Bilgisayar Programcılığı Eğitimine Nesne Yönelimli Metot İle Başlama

Barboros Can

Çankaya Üniversitesi, Bilgisayar Mühendisliği Bölümü

Öğretmenler Cad. No:14 Yüzüncüyıl, 06530, Ankara

bcan@cankaya.edu.tr

Özet. 1990'ların başından bu yana bilgisayar dünyasında nesne-yönelimli programlama (object-oriented programming) oldukça yaygın bir kullanım alanı bulmuştur. Bununla beraber, üniversite ya da lise düzeyinde, bilgisayar programcılığı eğitimine yapısal programlama (structured programming) ile başlanmaktadır. Bu programlama yöntemine alışan öğrenciler, psikoloji biliminde negatif transfer (ters etki) olarak adlandırılan durumun ortaya çıkması sonucu, nesne-yönelimli programlamaya geçişte zorlanmaktadırlar. Bu sorunu gidermek için programcılık eğitimine nesne-yönelimli programcılıktan başlanması önerilmektedir. Bu yazı, programcılık eğitimine nesne-yönelimli bir programlama dili ile başlanmasının getireceği zorlukları ve kolaylıkları ortaya koymayı amaçlamaktadır.

1 Giriş

Üniversitede nesne-yönelimli programlama derslerinde, daha önce yapısal-programlama yöntemiyle program yazmış olan öğrencilerin nesne-yönelimli tasarım ve programlama ile ilgili bazı basit, temel kavramları anlamakta zorlandıkları gözlenmektedir. Öğrencilerin geçmişte edindikleri bilgiler yeni öğrenimlerini zorlaştırmakta, yani öğrenmede negatif transfer (ters etki) [1] olarak adlandırılan durum ortaya çıkmaktadır. Bu çalışmada, sözü edilen soruna çözüm olarak, programcılık eğitimine doğrudan nesne-yönelimli programlamadan başlanması önerilmektedir. Bertrand Meyer'in [2] de dediği gibi: "Eğer nesne-yönelimli tasarım hedeflerinin ve tekniklerinin getireceklerinde hemfikirseniz, bunu öğrencilerinize aktarmakta gecikmeniz için hiçbir sebep yoktur; bu yaklaşımı yazılım geliştirmede ilk yöntem olarak kullanmalısınız.". Bu yüzden, bilgisayar programlama deneyimi olmayan, lise düzeyindeki öğrencilere programcılık eğitimi verilmesi gerektiğinde nesne-yönelimli programcılık yöntemi seçildi.

1.1 Programlama Dili

Yaygın olarak mühendislik fakültelerinde ve meslek yüksek okullarında, uzun süredir programcılığa giriş derslerinde programlama dili olarak "C" ya da "Pascal" öğretilmektedir. Bu dillerin, modern programlama dillerinin yetenekleri ile karşılaştırıldıklarında, belli alanlarda oldukça geri kaldıkları görülmektedir. Yapısal programlama dilleri, modern programlama dillerinin getirdiği güvenli işaretleme (safe reference), atık toplayıcı (garbage collection) ve istisna kontrolü (exception handling) gibi imkanları bize sunamamaktadırlar. Bunlardan daha önemlisi, nesne yönelimli programlama dilleri bize yapısal programlama dillerinden çok daha iyi bir tasarım ve modelleme yapma imkanı vermektedir.

Bunların yanında seçeceğimiz dilin pratik bir yanı da olmalı. Programcılığa giriş dersi, Bilgisayar Mühendisliği eğitimi almayan öğrenciler tarafından da alınmaktadır. Bu öğrenciler nesne-yönelimli programlama tekniklerini öğrendikten sonra, kariyerlerinin ileri safhalarında tekrar bilişim teknolojileri ile ilgili bir sistem tasarlamak zorunda kalırlarsa bunları kullanabilmelidirler. Sayılan sebeplerden, seçilecek dil genel amaçlı, kolaylıkla ulaşılabilen, yaygın kullanımlı olmalıdır. Java bilgisayar programlama dili bu kriterleri karşılamaktadır. Bu nedenle Java, programlama dili olarak seçildi.

