

ORACLE®

Lesson 2-7: The Optional Class

The Problems Of null

- Certain situations in Java return a result which is a null
 - Reference to an object that is not initialised

Avoiding NullPointerExceptions

```
String direction = gpsData.getPosition().getLatitude().getDirection();
```

```
String direction = "UNKNOWN";
```

```
if (gpsData != null) {  
    Position p = gpsData.getPosition();  
  
    if (p != null) {  
        Latitude latitude = p.getLatitude();  
  
        if (latitude != null)  
            direction = latitude.getDirection();  
    }  
}
```

Optional Class

Helping To Eliminate the NullPointerException

- Terminal operations like `min()`, `max()`, may not return a direct result
 - Suppose the input stream is empty?
- `Optional<T>`
 - Container for an object reference (null, or real object)
 - Think of it like a stream of 0 or 1 elements
 - Guaranteed that the `Optional` reference returned will not be null

Optional ifPresent()

Do something when set

```
if (x != null) {  
    print(x);  
}
```

```
opt.ifPresent(x -> print(x));  
opt.ifPresent(this::print);
```

Optional filter()

Reject certain values of the Optional

```
if (x != null && x.contains("a")) {  
    print(x);  
}
```

```
opt.filter(x -> x.contains("a"))  
    .ifPresent(this::print);
```

Optional map()

Transform value if present

```
if (x != null) {  
    String t = x.trim();  
    if (t.length() > 0)  
        print(t);  
}
```

```
opt.map(String::trim)  
    .filter(t -> t.length() > 0)  
    .ifPresent(this::print);
```

Optional flatMap()

Going deeper

```
public String findSimilar(String s)
```

```
Optional<String> tryFindSimilar(String s)
```

```
Optional<Optional<String>> bad = opt.map(this::tryFindSimilar);  
Optional<String> similar = opt.flatMap(this::tryFindSimilar);
```


Update Our GPS Code

```
class GPSData {  
    public Optional<Position> getPosition() { ... }  
}
```

```
class Position {  
    public Optional<Latitude> getLatitude() { ... }  
}
```

```
class Latitude {  
    public String getDirection() { ... }  
}
```

Update Our GPS Code

getPosition and
getLatitude return
an Optional

```
String direction = Optional  
    .ofNullable(gpsData)  
    .flatMap(GPSData::getPosition)  
    .flatMap(Position::getLatitude)  
    .map(Latitude::getDirection)  
    .orElse("None");
```

getDirection
returns a String

Create new Optional
with a reference that
could be null

If getDirection returns a null
return "None", otherwise the
actual direction

Section 7

Summary

- Optional class eliminates problems of `NullPointerException`
- Can be used in powerful ways to provide complex conditional handling

Lesson 2: Summary

Lesson 2

Introduction To Streams

- Streams provides a straight forward way for functional style programming in Java
- Streams can either be objects or primitive types
- A stream consists of a source, possible intermediate operations and a terminal operation
 - Certain terminal operations return an `Optional` to avoid possible `NullPointerException` problems

ORACLE®