**DAY-41 EXCEPTİONS-2**

EXCEPTİON NEDİR?

Türkcesi istisnadır. Java çözemediği bir sorunla karşılaştığında exception classı devreye gırer. Exception classı bize karşılaşılan sorunun cinsini , sorunun nereden kaynaklandığını ve sebebini verir.

Exception fırlatıldığında kodun çalışması durdurulur(stop exception)

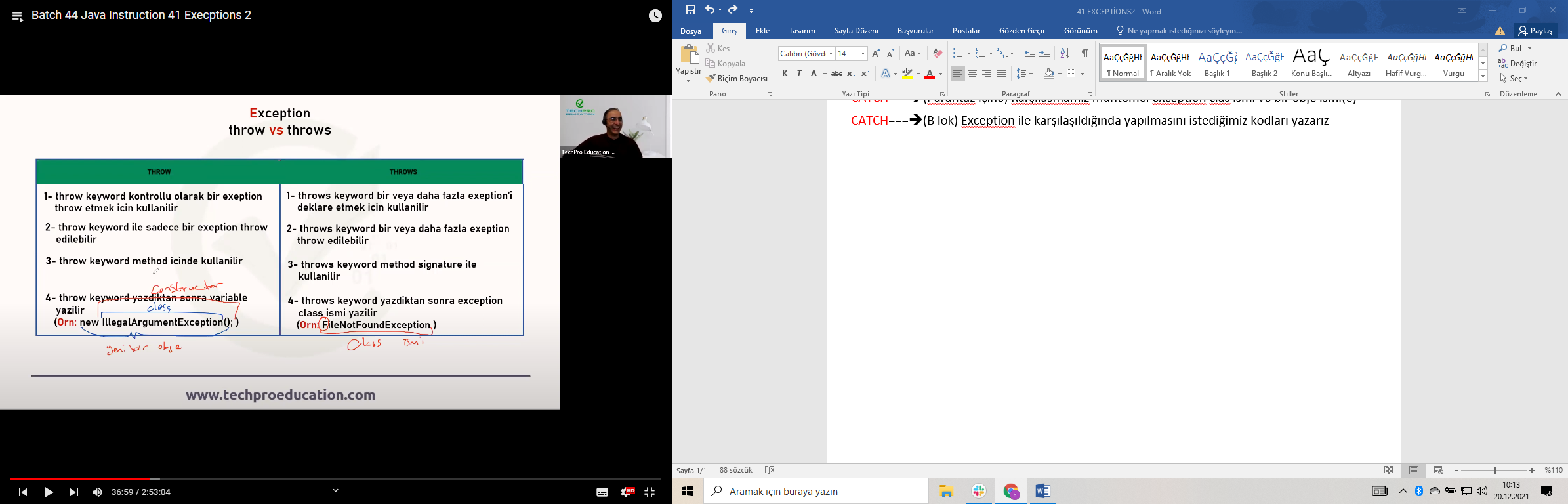
Eger sorunla karşılaşılırsa java exceptıon oluşturacak bir kod için bizden karar vermemizi ister. Hiçbir şey yapmazsak kod çalışmaz-.

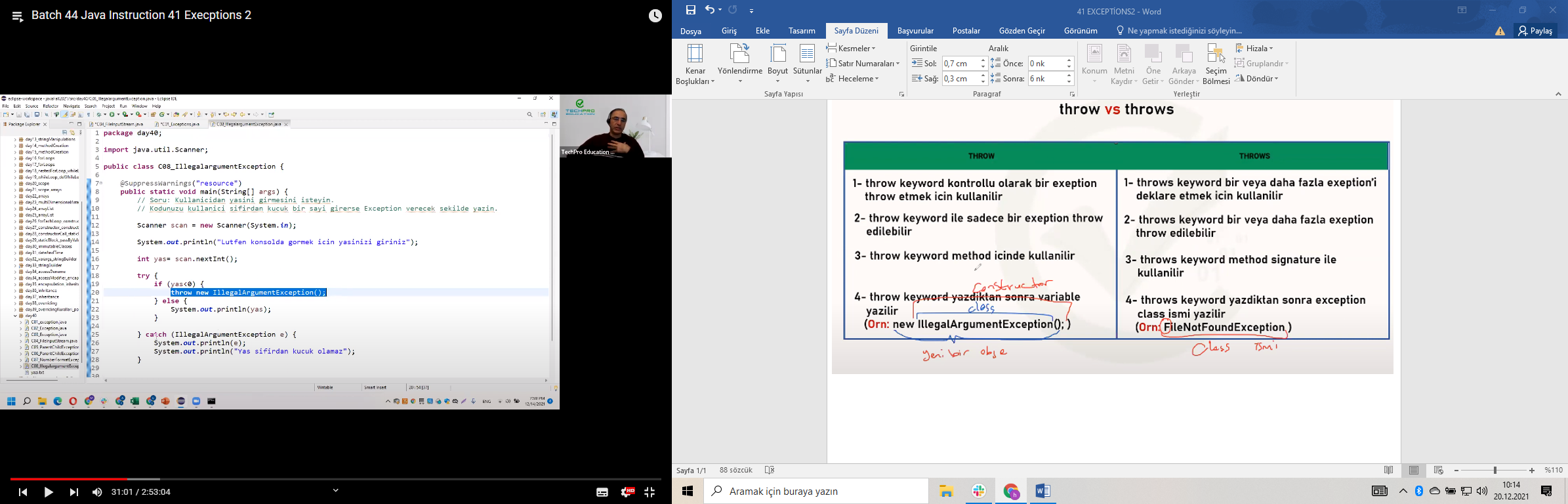
Kodumuzun çalışmaya devam etmesini istiyorsak exception oluşturması muhtemel kodlarımızı TRY-CATCH bloğu içinde yazmalıyız.

