Doçentlik Sınavı, Çok Biçimlilik ve Java

Kemal TURHAN[a]

[a]Karadeniz Teknik Üniversitesi, Tıp Fakültesi, Tıp Eğitimi ve Bilişimi Anabilim Dalı, Trabzon.

An Examination, Polymorphism and Java

Abstract

The object-oriented approach goes a step further by providing tools for the programmer to represent elements in the problem space. To illustrate the basic concepts in object oriented programming such as abstracting and polymorphism we presented an example how we might go about handling a real world situation and how we could make the computer more closely model in real world. In this study, it was discussed with an example how Object Oriented Thinking (OOT) can help medical informatics students to learn about using Java Programming Language for solving a real world problem.

Key Words

Polymorphism, Java, Object Oriented Thinking

Özet

Nesne yönelimli yaklaşım programcıların problem çözme yöntemlerini geliştirecek önemli araçlar sağlamaktadır. Object Oriented Thinking (OOT)'nin çok biçimlilik, genelleme gibi temel OOT kavramlarını açıklayabilmek için gerçek yaşamda karşılaşılan bir durumu çözmek ve bilgisayarı gerçek yaşama daha uygun hale getirmek amacıyla örnek problem verilmiştir. Tıp bilişimi öğrencilerinin her hangi bir problemi çözerken OOT'den nasıl yararlanılabileceği sunulan örnek yardımıyla tartışılmıştır.

Anahtar Kelimeler:

Çok biçimlilik, Java, Nesne Yönelimli Düşünme.

1. Giriş

Object Oriented Thinking (OOT) temelinde problem uzayındaki her şeyin bir nesne olduğu düşüncesi yatmaktadır. Gerçek yaşamda da her şeyin bir nesne olduğu düşünülürse, OOT'nin tüm yaşam alanında uygulama şansı bulabilecek, son derece önemli bir felsefi araç olduğu açıktır. Yeni başlayan öğrenciler, yapısal (geleneksel) düşünme ve problem çözme alışkanlığını kazanmış uygulama geliştiricileri, sistem analistleri, sistem tasarımcıları ve yazılım mühendisleri için son derece önemli olan OOT kavramlarını anlatmak ve bu beceriyi kazanmalarını sağlayarak, kişisel problem alanlarında etkili kullanmalarını sağlamak güç olabilmektedir.

Geleneksel programlama anlayışına göre, bilgisayarlar çeşitli koşullara bağlı olarak prosedürel bir yapı içinde istenilen işlemleri yapan araçlar olarak kullanılmaktadır. Program satırları incelendiğinde problemi anlamak çoğunlukla mümkün olmamaktadır. Problemi bilgisayarın işlem yapma sürecine uygun hale getirmek kodlar üzerinden problemi anlamayı güçleştirebilmektedir (1).

Object Oriented Programming'in (OOP) programlama dünyasında meydana getirdiği en temel değişim, bilgisayarın problem dünyasının içine çekilmesi denilebilir. Programın, gerçek yaşamın modellenmiş nesneleri aracılığı ile probleme adapte olması sağlanmaktadır. Dolayısıyla, kod incelendiğinde, hem problem hem de çözüm aynı anda anlaşılır olabilmektedir. Projenin büyüklüğü ne olursa olsun yazılımda herhangi bir sorun çıktığında ilgili nesneler incelendiğinde sorunu anlamak çok daha kolay olacaktır (2). Uygulama geliştirilirken kullanılan programlama dili nesne yönelimli (Java, C# veya Delphi) olsa da, programlama anlayışında tam bir dönüşüm gerçekleştirilmedikçe bu araçlarla da geleneksel programlama yöntemlerini sürdürmek mümkün. Kullanılan dilin özellikleri ne kadar bilinirse bilinsin OOP için gerekli olan felsefi alt yapı (OOT) öğrencide oluşturulmamışsa nesneler dünyasında gerçek yaşamı taklit etmek mümkün olmayacaktır.

2. Doçentlik Sınavı : Gerçek Yaşam Problemi

Birçok akademisyen gibi bu makalenin yazarının da akademik aşama kaydedebilmek için geçmek zorunda olduğu doçentlik sınavı tam bir gerçek yaşam problemidir. Bu problemi OOT ve Java (OOP) ile modelleyerek çözüme ulaştırabilmek mümkündür.

