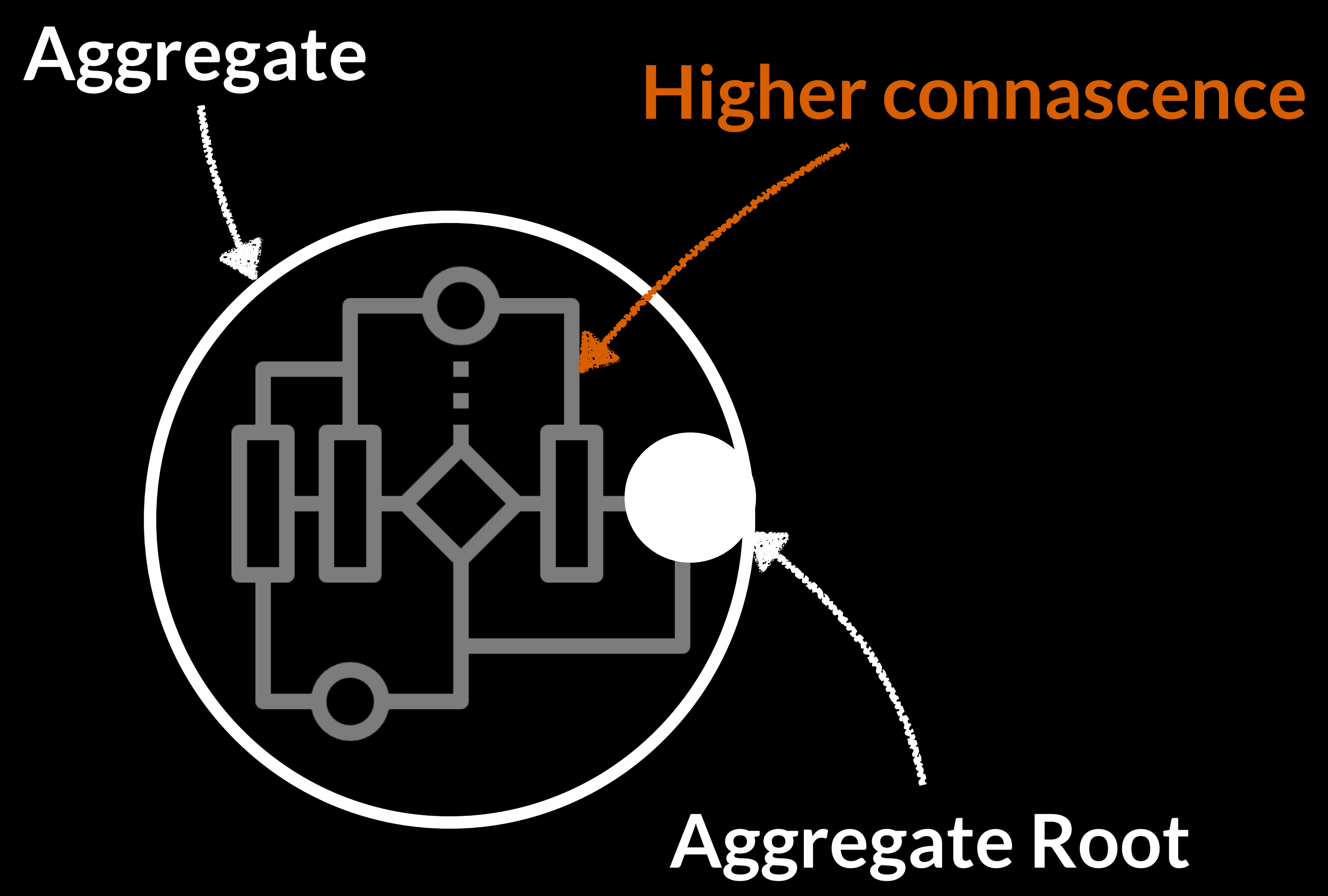
| has a root and a boundary

Coherence

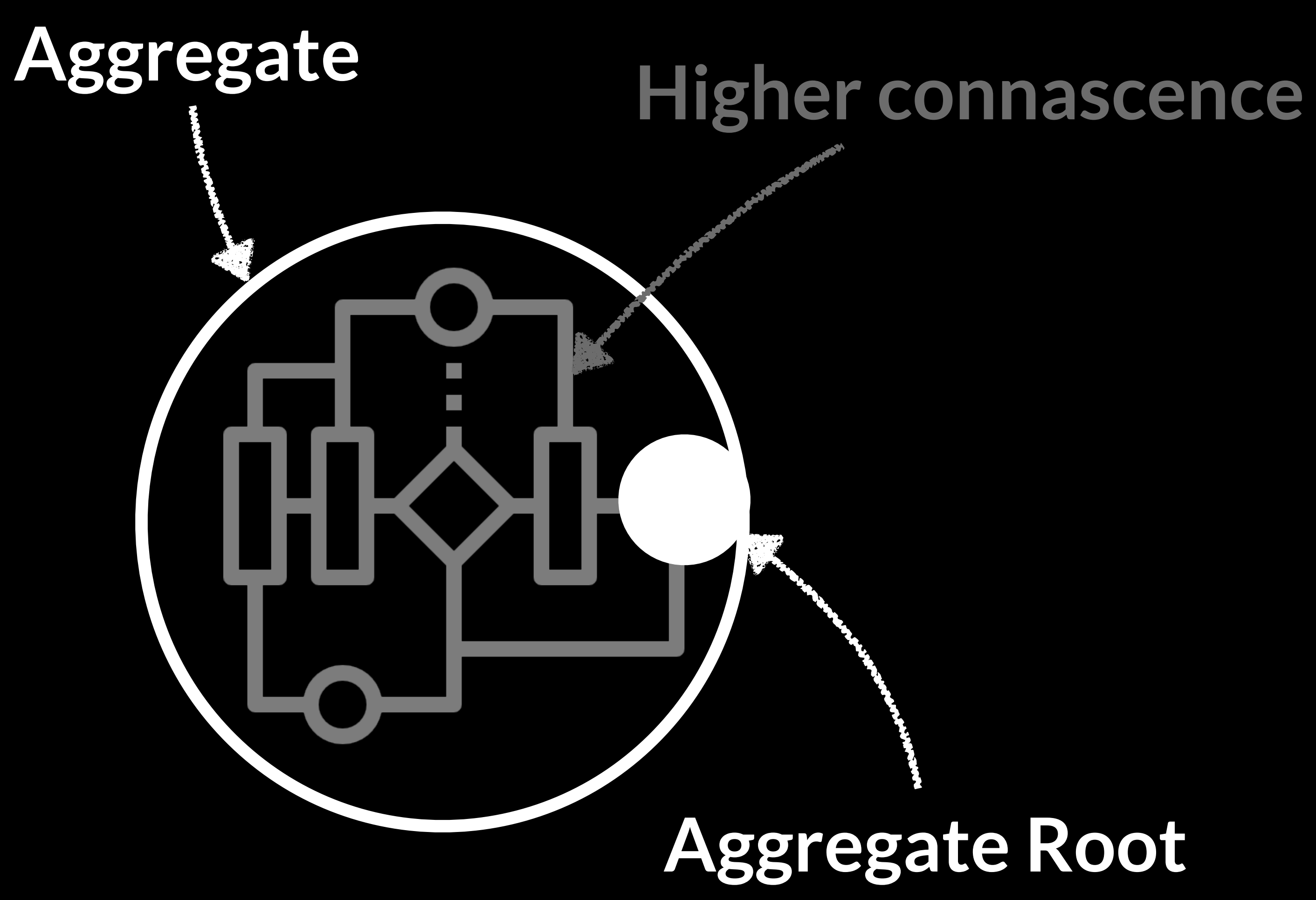
| the quality of forming a unified whole

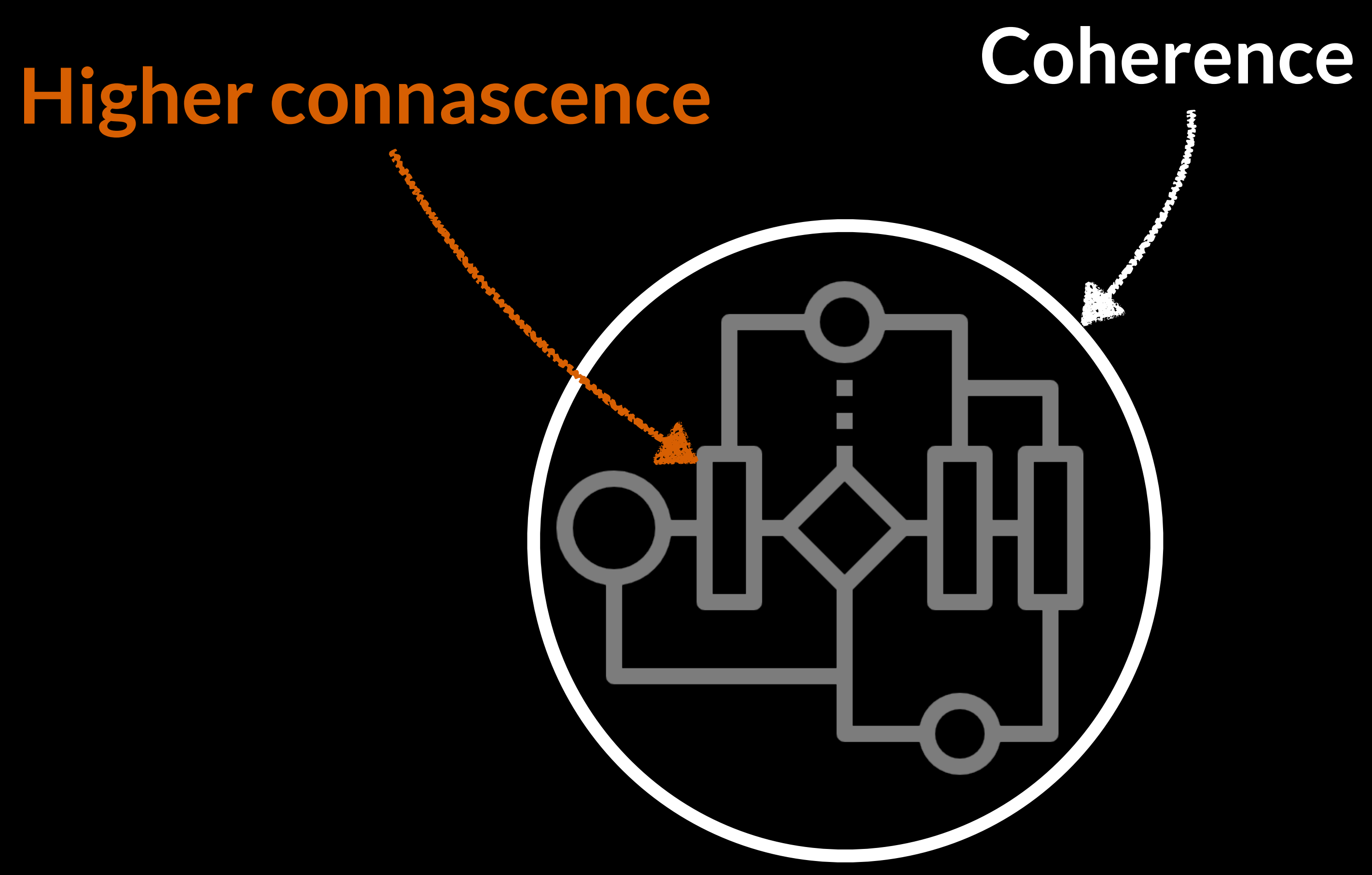
| the quality of being logical and consistent