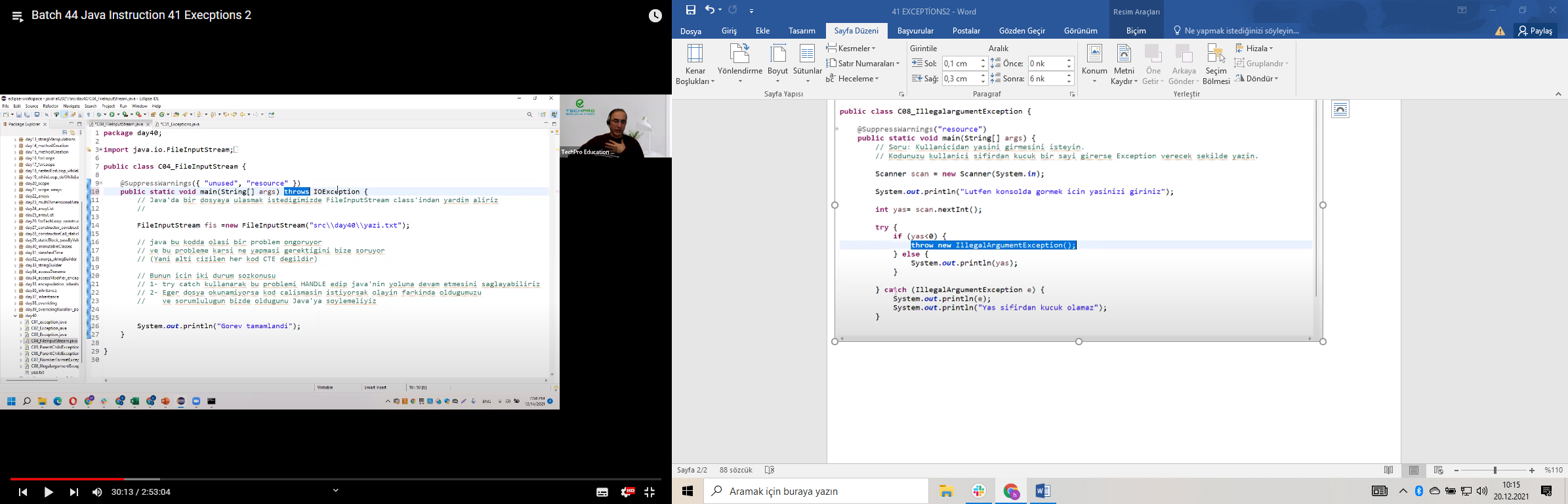
TRY ===🡺 Çalışmasını istediğimiz kodlar

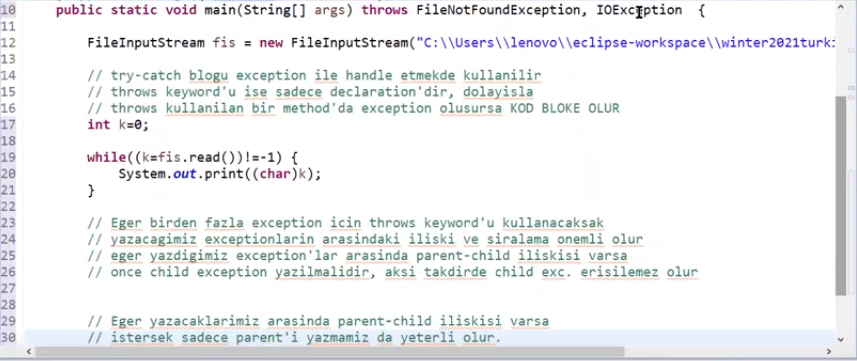
CATCH===🡺(Parantaz içine) Karşılasmamız muhtemel exception clas ismi ve bir obje ismi(e)

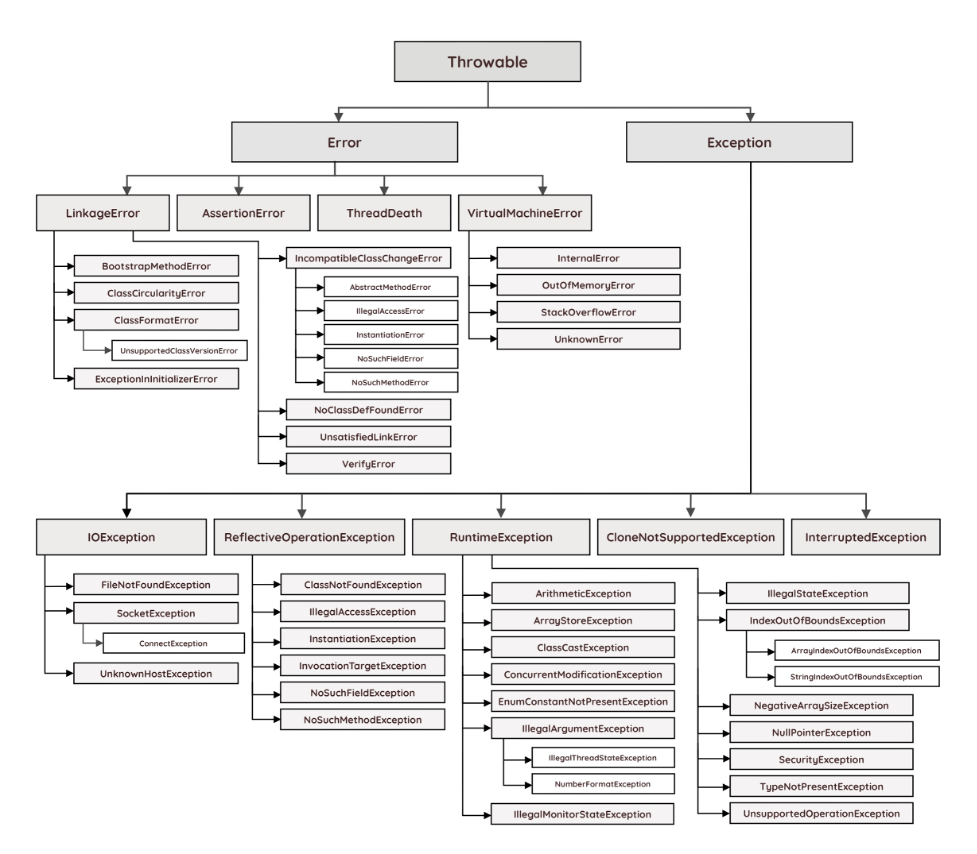
CATCH===🡺(Blok) Exception ile karşılaşıldığında yapılmasını istediğimiz kodları yazarız







