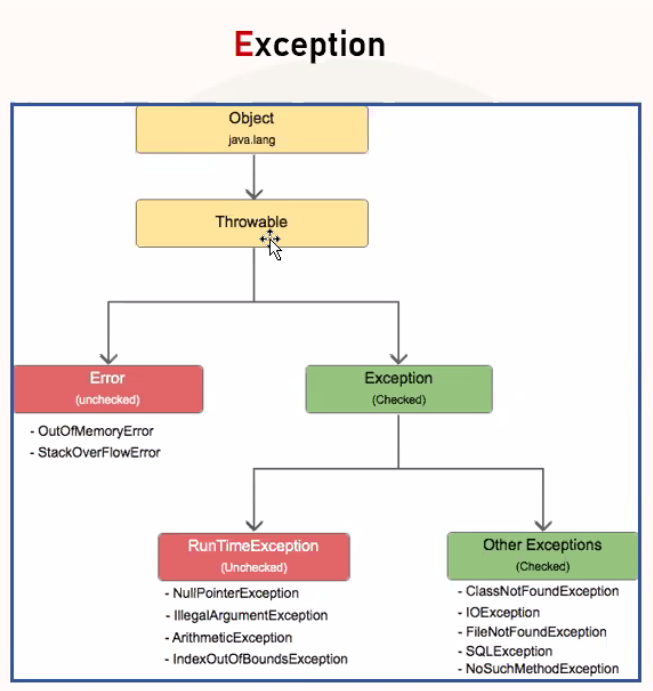
***Day 42 SOCRATIVE-EXCEPTIONS && ABSTRACT CLASSES***

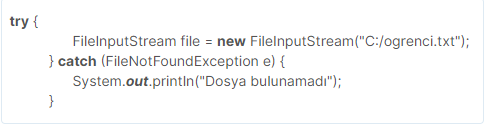
***SOCRATIVE- EXCEPTIONS***

****

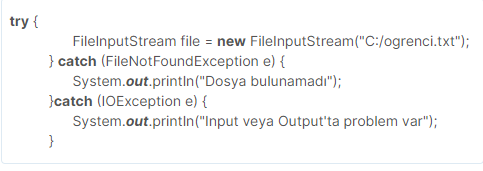
***TRUE/FALSE***

* Kodumuzun çalışması için **Compile Time Exception**'lar handle edilmelidir.
* **Complitime exceptions**’ın diğer Adı **Checked exceptions**’dır.
* **Compile Time Exception**'lar method parantezinden sonra **try-catch block** kullanılarak handle edilebilirler.
* Compile Time Exception'lar **throws** keyword'u kullanılarak giderilebilir. *(Bu şekilde Java altını kırmızı çizmez, handle edildiği anlamına gelmez, hata olduğunu biliyorum sorumluluk bende hata verebilirsin anlamına gelir)*
* **IOException, FileNotFoundException** **Compile Time Exception**'lardan birisidir.
* **FileNotFoundException** herhangi bir sebeple dosyanın adresi yanlış yazılmış veya dosya silinmiş olduğunda dosyaya ulaşılamadığı zaman ortaya çıkar.
* **IOException** Input veya Output yapılırken oluşan herhangi bir problemde ortaya çıkar.
* **FileNotFoundException**, **IOException** ‘in child class'idir.

**Try catch bloğu**

****

**Try body si içerisinde dosya okunmak üzere link verilmiş Java dosyayı bulamama ihtimali için (FileNotFoundException) “Dosya bulunamadı uyarısı vermesini istiyoruz.**



İki tane **catch** yapıldığında daha geniş kapsamlı olan daha çok hatayı bulan altta olmalıdır. Tersi yapıldığında Java alttakine ulaşılamayan kod hatası verir.

**ArithmeticException**:

**int** num1 = 6;

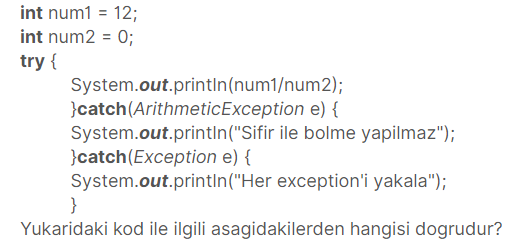
**int** num2 = 0;

System.**out**.println(num1**/**num2);

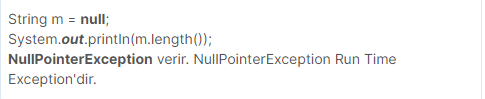
> **ArithmeticException** verir.

>Kod çalışmadan bunun hata olup olmadığını Java anlamaz CTE vermez

>**ArithmeticException** bir runtime exceptiondir

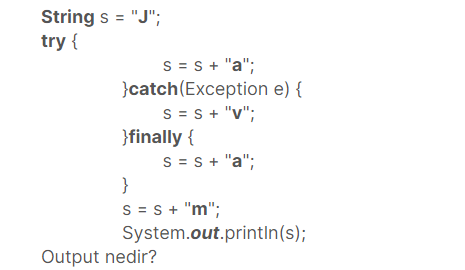


İlk **cath** bloğu olan daha spesifik AritmeticException uyduğu için “Sıfır ile bölme yapılmaz” yazdırır.

