Özet

- **GİRİŞ**
- Yazılım geliştiricilerinin birbiri ardına yazdığı kod satırlarında birçok işlemler yapılmaktadır. Bu işlemler yapılırken hata olmadan her zaman istenilen veya beklenilen sonuçlar ortaya çıkmamaktadır. Bazen yazılan kodlar tamamen farklı işlemler yapabilir. Bu durumlarda, kod satırlarında bir sonraki görevin sebebini anlamak veya hatanın nereden kaynakladığını tespit etmek için *hata ayıklama* işlemi yapılabilir. Bu işlemler hata ayıklama aracı ile gerçekleştirilebilir.
- Hata ayıklama araçları, tüm sorunlara veya kodlarda yer alan hataları açığa çıkarabilecek bir şey değildir. Hata ayıklama, bir programlama hatası yapıldığında kesin hata yapılan noktayı bize göstermemektedir. Visual studio gibi programlarda yer alan hata ayıklama araçları kod satırlarını adım adım çalıştırarak hatayı tespit etmenizi sağlar. Bu araç ile kodlarda düzeltmeleri göstermekte ve geçici değişiklikler yapmanıza izin vermektedir. Ayrıca yazılan kodlarda hatalara karşı ne kadar önlem alsak da başka sebeplerden kaynaklanan hatalar hep olacaktır. Örneğin bir metin kutusuna kullanıcının bilgi girmesi gerekirken boş bırakması bir hataya sebep olacaktır. Bu tür durumlarda *hata kontrolü* yaparak sistem kullanıcılarının hatayı görmesini engellememiz gerekmektedir. İşte bu ünitede hata ayıklama araçları, hata ayıklama ve iz bulma sınıfları ve istisnai durumlar başlıkları altında konular örnekli olarak açıklanmıştır.
- **HATA AYIKLAMA ARAÇLARI**
- Hata, yazılan kodların derlemesinde ortaya çıkmaktadır. Hatalar genellikle kullanıcı kaynaklıdır. Örneğin, kullanıcının yazılan sistemde metin girilmesi gereken alana sayı bilgisi girildiği durumda sistemde hata meydana gelecektir. Yazılım geliştiriciler bu yüzden oluşabilecek durumlar için hataları kontrol edebilen kod satırları yazarak sistemin hata vermesini önler.
- **Hata Türleri**
- Hatalar; yazılım, çalışma zamanı ve mantıksal hatalar olmak üzere üç çeşittir.
- **Debug Menüsünün Elemanları**
- Bu menün elemanları incelendiğinde bunlar; start, restart, stop debugging, step into, step over, step out ve watch'dur.
- **TaskList (Görev Listesi Penceresi)**
- TaskList penceresinde, sistem içerisinde tanımlanmayan fonksiyon, değişken, metot veya yazım hataları gösterilir.
- **Breakpoint (Kesme Noktası) ve Breakpoint Penceresi**
- Kodların derlenmesi sırasında sistemi istediğiniz bir noktada durdurmak için kod satırına breakpoint (kesme noktası) ekleyebilirsiniz. Böylelikle sistem sizin işaretlemiş olduğunuz breakpoint noktasında duracaktır. Klavyeden F9 kısayol tuşu veya sağ tıkla açılan menüde Insert Breakpoint komutu ile kod satırına breakpoint ekleyebilirsiniz.
- **Output Penceresi**
- Sistemin derlenme aşamalarının gösterildiği penceredir. Klavyeden kısayol tuşu Ctrl+Alt+O'dur. Output penceresinde bulunan açılır listede dört seçenek vardır. Bunlar; derleme, derleme sırası, hata ayıklama ve paket yöneticisidir.
- **HATA AYIKLAMA VE İZ BULMA SINIFLARI**
- **Trace ve Debug Sınıfları**
- Dosyalama mekanizmasının (Log) oluşturulması ve yönetilmesi işlemleri trace ve debug sınıflarının kullanımı ile yapılmaktadır. Bu sınıfları kullanabilmek için using deyimi ile birlikte System.Diagnostics isim uzayının uygulamanın başına yazılması gerekmektedir. Proje geliştirme aşamasında hata mesajlarını, proje uygulanmasında ise gerekli dosyalama bilgilerini gerekli yerlere yazdırmak açısından bu iki sınıf çok fazla kolaylık sağlar. Bu iki sınıfın tüm özellikleri aynıdır. İki sınıf arasındaki tek fark Debug sınıfının Realese uygulaması içerisinde derlenmesidir.

Özet (devamı)

- **İSTİSNAİ DURUMLAR**
- Bir projenin hatasız yazılmış olması o projenin hiç hata vermeyeceği anlamına gelmez. Gerek kullanıcı gerekse diğer durumlardan dolayı doğru yazılmış kodlarda hata verebilir. Bu yüzden hataların oluşumunu her zaman kontrol altında tutamayabiliriz. Fakat oluşabilecek hatalara karşı önlemler alabiliriz.
- **Try – Catch Kullanımı**
- Try-catch tüm programcılar tarafından en sık kullanılan hata ayıklama kod bloğudur. Oluşabilecek hata kodları try bloğunda, hata durumunda işletilecek kodlar ise catch bloğunda yazılır. Eğer hata olmazsa catch bloğundaki kodlar çalıştırılmaz. Hata olması durumunda hata oluşan kod satırından catch bloğuna kadar olan kod satırları işletilmez. Catch bloğundaki kod satırları işletilir.
- **Finally Kullanımı**
- Hata oluşması durumunda kullanılacak kod bloğu yoksa programın hata vermesini engellemek için finally bloğu kullanmaktadır. Finally bloğunun özelliği hata olsun ve olmasın çalışan bir bloktur.
- **Throw Rethrow Kullanımı**
- **Throw kullanımı**
- Throw hatayı gönderme anlamına gelir. Bazen hatalar bilerek oluşturulmak istenilebilir. İşte bunun için *throw* komutu kullanılır. Kod bloğunda eğer throw ifadesi kullanılmışsa program ifadenin kullanıldığı noktada durarak istenilen istisnayı üretir.
- **Rethrow kullanımı**
- Hatayı yeniden gönderme anlamında kullanılır. Eğer projede birden fazla catch bloğunun kullanıldığı zaman rethrow kullanılır. Bu durum bir catch bloğunda yakalanan istisnai durum diğer catch bloğunda da yakalanabilsin diye kullanılır.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi hata türüdür?
   a) Mantıksal hata
   b) Zamansal hata
   c) Verisel hata
   d) Görsel hata
   e) İşlemsel hata

2. Değişken, yapı ve fonksiyon adları ve komutların yanlış yazımından kaynaklı oluşan hatalar hangisidir?
   a) Mantıksal hata
   b) Zamansal hata
   c) Yazılım hatası
   d) Görsel hata
   e) İşlemsel hata

3. Aşağıdakilerden hangisi Debug menüsünün elemanlarından değildir?
   a) Start
   b) Restart
   c) Step Into
   d) File
   e) Watch

4. Aşağıdakilerden hangi adımlar izlenerek TaskList penceresi açılır?
   a) Project->Other Windows->Task List
   b) Project->Other->Task List
   c) Project->Windows->Task List
   d) Project->Options->Task List
   e) Project->Select->Task List

5. Aşağıdakilerden hangi koşul Breakpoint kullanımında ayarlanamamaktadır?
   a) Breakpoint istenilen bir sayıda projeyi durdurur.
   b) İstenilen bir sayının katlarında projeyi durdurur.
   c) Kod derlendiği zaman her breakpoint'e geldiği zaman proje durur.
   d) İstenilen bir sayı ve üzerindeki sayılar için proje durur.
   e) Kod derlendiği zaman her breakpoint'e geldiği zaman proje devam eder.

6. Sistemin derlenme aşamalarının gösterildiği pencere aşağıdakilerden hangisidir?
   a) Output Penceresi
   b) Breakpoint Penceresi
   c) Görev Listesi Penceresi
   d) Proje Penceresi
   e) Step out Penceresi

7. Debug ve Realese sınıfları arasındaki fark aşağıdakilerden hangisidir?
   a) Debug sınıfının Realese uygulaması içerisinde derlenmesi
   b) Realese sınıfının Debug uygulaması içerisinde derlenmesi
   c) Using deyimi ile birlikte kullanılır
   d) Proje uygulamasında gerekli dosyalama bilgilerinin yazdırılması
   e) Dosyalama mekanizmasının oluşturulmasında kullanılır

8. Her koşulda çalışacak kod satırları hangi blokta yazılır?
   a) Catch
   b) Finally
   c) Try
   d) Throw
   e) Rethrow

9. Trace ve Debug sınıflarını kullanabilmek için aşağıdaki uzay isimlerinden hangisi proje eklenmelidir?
   a) IO
   b) Windows.Forms
   c) Data
   d) Diagnostics
   e) Database

10. Aşağıdakilerden hangisi hata ayıklama işlemini sonlandırır?
    a) Step Into
    b) Step Over
    c) Step Out
    d) Watch
    e) Start

Cevap Anahtarı
1.a, 2.c, 3.d, 4.a, 5.e, 6.a, 7.a, 8.b, 9.d, 10.e