****

****

**FİLENOTFOUNDEXCEPTİON**

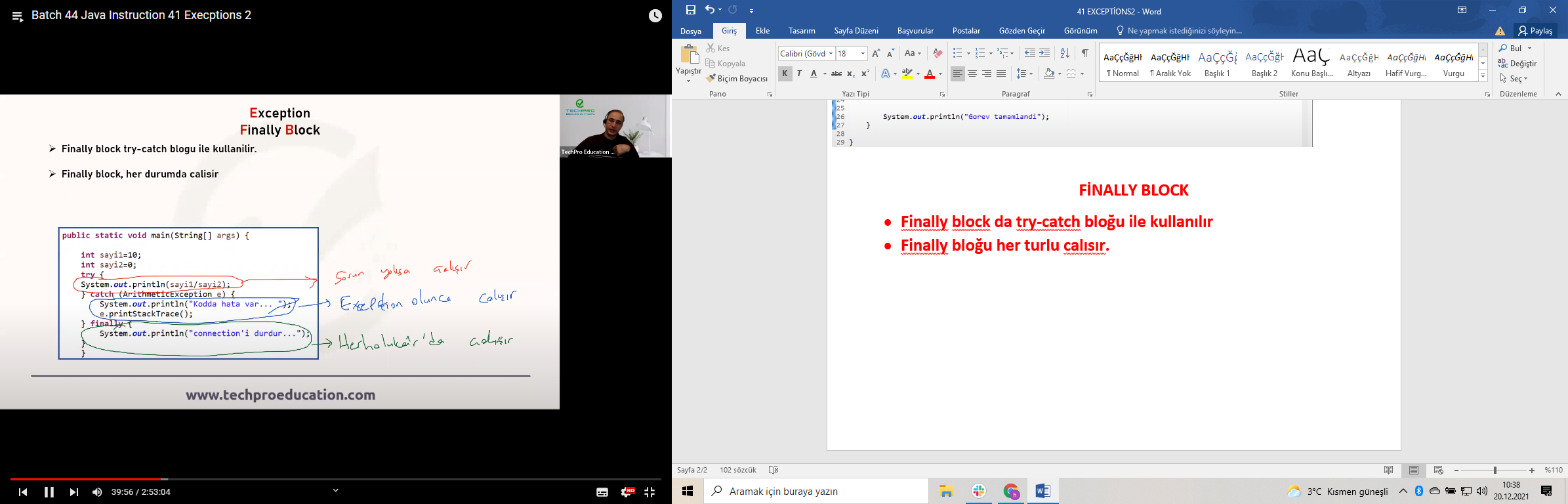
Program bir dosyayı okumaya çalışıyorken dosya bulunmazsa oluşur. IOEXCEPTİON in subclassıdır**.**

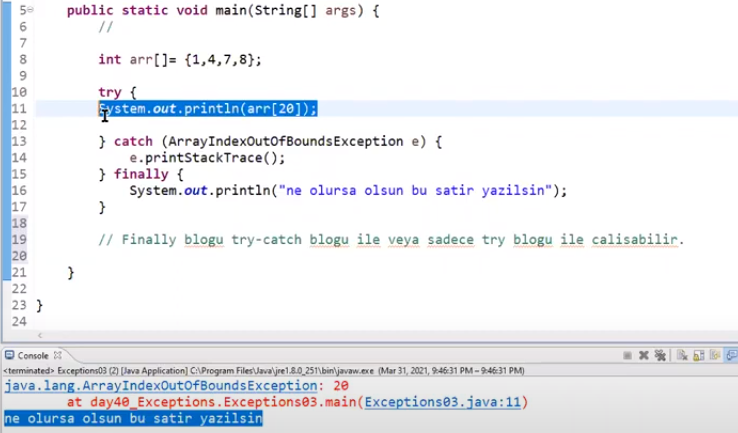
**IOEXCEPTİON**

Programımızdakı bir file a İNPUT/OUTPUT yazılıyorsa ve program çalışırken bir problem çıkarsa oluşur. Kod yazılırken mutlaka HANDLE edilmelidir.

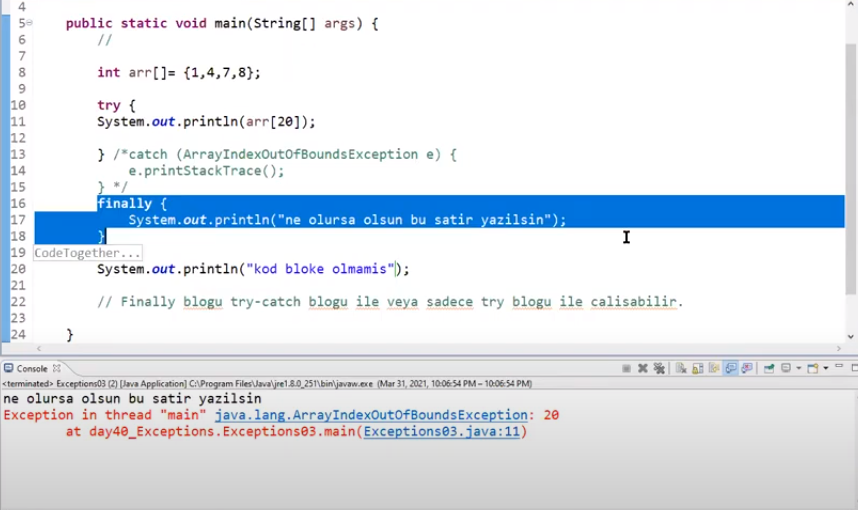
**FİNALLY BLOCK**

* Finally block da try-catch bloğu ile kullanılır
* Finally bloğu her turlu calısır.
* Öyle bir kod yazalım ki exception olsada çalışsın olmasada çalışsın.
* Birden fazla CATCH BLOGU olabilir.
* (Öngörmediğimiz bir hata olursa ve kod date-base i bozmasın ıstersek kullanırız bir nevi SİGORTA
* Birden fazla CATCH bloğu varsa CHİLD olan önce yazılmalıdır.
* Finally block catch olmadan sadece try ile de kullanılır
* TRY-CATCH-FİNALLY ----YA DA----TRY-FİNALLY