2 Eğitim

Eğitimi iki aşamalı bir süreç olarak algılamamızda yarar vardır. Birinci aşamada nesne-yönelimli programlama ve modellemenin temelleri aktarılmalı, konular günlük hayattan örneklerle somutlaştırılmalıdır. Algoritma kurma yeteneğinin geliştirilmesi için akış diyagramları ve südokod (pseudocode) kullanılmalıdır. Doğrudan kodlama yapılmamalıdır.

İkinci aşamada ise kodlamaya geçilmeli ve Java dilinin özellikleri kullanılmaya başlanmalıdır. Unutulmaması gereken en önemli nokta Java dili eğitimi değil, nesne-yönelimli programlama eğitimi verildiğidir. Java bu iş için kullanılan bir

araçtır ve amaç durumuna gelmemelidir.

Üniversite düzeyindeki öğrencilerle yapılan çalışmalarda gözlemlendiği kadarıyla, öğrencilerin zorlandıkları kısım, Java programlama dilinin sentaksı (syntax) ve inceliklerinin öğrenilmesinden çok, nesne-yönelimli programcılık tekniklerinin uygulanması ve doğru tasarımların yapılması olmuştur. Jason Hong'da [3] "Java'nın programcılığa başlama dili olarak kullanılması" ile ilgili makalesinde, öğrencilerin en çok tasarım sırasında zorlandıklarını belirtmektedir. Bunlarda bize eğitimin en önemli ve üzerinde durulması gereken sürecinin birinci aşama olduğunu göstermektedir.

2.1 Genel Zorluklar

Lise düzeyindeki öğrencilerle yapılan çalışmada, birinci aşama yaklaşık dört hafta sürdü. Bu süre de, öğrencilere öncelikle gerçek yaşamın nesnelerden ve bu nesneler arasındaki ilişkilerden meydana geldiği öğretildi. Öğrencilere örnek olarak bir benzin istasyonun nasıl nesnesel olarak modelleneceği gösterildi. Daha sonra her öğrenciden kendi seçtiği bir sistemi modellemesi istendi. Öğrencilerin yapmış olduğu çalışmalardan bazıları sınıf içinde tartışılarak modellemelerin daha da geliştirilmesi sağlandı. Yapılan bu uygulamalar, öğrencilerin tasarım becerilerini arttırmıştır.

Kodlama aşamasına geçildiğinde ise Java'da ekrana bir çıktı vermek pek zor olmasa da, klavyeden bilgi girmek karşılaşılan ilk zorluk oldu. Klavyeden giriş için System.in.read() metodu kullanılabilir, ya da DataInputStream, InputStreamReader sınıfından yaratılacak bir örnek nesne ile giriş yapılabilir. Hangi yöntem kullanılırsa kullanılsın, sonuçta mutlaka IOException kontrolü yapılmak zorundadır. Bu da başlangıç aşamasında olan öğrenciler için işleri daha da karmaşık hale getirmektedir. Bunu önlemek için öncelikle değişkenlere veri atama kodun içinden yapılmıştır. Bu pek pratik bir yöntem değildi fakat daha sonra, dışardan veri girilmesini sağlayacak hazır bir sınıf kodu oluşturuldu ve bu kullanılmaya başlandı. Görsel programlamaya geçildiğinde ise işimizi daha da kolaylaştıran, JOptionPane sınıfından metotlar kullanılmaya başlandı.