Aggregate

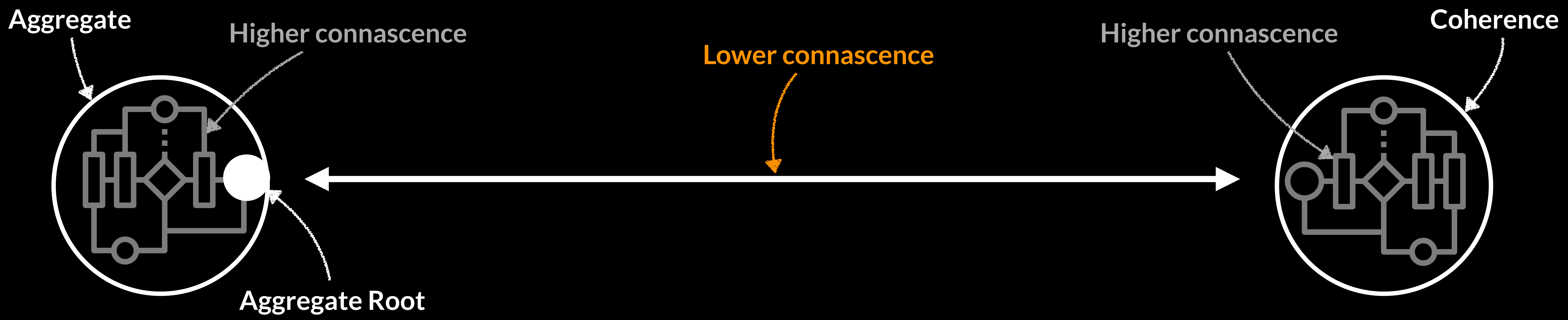


Coherence

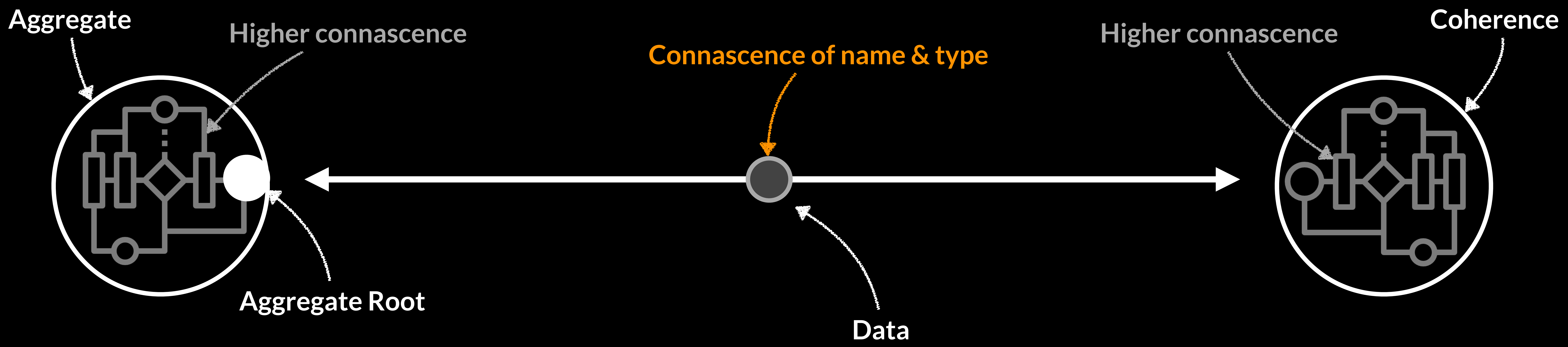

Aggregate
Higher connascence
Aggregate Root


Higher connascence
Coherence

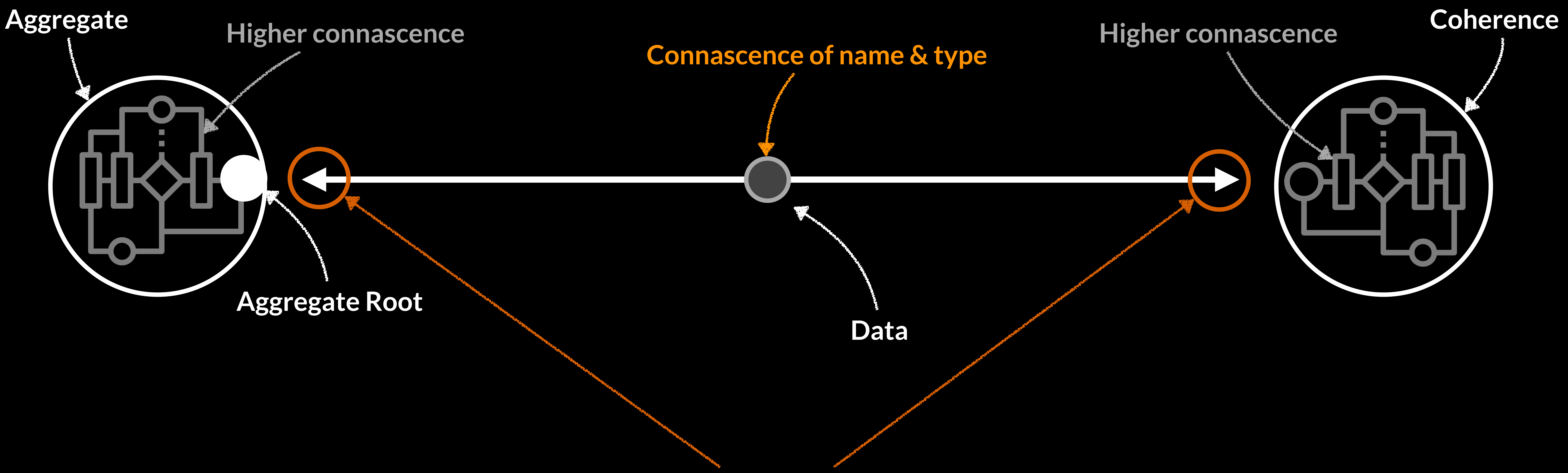
Connascence



Connascence



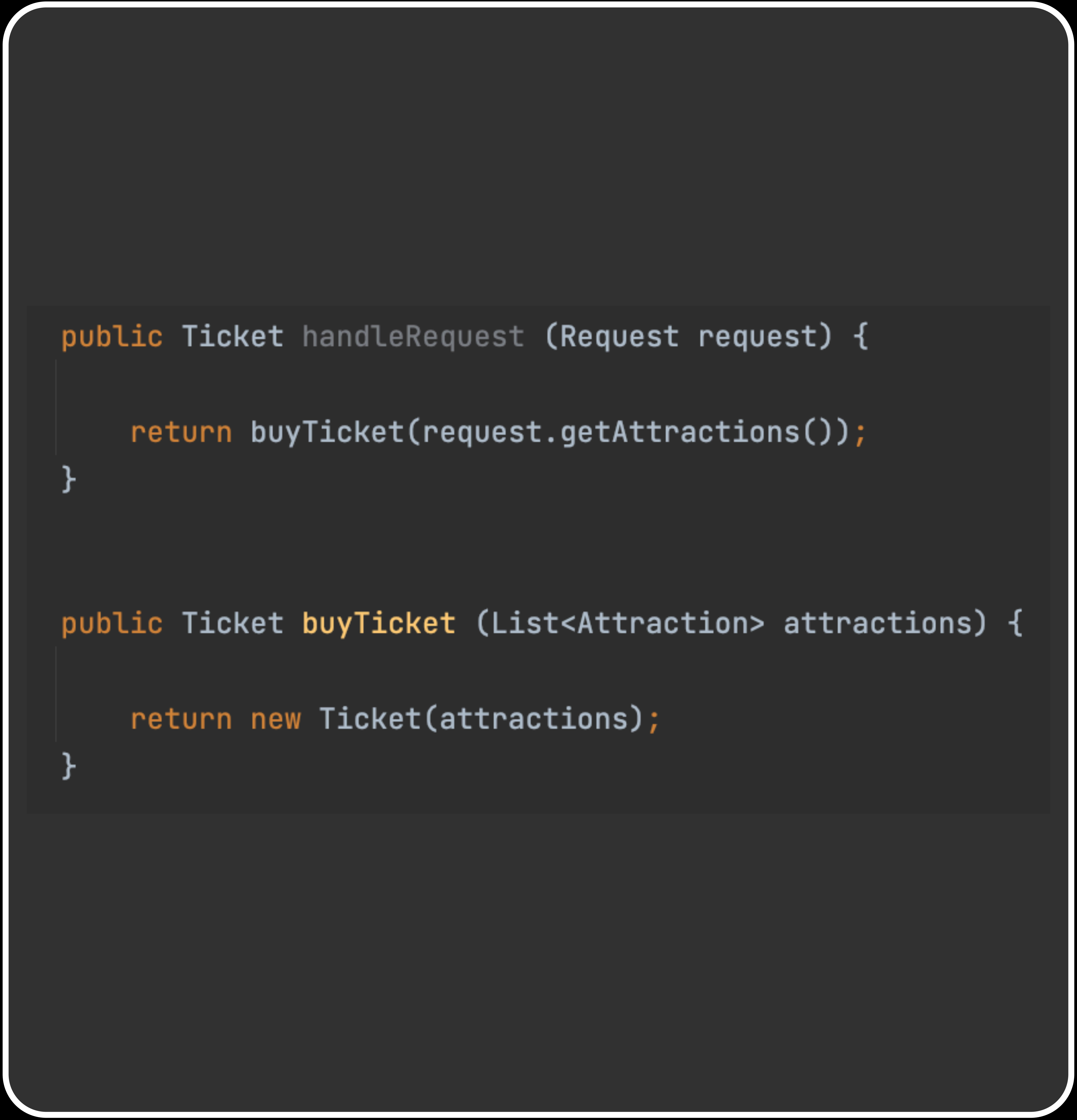
Connascence



Connascence of location?

Connascence

- of location?

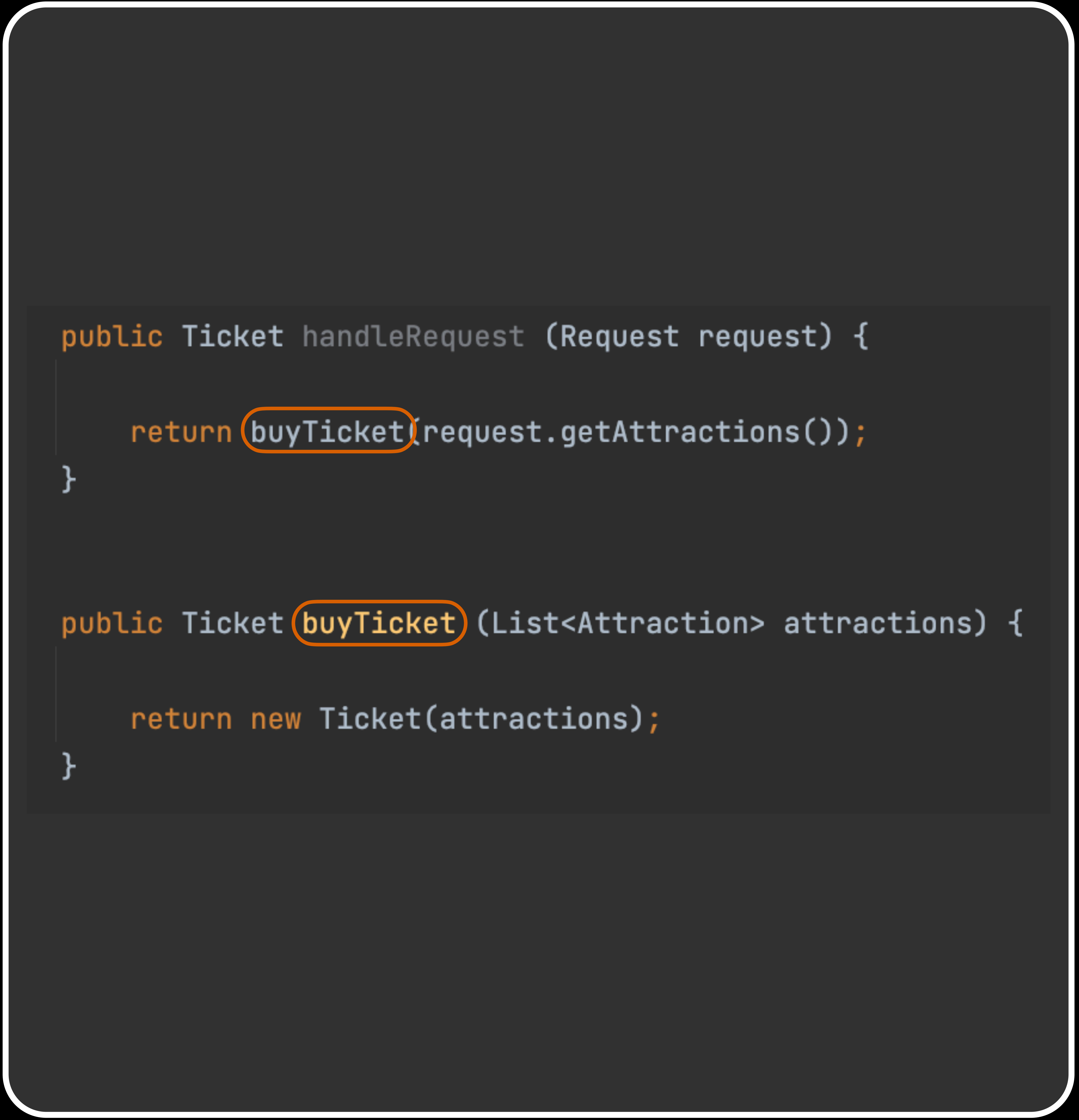


```java
public Ticket handleRequest (Request request) {

    return buyTicket(request.getAttractions());
}


public Ticket buyTicket (List<Attraction> attractions) {

    return new Ticket(attractions);
}
```

Connascence

of location?

- same class

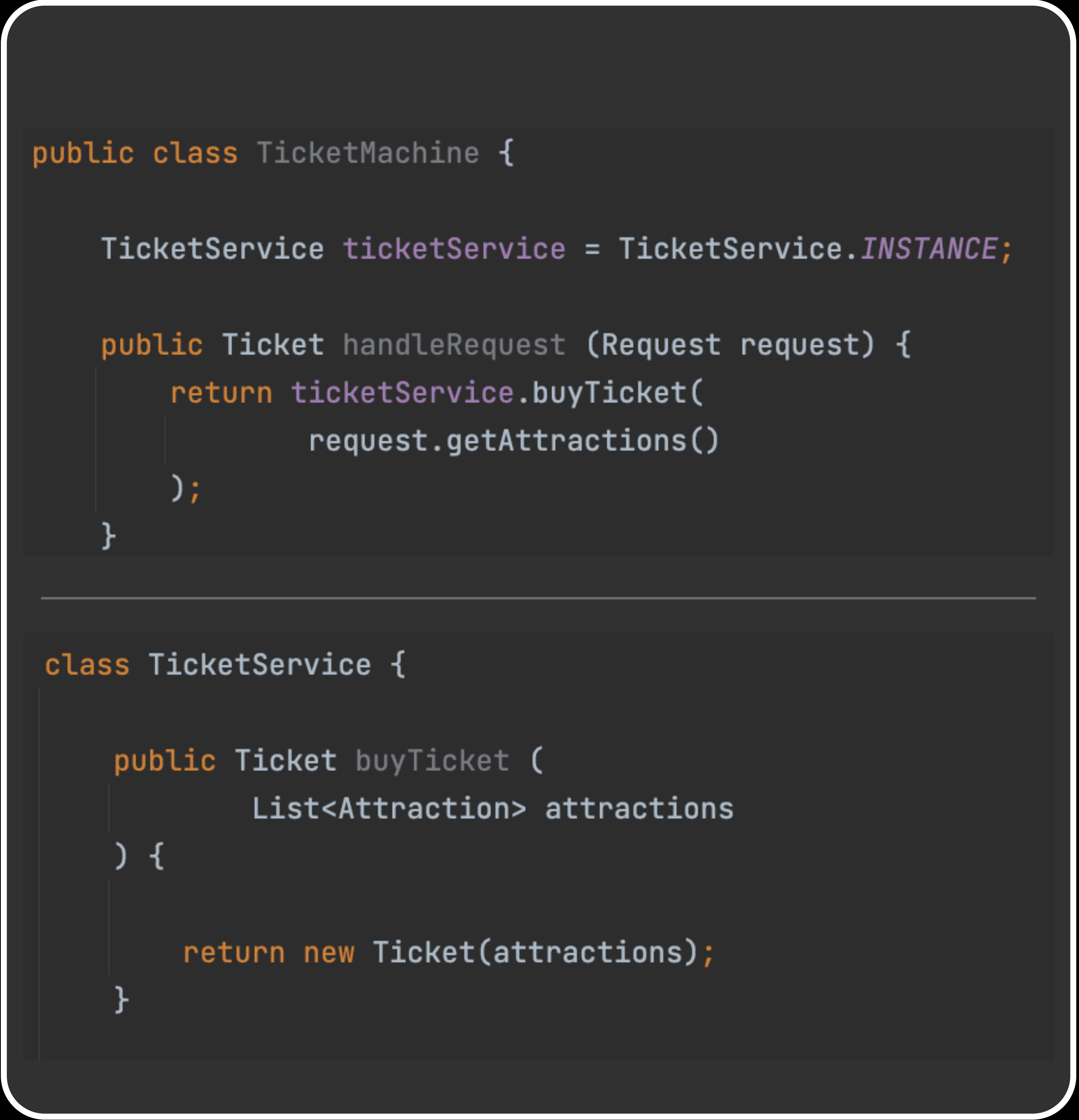


```
public Ticket handleRequest (Request request) {
    return buyTicket(request.getAttractions());
}


public Ticket buyTicket (List<Attraction> attractions) {
    return new Ticket(attractions);
}
```

Connascence

of location?

