NESNE YÖNELİMLİ PROGRAMLAMA 2(Object Oriented Programming 2/OOP)

Öğr. Gör. Celil ÖZTÜRK
Marmara Üniversitesi
Teknik Bilimler Meslek Yüksekokulu

İçerik

- Dosya İşlemleri

Dosyalarla Çalışmak

KARŞILAŞILABİLECEK AYKIRI DURUMLAR

- java.io.IOException: Genel G/Ç hatalarını simgeler.
- java.io.EOFException extends IOException: Beklenmedik bir nedenle dosyanın veya akının sonuna gelindiğini gösterir.
- java.io.FileNotFoundException extends IOException: İstenen dosyanın verilen yolda bulunamadığını gösterir.
- java.lang.SecurityException extends java.lang.RuntimeException: İstenen dosya işleminin güvenlik kısıtlamaları nedeniyle yapılamadığını gösterir.

DOSYA İŞLEMLERİ HAKKINDA GENEL BİLGİ

- Java dilinde dosya işlemleri iki ana gruba ayrılmıştır:
  - Dosya yönetim işlemleri: Dizin ve dosyaları oluşturmak, yeniden adlandırmak, silmek, vb.
  - G/Ç işlemleri.
- G/Ç işlemleri sadece dosyalar üzerinde değil, farklı kaynaklar üzerinde de yapılır: TCP soketleri, web sayfaları, komut satırı, vb.
  - Bu nedenle G/Ç işlemleri, dosya işlemlerinden ayrılmıştır ve adı geçen farklı kaynaklar üzerinde de aynen dosyalar üzerinden icra ediliyormuş gibi yapılmaktadır.
- Nesneye yönelik düşünme biçiminin iyi bir uygulaması olan bu yaklaşım, karmaşıklığı arttırmıştır.

DOSYA YÖNETİMİ

- Diskteki dizin ve dosyaları simgeleyen java.io.File sınıfı ile yapılır.
- Bir File nesnesi oluşturmak, diskte fiziksel bir nesne oluşturmak demek DEĞİLDİR!
- Bir dosya nesnesi oluşturmak
  - File(String dosyaAdi) kurucusu ile.
  - dosyaAdi, dosyanın hem yolunu hem de adını içermelidir.
  - Tam yol veya göreceli yol
- Göreceli yol kullanırken dikkat: Geliştirme ortamları kaynak kod ve derlenmiş kodu ayrı dizinlerde tutabilir.

DOSYA YÖNETİMİ

- Dizin ayıracı:
  - Windows'ta \, ancak karakter katarı içerisinde \\ kullanımı.
  - Unix'te /
- File(String dizin, String isim) ve File(File dizin, String isim) kurucuları ile:
  - Verilen dizinin altındaki verilen isimli bir dosya veya dizini simgeler.
  - File(String) kurucusunun kuralları aynen geçerlidir.

DOSYA YÖNETİMİ

java.io.File sınıfının bazı metotları:

- boolean exists( ); dosya/dizinin fiziksel mevcudiyet durumunu döndürür
- boolean isFile( ); nesnenin bir dosya olup olmadığını döndürür
- File getParentFile( ); bu dosya/dizinin üst dizinini döndürür
- String getCanonicalPath( ) throws IOException; nesnenin kendi adı dahil olmak üzere dosya/dizinin tam yolunu döndürür
- boolean canRead( ); nesneye okuma hakkı var mı?
- boolean canWrite( ); nesneye yazma hakkı var mı?
- boolean createNewFile( ); Dosyayı fiziksel olarak da oluşturur

DOSYA YÖNETİMİ

java.io.File sınıfının bazı metotları:

- boolean mkdir( ); Dizini fiziksel olarak da oluşturur
- boolean mkdirs( ); Dizini gerekli üst dizinlerle birlikte fiziksel olarak da oluşturur
- boolean renameTo(File newName); nesneyi yeniden adlandırır.
- boolean delete( ); nesneyi siler.