Derslerde kullanılacak, bizim istediğimiz eğitim programına uygun kitap bulmakta bir başka zorluktu. Piyasada nesne-yönelimli programcılığı, Java dilinde programcılığı ya da ikisini birden anlatan çok çeşitli kitaplar olmasına rağmen, tüm ihtiyaçlarımızı karşılayan tek bir kitap bulunamadı. Özellikle Türkçe kitaplar bulmak daha büyük bir sorun olarak karşımıza çıktı. Bu yüzden Timothy Budd [4] ve Martin Kalin'in [5] kitaplarından aynı anda yararlanılmasına karar verildi. Budd'ın kitabı eğitimin teori kısmını kapsaması bakımından ders kitabı olarak, Kalin'in kitabı ise Java dilinin inceliklerini içermesi yönü ile bir referans kitabı olarak kullanıldı.

Bunların dışında, internette Java ve nesne-yönelimli programcılıkla ilgili çok fazla site ve haber grubu bulunmaktadır. Bu kaynakların büyük bir çoğunluğu ücretsiz ve serbest yazılım kodlar içermektedir. Bu kaynak kodların bazıları öğrenciler için faydalı olmakla birlikte; bazı kodlar kötü yazılım örneğidir.

2.2 Kavramsal Zorluklar

Java programlama dilini öğrenirken, öğrencilerin karşılaştığı en büyük zorluk aynı anda birçok kavramı anlamak zorunda olmalarıydı. Örnek olarak aşağıdaki kod parçasını ele alalım.

```
public static void main(String [] args)
```

Bu bir tek satırda bilinmesi gereken o kadar çok kavram vardır ki, programcılığa yeni başlayan birisinin, hepsinin aynı anda öğrenmesi mümkün değildir. Bu yüzden eğitimin ikinci aşamasının ilk başlarında, programın doğru şekilde çalışabilmesi için, nedenleri üzerinde durulmadan bu şekilde bir sentaks kullanılması gerektiği söylenip geçildi. Öğrencilerin sentaksla fazla zaman kaybetmeden, rahat bir ortamda program geliştirebilmeleri için JCreator[1] yazılımının ücretsiz sürümünden faydalanıldı.

Veri tipleri kavramında ise karşımıza basit veri tipleriyle, nesnelere referans (wrapper class) olan tiplerin farkının anlaşılması güçlüğü çıkmaktadır. Gerçektende bilgisayarın hafızasında bir tam sayı tutmak istediğimizde bunu "int" basit veri tipiyle yapabiliyorken, "Integer" nereden çıkmıştır ve nasıl kullanılmaktadır. Bu karmaşayı daha fazla arttırmamak için ilk başlarda sadece basit veri tipi kullanılmalı, örnek nesnelerin nasıl türetildiği anlatıldıktan sonra bu iki tip birlikte kullanılmaya başlanmalıdır.

Sınıf (class) ve örnek (instance) ayırımının yapılması ise oldukça zaman alan bir durumdur. Aslında şu ana kadar söylenenler Java dilinin kendisini anlamaktan kaynaklanan zorluklardı. Bunlardan daha önemli olan ve anlaşılması en

[1] Xinox Software firmasının tescilli markasıdır. http://www.jcreator.com adresinden indirilebilir.

güç olan kavram ise, daha öncede belirtildiği gibi, nesne-yönelimli bir tasarım yapılmasıdır. Birçok öğrenci Java'da program yazsa da, ortaya çıkarılan programlar modülerlik, soyutlama ve tekrar kullanım özellikleri bakımından oldukça yetersiz olmaktadır. Bunun sebebi ise yazılımın tasarım safhasının iyi çalışılmaması ve anlaşılmamasıdır.

Programcılığa yeni başlamış bir öğrenciden çok iyi bir tasarım beklenemez fakat, bu öğrenciler zaman içerisinde iyi tasarım yapması için teşvik edilmelidir. Bunun için öğrencilere iyi tasarımın nasıl olduğunu söylemektense, örnek kodlarla göstermek daha yararlı olacaktır.

