**Java 8 Optional Yeniliği**

Bir Java developer’ın korkulu rüyası NullPointerException exception ile uğraşmaktır. null değer almamak için Java 8 içerisinde Optional sınıfı getirilmiştir.

Optional type’inden objeler, null olma ihtimali olan alanları kolay handle etmek için oluşturulmuştur.

**Optional create etmek …..**

Bir Optional objesi, Optional class’ının çeşitli statik metodlarıyla oluşturulmaktadır. Bunlar empty, of ve ofNullable ‘dir.

**empty :**Taze bir Optional nesnesi oluşturur.

**of :** Bir nesneyi Optinal ile sarmalar. Parametre olarak null değer kabul etmez.

**ofNullable :** Bir nesneyi Optinal ile sarmalar. Parametre olarak null değer kabul eder.

**Örneğin**

Optional<Double> empty = Optional.empty(); **(1)**

Optional<String> of = Optional.of("Merhaba Dünya"); **(2)**

Optional<String> ofNull = Optional.of(null); **(3)**

Optional<Integer> ofNullable = Optional.ofNullable(null); **(4)**

1. Değer içermeyen Optional
2. String türünden değer içeren Optional
3. Optional#of null kabul etmez. exception fırlatır.
4. Optional#ofNullable null kabul eder. exception fırlatmaz.

**.ifPresent() – Varsa yap, yoksa yapma**

Eğer bir Optional içerisinde sadece data varsa (null değilse) bir işin yapılması isteniyorsa .ifPresent() metodu kullanılabilir.  .ifPresent() metodu Consumer<T> fonksiyonel interface türünden bir obje kabul etmektedir.

Örneğin bir sayının karesini almaya çalışalım. Kullanılan değişken null değerini referans ediyorsa NullPointerException exception’ı alınacaktır.

Integer numara = null;

Double karesi = Math.pow(numara , 2); **(1)**

System.out.println("Sonuç: " + karesi);

1. Exception in thread “main” java.lang.NullPointerException

Bu istisna için if statement ile karşı önlem alınabilir.

Integer numara = null;

if(numara != null) {

Double karesi = Math.pow(numara , 2);

System.out.println("Sonuç: " + karesi);

}

Fakat if statement ile birlikte ! , == , != ifadelerini kullanmak akıcı bir geliştirim deneyimi sunmaz. Ayrıca bu durum hata yapılmasına daha açıktır. Bunun yerine Optional.ifPresent() metodunu kullanabiliriz.

Integer numara = null;

Optional<Integer> opt = Optional.ofNullable(numara);

opt.ifPresent(num -> {

Double karesi = Math.pow(num , 2);

System.out.println("Sonuç: " + karesi);

});

**.map() – Dönüştürme**

Optional objelerinin sarmaladığı data üzerinde casting yapılabilmektedir. Bir önceki örneği bu şekilde yeniden yazabiliriz.

Integer numara = null;

Optional<Integer> opt = Optional.ofNullable(numara);

opt

.map(num->Math.pow(num,2))

.ifPresent(System.out::println);

**.filter() – Filtreleme**

Optional objelerinin sarmaladığı veriler üzerinde süzme işlemi de yapılabilmektedir.

Örneğin aşağıdaki kod parçası yerine;

String message = null;

if (message != null)

if (message.length() > 5)

System.out.println(message);

Aşağıdaki Optional karşılığını kullanabiliriz.

String message = null;

Optional<String> opt = Optional.ofNullable(message);

opt

.filter(m -> m.length() > 5)

.ifPresent(System.out::println);

**.orElse() – Varsa al, yoksa bunu al**

.orElse() metodu daha çok ternary (üçlü) şart ihtiyacı olduğu durumlarda ihtiyaç duyulabilir. Daha akıcı bir geliştirim deneyimi sunar.

numara null değilse numarayı döndür, null ise 0 döndür.

Integer numara = null;

int result = (numara != null) ? numara : 0;

Yukarıdaki üçlü şartı .orElse()  method’u ile birlikte kullanabiliriz.

Integer numara = null;

Optional<Integer> opt = Optional.ofNullable(numara);

int result = opt.orElse(0);

**.orElseGet() – Varsa al, yoksa üret**

Bu metod .orElse() metoduna çok benzer, fakat .orElseGet() metod parametresi olarak Supplier fonksiyonel arayüzü türünden nesne kabul eder.

List<String> names = Arrays.asList("Ali","Veli","Selami");

Optional<List<String>> opt = Optional.ofNullable(names);

names = opt.orElseGet(()-> new ArrayList()); **(1)**

names = opt.orElseGet(ArrayList::new); **(2)**

1. Lambda ile
2. Metod referans ile

**.orElseThrow() – Varsa al, yoksa fırlat**

Optional objesi bir değeri içeriyorsa (null olmayan) o değeri döndürür, null ise de sağlanan exception objesi fırlatır. .orElseThrow() metodu Supplier türünden bir obje kabul eder.

Integer numara = null;

Optional<Integer> opt = Optional.ofNullable(numara);

int result = opt.orElseThrow(RuntimeException::new); **(1)**

1. Varsa döndürür, yoksa yeni bir RuntimeException exception’ı fırlatır.