- same class



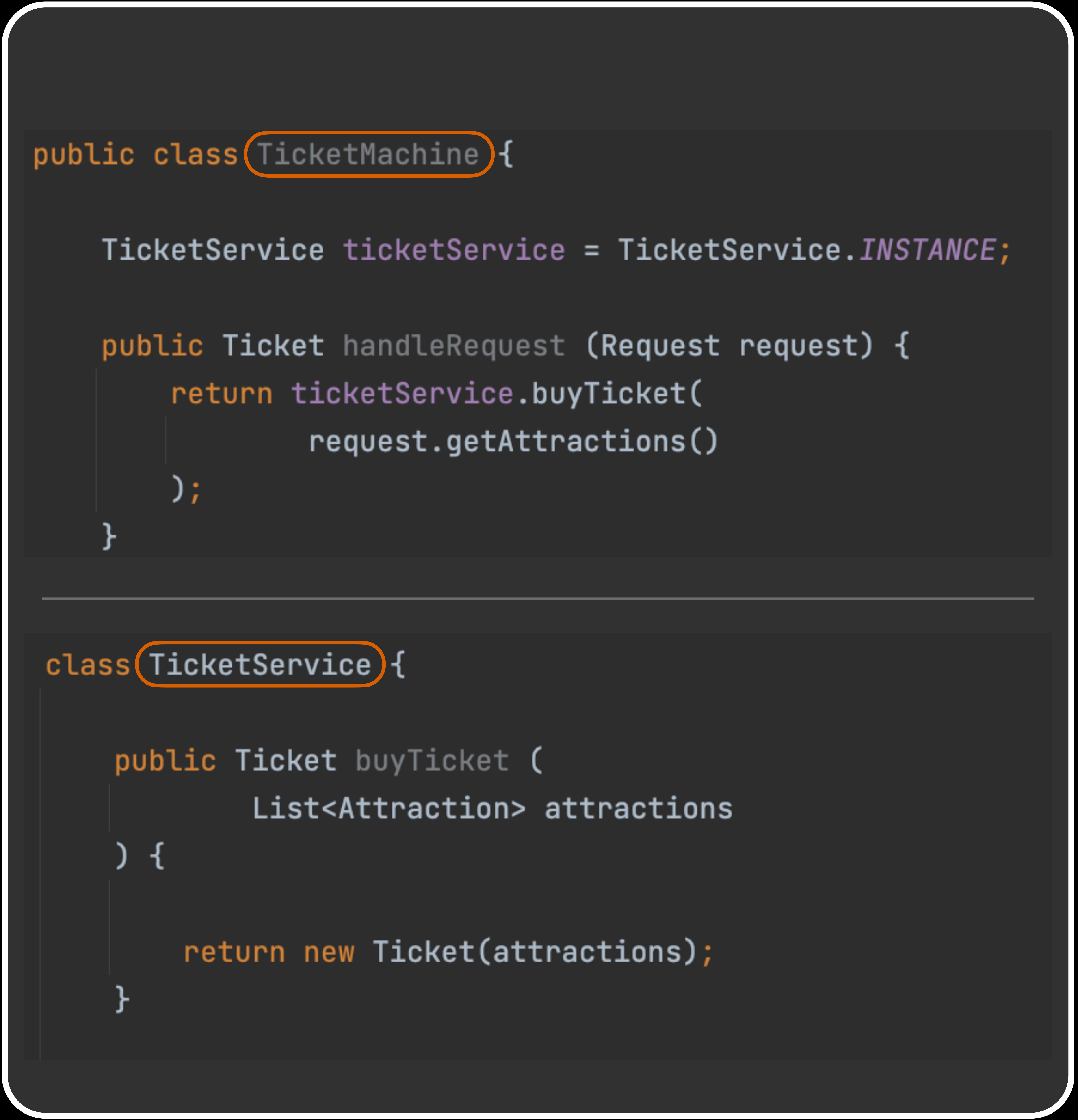
```
public class TicketMachine {

    TicketService ticketService = TicketService.INSTANCE;

    public Ticket handleRequest (Request request) {
        return ticketService.buyTicket(
                request.getAttractions()
        );
    }
```

```
class TicketService {

    public Ticket buyTicket (
            List<Attraction> attractions
    ) {

        return new Ticket(attractions);
    }
```

Connascence

of location?

- same class
- same package



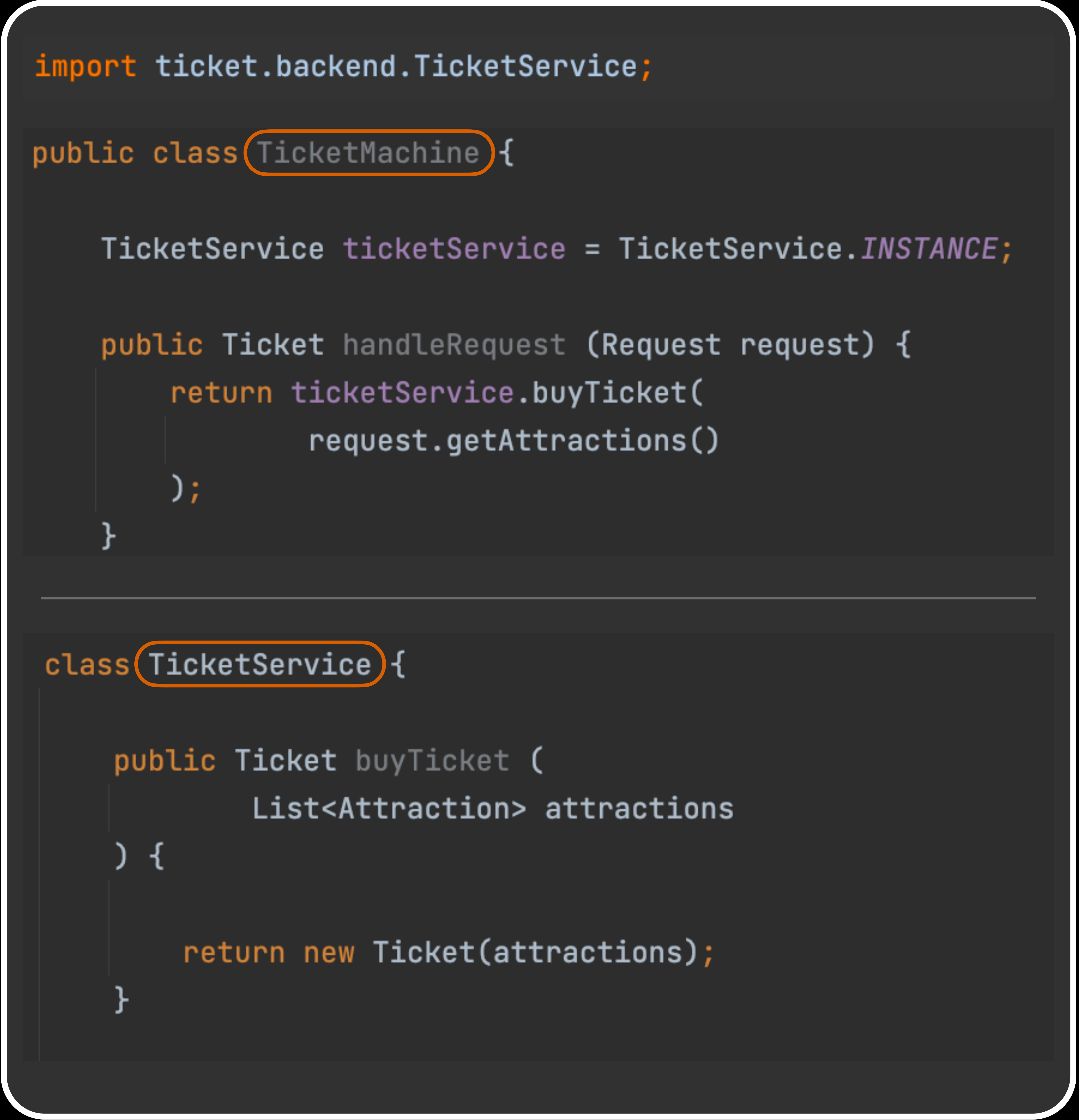
```
public class TicketMachine {

    TicketService ticketService = TicketService.INSTANCE;

    public Ticket handleRequest (Request request) {
        return ticketService.buyTicket(
                request.getAttractions()
        );
    }
```

```
class TicketService {

    public Ticket buyTicket (
            List<Attraction> attractions
    ) {

        return new Ticket(attractions);
    }
```

Connascence

of location?

- same class
- same package
- same classpath / module path



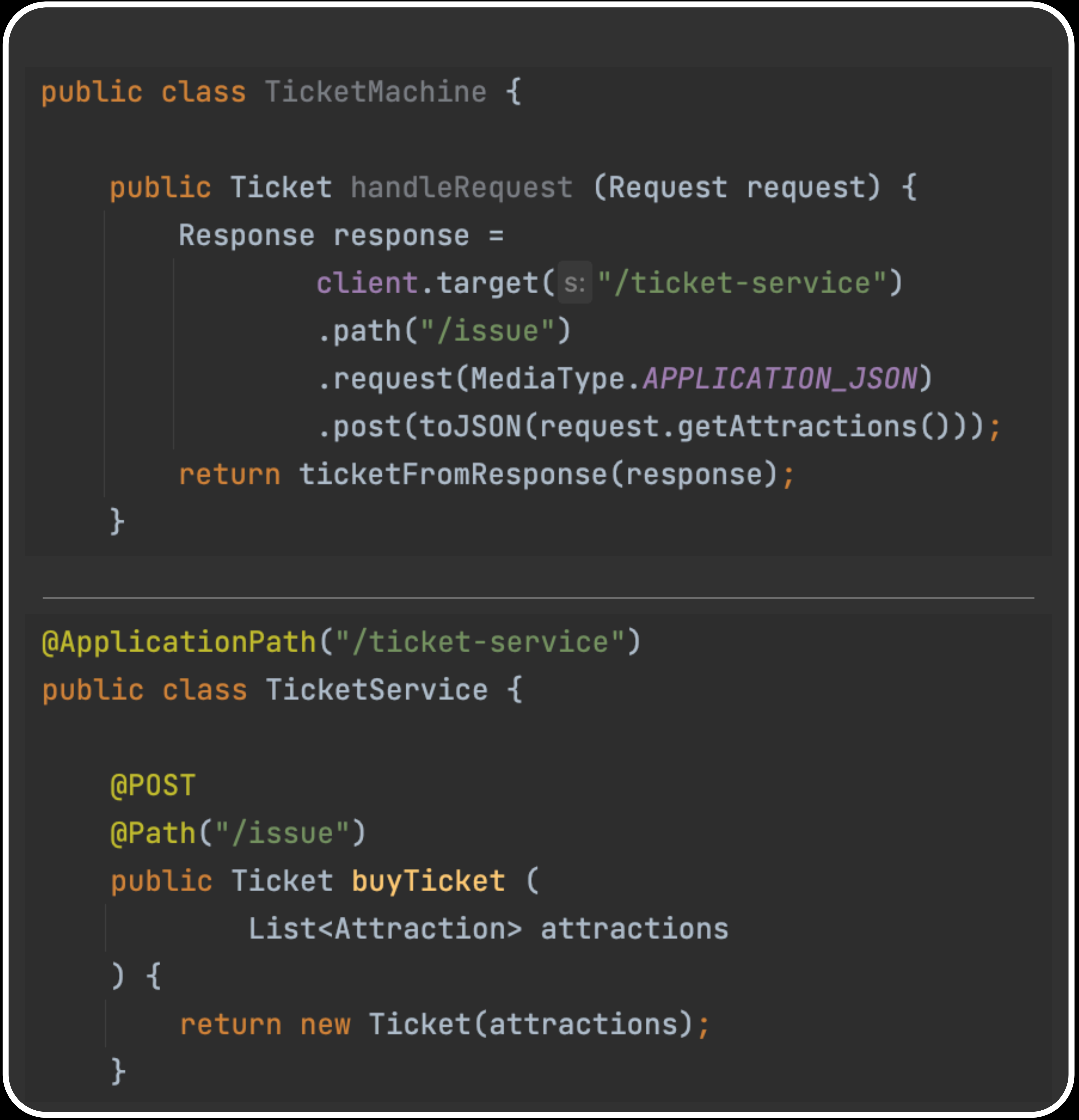
```
import ticket.backend.TicketService;

public class TicketMachine {

    TicketService ticketService = TicketService.INSTANCE;

    public Ticket handleRequest (Request request) {
        return ticketService.buyTicket(
                request.getAttractions()
        );
    }
```

```
class TicketService {

    public Ticket buyTicket (
            List<Attraction> attractions
    ) {

        return new Ticket(attractions);
    }
```

Connascence

of location?