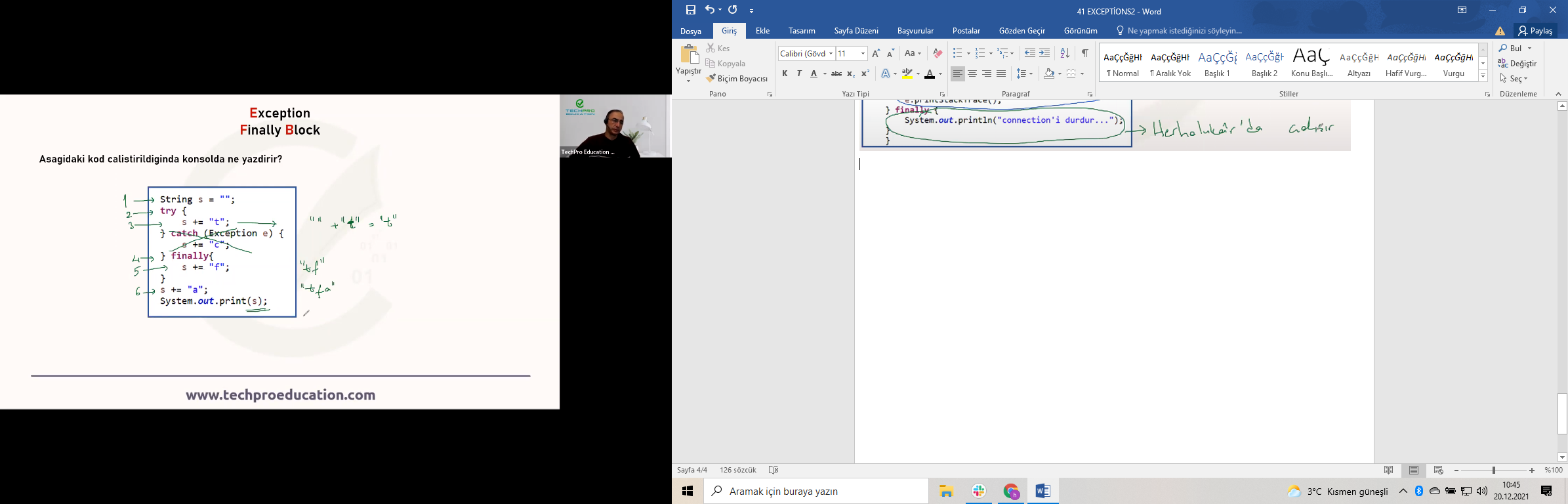
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

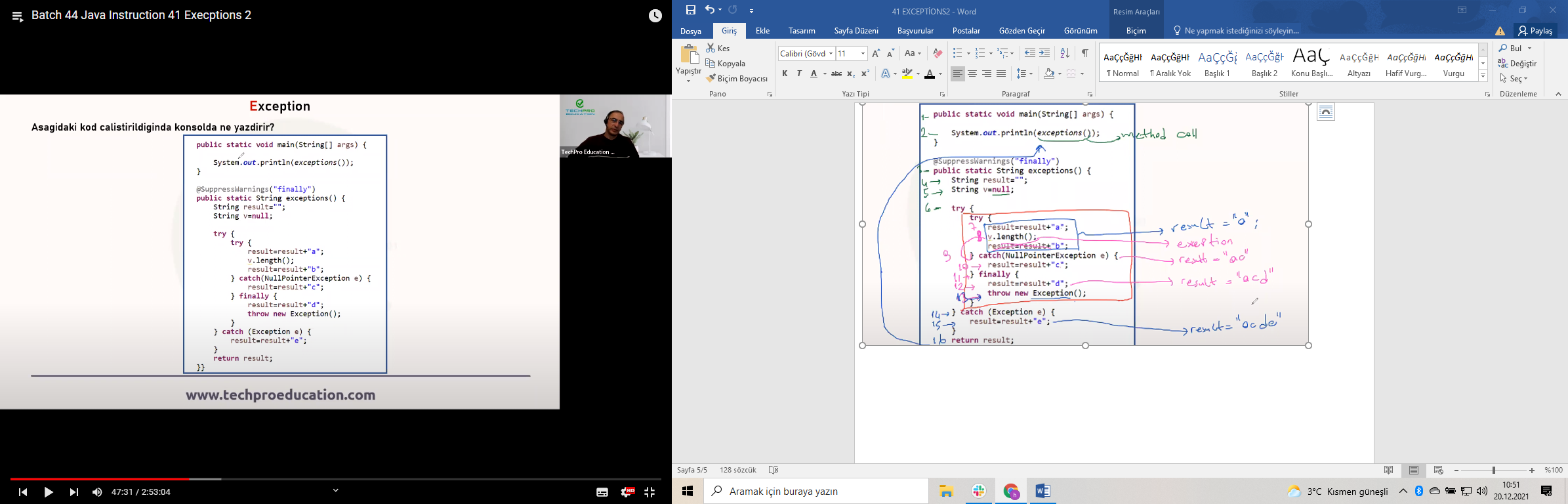
**Kod bloke oluyor ama yınede fınally blok çalışıyor!!**

**ÖRNEK**



**Concat yapar- try calısır-cacch çalışmaz- fınally calısır!**

**ÖRNEK**



Method call

“”