***NullPointerException***

Null in uzunluğunu bulmaya çalışırsak **NullPointerException** verir

***Finally***



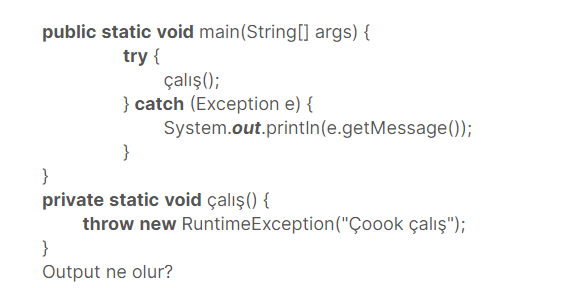
Exception olmadığı için catch çalışmaz Finally mutlaka sonunda çalışır //**Jaam**

**Throw-Throws**

Throw method body si içine yazılır, throws method parantezi ile curly brace arasına yazılır.

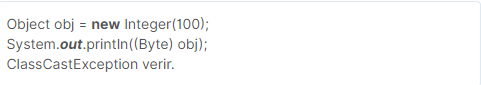
**Throw’u** method body'si içinde **exception** almak istediğimiz yerde istediğimiz kadar kullanabiliriz **throws** ise mecburen method parantezinden hemen sonra bir kere kullanılabilir

1 **throw** bir tane exception üretir, 1 **throws** birden çok exception üretebilir.



**Try** bloğunda çalış metoduna gidilecek çalış metodu **Runtime** exception verecek hata mesajı olarak Çook çalış yazacak, sonra catch exception yakalayacak ve hatanın üzerindeki mesajı verecek (e. getMessage ())

***ClassCastException***



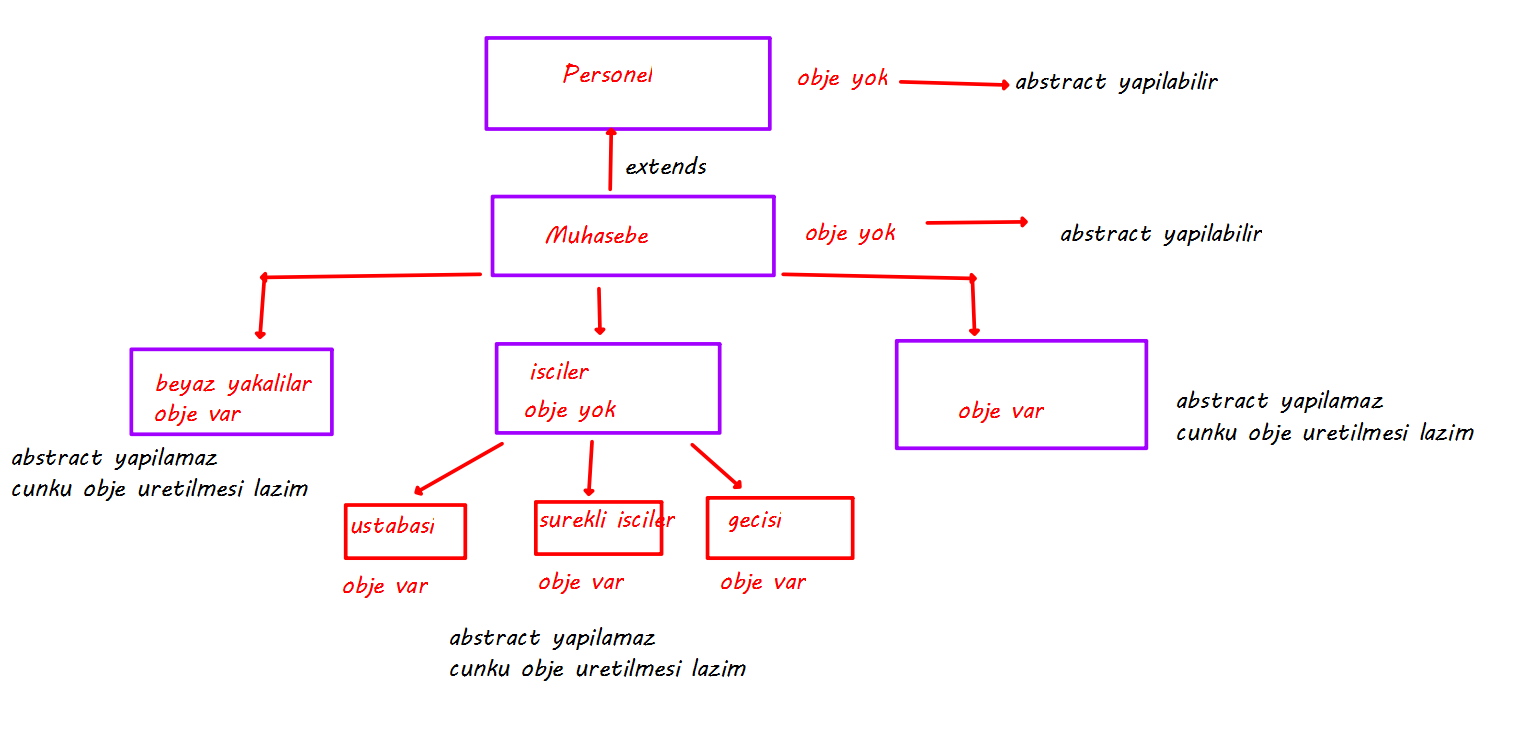
**Data casting** primitive data turu ile yapılır. **Wrapper** classlar immutable dır cevirilemezler. **Non primitive** bir obje de data casting olmaz **ClassCastException**verir.