Notlar:

- Tüm bu metotları ezberlemek zorunda değilsiniz.
- Fiziksel: Dosya/dizinin diskte gerçekten yer alması

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

- Herhangi bir G/Ç kaynağı, Java dilinde bir akı (stream) ile temsil edilir.
  - Dosya, bellek, komut satırı, ağ, … Java'da hepsi birer G/Ç kaynağıdır.
- Okunabilirlik ve başarıma göre temel çalışma biçimleri:
  - İkili (Binary) düzen: Hızlı ancak bir insan tarafından okunamaz.
  - Karakter düzeni: Bir insan tarafından okunabilir ama daha yavaş.
  - Kayıt düzeni: Bileşik kayıtlar şeklinde G/Ç işlemleri (Pascal: record, C:Struct, Java: nesneler

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

- Erişim düzenine göre temel çalışma biçimleri:
  - Sıralı/ardışıl erişim (sequential access): Tüm kayıtlar sıra ile okunur.
  - Rastgele (random) erişim: Herhangi bir kayıda doğrudan erişilebilir.
- Java, farklı çalışma biçimleri için farklı akıları, birbirleri ile zincirleyerek kullanır.
- Derste, nesneleri bir bütün olarak işlemeye yarayan kayıt düzeni işlenecektir.
  - Java terminolojisinde bu işleme serileştirme (Serialization) adı verilir.

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

- Serileştirme – Çıktı işlemleri:
  - Nesneleri bir dosyaya yazma işlemleri
  - Serileştirilecek nesneler, java.io.Serializable arayüzünü gerçeklemelidir.
  - Programcının arayüzdeki metotları yeniden tanımlamasına gerek yoktur.
  - ObjectOutputStream ve FileOutputStream nesneleri zincirlenerek kullanılır.
- Aynı akıya birden fazla nesne kaydedilebilir.
  - Zaten birbirleri ile ilişkili nesneler aynı dosyaya kaydedilmelidir, aksi halde ‘işaretçi kırılması’ olayı yaşanır.

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

Örnek kayıt: Arkadas sınıfı

```
package not04a;
public class Arkadas implements java.io.Serializable {
private static final long serialVersionUID = 1L;
private String isim, telefon, ePosta;
public Arkadas( String name ) { this.isim = name; }
public String getIsim( ) { return isim; }
public String getTelefon( ) { return telefon; }
public void setTelefon( String telefon ) {
this.telefon = telefon; }
public String getEPosta( ) { return ePosta; }
public void setEPosta( String posta ) { ePosta = posta; }
public String toString( ) {
return isim + " - " + telefon + " - " + ePosta;
}
}
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

- **Kayıtları dosyaya yazan program:**

```
package not04a;

import java.util.*;

import java.io.*;

public class ArkadasOlustur {

public static void main(String[] args) {

Integer arkadasSayisi;

Arkadas[ ] arkadaslar;
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

```
Scanner giris = new Scanner( System.in );
System.out.println("Bu program arkadaşlarınızın iletişim " +
" bilgilerini diskteki bir dosyaya kaydeder.");
System.out.print("Kaç arkadaşınızın bilgisini gireceksiniz? ");
arkadasSayisi = giris.nextInt( );
giris.nextLine( );
arkadaslar = new Arkadas[arkadasSayisi];
for( int i = 0; i < arkadasSayisi; i++ ) {
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

```
System.out.print((i+1)+". arkadaşınızın ismi nedir? ");
arkadaslar[i] = new Arkadas( giris.nextLine() );
System.out.print("Bu arkadaşınızın telefonu nedir? ");
arkadaslar[i].setTelefon( giris.nextLine() );
System.out.print("Bu arkadaşınızın e-posta adresi nedir? ");
arkadaslar[i].setEPosta( giris.nextLine() );
}
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

```
try {
String dosyaAdi = "arkadaslar.dat";
ObjectOutputStream yazici = new ObjectOutputStream(
new FileOutputStream( dosyaAdi ) );
yazici.writeObject( arkadasSayisi );
for( Arkadas arkadas : arkadaslar )
yazici.writeObject( arkadas );
System.out.println("Girilen bilgiler " + dosyaAdi +
" adlı dosyaya başarıyla kaydedildi.");
}
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

