TALOS

JAVA 8 IN ACTION

Agenda

1. Intro
2. Functional interfaces
3. Method references & Lambdas
4. Optional
5. Changes in the interfaces
6. Stream
7. Time API

TALOS

1. Intro
Part 1

```
private String getAddressName(AbstractOrderEntryModel abstractOrderEntryModel)
  {
    if (abstractOrderEntryModel == null)
    {
      return DEFAULT_ADDRESS_NAME;
    }
    AddressModel addressModel = abstractOrderEntryModel.getDeliveryAddress();
    if (addressModel == null)
    {
      return DEFAULT_ADDRESS_NAME;
    }
    String streetName = addressModel.getStreetName();
    if (StringUtils.isBlank(streetName))
    {
      return DEFAULT_ADDRESS_NAME;
    }
    return streetName;
  }
```

TALOS

1. Intro
Part 1

```
private String getAddressName(AbstractOrderEntryModel abstractOrderEntryModel)
  {
    if (abstractOrderEntryModel == null)
    {
        return DEFAULT_ADDRESS_NAME;
    }
    AddressModel addressModel = abstractOrderEntryModel.getDeliveryAddress();
    if (addressModel == null)
    {
        return DEFAULT_ADDRESS_NAME;
    }
    String streetName = addressModel.getStreetName();
    if (StringUtils.isBlank(streetName))
    {
        return DEFAULT_ADDRESS_NAME;
    }
    return streetName;
  }
```

TALOS

1. Intro
Part 1

```
private String getAddressName(AbstractOrderEntryModel abstractOrderEntryModel)
  {
    if (abstractOrderEntryModel == null)
    {
      return DEFAULT_ADDRESS_NAME;
    }
    AddressModel addressModel = abstractOrderEntryModel.getDeliveryAddress();
    if (addressModel == null)
    {
      return DEFAULT_ADDRESS_NAME;
    }
    String streetName = addressModel.getStreetName();
    if (StringUtils.isBlank(streetName))
    {
      return DEFAULT_ADDRESS_NAME;
    }
    return streetName;
  }
```

TALOS

1. Intro
Part 2

```
Comparator<FutureStockData> sorter = new Comparator<FutureStockData>()
        {
            @Override
            public int compare(FutureStockData o1, FutureStockData o2)
            {
                Date date1 = o1.getDate();
                Date date2 = o2.getDate();
                if (isNull(date1) && isNull(date2))
                {
                    return 0;
                }
                else if (isNull(date1))
                {
                    return -1;
                }
                else if (isNull(date2))
                {
                    return 1;
                }
                return date1.compareTo(date2);
            }
        };
```

TALOS

1. Intro
Part 2

```
Comparator<FutureStockData> sorter = new Comparator<FutureStockData>()
        {
            @Override
            public int compare(FutureStockData o1, FutureStockData o2)
            {
                Date date1 = o1.getDate();
                Date date2 = o2.getDate();
                if (isNull(date1) && isNull(date2))
                {
                    return 0;
                }
                else if (isNull(date1))
                {
                    return -1;
                }
                else if (isNull(date2))
                {
                    return 1;
                }
                return date1.compareTo(date2);
            }
        };
```

TALOS

1. Intro
Part 2

```
Comparator<FutureStockData> sorter = new Comparator<FutureStockData>()
        {
            @Override
            public int compare(FutureStockData o1, FutureStockData o2)
            {
                Date date1 = o1.getDate();
                Date date2 = o2.getDate();
                if (isNull(date1) && isNull(date2))
                {
                    return 0;
                }
                else if (isNull(date1))
                {
                    return -1;
                }
                else if (isNull(date2))
                {
                    return 1;
                }
                return date1.compareTo(date2);
            }
        };
```

TALOS

1. Intro
Part 3

```
List<AbstractOrderModel> orders = createOrders();
Set<String> result = new HashSet<>();
for (AbstractOrderModel order : orders)
{
   for (AbstractOrderEntryModel entry : order.getEntries())
   {
      AddressModel addressModel = entry.getDeliveryAddress();
      String zipCode = addressModel.getZipCode();
      if (StringUtils.isNotBlank(zipCode) && !ZIP_4.equals(zipCode))
      {
         result.add(zipCode);
      }
   }
}
```