Null

a

nullpoınterexceptıon

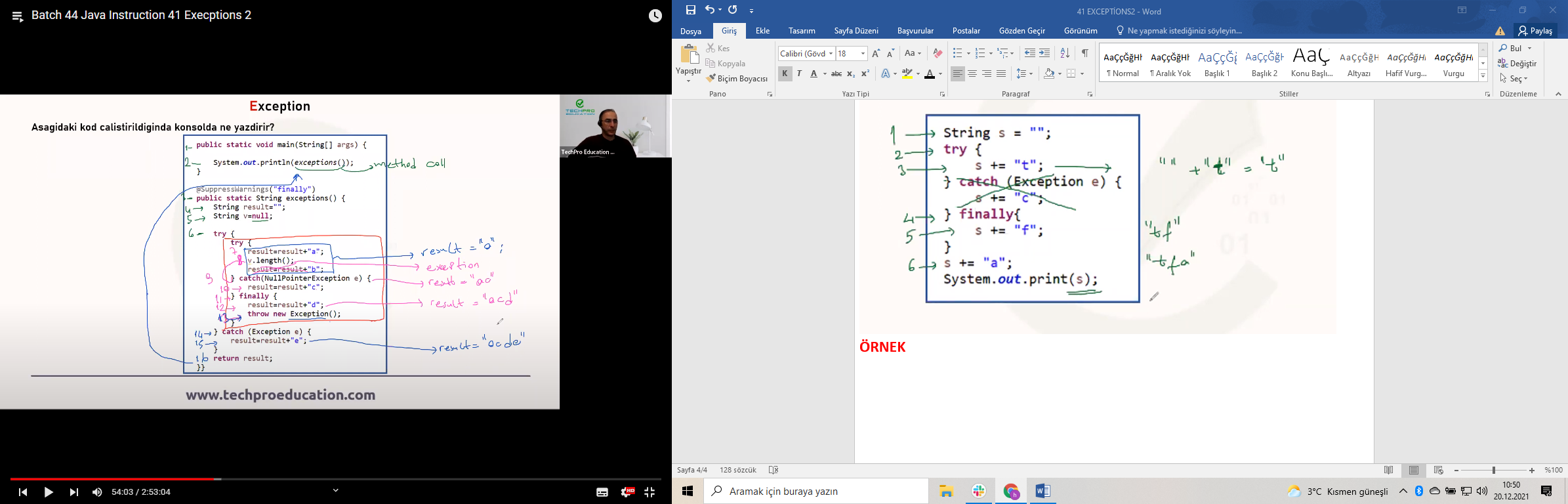
a+c=ac

ac+d=acd

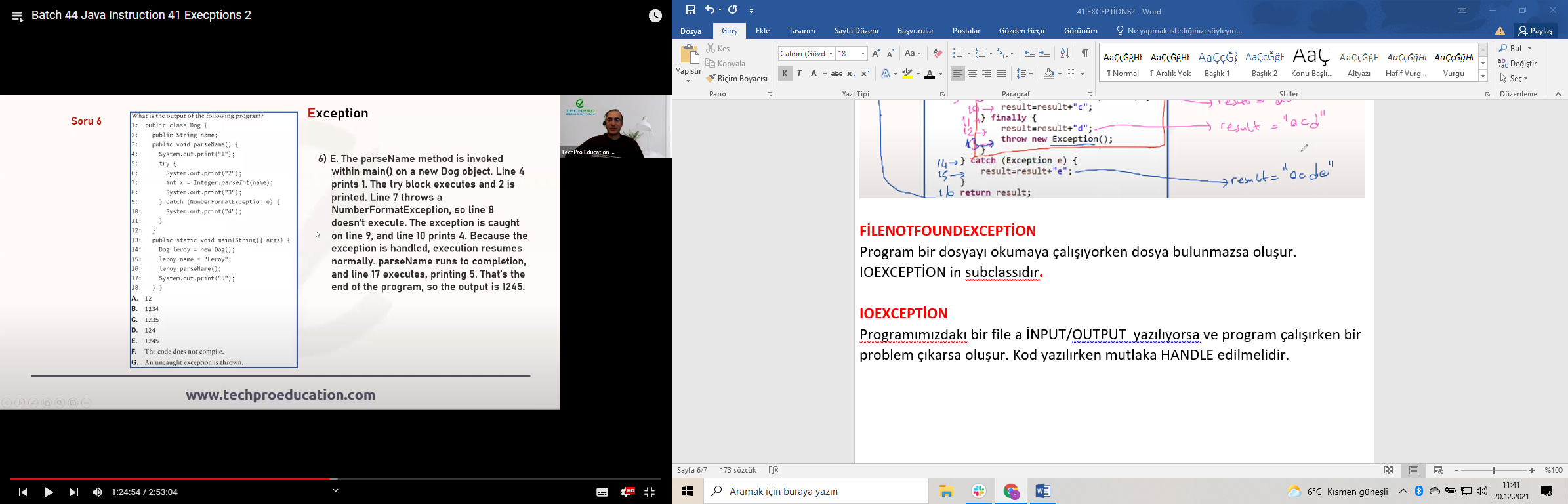
exceptıon fırlattı

catch de exceptıon yakaladı

acd+e=acde…



**ÖRNEK**



ERROR ILE EXCEPTION FARKLARI

* ERROR DA HATALAR ONGORULEMEZ
* HANDLE EDILMEZ
* TRY-CATCH MANTIGI YOKTUR.
* JAVA ERROR A KARŞI BİLGİ SAHİBİ OLMADIGI İÇİN CHECK YAPMA İTHİMALİ YOKTUR . BU YÜZDEN UNCHECKEDDIR.
* HER ZAMAN RUNTİME DA ORTAYA ÇIKAR.YAZMADAN BİLEMEZ.
* JAVA DA HATALAR =🡺 “java.lang.Error” paketi olarak tanımlanır