Tablo-1Senaryo Dokümantasyonu

Sınav Senaryosu	Doçentlik Sınavı
Birinci Aktör	Doçent Adayı
İlgililer ve Beklentileri	Jüri üyeleri, adaya soracakları soruları bilmesini beklemektedirler.
Ön Koşullar	Aday YÖK kriterlerine göre gerekli diploma, yayın ve yabancı dil koşullarını sağlamalı. Jüri üyelerinin oy çokluğunu sağlamış olmalı.
Son Koşullar	Aday yapılacak sınavda başarılı olursa doçent olacaktır.
Ana Akış	Aday belirlenen ilde sınava davet edilir. Önceden belirlenmemiş bir zaman süresince aday sınava alınır. Her bir jüri üyesi kendilerine has objektif kriterlerine göre belli olmayan sayıda ve zorluk derecesinde adaya soru sorarlar. Aday bu sorulara doğru cevap verir veya veremez. Sınav biter ve jüri kararını verir.
Alternatif Akış	Alternatif bir akış yoktur.

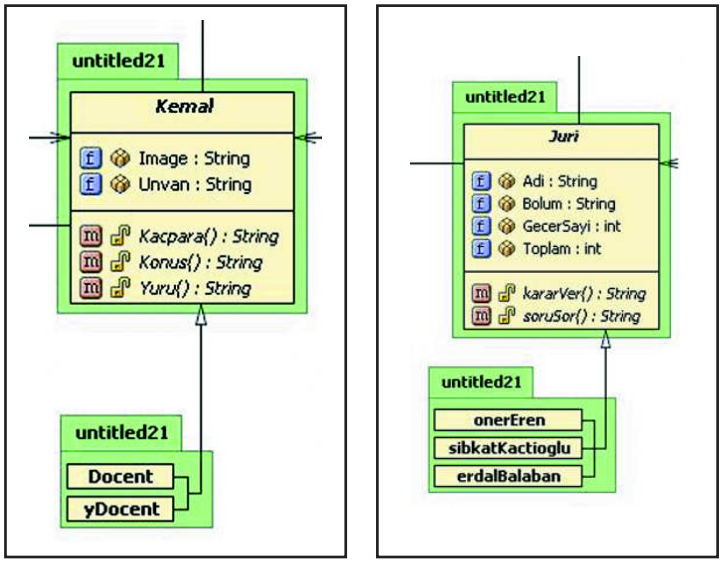
Senaryo dokümanından yararlanarak nesneler ve bu nesnelerin bir biri ile olan etkileşimleri ve etkileşimde kullanacakları mesajlar belirlenebilir.

Senaryodan anlaşıldığı üzere problemde iki temel nesne ve bunların etkileşimleri söz konusu Şekil-1).

Senaryonun gerçeklenmesi için gerekli olan nesnelerin senaryoda yerlerini almalarını sağlayacak iki tür (class) şekil-1'de görülmektedir. Her iki türde de dikkat edilirse bir genelleme söz konusu. Kemal, yani adayın iki durumu söz konusu (Docent, yDocent). Aynı şekilde Juri'nin ise (onerEren, sibkatKactioglu, erdalBalaban) gibi senaryo için değişik jüri üyelerini temsil edecek alt türleri görülmektedir. Burada Kemal de Juri de soyut (Abstract) tür olarak tanımlanmıştır. Bunun nedeni; sonraki bölümde göreceğimiz gibi soyut sınıfta yer alan Kacpara(), Konus(), Yuru() ve kararVer(), soruSor() gibi mesajlara, soyut türe ait nesnelerin senaryo akışındaki duruma göre farklı tepkiler verebilmelerini sağlamaktır. Bunu soyut sınıf üzerinden nesneye ulaşarak sağlayabiliriz. Böylece sınav bittiğinde aday doçent olmuşsa mesajlara vereceği tepkilerin farklılaşması ve soru soran üyenin kendi sitiline göre soru sorması kendiliğinden sağlanabilir. Bu bizi OOT ve OOP'nin önemli olan çok biçimlilik kavramına (polimorphizm) götürür. Çok biçimliliğin uygulaması sonraki bölümde verilmiştir.