TALOS

1. Intro
Part 3

```
List<AbstractOrderModel> orders = createOrders();
Set<String> result = new HashSet<>();
for (AbstractOrderModel order : orders)
{
  for (AbstractOrderEntryModel entry : order.getEntries())
  {
    AddressModel addressModel = entry.getDeliveryAddress();
    String zipCode = addressModel.getZipCode();
    if (StringUtils.isNotBlank(zipCode) && !ZIP_4.equals(zipCode))
    {
      result.add(zipCode);
    }
  }
}
```

TALOS

1. Intro
Part 3

```
List<AbstractOrderModel> orders = createOrders();
Set<String> result = new HashSet<>();
for (AbstractOrderModel order : orders)
{
   for (AbstractOrderEntryModel entry : order.getEntries())
   {
      AddressModel addressModel = entry.getDeliveryAddress();
      String zipCode = addressModel.getZipCode();
      if (StringUtils.isNotBlank(zipCode) && !ZIP_4.equals(zipCode))
      {
         result.add(zipCode);
      }
   }
}
```

TALOS

1. Intro
Part 3

```
List<AbstractOrderModel> orders = createOrders();
Set<String> result = new HashSet<>();
for (AbstractOrderModel order : orders)
{
   for (AbstractOrderEntryModel entry : order.getEntries())
   {
      AddressModel addressModel = entry.getDeliveryAddress();
      String zipCode = addressModel.getZipCode();
      if (StringUtils.isNotBlank(zipCode) && !ZIP_4.equals(zipCode))
      {
         result.add(zipCode);
      }
   }
}
```

TALOS

1. Intro
Part 4

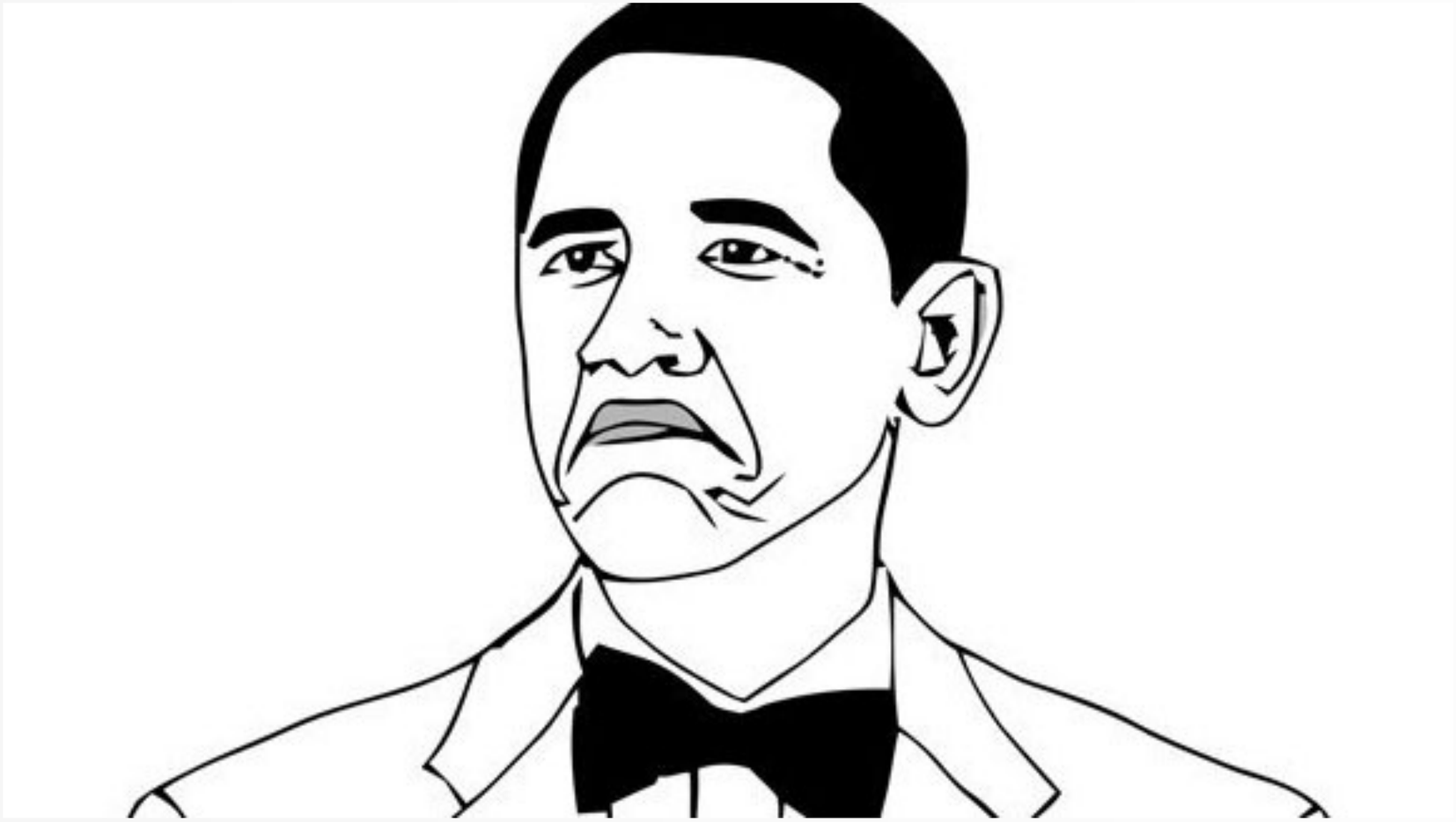
How do we encapsulate behavior?