```
giris.close();
catch( IOException e ) {
System.out.println("Dosyaya kayıt işlemi sırasında"+
" bir hata oluştu.");
e.printStackTrace();
}
}
}
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

- Serileştirme – Girdi işlemleri:
  - Nesneleri bir dosyadan okuma işlemleri
  - ObjectInputStream ve FileInputStream nesneleri zincirlenerek kullanılır.
  - Okunan nesneler Object türünde olduğu için, ait oldukları tipe dönüştürülmek zorundadır (typecasting).
  - Okunan nesneler bir dizide saklanacaksa kaç nesne okunacağı bilinmek zorundadır.
  - Dinamik olarak boyutu değişebilen veri yapılarında buna gerek yoktur.

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

- **Kayıtları dosyadan okuyan program:**

```
package not07a;
import java.io.*;
public class ArkadasGoster {
public static void main( String[] args ) {
String dosyaAdi = "arkadaslar.dat";
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

```
try {
ObjectInputStream okuyucu = new ObjectInputStream(
new FileInputStream( dosyaAdi ) );
Integer kayitSayisi = (Integer)okuyucu.readObject();
for( int i = 0; i < kayitSayisi; i++ ) {
Arkadas arkadas = (Arkadas) okuyucu.readObject();
System.out.println(arkadas);
}
}
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

```
catch( IOException e ) {
System.out.println("Dosya okuma işlemleri sırasında" +
" bir hata oluştu.");
e.printStackTrace();
}
catch( ClassNotFoundException e ) {
System.out.println("Okunan kayıtları işlerken " +
"bir hata oluştu.");
e.printStackTrace();
}
}
}
```

AKILAR (STREAM) İLE DOSYA İŞLEMLERİ

•Bir akının sonuna gelip gelmediğimizi sınamanın doğru çalışan bir yolu halen bulunmamakta. Bu nedenle aşağıdaki gibi bir kod **yazamıyoruz:**

```
try {
        ObjectInputStream okuyucu = new ObjectInputStream(
                new FileInputStream( dosyaAdi ) );
        Arkadas arkadas = (Arkadas) okuyucu.readObject();
        while okuyucu.hasNext() {
                System.out.println(arkadas);
                arkadas = (Arkadas) okuyucu.readObject();
}
okuyucu.close(); }
```

Çalışmaz !

DÜZELTİLMEMİŞ HATALAR

- int ObjectInputStream.available( ), metodu var ama hatalı
  - http://www.coderanch.com/t/378141/java/java/EOF-ObjectInputStream
- Üstelik, readObject() metodu dosya sonuna gelinince null döndüremiyor.
  - http://stackoverflow.com/questions/2626163/java-fileinputstream-objectinputstream-reaches-end-of-file-eof
- Bir aykırı durum olmasına izin verip catch bloğunda döngüyü sonlandırma yoluna gidilebilir ama tercih edilmez.
  - Aykırı durum işleme, denetim akışını düzenleme amacıyla icat edilmemiştir

DÜZELTİLMEMİŞ HATALAR

- Daha iyi bir çözüm, saklanması gereken tüm nesneleri bir veri yapısında tutmak ve o veri yapısıyla serileştirme işlemlerini yapmaktır.
  - Sonraki yansıda gösterilecektir.
  - Veri yapısındaki nesneler java.io.Serializable arayüzünü gerçeklemeyi sürdürmelidirler.
  - A→B ilişkisi varsa her iki sınıf da java.io.Serializable arayüzünü gerçeklemelidir.

