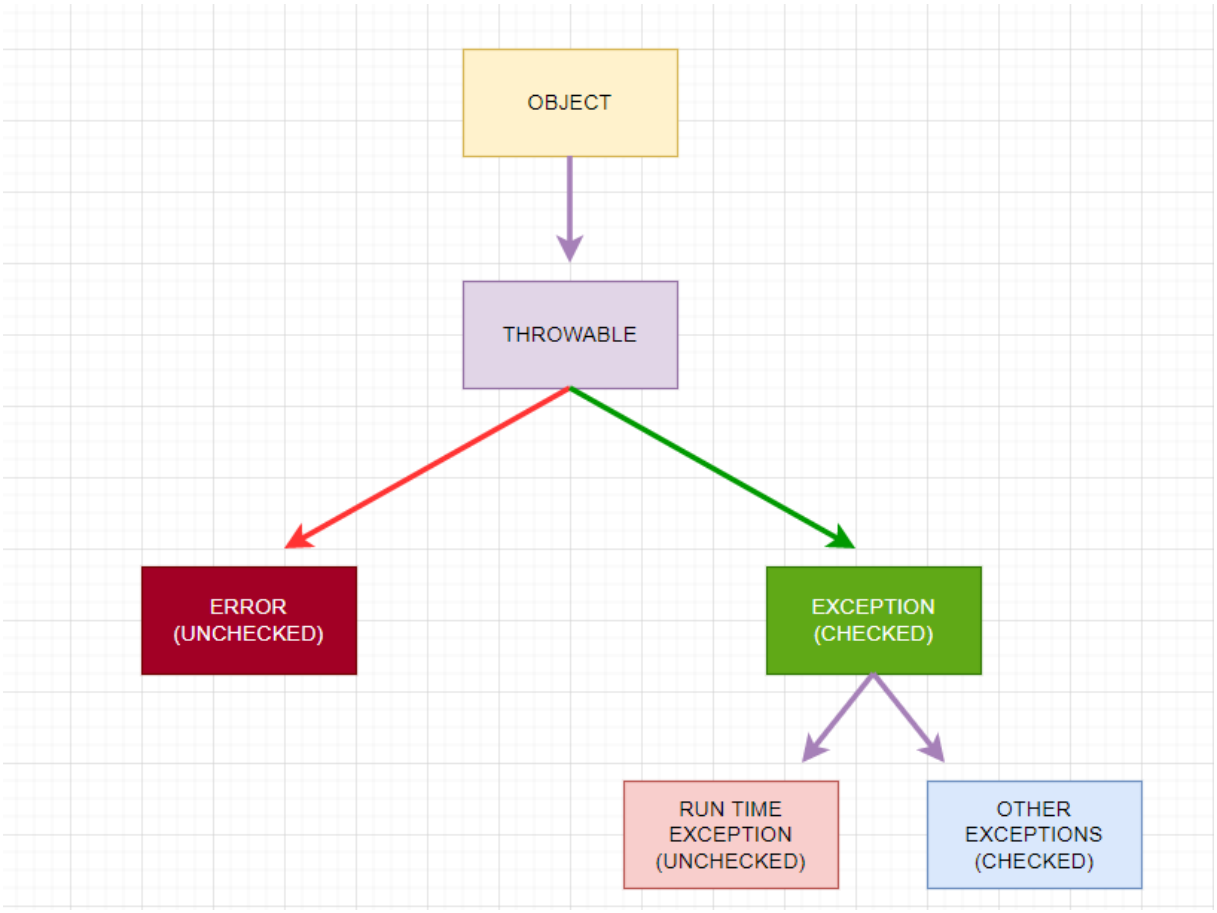
Checked vs Unchecked Exception

Java'da Checked ve Unchecked Exceptionları derleyici tarafından kontrol edilebilme ve derleyici tarafından kontrol edilememe olarak birbirinden ayrılır. Checked Exceptionlar derleyici tarafından kontrol edilebilen "Exception" olurken, Unchecked Exceptionlar derleyici tarafından kontrol edilemeyen "Exception" olarak tanımlanır.