TALOS

1. Intro
Part 4

How do we encapsulate behavior?

>> A private method!

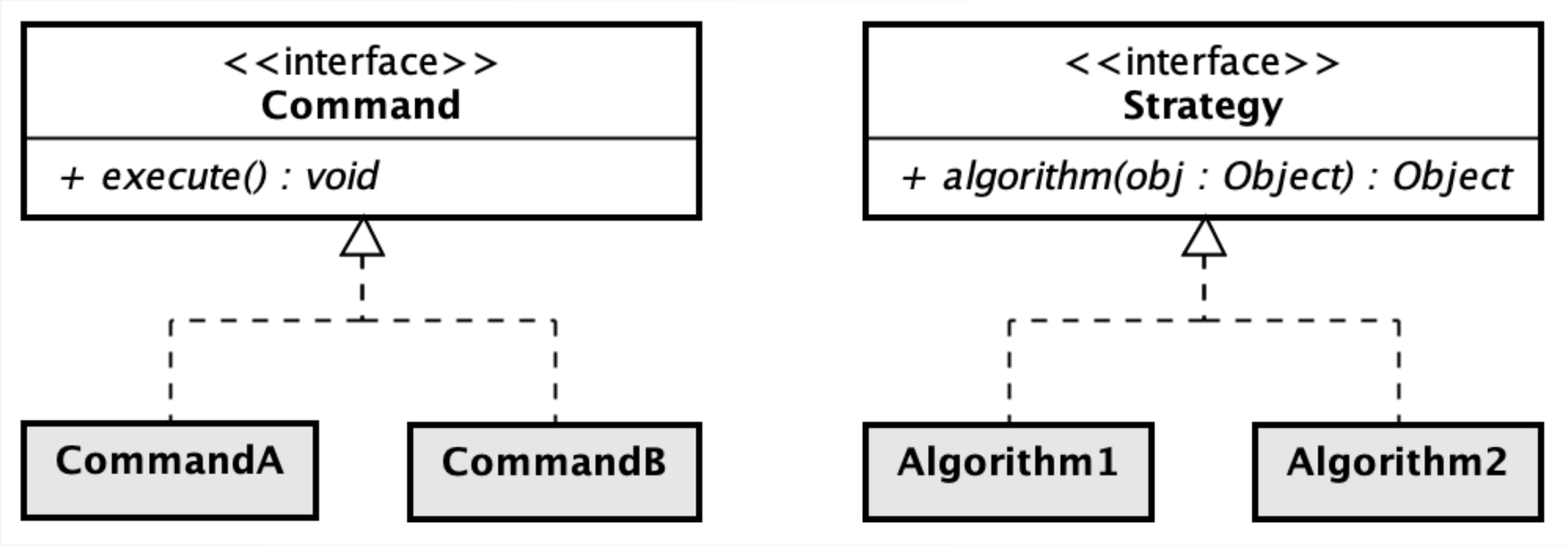


TALOS

1. Intro
Part 4

How do we encapsulate behavior?