- same class
- same package
- same classpath / module path



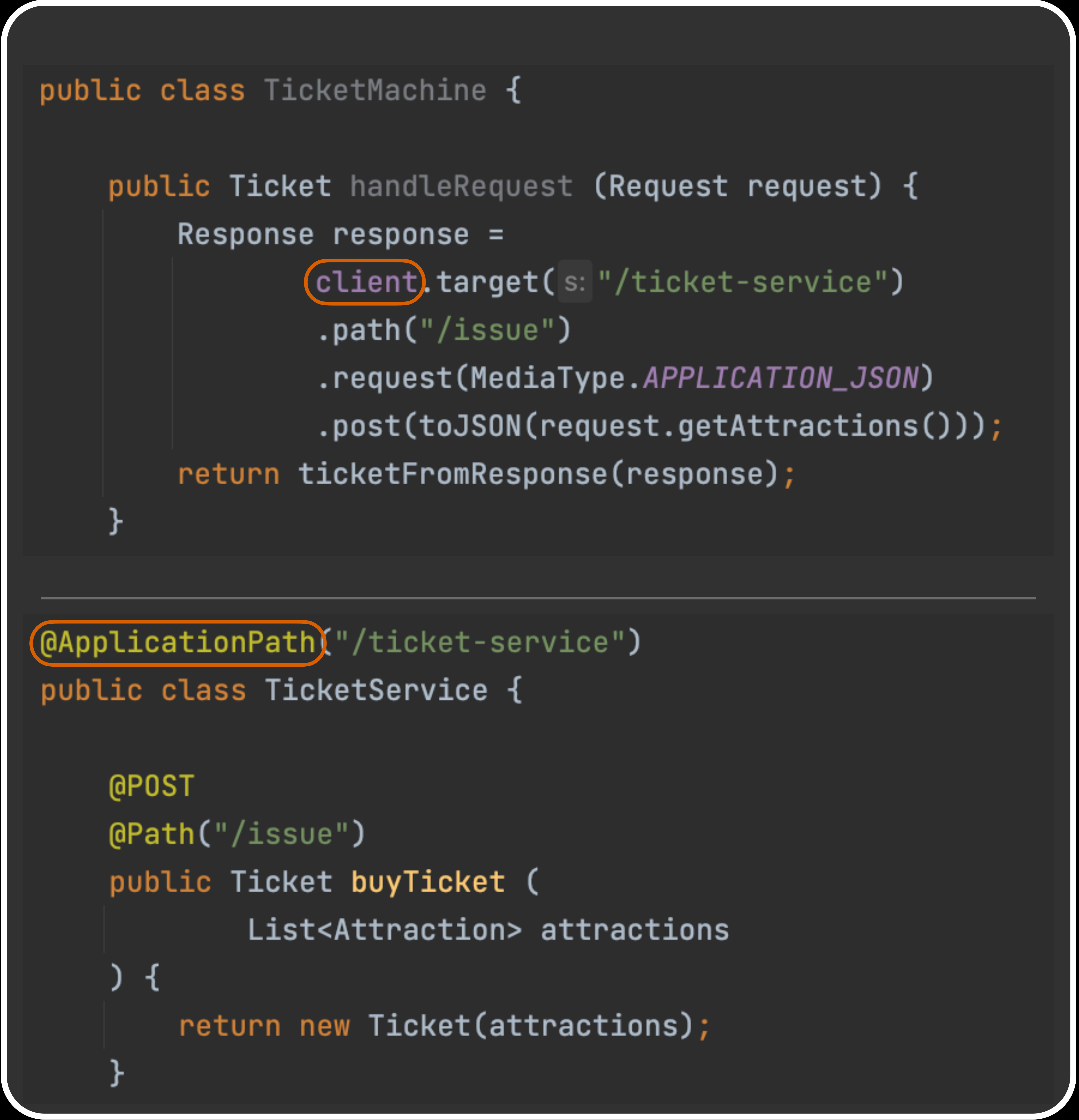
```
public class TicketMachine {

    public Ticket handleRequest (Request request) {
        Response response =
                client.target( s: "/ticket-service")
                .path("/issue")
                .request(MediaType.APPLICATION_JSON)
                .post(toJSON(request.getAttractions()));
        return ticketFromResponse(response);
    }
```

```
@ApplicationPath("/ticket-service")
public class TicketService {

    @POST
    @Path("/issue")
    public Ticket buyTicket (
            List<Attraction> attractions
    ) {
        return new Ticket(attractions);
    }
```

Connascence

of location?

- same class
- same package
- same classpath / module path
- same configuration



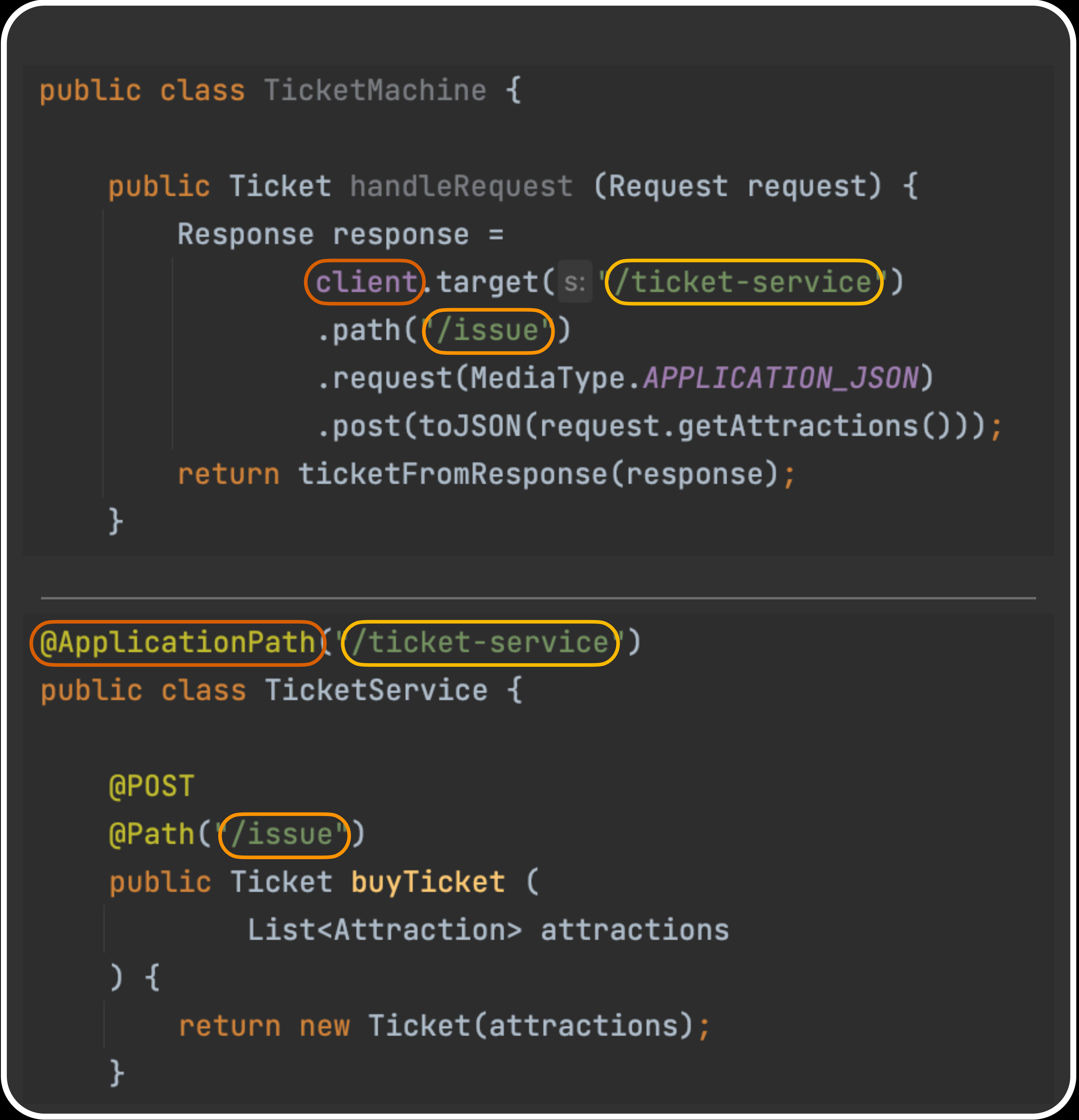
```java
public class TicketMachine {

    public Ticket handleRequest (Request request) {
        Response response =
                client.target( s: "/ticket-service")
                .path("/issue")
                .request(MediaType.APPLICATION_JSON)
                .post(toJSON(request.getAttractions()));
        return ticketFromResponse(response);
    }
```

```java
@ApplicationPath("/ticket-service")
public class TicketService {

    @POST
    @Path("/issue")
    public Ticket buyTicket (
            List<Attraction> attractions
    ) {
        return new Ticket(attractions);
    }
```

Connascence

of location?

- same class
- same package
- same classpath / module path
- same configuration



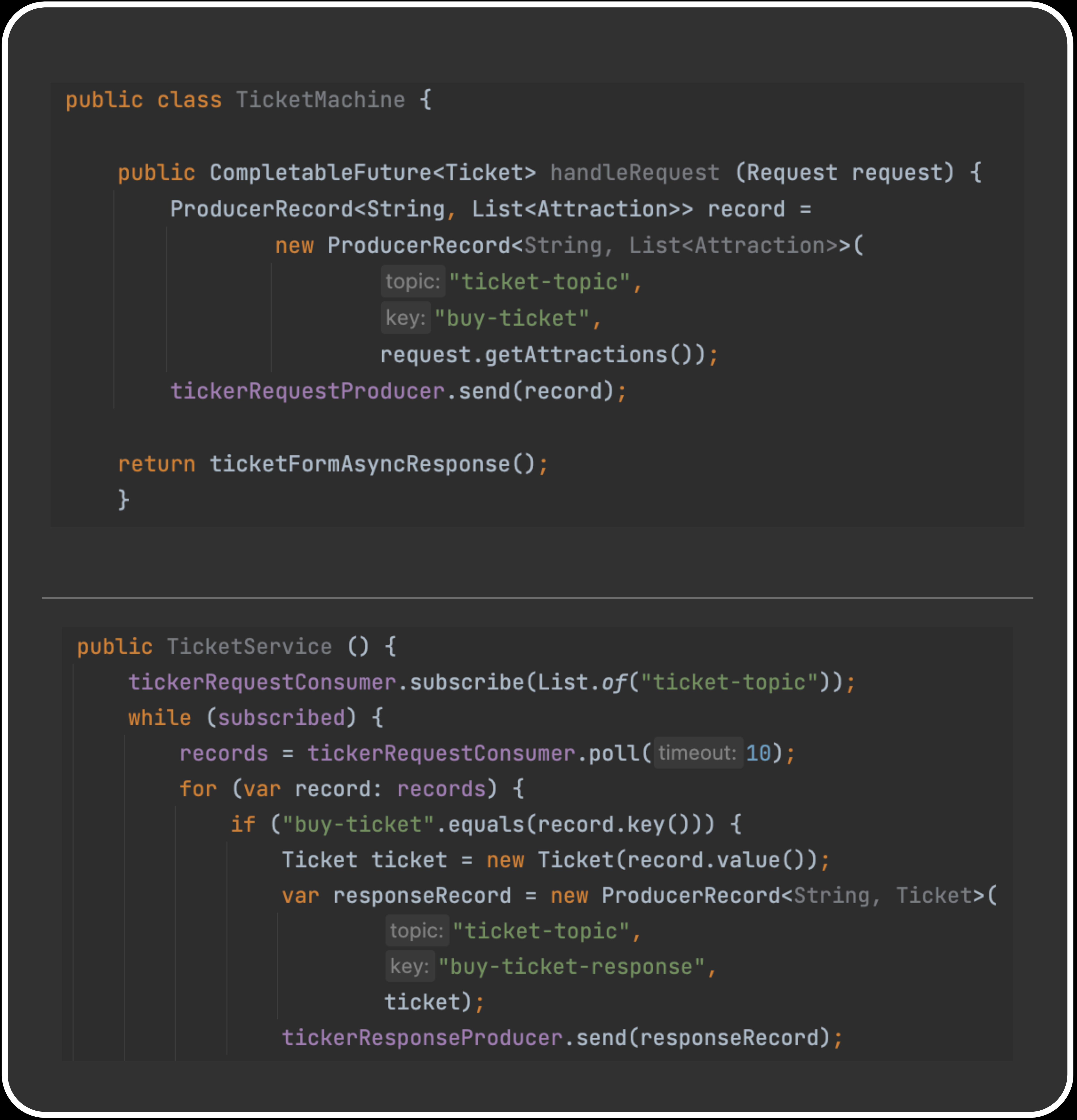
```java
public class TicketMachine {

    public Ticket handleRequest (Request request) {
        Response response =
                client.target( s: "/ticket-service")
                    .path("/issue")
                    .request(MediaType.APPLICATION_JSON)
                    .post(toJSON(request.getAttractions()));
        return ticketFromResponse(response);
    }
```

```java
@ApplicationPath("/ticket-service")
public class TicketService {

    @POST
    @Path("/issue")
    public Ticket buyTicket (
            List<Attraction> attractions
    ) {
        return new Ticket(attractions);
    }
```

Connascence

of location?