3. Doçentlik Sınavı Benzetimi : Java Uygulaması

Soyut türlerin farklı biçimlerinin tanımlanması gerekmektedir. Bunun için OOP'de olan miras Şekil-2) (*class yDocent extends Kemal*) alma özelliğinden yararlanabiliriz.



Şekil-1. Problemin aktörleri

3.1. Aday(Kemal) Sınıfın Java'da Gerçekleştirimi

```
public abstract class  Kemal {
   String Unvan;
   String Image;
   public abstract String Yuru();
   public abstract String Konus();
   public abstract String Kacpara();
}
```

Şekil-2. Aday türü için soyut sınıf tanımı

Java kod parçalarından da anlaşılacağı üzere soyut sınıfımızda (abstract class) *Yuru()*, *Konus()* ve *Kacpara()* gibi üç metot bulunmaktadır. Bu metotlar soyut sınıftan miras aldıkları için sınava giren adayın durumuna göre farklı çalıştırılacaktırŞekil-2, şekil-3). Benzer şekilde *unvan* ve *image* isimli özelliklerde sonuca göre değişecektir.

```
class yDocent extends Kemal {
  yDocent(){
    Unvan="Yrd. Doç. Dr.";
    Image="images/yDoc.gif";
  }
  public String Yuru(){
    return "Kenardan yürü...";
  }
  public String Konus(){
    return "Dikkatli konuş...";
  }
  public String Kacpara(){
    return "X YTL";
  }
}

class Docent extends Kemal {
  Docent(){
    Unvan="Doç. Dr.";
    Image="images/Doc.gif";
  }
  public String Yuru(){
    return "Yürü bakalım meydan DOÇENT görsün..";
  }
  public String Konus(){
    return "İstediğini söyle kasılabilirsin...";
  }
  public String Kacpara(){
    return "X YTL"+" + 300 YTL daha kazanacaksın.";
  }
}
```

Şekil-3. Çok biçimlilik için soyut sınıftan miras alan iki tür (Docent ve yDocent).

3.2. Juri Sınıfının Java'da Gerçekleştirimi

Jüri sınıfından miras alan her nesne *Toplam* ve *GecerSayi* isimli iki nesne özelliğine sahiptir. Bu değişkenlerden *Toplam* üyenin kaç soru sorduğunu, *GecerSayi* ise adayın zorluk derecesine göre kaç soruyu doğru bildiğini saklamaktadır. Kodlar incelenirse iki üyenin *kararVer()* metotları farklıdır. Bir üye, aday soruların yarısını doğru bilirse olumlu karar verirken diğer üye ise karar verdiği andaki *Moralmetre* isimli özelliğinden etkilenerek karar vermektedir. Burada jüri üyelerinin soru sorma ve karar verme sitilleri çok daha karmaşık hale getirilerek bunlar metot gövdelerinde tanımlanabilir. Makalenin konusu açısından amaç sadece çok biçimliliği uygulamada gösterebilmektir.

```
6  public abstract class Juri    {
7     String Adi;
8     String Bolum;
9     int Toplam;
10    int GecerSayi;
11    public abstract String soruSor();
12    public abstract String kararVer();
13 }
```

Şekil-4. Jüri soyut sınıfının Java karşılığı.

```
41 class erdalBalaban extends Juri {
42   String[] Sorular={"1. soru","2. soru","3. soru","4.soru","5.soru",
43                     "6. soru","7. soru","8. soru","9.soru","10.soru"};
44   int[] Zorluk={8,9,3,10,9,4,8,3,1,1};
45
46   public String soruSor(){
47     Random r1 =new Random();
48     int sn;
49     sn= r1.nextInt(10);
50     Toplam++;
51     if (Zorluk[sn]<=5) {GecerSayi++;};
52     return Zorluk[sn]+" "+Sorular[sn];
53
54   }
55 public String kararVer() {
56    if (GecerSayi>=Toplam/2) {
57      return "Geçti";
58    }
59    else {
60      return "Kaldı";
61    }
62  }
63 }
```