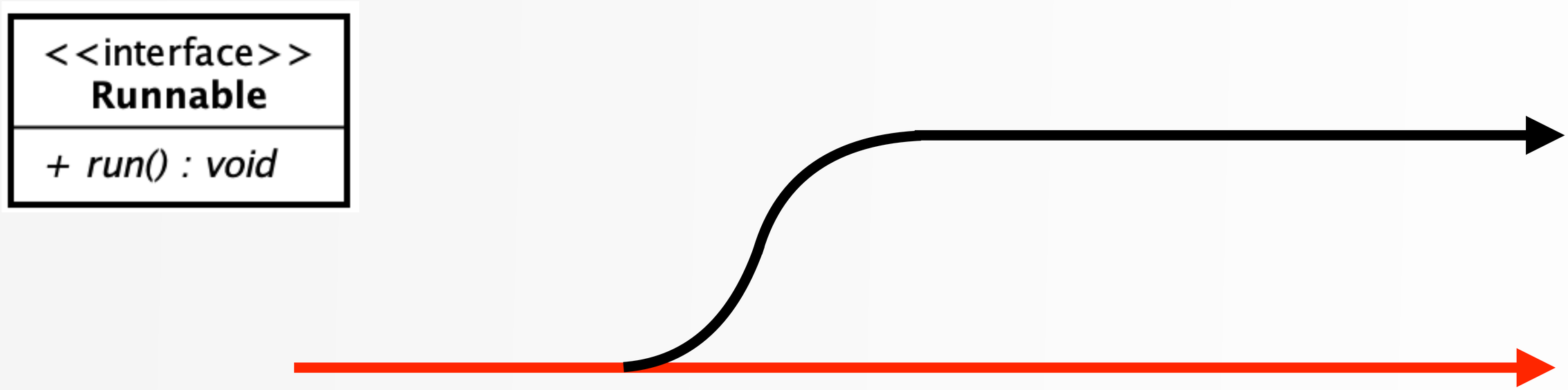
>> Design Patterns!