- same class
- same package
- same classpath / module path
- same configuration



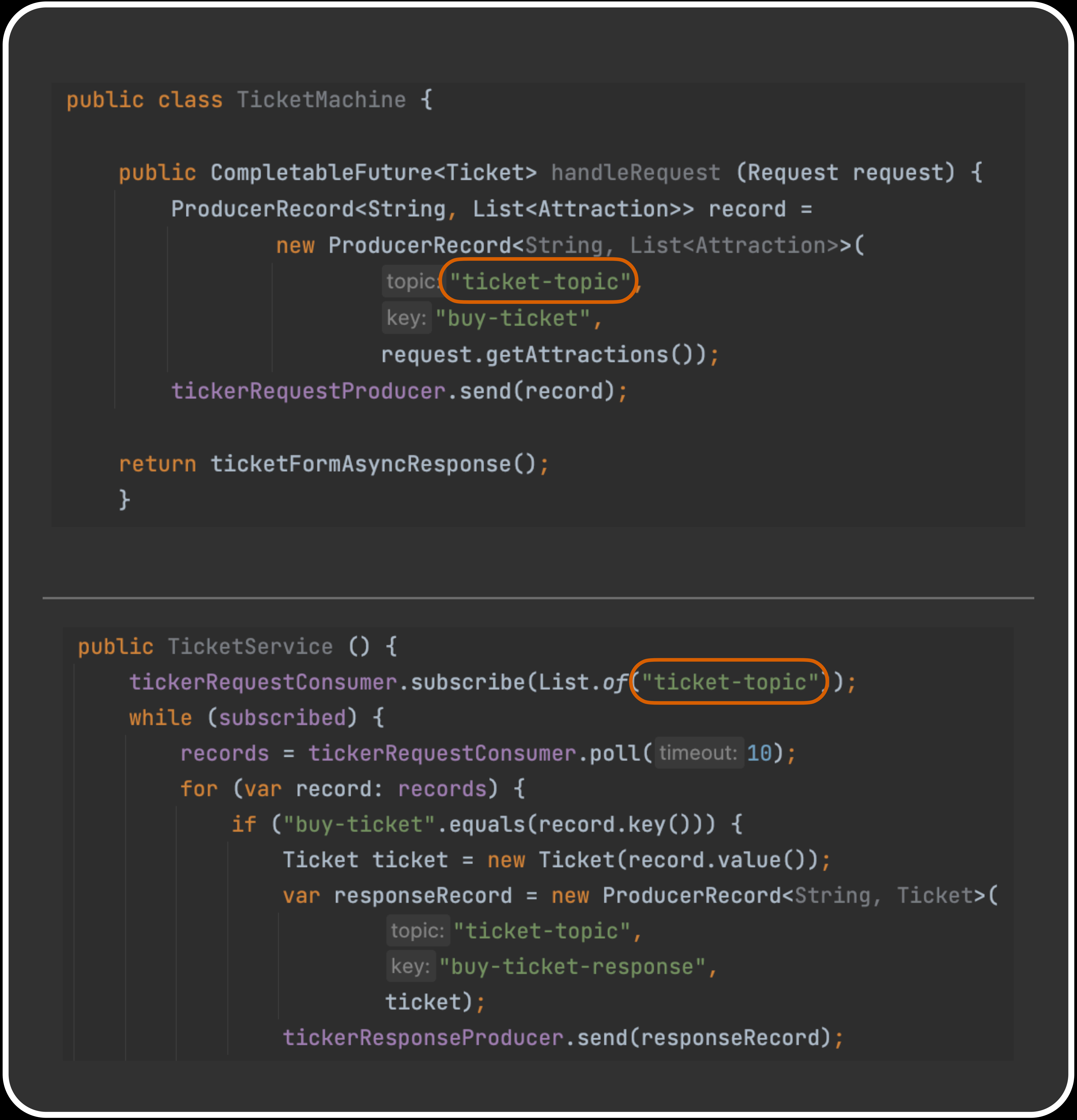
```java
public class TicketMachine {

    public CompletableFuture<Ticket> handleRequest (Request request) {
        ProducerRecord<String, List<Attraction>> record =
                new ProducerRecord<String, List<Attraction>>(
                        topic: "ticket-topic",
                        key: "buy-ticket",
                        request.getAttractions());
        tickerRequestProducer.send(record);

    return ticketFormAsyncResponse();
    }
```

```java
public TicketService () {
    tickerRequestConsumer.subscribe(List.of("ticket-topic"));
    while (subscribed) {
        records = tickerRequestConsumer.poll( timeout: 10);
        for (var record: records) {
            if ("buy-ticket".equals(record.key())) {
                Ticket ticket = new Ticket(record.value());
                var responseRecord = new ProducerRecord<String, Ticket>(
                        topic: "ticket-topic",
                        key: "buy-ticket-response",
                        ticket);
                tickerResponseProducer.send(responseRecord);
```

Connascence

of location?

- same class
- same package
- same classpath / module path
- same configuration
- same stream/topic



```java
public class TicketMachine {

    public CompletableFuture<Ticket> handleRequest (Request request) {
        ProducerRecord<String, List<Attraction>> record =
                new ProducerRecord<String, List<Attraction>>(
                        topic: "ticket-topic",
                        key: "buy-ticket",
                        request.getAttractions());
        tickerRequestProducer.send(record);

    return ticketFormAsyncResponse();
    }
```

```java
public TicketService () {
    tickerRequestConsumer.subscribe(List.of("ticket-topic"));
    while (subscribed) {
        records = tickerRequestConsumer.poll( timeout: 10);
        for (var record: records) {
            if ("buy-ticket".equals(record.key())) {
                Ticket ticket = new Ticket(record.value());
                var responseRecord = new ProducerRecord<String, Ticket>(
                        topic: "ticket-topic",
                        key: "buy-ticket-response",
                        ticket);
                tickerResponseProducer.send(responseRecord);
```

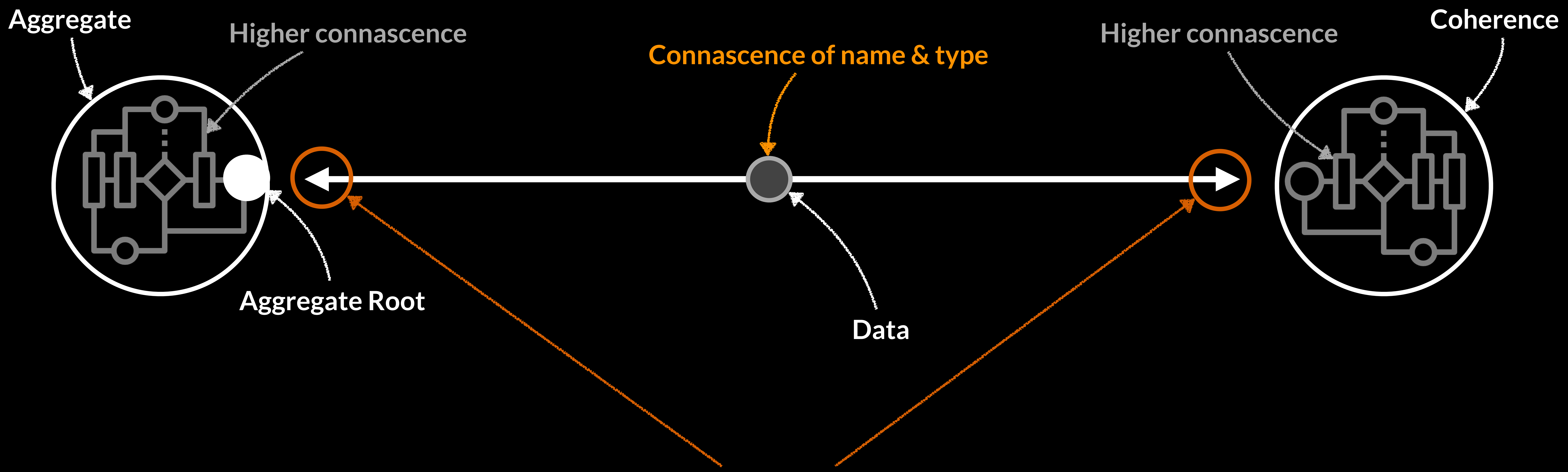
Connascence

of location?

- same class
- same package
- same classpath / module path
- same configuration
- same stream/topic

Location awareness

Connascence



Connascence of location?

Messaging

Messaging

Messaging

COMMAND ●

EVENT ●

QUERY ●

Messaging

COMMAND ● Route to single handler

Provides confirmation/result

EVENT ●

QUERY ●

Messaging

COMMAND	●	Route to single handler Provides confirmation/result
EVENT	●	Distribute to all logical handlers No results
QUERY	●	

Messaging

COMMAND	●	Route to single handler Provides confirmation/result
EVENT	●	Distribute to all logical handlers No results
QUERY	●	Route with load balancing Provides result

Messaging

COMMAND	●	Route to **single** handler Provides confirmation/result
EVENT	●	Distribute to **all** logical handlers No results
QUERY	●	Route to **one or many** handlers Provides merged result

Messaging

COMMAND	●	Route to single handler Provides **confirmation**/result
EVENT	●	Distribute to all logical handlers **No results**
QUERY	●	Route to one or many handlers Provides **merged result**

Messaging

COMMAND ●

EVENT ●

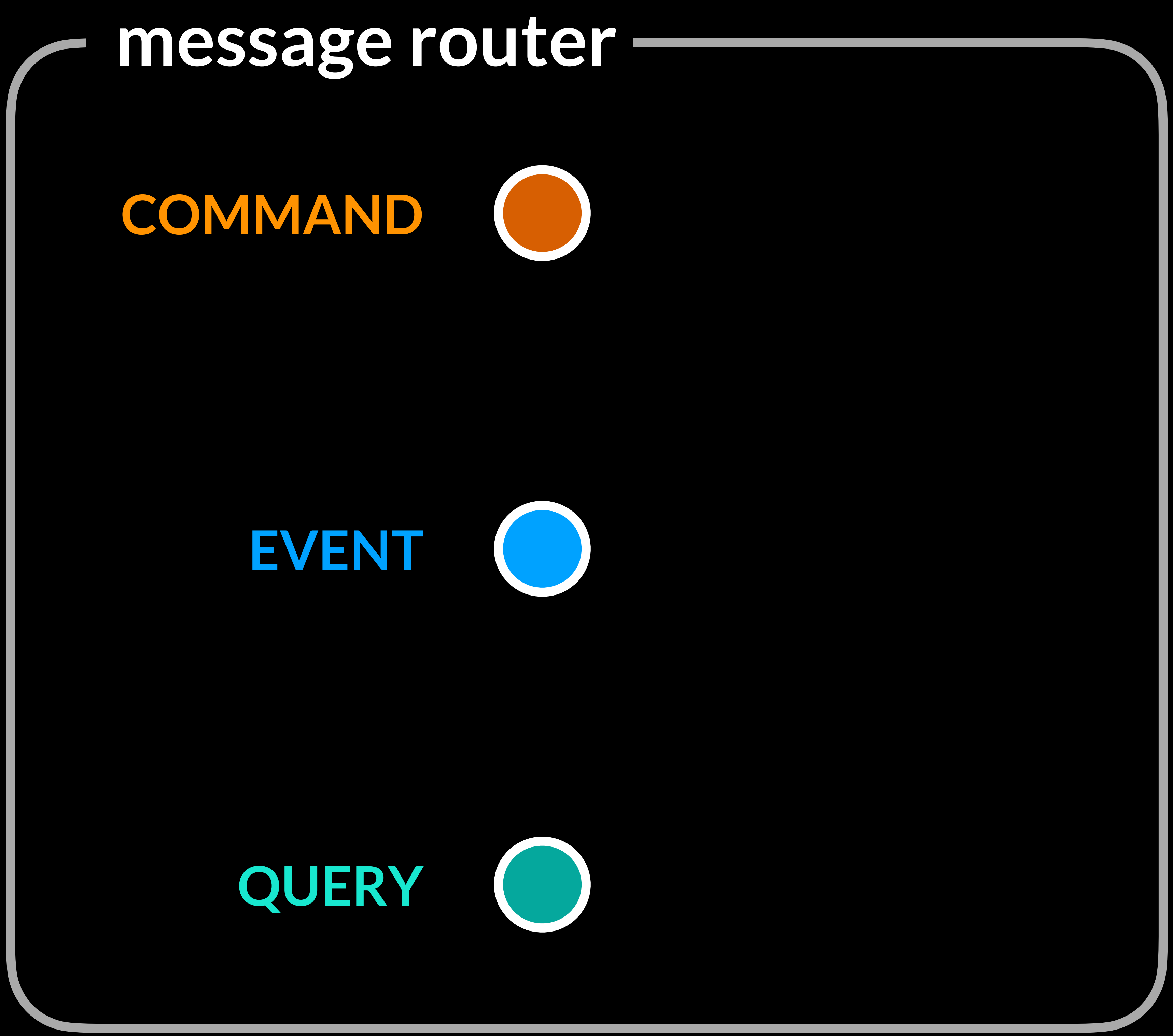
QUERY ●

```
public class IssueTicketCommand {
    AttractionId mainAttractionId;
    List<AttractionId> attractions = new ArrayList<>();
}
```