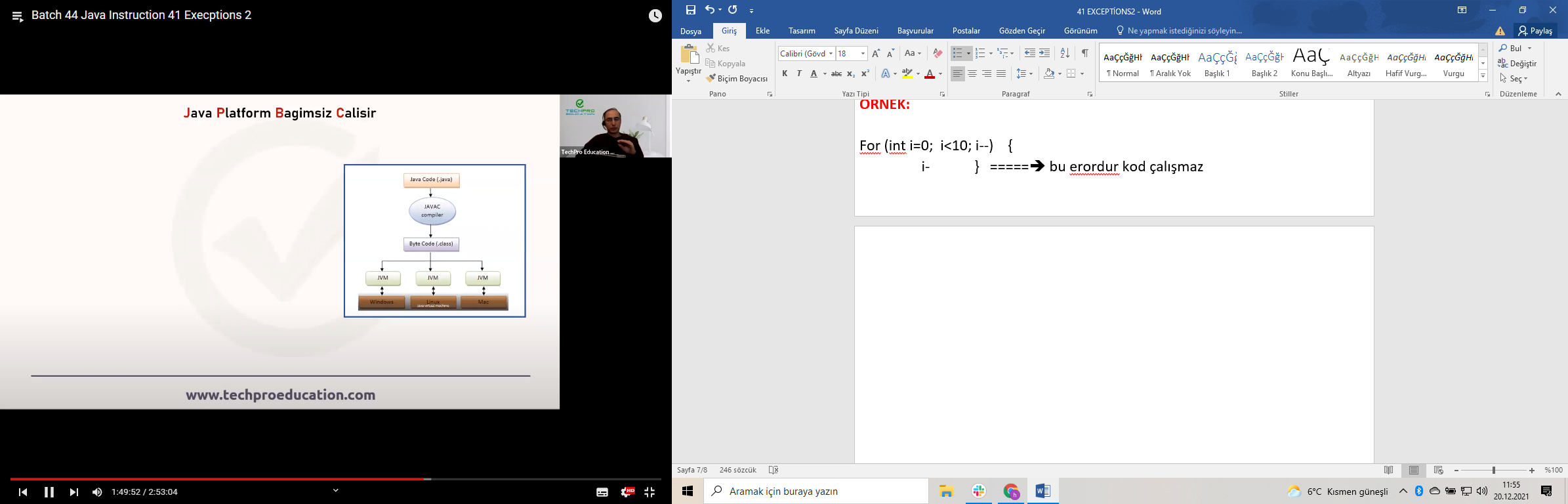
**ÖRNEK:**

String str=”asd; 🡺 error hatadır duxzeltılmeye ıhtıyacıvar .syntax hataları gibi

For (int i=0; i<10; i--) {

Syso(i); } =====🡺 bu erordur kod çalışıyor ancak , -sonsuza gıder.

**JAVA BAGIMSIZDIR**



**Write once run everywhere**

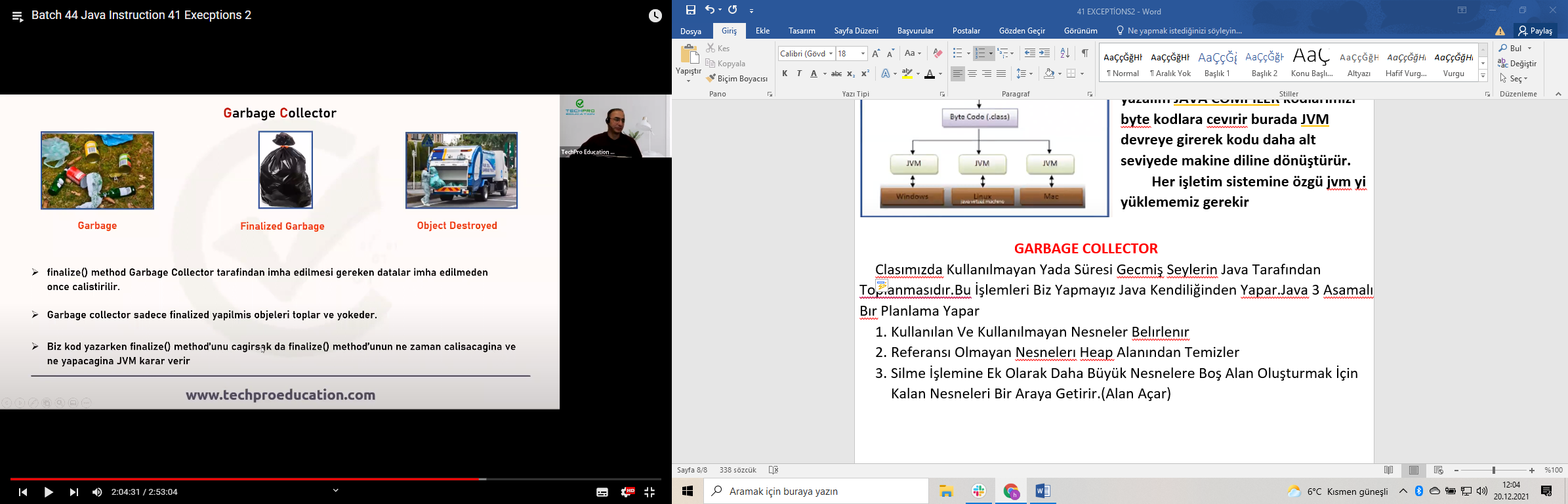
**Jvm= java sanal makınası byte codlar ıle ısletım msıstemı arasında tercümanlık yapıyor.**

**Hangi platformda yazarsak yazalım JAVA COMPİLER kodlarımızı byte kodlara**

* 1. **cevırir burada JVM devreye girerek kodu daha alt seviyede makine diline dönüştürür.**

**Her işletim sistemine özgü jvm yi yüklememiz gerekir.**

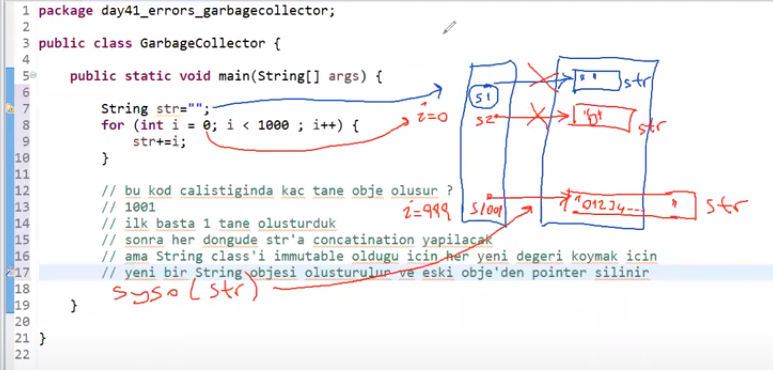
**GARBAGE COLLECTOR**

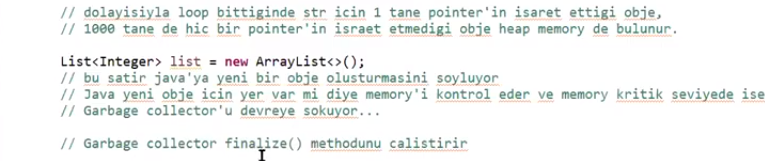


Clasımızda Kullanılmayan Yada Süresi Gecmiş Seylerin Java Tarafından Toplanmasıdır. Bu İşlemleri Biz Yapmayız Java Kendiliğinden Yapar. Java 3 Asamalı Bır Planlama Yapar