DÜZELTİLMEMİŞ HATALAR

- Veri yapıları ile serileştirme (diske yazma) işlemleri:

```
package oop04b;
import java.util.*;
import java.io.*;
@SuppressWarnings("resource")
public class ArkadasOlustur { public static void main(String[] args) { LinkedList arkadaslar = new
LinkedList();
Scanner giris = new Scanner( System.in );
System.out.println("Bu program arkadaşlarınızın iletişim " + " bilgilerini diskteki bir dosyaya kaydeder.");
```

DÜZELTİLMEMİŞ HATALAR

```
System.out.print("Kaç arkadaşınızın bilgisini gireceksiniz? ");
int arkadasSayisi = giris.nextInt( );
giris.nextLine( );
for( int i = 0; i < arkadasSayisi; i++ ) {
        System.out.print((i+1)+". arkadaşınızın ismi nedir? ");
        Arkadas arkadas = new Arkadas( giris.nextLine() );
        System.out.print("Bu arkadaşınızın telefonu nedir? ");
        arkadas.setTelefon( giris.nextLine() );
        System.out.print("Bu arkadaşınızın e-posta adresi nedir? ");
        arkadas.setEPosta(giris.nextLine() );
        arkadaslar.add(arkadas);
}
```

DÜZELTİLMEMİŞ HATALAR

```
try {
            String dosyaAdi = "arkadaslarAlt.dat";
            ObjectOutputStream yazici = new ObjectOutputStream(
            new FileOutputStream( dosyaAdi ) );
            yazici.writeObject( arkadaslar );
            yazici.close();
            System.out.println("Girilen bilgiler " + dosyaAdi +
            " adlı dosyaya başarıyla kaydedildi.");
            }
catch( IOException e ) {
            System.out.println("Dosyaya kayıt işlemi sırasında"+
            " bir hata oluştu.");
            e.printStackTrace();
                        }
            }
}
```

DÜZELTİLMEMİŞ HATALAR

- **Veri yapıları ile serileştirme (diskten okuma) işlemleri:**

```
package oop04b;
import java.io.*;
import java.util.*;
public class ArkadasGoster {
        public static void main( String[] args ) {
        String dosyaAdi = "arkadaslarAlt.dat";
try {
        ObjectInputStream okuyucu = new ObjectInputStream( new
FileInputStream( dosyaAdi ) );
        @SuppressWarnings("unchecked")
        LinkedList<Arkadas>
        arkadaslar = (LinkedList<Arkadas>)okuyucu.readObject();
        for( Arkadas arkadas : arkadaslar ) {
System.out.println(arkadas); }
okuyucu.close(); }
```

DÜZELTİLMEMİŞ HATALAR

```
catch( IOException e ) {
System.out.println("Dosya okuma işlemleri sırasında" +
" bir hata oluştu.");
e.printStackTrace();
}
catch( ClassNotFoundException e ) {
System.out.println("Okunan kayıtları işlerken " +
"bir hata oluştu.");
e.printStackTrace();
}
}
}
```

Farklı bir örnek

```
import java.io.*;
import java.util.*;
public class DosyaIşlemleri implements java.io.Serializable {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
            //Dosya yazma işlemi....
        String str = "Bunu dosyaya yazdir";
        File file = new File("dosya.txt");
        if (!file.exists()) { file.createNewFile();
        } FileWriter fileWriter = new FileWriter(file, false);
        BufferedWriter bWriter = new BufferedWriter(fileWriter);
        bWriter.write(str);
        bWriter.close();

            //Dosya okuma işlemi buradan başlıyor..
        FileReader fileReader = new FileReader(file);
        String line;
        BufferedReader br = new BufferedReader(fileReader);
        while ((line = br.readLine()) != null) {
         System.out.println(line);
        }
        br.close();
    }}
```

Kaynaklar

- Yrd.Doç.Dr. Yunus Emre SELÇUK'un ders notları
- Dr. Öğr. Üyesi Aysun ALTIKARDEŞ Nesne Yönelimli Programlama 2 Ders Notları
- https://mesutpek.com.tr/java-da-dosya-islemleri-okuma-yazma/