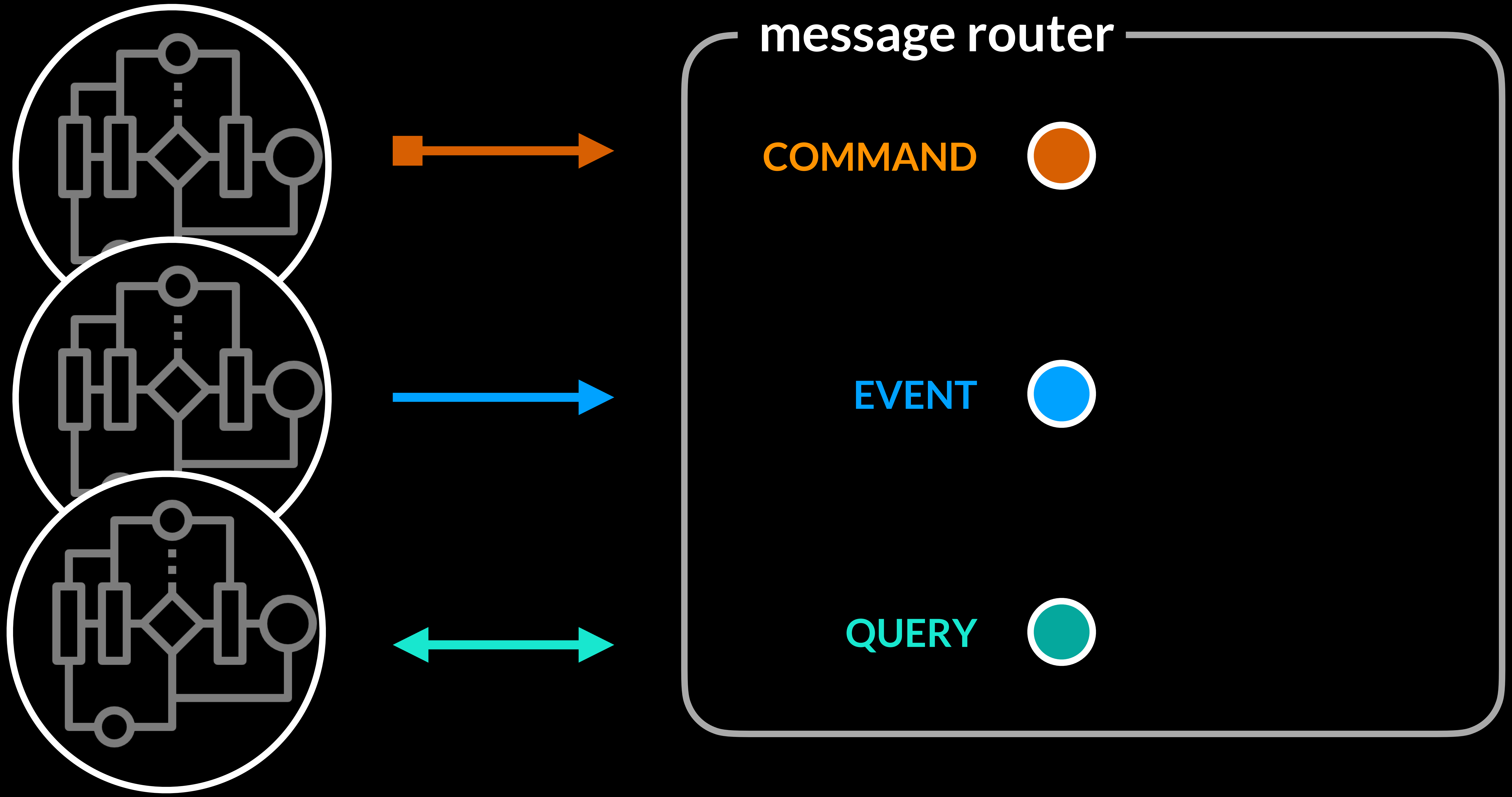
```
public class TicketIssuedEvent {
    TicketId ticketId;
    AttractionId mainAttractionId;
    List<AttractionId> attractions = new ArrayList<>();
}
```

```
public class TicketAttractionsQuery {
    TicketId ticketId;
}


public class TicketAttractionsResponse {
    AttractionId mainAttractionId;
    List<AttractionId> attractions = new ArrayList<>();
}
```

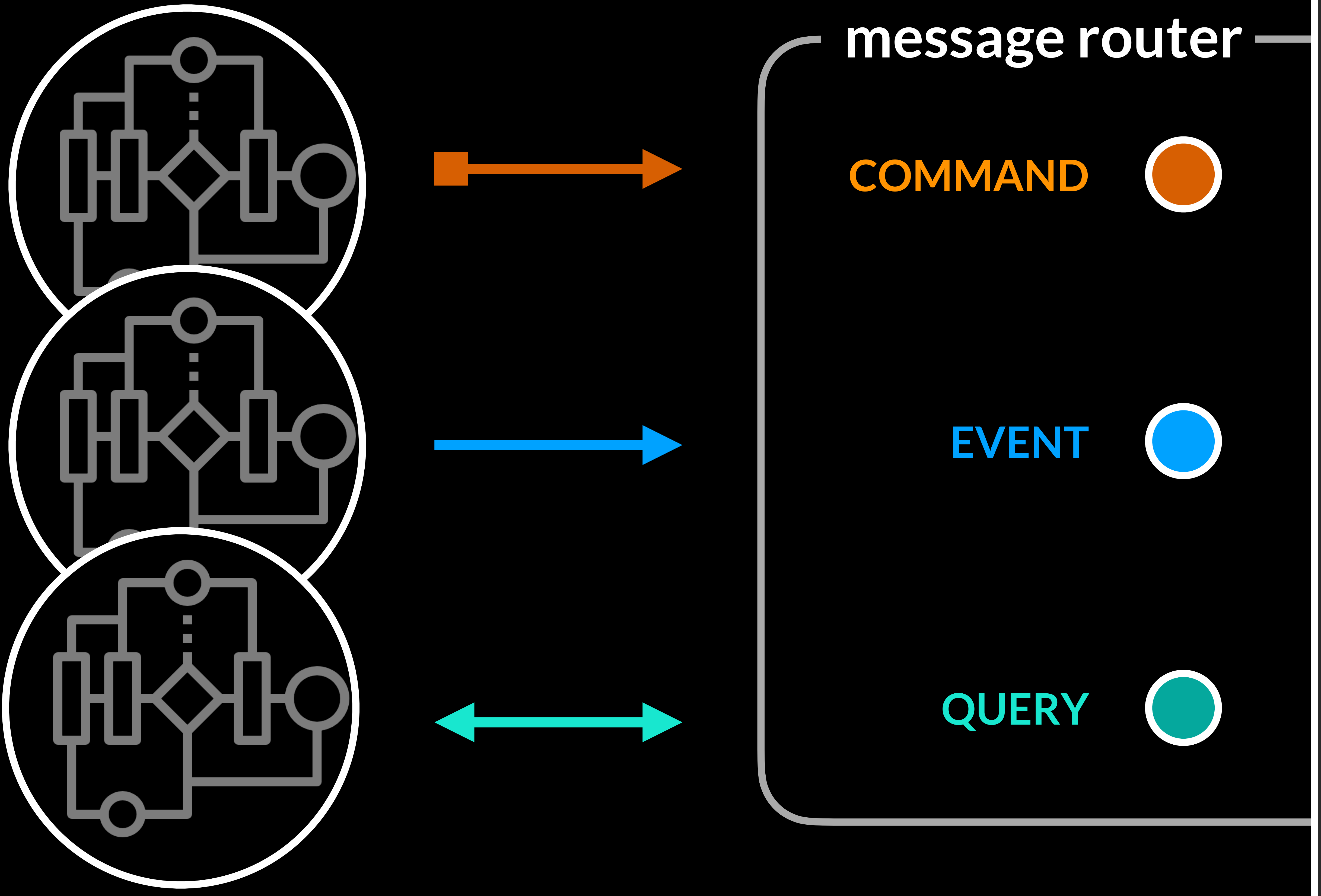
Messaging



Messaging

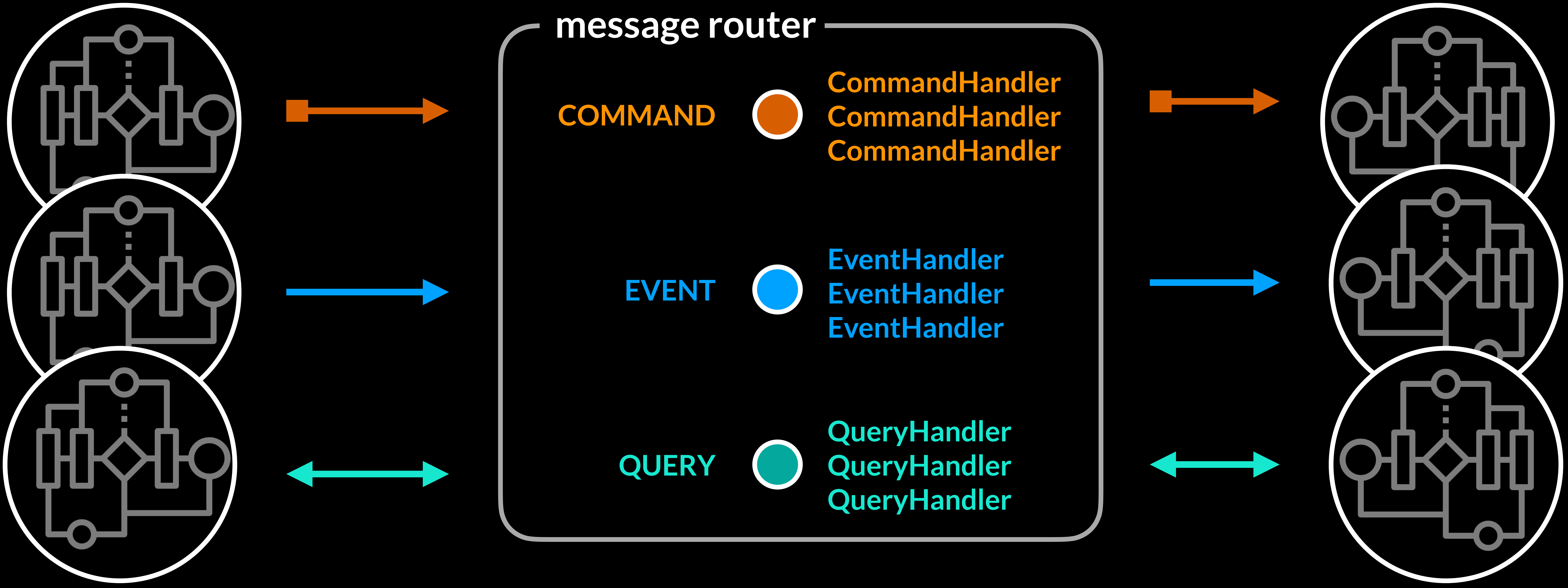


Messaging

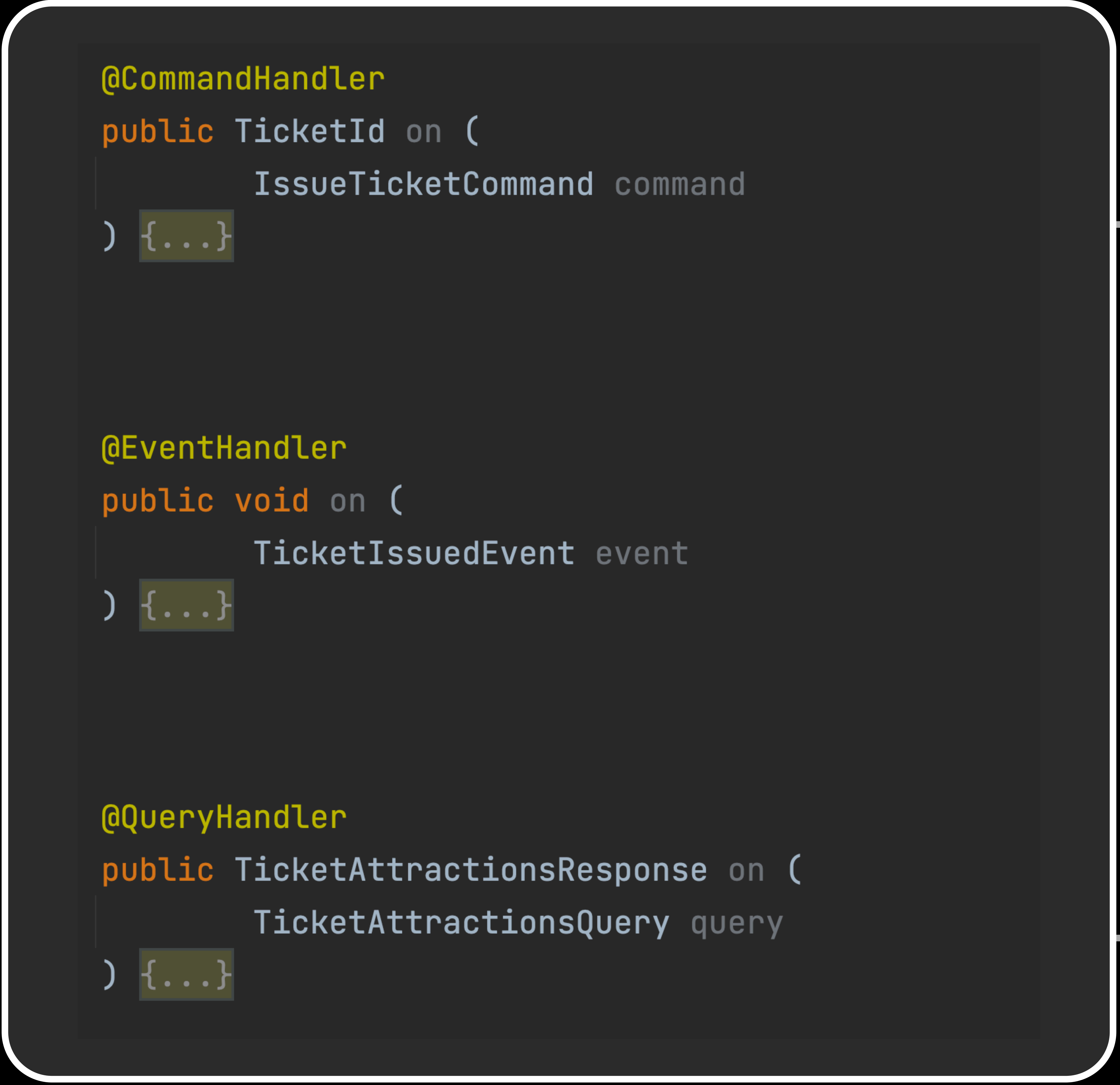


```
IssueTicketCommand command = new IssueTicketCommand(
        request.getMainAttraction(),
        request.getAttractions()
);
TicketId ticketId = commandGateway.sendAndWait(command);


TicketIssuedEvent event = new TicketIssuedEvent(
        ticketId,
        request.getMainAttraction(),
        request.getAttractions()
);
eventGateway.publish(event);


TicketAttractionsQuery query = new TicketAttractionsQuery(
        ticketId
);
queryGateway.query(
        query,
        ResponseTypes.instanceOf(
                TicketAttractionsResponse.class
        )
);
```