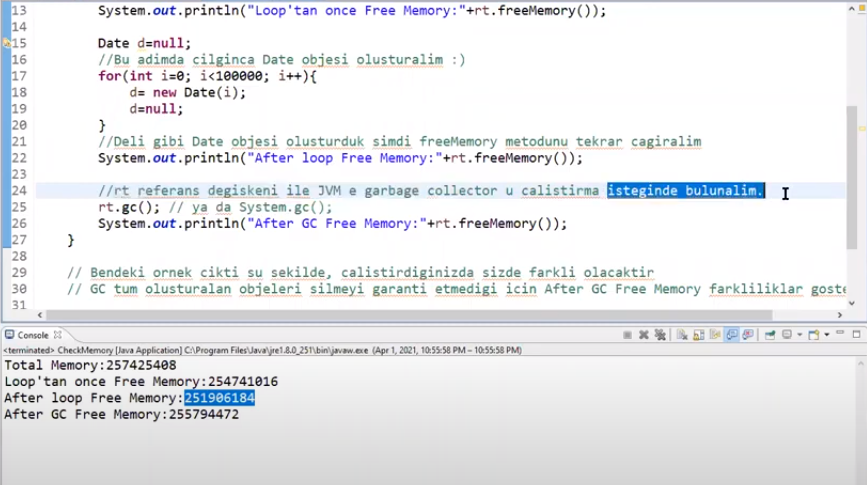
1. Kullanılan Ve Kullanılmayan Nesneler Belırlenır(İŞARETLE)
2. Referansı Olmayan Nesnelerı Heap Alanından Temizler(SİLME)
3. Silme İşlemine Ek Olarak Daha Büyük Nesnelere Boş Alan Oluşturmak İçin Kalan Nesneleri Bir Araya Getirir.(SIKISTIRMA VE SİLME🡪 ALAN AÇMA)

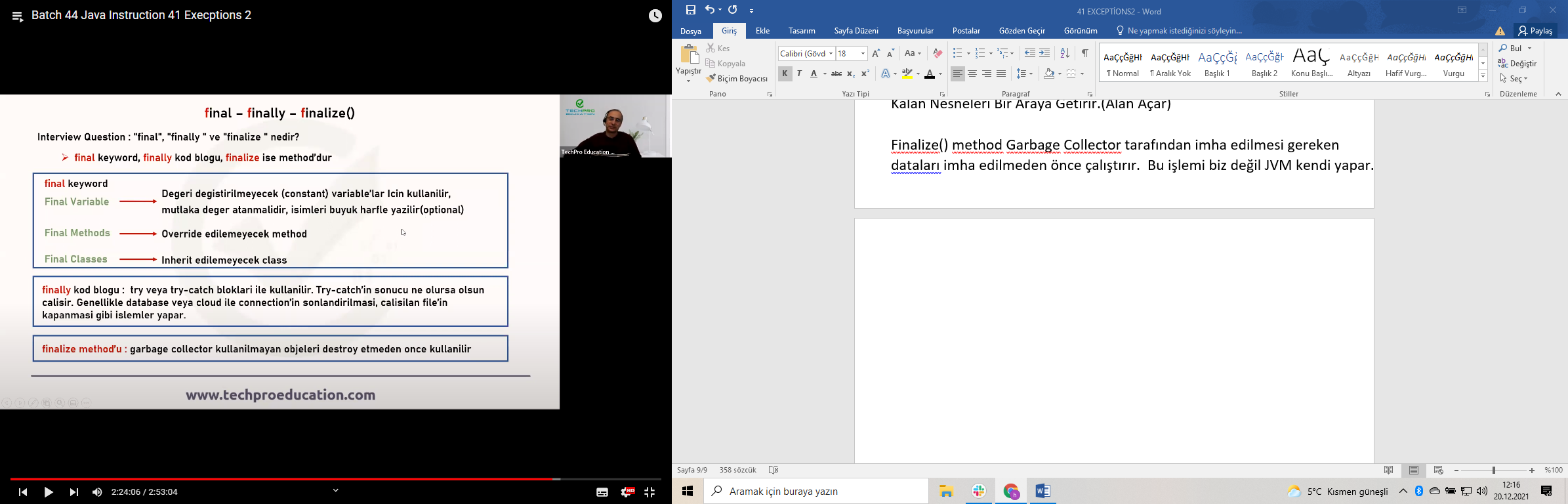
Finalize() method Garbage Collector tarafından imha edilmesi gereken dataları imha edilmeden önce çalıştırır. Bu işlemi biz değil JVM kendi yapar.



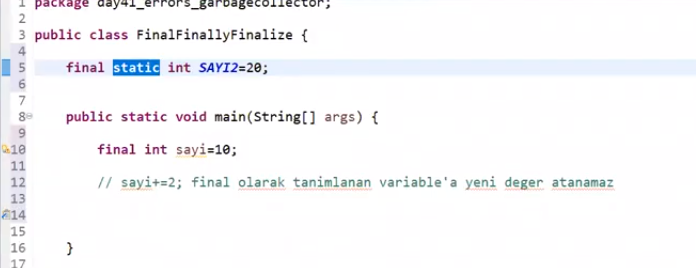


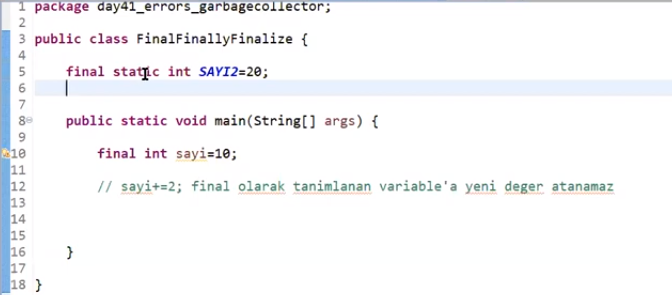
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***