TALOS

1. Intro
Part 4

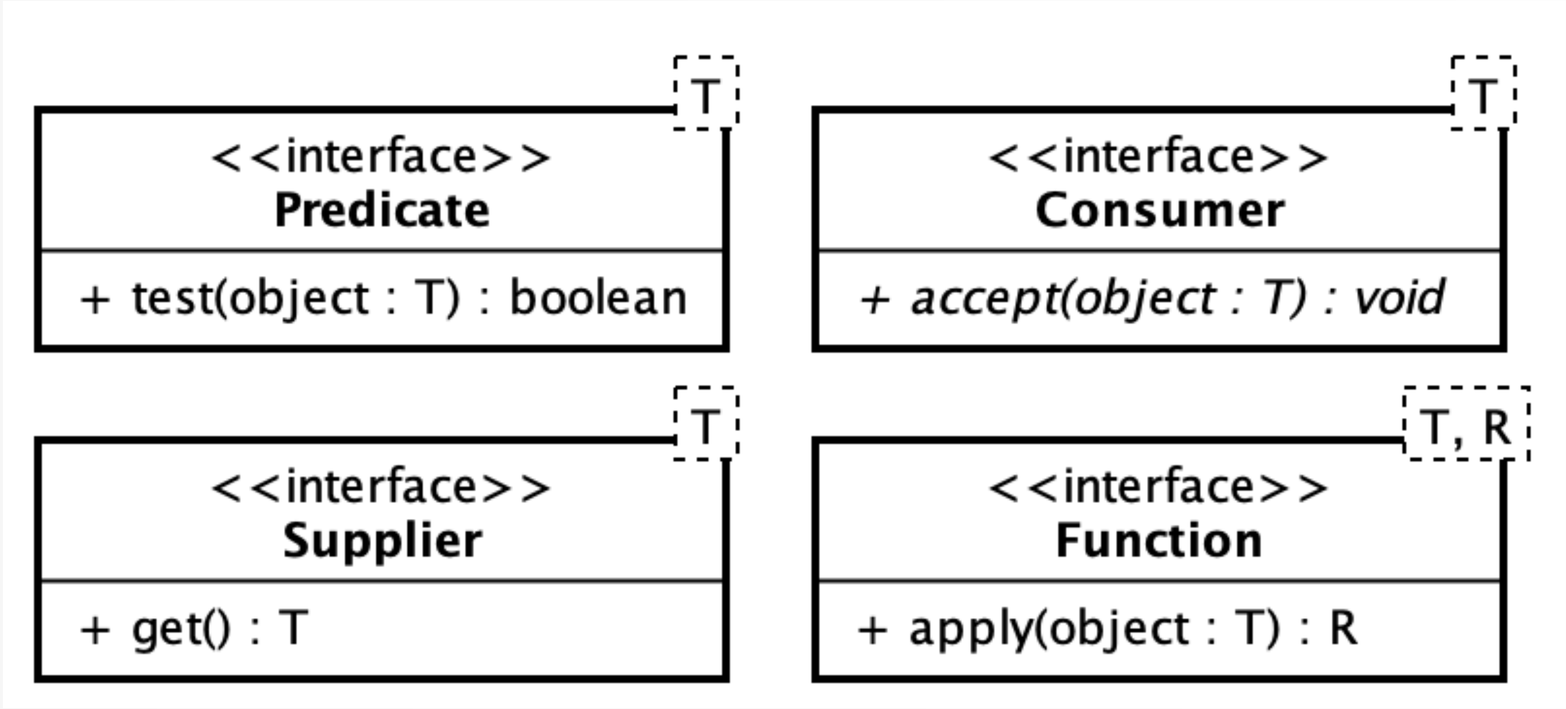
How do we encapsulate behavior?



TALOS

2. Functional Interfaces

INTERFACES WITH EXACTLY **ONE ABSTRACT METHOD**



@FunctionalInterface

TALOS

3. Method references & Lambdas

ALLOWS YOU TO CREATE OBJECTS OF FUNCTIONAL INTERFACES.

TALOS

3. Method references & Lambdas

ALLOWS YOU TO CREATE OBJECTS OF FUNCTIONAL INTERFACES.

METHOD REFERENCES

```
Class::instanceMethod
object::instanceMethod
Class::staticMethod
Class::new
Class[]::new
```

LAMBDAS

```
arg -> returns
arg -> {
            body;
            return x;
        }
(arg1, arg2) -> returns
```

TALOS

4. Optional

IT IS EITHER A WRAPPER FOR AN OBJECT OR FOR NO OBJECT. THEREFORE, IT IS A SAFER ALTERNATIVE TO HANDLE AN OBJECT THAT COULD BE NULL.

TALOS

4. Optional

IT IS EITHER A WRAPPER FOR AN OBJECT OR FOR NO OBJECT. THEREFORE, IT IS A SAFER ALTERNATIVE TO HANDLE AN OBJECT THAT COULD BE NULL.