Exceptionları daha iyi yorumlamak için ve Checked, Unchecked olarak belirtmek için aşağıda bulunan hiyerarşi diagramı üzerinden örnekler ile birlikte ilerleyebiliriz.



CHECKED EXCEPTION

Checked Exceptionlar, Exception sınıfına ait alt sınıflardır(sub-class). Derleyici tarafından kontrol edildiği için, biz bu exceptionları kullanırken kodumuzda belirtmek zorundayız. Aksi halde kodumuzu ayağa kaldıramayız ve hatalar alırız. Yani programımızı derleyemeyiz. Checked Exception örnekleri şöyledir;

- ClassNotFoundException
- IOException
- FileNotException
- SQLException
- NoSuchMethodException

ClassNotFoundException, örneğine şöyle bir örnek verebiliriz ve bu durumun nasıl yazılması gerektiğini görebiliriz. Aşağıdaki örnekte kod ve kod arkasından fırlatılan hata mesajı gösterilmiştir.

```
class Deneme{
    public static void main(String[] args) throws ClassNotFoundException {

            Class temp = Class.forName("deneme");
    }
}
```

```
java: unreported exception java.lang.ClassNotFoundException; must be caught or declared to be thrown
```

Bunu kod içinde belirtmemiz gerektiği için kodumuzu yeniden şöyle yazmamız gerecek. “deneme” isimli bir sınıfımız olmadığı için bizim yazdığımız println fonksiyonu devreye girecek ve program hata fırlatmayacak.

```
class Deneme{
    public static void main(String[] args) throws ClassNotFoundException {


        try {
            Class temp = Class.forName("deneme");
        } catch (ClassNotFoundException e) {
            System.out.println("deneme adında bir sınıf mevcut değil.");
        }

    }

}
```

```
deneme adında bir sınıf mevcut değil.
```

Görüldüğü üzere program çalışıyor ve istediğimiz mesaj bloğuna gidiyor. Bunu diğer Checked Exceptionlar içinde yapabiliriz. Bu bize geliştirdiğimiz projelerde oluşabilecek hata yerlerinde önceden önlem alarak sağlıklı bir şekilde geliştirme yapmamızı sağlayacaktır.

"throws ……" anahtar kelimesi ile birlikte oluşabilecek hatayı kabullendiğimizi belirtiriz ve buna uygun try-catch bloğu yazarız. Bu işleme "Exception Handling" (Hata Yakalama) adı verilir.

UNCHECKED EXCEPTION

Unchecked Exceptionlar, yukarıda bulunan hiyerarşiye göre RunTimeException sınıfını baz alırlar. Checked Exceptionlara göre derleme sırasında herhangi bir sorunla karşı karşıya kalmayacağımızı söylebiliriz. Ama çalışma sırasında bunlar sorun oluşturacaktır ve Exception atacaktır. Mesela herhangi bir matematiksel işlem için genel matematik kurallarına aykırı bile olsa bize derleme sırasında herhangi bir hata fırlatmayacaktır. Sonrasında ilgili Exception fırlatmasını yapacaktır. Unchecked Exceptionlara örnek olarak;

- NullPointerException
- ArrayIndexOutOfException
- IllegalArgumentException verilebilir.

Derleme sonrası ortaya çıkan hatalardan sonra Checked Exceptionlar gibi bizimde bu Exceptionları yakalamak için handle etmemiz gerekir. Bunu yine Checked Exceptionlarda olduğu gibi try-catch bloğu ile yapabiliriz.

KAYNAKÇA

- Baeldung – Checked and Unchecked Exceptions in Java
- GeeksforGeeks - Checked and Unchecked Exceptions in Java
- https://app.diagrams.net/ → Diagram için.