Şekil-5.Jüri sınıfından miras alan bir jüri üyesi (ErdalBalaban)

```
15 class sibkatKactioglu extends Juri {
16   String[] Sorular={"1. soru","2. soru","3. soru","4.soru","5.soru",
17                     "6. soru","7. soru","8. soru","9.soru","10.soru");
18   int[] Zorluk={4,5,3,10,1,4,5,3,10,1};
19   Random r1 =new Random();
20   int Moralmetre;
21
22 public String soruSor(){
23     int sn;
24     sn= r1.nextInt(10);
25     Toplam++;
26     if (Zorluk[sn]<=5) {GecerSayi++;};
27     return Zorluk[sn]+" "+Sorular[sn];
28   }
29
30 public String kararVer() {
31    Moralmetre= r1.nextInt(4);
32    if   (Moralmetre==0 ) {Moralmetre=1;};
33    if (GecerSayi>=Toplam/Moralmetre) {
34      return "Geçti";
35    }
36    else {
37      return "Kaldı";
38    }
39  }
40 }
```

Şekil-6 .Jüri sınıfından miras alan bir jüri üyesi (sibkatKactioglu)

3.3. Sınavın Benzetimi

Artık senaryonun gerçekleşmesi için gerekli olan gerçek yaşam nesneleri hazırdır. Sınav başladığında jüri üyeleri program tarafından rasgele seçilerek, her üye kendi soruları içinden rasgele zorluk derecesinde soru seçecektir. Sınav bittiğinde jüri üyeleri kararlarını bildirecekler ve sonunda adayın son durumu belli olacaktır.

Bu bölümde çok biçimlilikle (1, 2, 3) ilgili en önemli kod satırları verilecektir.

Aday nesnesinin inşa edilmesi için gerekli kod;

Kemal KemalTURHAN=new yDocent0;

Jüri üyelerinin benzetimde görev yapmaları için *Juries* isimli tek boyutlu bir dizi onlara ev sahipliği yapmaktadır. Görüldüğü gibi dizinin tipi *Juri (Juri[] Juries)* sınıfından belirlendi. Burada yapılan, problemin çözümü için bir genellemedir(çok biçimlilik). Aynı şekilde aday da *Kemal* sınıfı ile ilişkilendirilerek benzer bir genelleme yapılmıştır. Genellemede amaç, bir türe miras aldığı sınıf üzerinden ulaşabilmektir(*Juries[s].soruSor()*). Ayrıca, *KemalTURHAN* isimli nesneyi *Kemal* isimli soyut sınıftan tanımlayarak *KemalTURHAN'ın* sınavda çıkacak sonuca göre *yDocent()* veya *Docent()* nesnesine referans olarak kullanabilmemizi, dolayısıyla çok biçimliliği sağlamaktadır.

Juri[] Juries = {new erdalBalaban(),new onerEren(),new sibkatKactioglu()} ;

Sınav artık başlatılabilir.

```
102     for (int i = 0; i < stop; i++) {
103
104       s=r1.nextInt(3);
105       Cevap=Juries[s].soruSor();
106       Sorular[i]=Juries[s].Adi+"> (Zorluk="+ Ceve
107
108     }
```

Şekil-7 : Sınav döngüsü.

Yukarıdaki kod parçası sınav anını göstermektedir. Burada en önemli kod satırı;

Cevap=Juries[s].soruSor();

s değişkeni rasgele program tarafından atandığı için o anki jüri üyesinin kim olacağı önceden bilinemez. Dolayısıyla *soruSor()* metodu çalışma anında(runtime, late binding) seçilen üye için çalışacaktır. Bu bize OOP'nin en önemli özelliklerinden birisini yani çok biçimliliği göstermektedir.

Artık karar verme zamanı gelmiştir ve jüri üyeleri kararlarını bildiriler.

jTextField1.setText(Juries[0].kararVer());

jTextField2.setText(Juries[1].kararVer());