2.3 Tasarımın Geliştirilmesi

Eğitimciler olarak çoğunlukla öğrencinin ortaya koyduğu programın çalışıp çalışmadığına ya da yazdığı kodun hatasız olup olmadığına bakarak başarı değerlendirmesi yapmaktayız. Bu değerlendirme sırasında öğrencinin kavramları anlamak, ihtiyaçları analiz etmek, tasarım yapmak, yapılan tasarımı kodlamak ve oluşturulan programı test etmek için ne kadar güç sarf ettiğini göz ardı etmekteyiz. Bu şekildeki değerlendirme ise öğrencinin eğitim süresince ne kadar ilerleme kaydettiğini ve kavramları ne kadar anladığını ortaya koyamamaktadır. Eğitimin asıl amacı, hedef kitlenin bilgi ve davranış düzeyi olarak ulaşılması istenen noktaya getirilmesidir. Yaptığımız başarı değerlendirmesi ise bunu tam olarak saptayamamaktadır.

Gözlemlendiği kadarıyla programcılık deneyimi az olan öğrencilerin çoğunluğu kavramları ve ihtiyaçları anlamak, tasarım yaratmak için yeterince zaman ayırmamaktadır. Kendilerine bir ödev verildiğinde, analiz ve tasarım safhalarını atlayıp, direkt olarak kodla boğuşmaya başlamaktadırlar. Bu da bizim öğretmek istediklerimizle çelişen bir durum oluşturmaktadır.

Tasarım yetisinin geliştirilmesi amacıyla, bu yazıda anlatılan çalışmada olduğu gibi, öğrencilere öncelikle günlük hayattan bir takım örnek problemler verilmeli. Bu problemleri kendi başlarına çözmeleri istenmeli ve ihtiyaç analizi, nesne-yönelimli tasarım ve kodlama safhalarını ayrı ayrı parçalar halinde ele almaları sağlanmalıdır. Tüm bunlar öğrenci tarafından gerçekleştirildikten sonra, verilen problemin aynı aşamalarının deneyimli bir programcı tarafından ortaya çıkarılmış örneği ile öğrencinin çalışmaları karşılaştırılmalıdır. İki yazılımın karşılaştırılması sonucu öğrenci eksiklerini görecek ve bunları gidererek kendini geliştirecektir.

3 Sonuçlar

Programcılık eğitimine nesne-yönelimli bir dil ile başlamak oldukça zaman ve güç isteyen bir çalışmadır. Eğer bu iş Java ile yapılacaksa unutulmaması gereken nokta Java dili eğitimi değil, nesne-yönelimli programcılık eğitimi verildiğidir. Önemli olan nesne-yönelimli programcılığın temellerinin verilmesidir. Programlama dilleri gelip geçicidir.

Konular mümkün olduğunca basit anlatılmalıdır. Nesne-yönelimli programcılık teknolojisi oldukça karmaşık ve zordur. Bu yüzden öğrencilere karşı esnek olunmalı ve konulardan konulara yavaşça geçilmeli, bol bol ödev verilmeli, alıştırma yaptırılmalıdır. İyi bir tasarımın ve kodun nasıl olması gerektiğini anlatmak yerine, iyi kodlar örneklerle gösterilmeli, bunların öğrencinin kendi yarattığı çalışmalarla karşılaştırılması sağlanmalıdır.

Kaynakça

1. James T. Kalker, "The Psychology of Learning", Prentice Hall, 1996.
2. B. Meyer, "Towards an Object-Oriented Curriculum", Journal of Object-Oriented Programming, Mayıs 1993.
3. Jason Hong, "The Use of Java as an Introductory Programming Language", ACM Crossroads Cilt 4, Sayı 4, s. 8-13, Bahar 1998.
4. Timothy Budd, "Understanding Object-Oriented Programming with Java", Addison-Wesley, 2000.
5. Martin Kalin, "Object-Oriented Programming in Java", Prentice Hall, 2001.