**ABSTRACT** **CLASSES**

Eger bir class'dan OBJE OLUSTURULMASINI istemiyorsak ve bu class'i sadece CHILD CLASS'larin yapmalari gereken gorevleri belirlemek icin kullaniyorsak o classı abstract class yapariz.

**Abstract** kelime anlamı sanal demek sanallaştırma anlamına gelebilir. Bir Class’ı **Abstract** yapmanın iki yolu vardır.

Classdan önce **abstract** yazmak veya **interface** oluşturmaktır.



Kendisinde obje barındırmayan ve child classların catısı durumunda olan classları abstract class yapabiliriz.(Personel Classı)

Bir parent class **Abstract** yapılınca da parent olmaya devam eder . **Abstract** classlarda **default constractor** bulunmasına ragmen obje üretilemez.

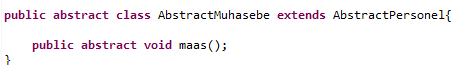
Obje oluşturdugumuz en alttaki class, parent classlardaki methodları override edince, oluşturulan objeler **en dogru** ve **specifik** ozelliklere kavuşur(Ustabaşı Classı). Aynı zamanda tüm child class’lar **ayni dinamik özelliklere** (method) sahip olurlar.

Klasık parent child ilişkisinde tum methodları override etme mecburiyeti yoktur. Java parent classlardaki tüm methodları override etmek mecburiyeti kazandırmak için **abstract** class yapısını oluşturmustur.

Yani bir classı abstract olarak tanımlarsanız child classlar tum abstract methodları override etmek zorunda kalır (orn bir ustabası classı maas methodu mesai metodu statu metodu yapmak zorunda)

Dolayisi ile abstract yaptıgımız parent classlar sadece kalıp görevi yapacaklar

**Abstract Class nasıl oluşturulur?**



Bir classı abstract yapmak için class dan once **abstract** yazmak yeterlidir.

**Class** ve **method** **abstract** olarak tanımladıgımda child classlar mecburen bu methodu **override** edecekler ve bu methodun body si hiçbir zaman çalışmayacak bu **yüzden abstract methodların** body’si yoktur.

Bir **child class** eger **abstract parentin** methodlarını override etmezse Java bunların override edilmesini ister ve CTE verir.

Abstraction; Full abstraction ya da Kısmi Abstraction olarak ikiye ayrılır.

Full Abstraction yapıldığı zaman sadece abstract methodlar olur (Interface).

Partial (Kısmi) abstraction da abstract methodlar ve concrete methodlar olur

**Concrete** bir **child class**, **parent'i** olan **abstract class**'lardaki tüm **abstract method**'lari **implement** etmelidir ama **concrete method**'lari implement etmek zorunda değildir. Opsiyonel olmasını istediğimiz özellikleri concrete method olarak oluşturabiliriz (Özel Sigorta).

Abstract class, genellikle ortak özellikleri olan nesneleri tek bir çatı altında toplamak için kullanılır.

**Abstract method nasıl oluşturulur?**

Abstract method oluşturmak için **abstract keyword**’u yazılmalıdır. Abstract method ’un body ’si olmaz (No implementation), body oluşturursanız CTE oluşur.

Abstract method **private**, **final** veya **static** olarak tanımlanamaz.

***SONUC***

***Abstraction Kuralları***

**1)** Concrete bir child class, parent’i olan abstract class’lardaki tüm abstract method’lari implement etmelidir, diğer turlu CTE oluşur.

**2)** Abstract method’lari implement etmek için öncelikle overriding yapılmalıdır**.**

**3)** Abstract class, concrete metotlara da sahip olabilir. Concrete methodlar implement edilmek ZORUNDA DEGILDIR. Parent-child ilişkisi ile kullanılabilirler veya istenirse override edilebilirler

**4)** Abstract bir child class, parent’i olan abstract class’lardaki method’lari implement etmek ZORUNDA DEGILDIR. Parent-child ilişkisi ile tüm method’lara ulaşabilir.

**5)** Abstract class’lar somutlaştırılamaz (can not be instantiated) yani abstract class’larda obje oluşturulamaz.

**6)** Abstract class’lar private veya final olarak tanımlanamaz