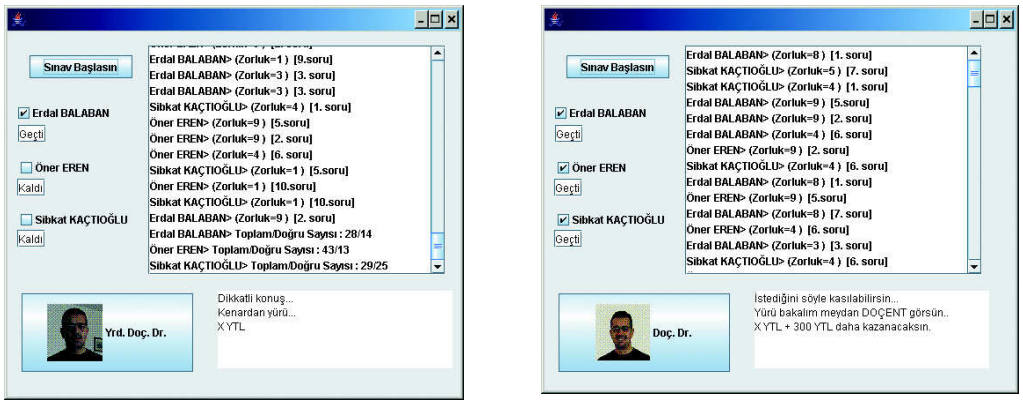
jTextField3.setText(Juries[2].kararVer());

Sınav bittiğinde adayın davranışlarında bir değişiklik olup olmadığına bakarak doçent olup olmadığını anlayabiliriz. Dikkat edilirse jüri üyeleri *Geçti* kararı vermişlerse *Doc* isimli değişken bir arttırılmaktadır. Bu değişken 1'den büyükse üç üye olduğu için çoğunluk sağlanmış olacaktır.

Yukarıdaki kod incelendiğinde programın yazarının hangi jüri üyesinin nasıl bir ruh haliyle soru soracağını ve sınav sonunda adayın nasıl konuşup yürüyeceğini veya kaç para kazanıyor olacağını bilmesi imkansız. Gerçek yaşamda da öyle değil midir ? İşte OOT ve OOP nin çok biçimlilik özelliği ile gerçek yaşama ne kadar yaklaşabildiğinin bir örneği.

```
123     if ( jTextField1.getText().equals("Geçti"))
124       {jCheckBox1.setSelected(true);Doc++;};
125     if ( jTextField2.getText().equals("Geçti"))
126       {jCheckBox2.setSelected(true);Doc++;};
127     if ( jTextField3.getText().equals("Geçti"))
128       {jCheckBox3.setSelected(true);Doc++;};
129
130     if (Doc>1) { KemalTURHAN= new Docent(); }
131       else
132             | { KemalTURHAN= new yDocent(); }
133
134      jKemal.setText(KemalTURHAN.Unvan);
135      jKemal.setIcon(new ImageIcon(KemalTURHAN.Image));
136
137      jTextArea1.setText(
138           KemalTURHAN.Konus()+'\n'+
139           KemalTURHAN.Yuru()+'\n'+
140           KemalTURHAN.Kacpara()
141          );
```

Şekil-8. Sınav sonucu.



Şekil-9. Sınavın Java ekran çıktıları.

4. Sonuç ve Öneriler

Bir bilgi sistemi proje geliştirme sürecinin her aşamasında yaşanan problemlere çözüm alternatifi olarak gündeme gelen OOT gerçektende çok önemli avantajlar sağlayabilir. Uygulama geliştirmenin dışında analiz ve tasarımda da UML gibi nesne yönelimli araçların kullanılması gerçek yaşamın çok daha kolay modellenmesini sağlayacaktır. Özellikle genç bilişimcilerin kendilerine sağlanan bu olanakları iyi kavramaları, yazılım sektörünün ülkemizde gelişmesine yardımcı olacaktır. Bu aynı zamanda bilişim projelerinde ortaya çıkan sorunların azalmasına ve daha az kaynak ve emek tüketilmesine neden olacaktır.

5. Yararlanılan Kaynaklar

[1] Eckel Bruce, Thinking in Java, 3rd ed. Revision 4.0, http://www.mindview.net/Books/TIJ/

[2] http://www.bolthole.com/OOP.html , (18 Ekim 2006'da erişildi.)

[3] Timothy A Budd, Understanding Object Oriented Programming with Java, Oregon State University Corvallis, Oregon, ‚July 1997.

Sorumlu Yazar; Yrd. Doç. Dr. Kemal TURHAN; Karadeniz Teknik Üniversitesi Tıp Fakültesi Tıp Eğitimi ve Bilişimi A.B.D., Trabzon. e-mail : kturhan@meds.ktu.edu.tr, kturhan_tr@yahoo.com, tel : 0 462 377 5680