**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***



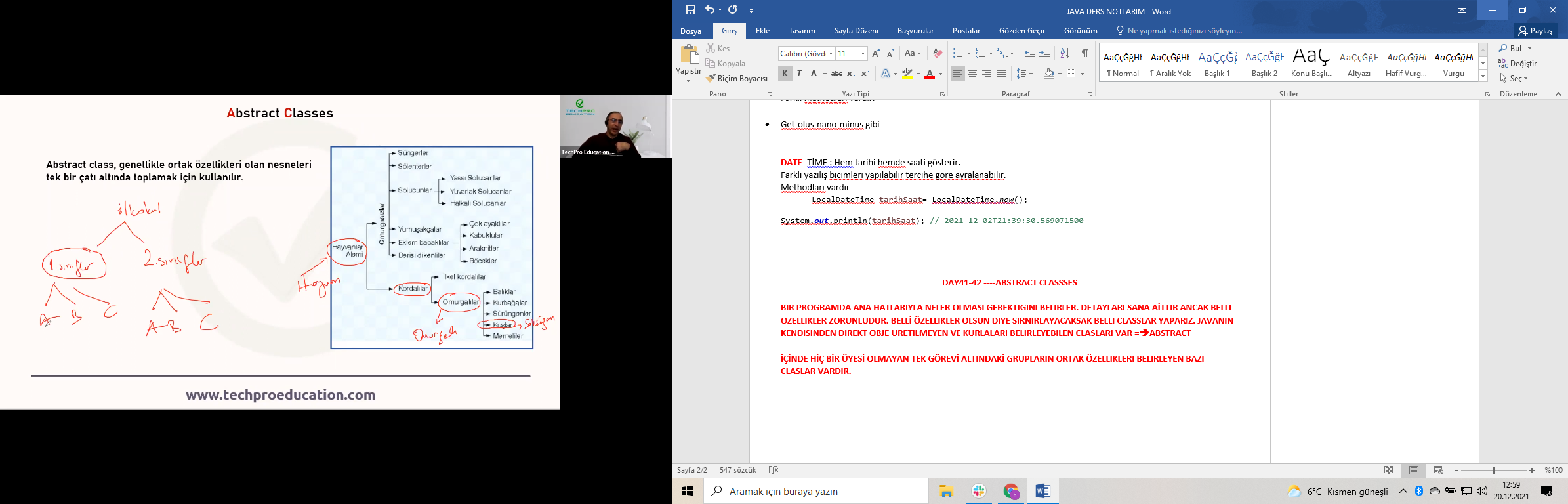


**DAY41-42 ----ABSTRACT CLASSSES**

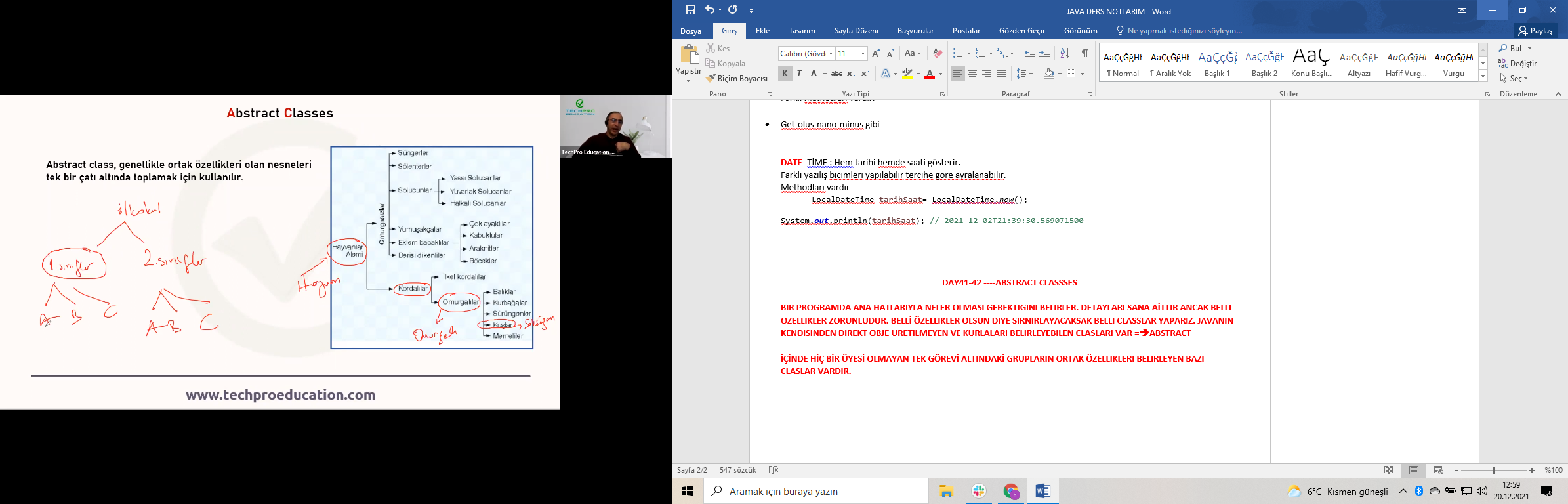
Bır programda ana hatlarıyla neler olması gerektıgını belırler. Detayları sana aittır ancak bellı ozellıkler zorunludur. Belli özellıkler olsun dıye sırnırlayacaksak bellı classlar yaparız. Javanın kendısınden dırekt obje uretılmeyen ve kuraları belırleyebılen clasları var =🡺abstract

İçinde hiç bir üyesi olmayan tek görevi altındaki grupların ortak özellıklerı belırleyen bazı claslar vardır.

Tüm child classlarda olmasını istedıgımız dınamık özellıklerı abstract classda abstract method olarak oluştururz.



Her parent class bu özelliği almaz. 1. Sınıf olupta a-b-c den birinde olmayan yok o yuzden 1.sınıflar clasında obje yoktur.

Abstract olmayan tüm child classlar abstract methodları override etmek zorundadır.

Böylece tüm child classlar aynı dinamik özelliklere sahip olurlar