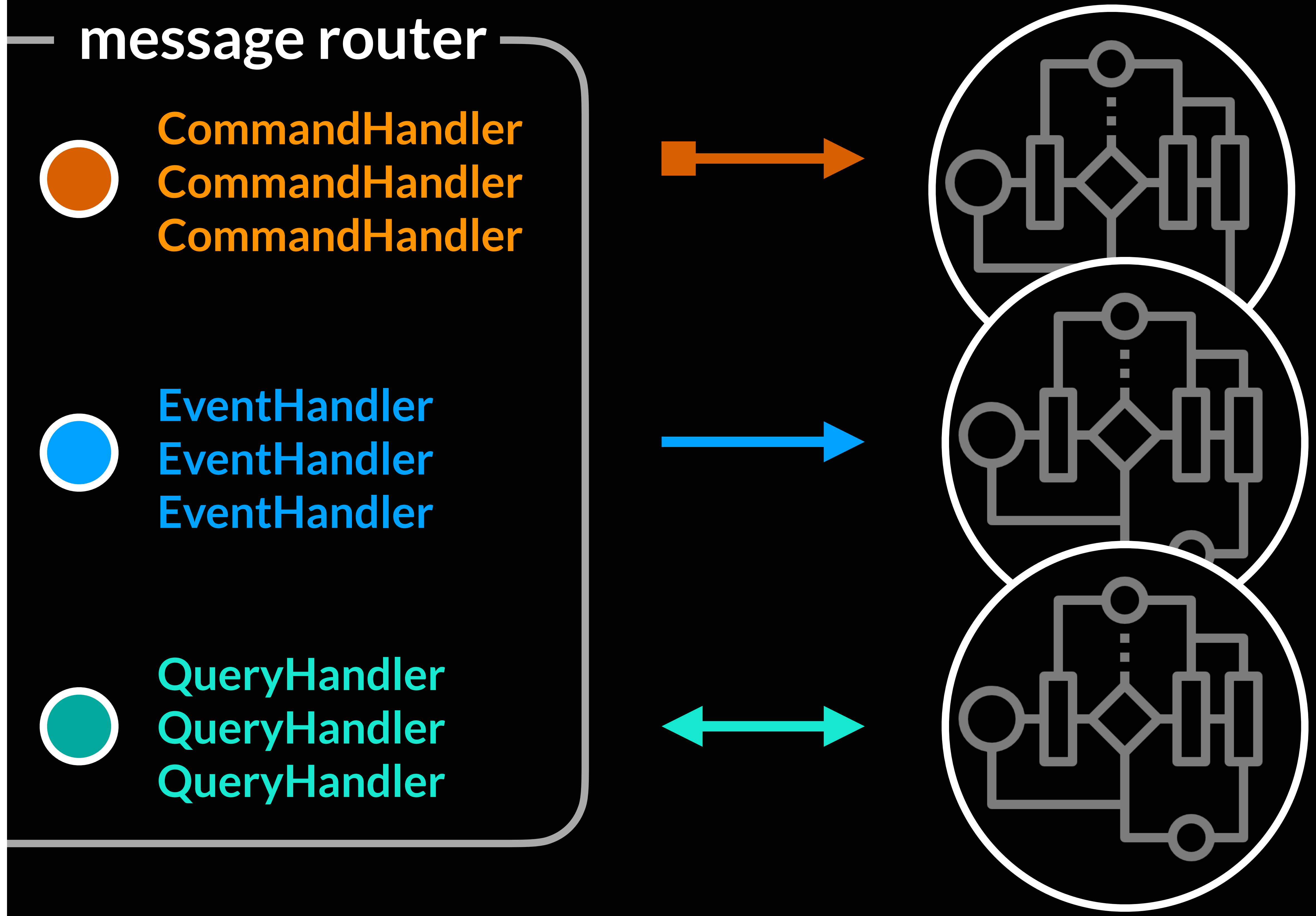
Messaging


message router
COMMAND
CommandHandler
CommandHandler
CommandHandler
EVENT
EventHandler
EventHandler
EventHandler
QUERY
QueryHandler
QueryHandler
QueryHandler

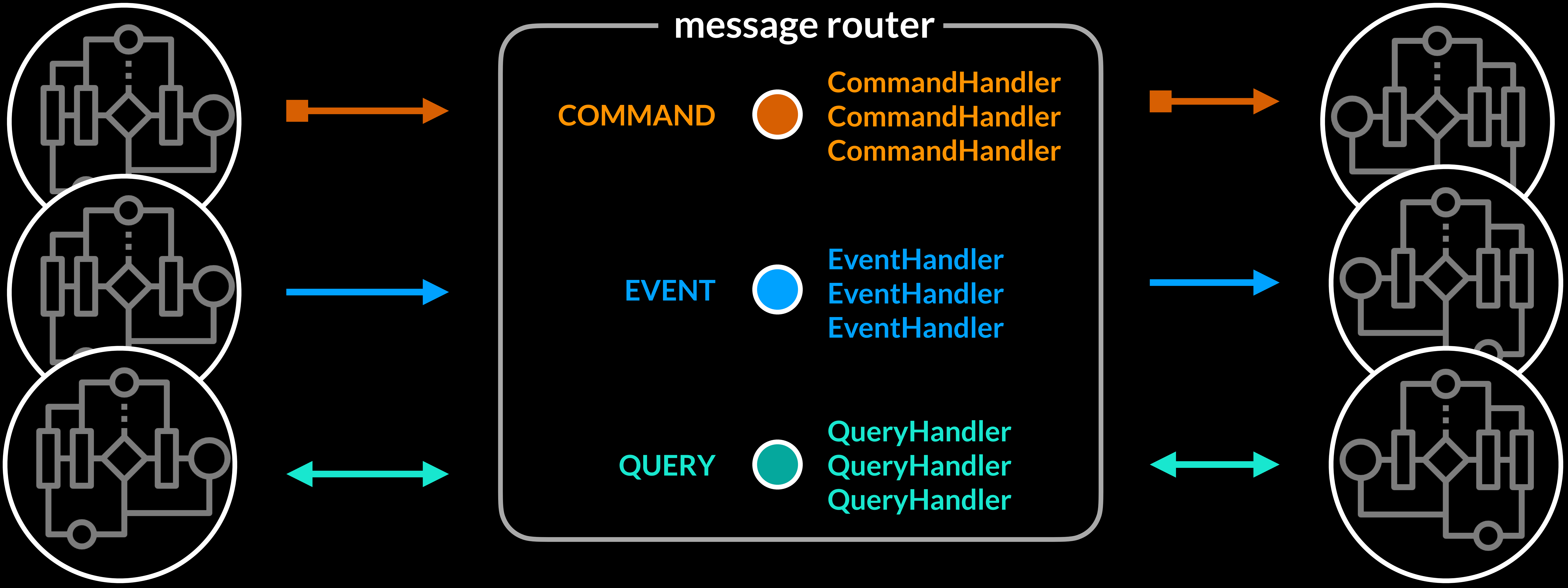
Messaging



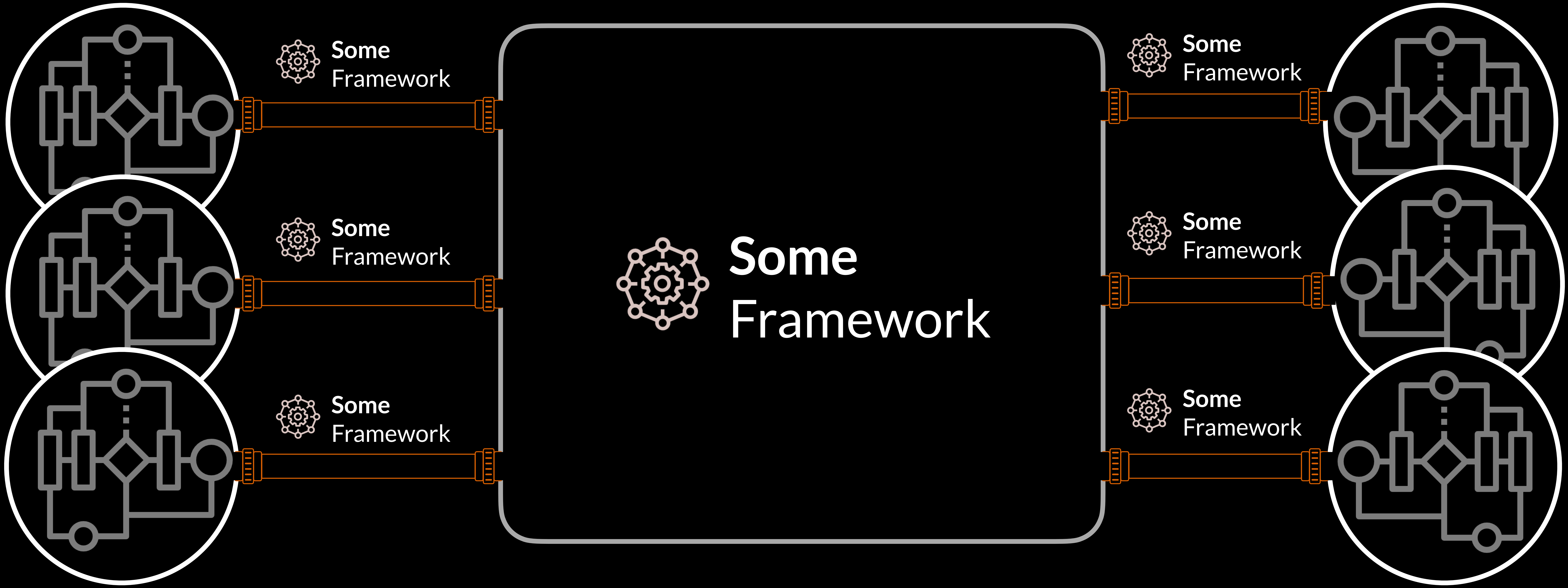
```
@CommandHandler
public TicketId on (
        IssueTicketCommand command
) {...}


@EventHandler
public void on (
        TicketIssuedEvent event
) {...}


@QueryHandler
public TicketAttractionsResponse on (
        TicketAttractionsQuery query
) {...}
```



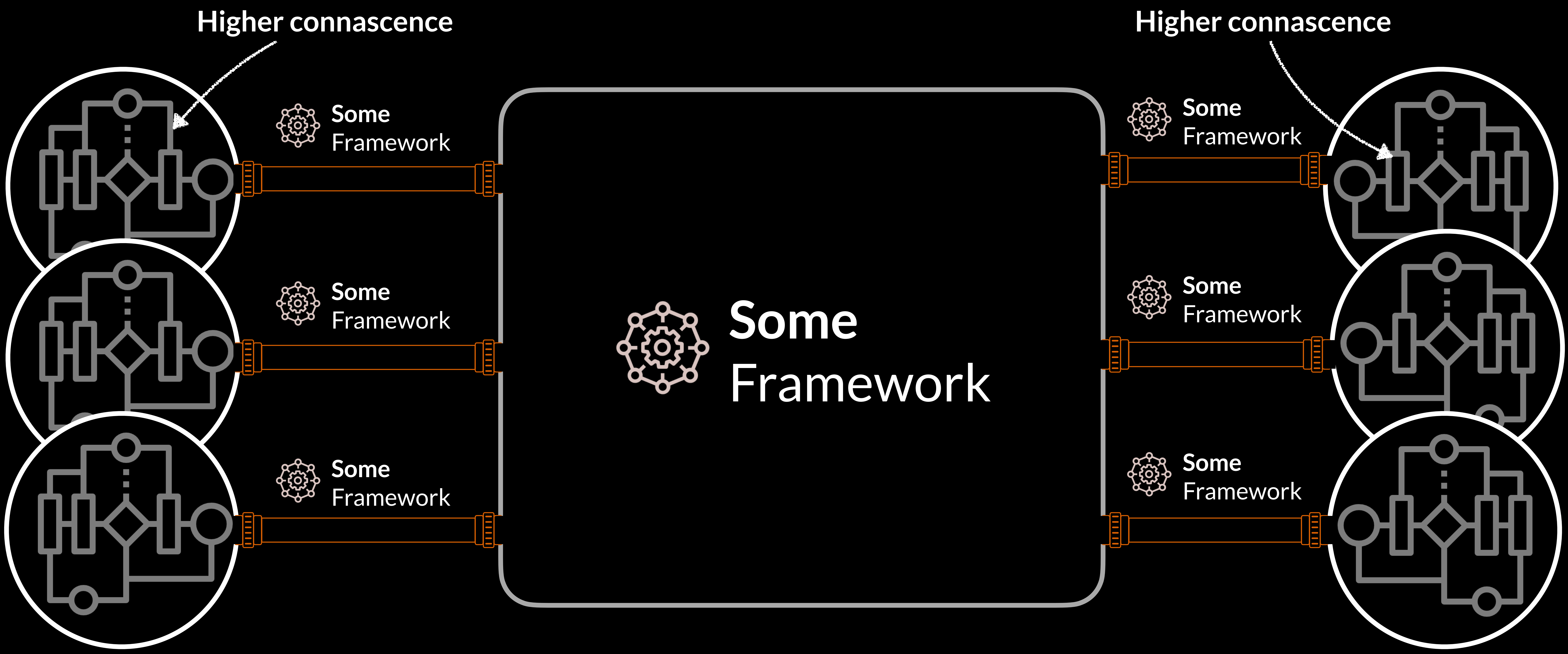
Messaging


message router
COMMAND
CommandHandler
CommandHandler
CommandHandler
EVENT
EventHandler
EventHandler
EventHandler
QUERY
QueryHandler
QueryHandler
QueryHandler

