**EXCEPTION HANDLING**

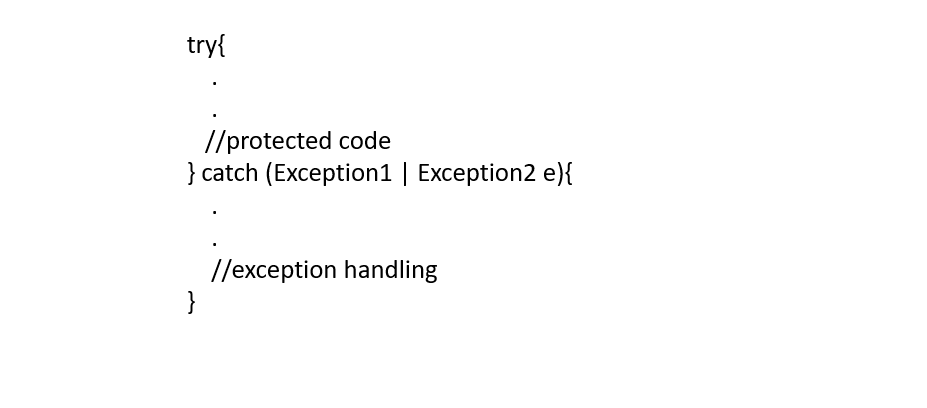
**Multi Catch**

Catch bloklarındaki duplication’u azaltamayı hedeflemektedir.

Bilindiği üzere bir try ifadesi birden fazla cath bloğu alabilmektedir.

Java 7 ile birlikte gelen multi catch özelliği sayesinde bir catch ifadesinde birden fazla exception tanımlanabilmektedir.

Multi catch grameri aşağıdaki gibidir;



**Hatalı Gösterimler:**

catch (FileNotFoundException e| SQLException e) : Aynı isimde olsa birden fazla isimlendirme tanımlayamayız.

catch (FileNotFoundException e| SQLException e2) : Farklı isimde olsa birden fazla isimlendirme tanımlayamayız.

catch (FileNotFoundException | IOException e2) : Aralarında hiyerarşi olduğu için bu kullanımda hatalıdır. Çünkü FileNotFoundException, IOException’ı extend etmektedir. Yani aynı catch bloğu içinde aralarında inheritance olan exceptionları kullanamayız.

**Example**: Aşağıdaki multi-catch yaklaşımlarını inceleyelim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

14: Doğru bir kullanımdır. Aralarında is-a ilişki yoktur. FileNotFoundException bir IOException olduğundan tanımlanmasında bir sakınca yoktur. Tek bir isimlendirme yapılmıştır.

15: İki isimlendirme yapıldığı için hatalıdır.

16: SQLException bir checked exceptiondır. Fırlatılma ihtimali olmadığı için kullanılması legal değildir.

Not: catch bloğunda bir checked exception bulunabilmesi için try bloğunda ilgili hatanın fırlatılma ihtimali olması lazım.

17: FileNotFoundException exception iki kere tanımlanmaya çalışılmış. Legal değildir.

18-19: Daha spesifik olan exception’ın üstte olması gereklidir.

**Try-With-Resources**

Bir başka problem ise finally bloklarındaki duplicationlar. Bir örnek ile inceleyelim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eski yaklaşımda finally bloğunda açmış olduğumuz kaynakları close metotuyla kapatmamız gerekmektedir. Çünkü bu tarz I/O işlemlerinde bağlantıların kapatılması bizim sorumluluğumuzda olmaktadır.

Try-with-resources yöntemi temel grameri;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Close işlemi finally bloğuna ihtiyaç duymadan otomatik olarak yapılmaktadır.

**Try-with-resources yaklaşımı Closeable interfacesini implemente eden sınıflarda uygulanır.**

BufferedReader’ın kapsamı (scope) try bloklarında geçerlidir. Catch ve finally blokları içerisinde geçerli değildir.

Close işlemi son oluşturulan objeden ilk objeye doğrudur.

**Suppressed Exception**

Suppressed bastırılmış, baskılanmış gibi anlamlara karşılık gelmektedir. Java 7 ile birlikte Throwable sınıfına eklenen getSuppressed() metodu ile gelen bir özelliktir. Birden fazla hatanın fırlatıldığı durumlarda tüm hata mesajlarını görebilmeyi sağlar.

Suppressed Exception grameri aşağıdaki gibidir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu