 **İSTANBUL ŞEHİR ÜNİVERSİTESİ 2019-2020 EĞİTİM ÖĞRETİM YILI GÜZ DÖNEMİ  
NESNE TABANLI PROGRAMLAMA DERSİ PROJE SINAVI**

**Adı Soyadı:   
Öğrenci Numarası:**

**QUIZ – 3**

Abstract Class ve Interface yapılarını inceleyen bu araştırmada örneklerle bu iki yapının 15 tane sınırlılıklarını tespit edeceksiniz. Karşılaştırarak sınırlılıklarından dolayı oluşan hataları, görüntüleri ile beraber açıklayacaksınız. Aşağıda ilk satır örnek olarak verilmiştir.

|  |  |  |
| --- | --- | --- |
| **ÇÖZÜMLER** | | |
| **ÖZELLİKLER** | **ABSTRACT CLASS** | **INTERFACE** |
| Abstract Class ve Interface’ler SEALED (mühürlenmiş) olamazlar. | Abstract Class’lar sealed olarak tanımlandığı zaman **“Bir soyut sınıf korumalı veya statik olamaz”** hatası verir. | Interface yapılar sealed olarak tanımlandığı zaman **“sealed değiştiricisi bu öge için geçerli değildir”** hatası verir. |
|  | Abstract Class içerisinde tanımlanan **abstract methodlar** **private**(özel) olamaz. | Method tanımlarında **private(özel)** yapılamaz. |
|  | Abstract Class içerisinde tanımlanan **abstract methodların gövdesi olmaz.** | Interface yapılar içerisinde method deklarasyonunda method gövdesi olmaz. |
|  | Abstact Class’tan **yeni bir nesne(örnek) türetilemez**. | Interface yapıdan yeni bir örnek üretilemez. |
|  | **Miras** alan Class içerisinde Abstact Class’ta belirlenmiş **methodlar implement**(uygulanma/ekleme) edilmelidir. | Class tanımlandıktan sonra interface içerisindeki methodlar implement edilmelidir. |
|  | Miras alma işlemi uygulanan Class’tan **yeni bir örnek(nesne) üretilirken Abstract Class** kullanılamaz. | Interface yapıda yeni bir örnek üretilirken interface new olarak kullanılamaz. |
|  | Abstract Class’lar **static(sabit)** olamaz. | Interface tanımlanırken static olamaz. |
|  | Abstract Class’lar **override(geçersiz kılma)** yapılamaz. | Interface override olarak tanımlanamaz. |
|  | Bir Class birden çok Abstract Class’tan **miras alma gerçekleştiremez.** | Birden çok interface bir class için kullanılabilir. |
|  | **Implement** edilen method **override** olarak tanımlanmalıdır. | Interface yapıda method override olarak tanımlanmaz. |
|  | Abstract Class’ta tanımlanan private field(özellik) için **getter ve setter oluşturulmalıdır.** | Interface içerisinde filed(özellik) tanımlanamaz. |
|  | Aynı adda ve aynı parametrelere sahip bir method tanımlanamaz. | Implement edilen aynı adda ve parametrelerde bir method varsa yenisi tanımlamanaz. |
|  | Abstract Class içerisinde tanımlalan abstract method **static(sabit) olamaz.** | Interface yapıda yapılan method deklarasyonunda static tanımı olamaz. |
|  | Abstract Class içerisinde abstracT olmayan method dolu olabilir. | Interface içinde hiçbir şekilde method dolu olamaz. |
|  | Abstrack Class hızlıdır. | Interface hız bakımından yavaştır. |
|  | Kalıtım sağlanması isteniyorsa Abstract Class kullanılmalıdır. | Interface ile kalıtım sağlanamaz. |

FARKLAR

* interface ve abstract class’lar new anahtar sözcüğü ile oluşturulamazlar.
* Bir sınıf birden fazla interface’i kalıtım olarak alabilir ama bir sınıfa bir tane abstract class kalıtım alınabilir.
* Interface içerisinde boş metodlar tanımlanabilir ama abstract class’larda hem boş metodlar tanımlanabilir hemde içi dolu metodlar tanımlabilir.
* Abstract sınıflar içerisinde metod gövdeleri tanımlanıp özellik değerleri ayarlandığı için genellikle sonradan üzerine ek geliştirilmek yapmak için kullanılıır ama interface de ise body ve değer set edilemediği için tamamen interface üzerinden tüm üyeleri implemente edilerek sıfırdan geliştirmeler yapılması gereken durumlarda kullanılır.
* Abstract class’lar içerisinde sadece abstract olarak işaretlenmiş metod ve özellikler implement edilmek zorundadır fakat interface içerisindeki tüm özellik ve metodlar implement edilmek zorundadır.
* Bir class bir tane abstract class’ı kalıtım olarak alabilir ama bir class istenilen sayıda interface’i kalıtım olarak alabilir.
* Interface içerisinde özellik ve metodlarda erişim belirleyiciler kullanılmaz herşey public olarak kabul edilir fakat abstract sınıflarda kullanılabilir.
* Abstract sınıflara diğer sınıf ve interface’ler kalıtım olarak geçilebilir fakat interface’e yine farklı interface’ler haricinde herhangi bir yapı kalıtım olarak geçilemez.
* Interfaceler çoklu kalıtımı sağlamaya yardımcı abstract classlar ise çoklu kalıtımı desteklemez.
* Interfacelerde metodların içerisini dolduramayız ama abstract classlarda doldurabiliriz Böylece bütün alt sınıfların belli bir özelliğe sahip olmasını sağlayabiliriz.
* Interface ile yapabildiğimiz herşeyi hatta daha fazlasını abstract classlar ile de yapabiliriz.
* Eğer türeteceğimiz classlarda belli başlı varsayılan özellikleri tekrar tekrar kopyala-yapıştır yapmak istemiyorsak o zaman abstract class kullanmamız gerekir. Çünkü abstract classlarla bir metodu tüm alt classlarda varsayılan metod şeklinde tanımlayabiliriz ve alt classlarda bunları tekrar yazmamıza gerek kalmaz kalıtımla aktarılmış olur.
* Kalıtım sağlamak istiyorsak abstract classlar kullanmamız gerekir.
* Abstract classları kullanmak hız açısından avantaj sağlar.
* Interface de yeni bir metod yazdığımız zaman bu interfaceden implement ettiğimiz tüm classlarda bu metodun içini tek tek doldurmak gerekiyor ancak abstract classlarda durum farklıdır burada bir metod tanımlayıp içini doldurduğumuzda abstract sınıfımızdan türetilmiş bütün sınıflar bu özelliği kazanmış olur.