- **Static methods**

  of(T value) : Optional<T>

  ofNullable(T value) : Optional<T>

  empty(): Optional<T>

- **Instance methods**

  get() : T

  isPresent() : boolean

  ifPresent(Consumer<T> action) : void

  map(Function<T, R> mapper) : Optional<R>

  flatMap(Function<T, Optional<R>> mapper) : Optional<R>

  filter(Predicate<T> predicate) : Optional<T>

  orElse(T other) : T

  orElseGet(Supplier<T> other) : T

  orElseThrow(Supplier<X> supplierException) : T

TALOS

5. Changes in the interfaces

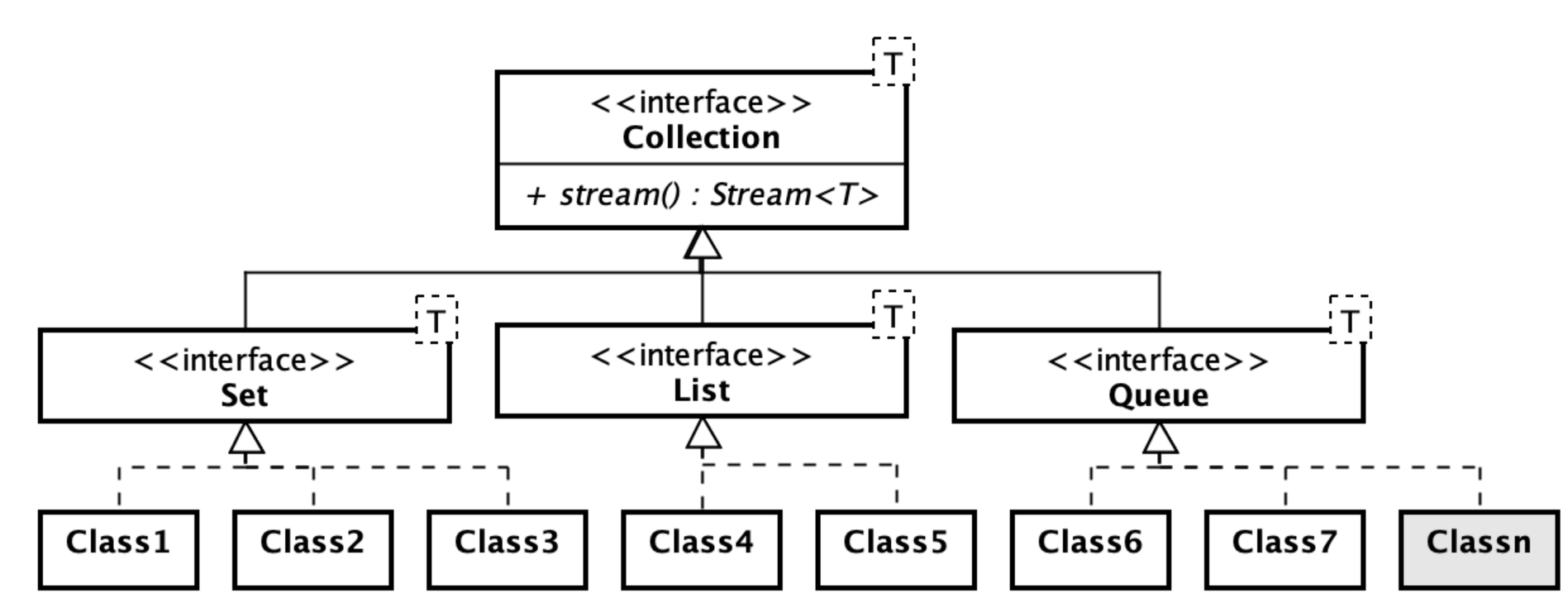
BINARY AND SOURCE COMPATIBILITY

DOES ANYONE KNOW WHAT IT MEANS?