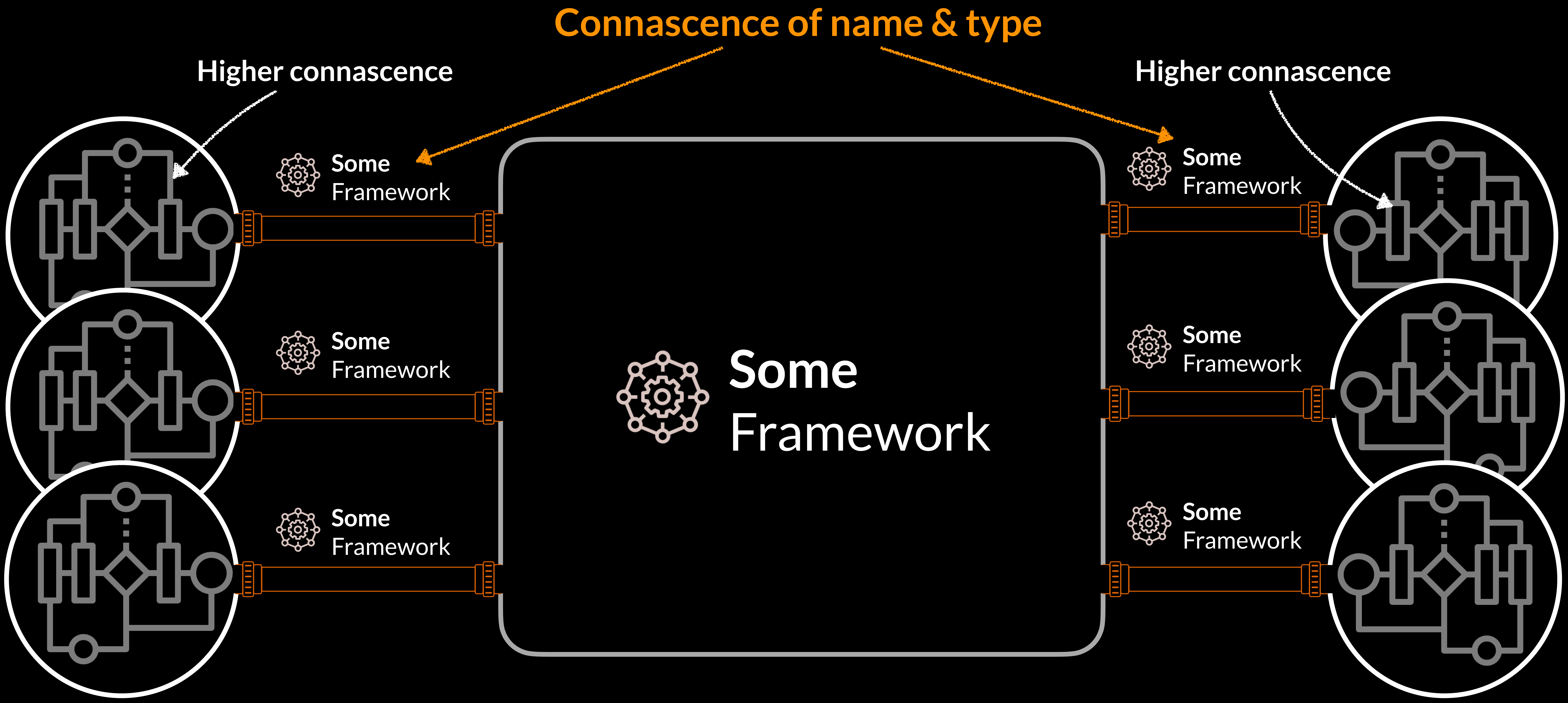
Messaging


Some Framework
Some Framework
Some Framework
Some Framework
Some Framework
Some Framework
Some Framework

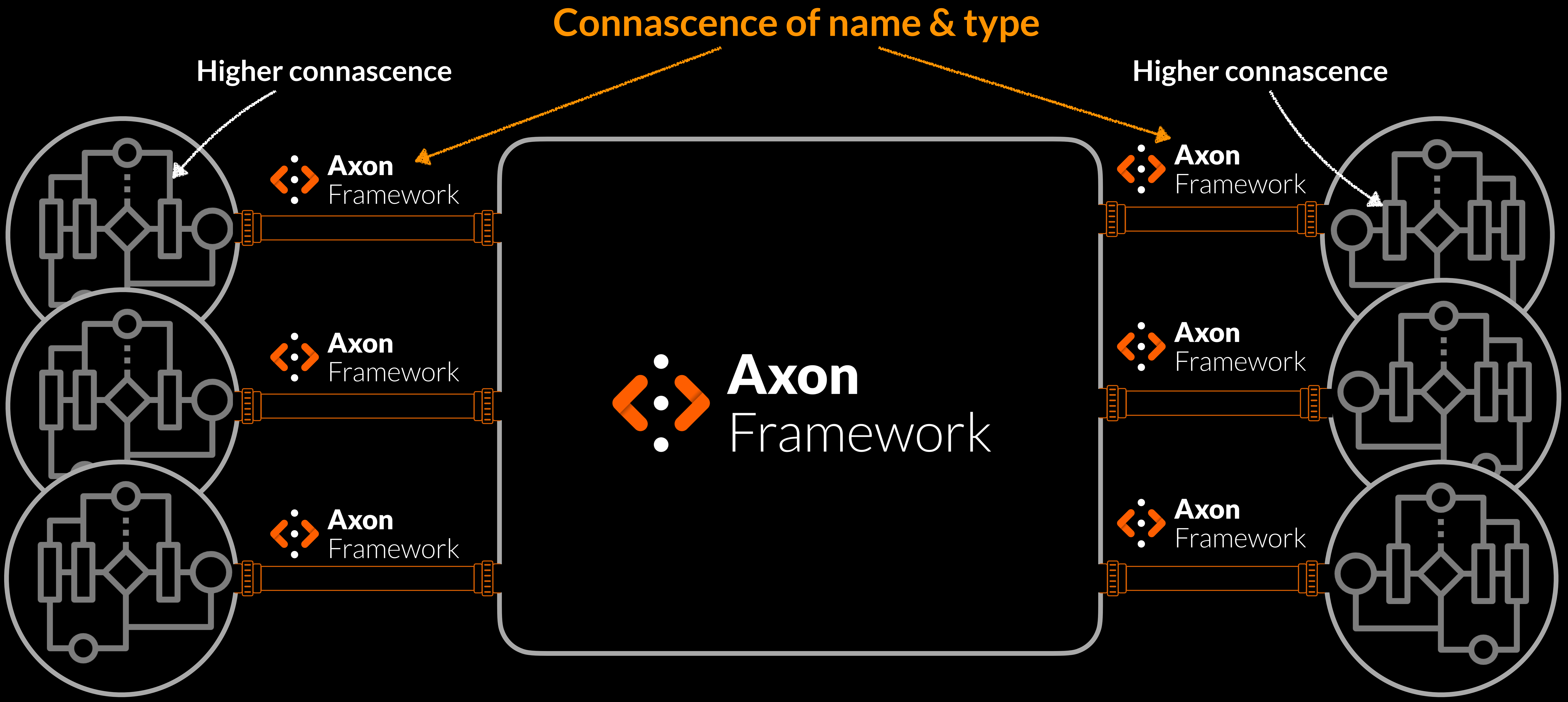
Connascence



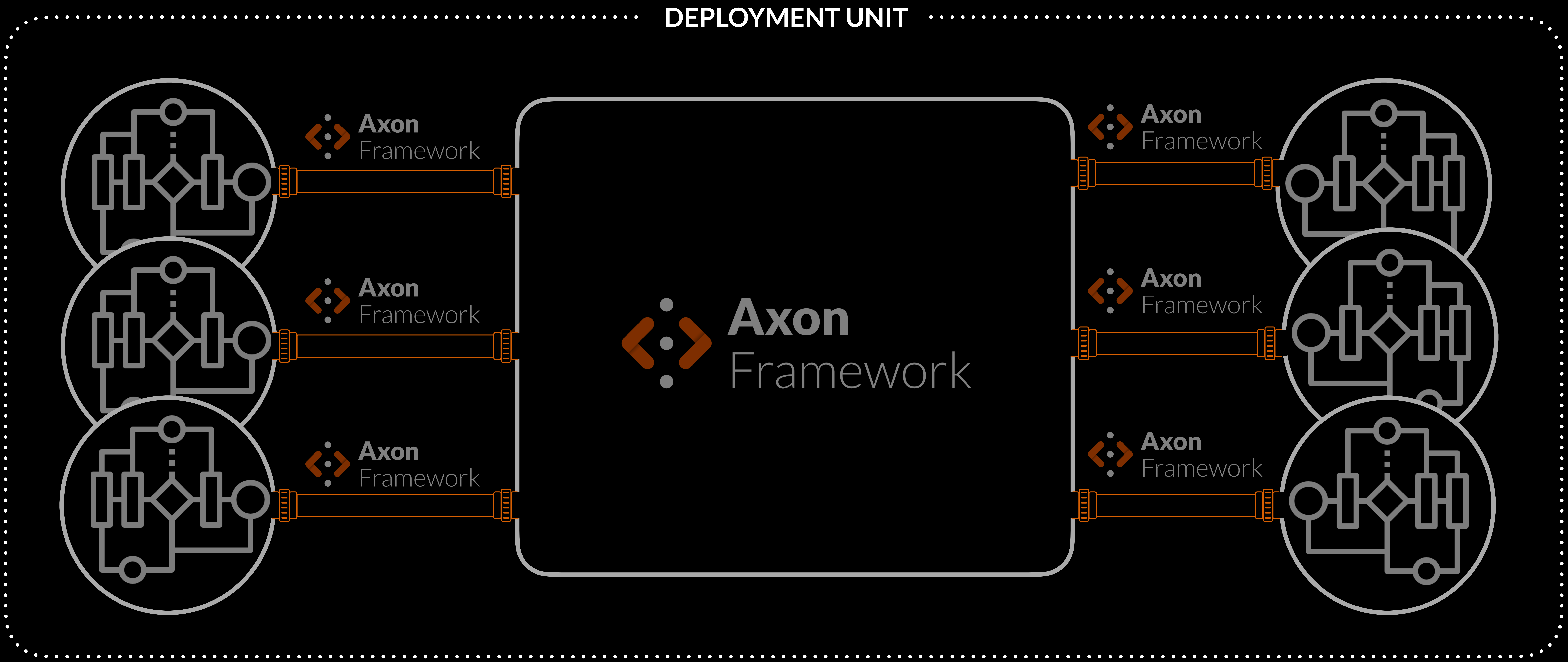
Connascence



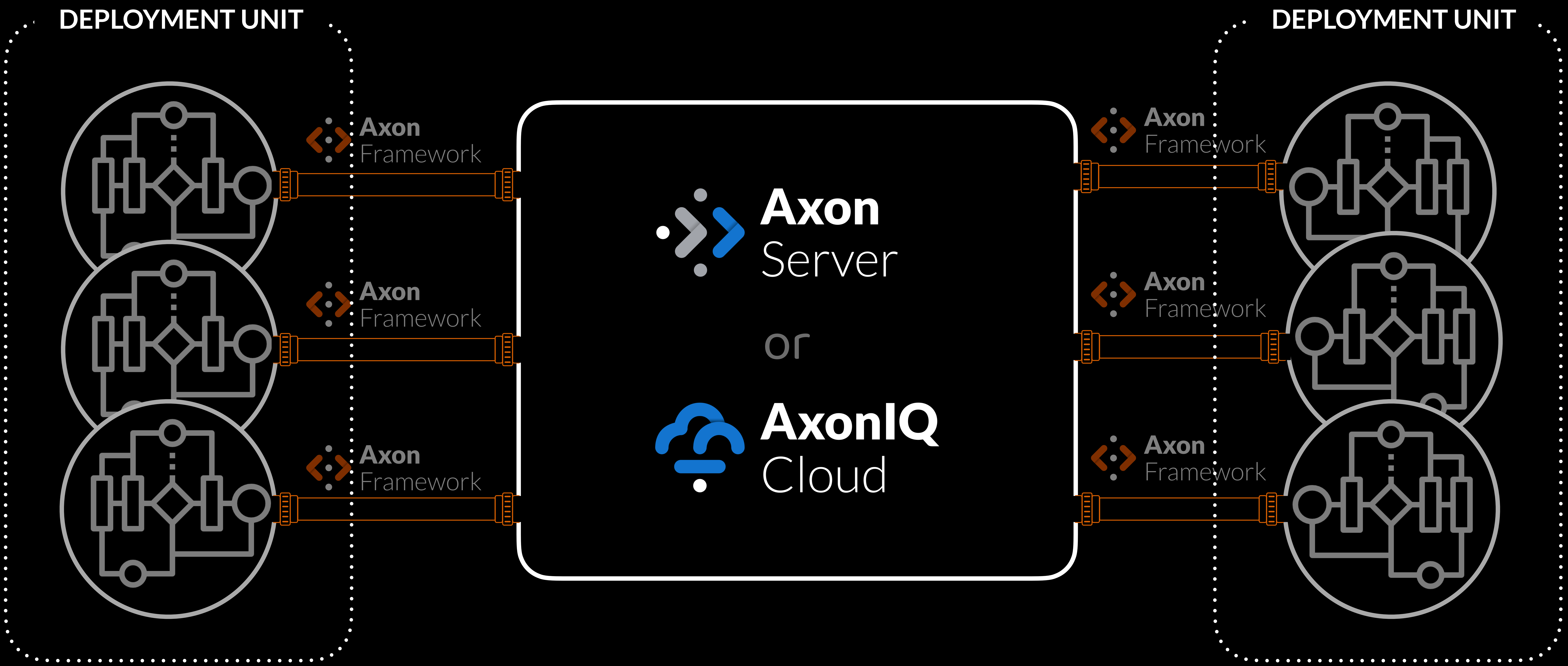
Connascence



Monolith


DEPLOYMENT UNIT
Axon Framework
Axon Framework
Axon Framework
Axon Framework
Axon Framework
Axon Framework
Axon Framework

Microservices



Open-source framework for building
DDD, ES and CQRS systems

https://developer.axoniq.io/axon-framework

Axon Framework

On premises, zero-configuration
message router and event store.

https://developer.axoniq.io/axon-server

Axon Server

Hosted and managed by AxonIQ
message router and event store.

https://developer.axoniq.io/axoniq-cloud

AxonIQ Cloud

Please send your feedback to

@MilenDyankov

@MilenDyankov@fosstodon.org

MilenDyankov@AxonIQ.io

axoniq.io

developer.axoniq.io

academy.axoniq.io

discuss.axoniq.io

github.com/axonframework

github.com/axoniq

@axon_iq