TALOS

5. Changes in the interfaces

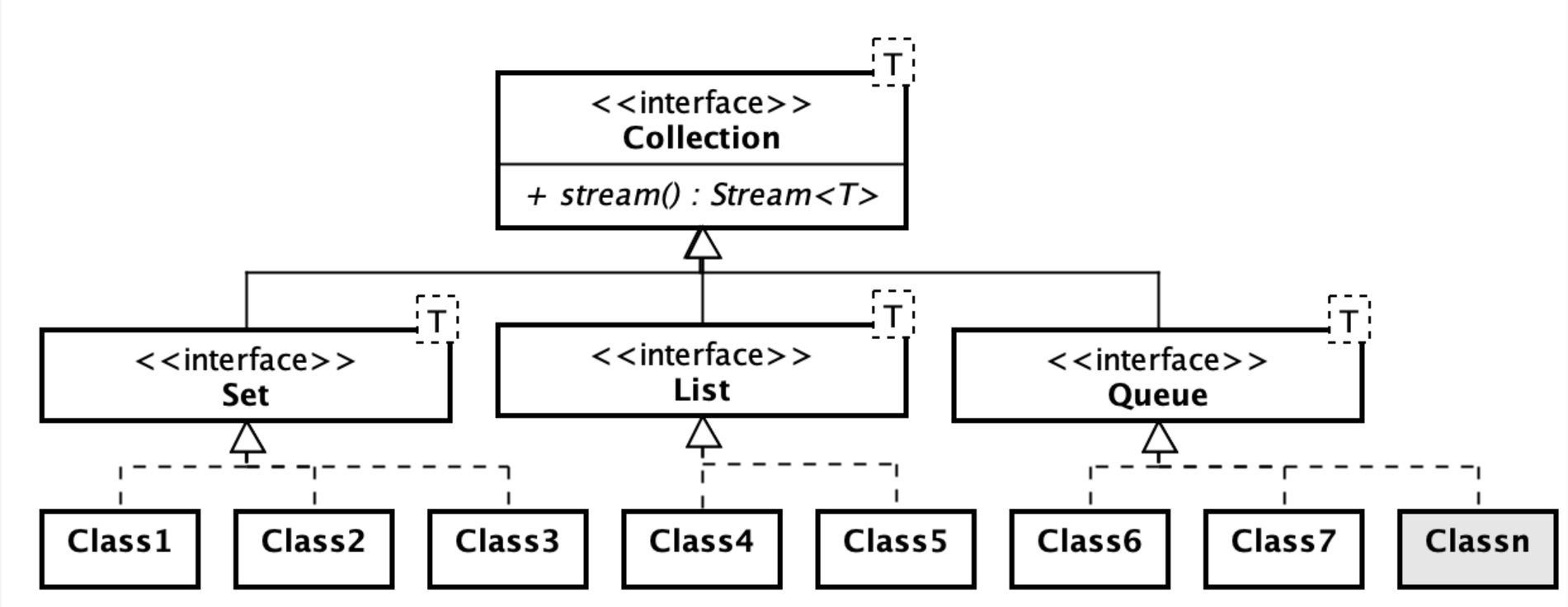
BINARY AND SOURCE COMPATIBILITY



TALOS

5. Changes in the interfaces

BINARY AND SOURCE COMPATIBILITY



- Static methods
- Default methods

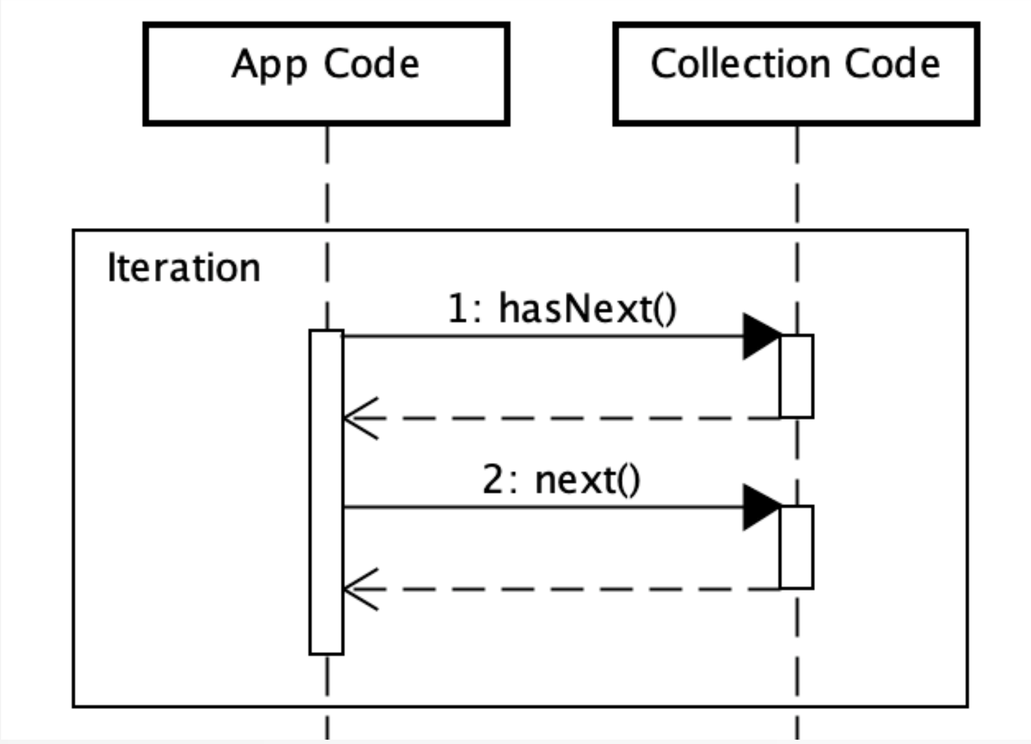
TALOS

6. Stream

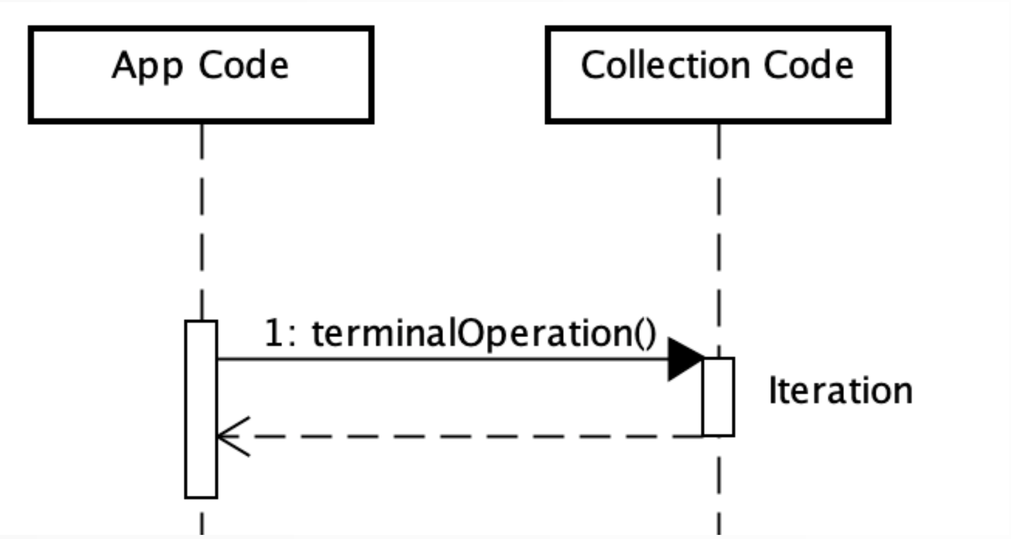
THE CREAM OF THE CROPS.

INTRODUCED IN JAVA 8 TO HANDLE COLLECTIONS IN A FUNCTIONAL STYLE.

EXTERNAL ITERATION



INTERNAL ITERATION



TALOS

6. Stream

THE CREAM OF THE CROPS.
INTRODUCED IN JAVA 8 TO HANDLE COLLECTIONS IN A FUNCTIONAL STYLE.

Intermediate operations

map(Function<T, R> mapper) : Stream<R>
flatMap(Function<T, Stream<R>> mapper) : Stream<R>
filter(Predicate<T> predicate) : Stream<T>
peek(Consumer<T> action) : Stream<T>

Terminal operations

allMatch(Predicate<T> predicate) : boolean
anyMatch(Predicate<T> predicate) : boolean
count() : long
reduce(BinaryOperator<T> accumulator) : Optional<T>
<R, A>**collect**(Collector<T, A, R> collector): R

TALOS

7. Time API

THE NEW TIME API HAS NEW IMMUTABLE OBJECTS AND SOLVE THE FLAWS OF THE PREVIOUS API (DATE, CALENDAR)

TALOS

7. Time API

THE NEW TIME API HAS NEW IMMUTABLE OBJECTS AND SOLVE THE FLAWS OF THE PREVIOUS API (DATE, CALENDAR)

Some classes

Instant
Duration
LocalDate
LocalTime
LocalDateTime
Period
ZonedDateTime

Partial dates

MonthDay
YearMonth
Year

Thread safe formatter

DateTimeFormatter

TALOS

THANK YOU!

TALOS