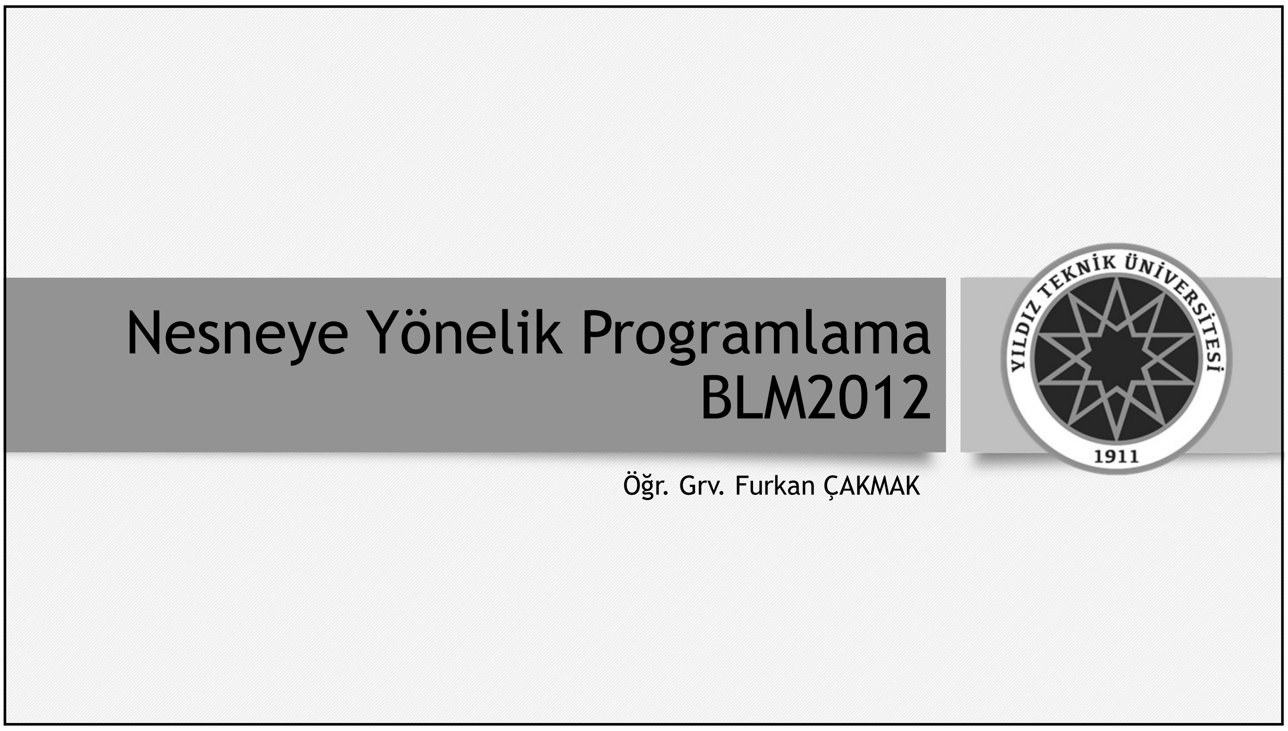
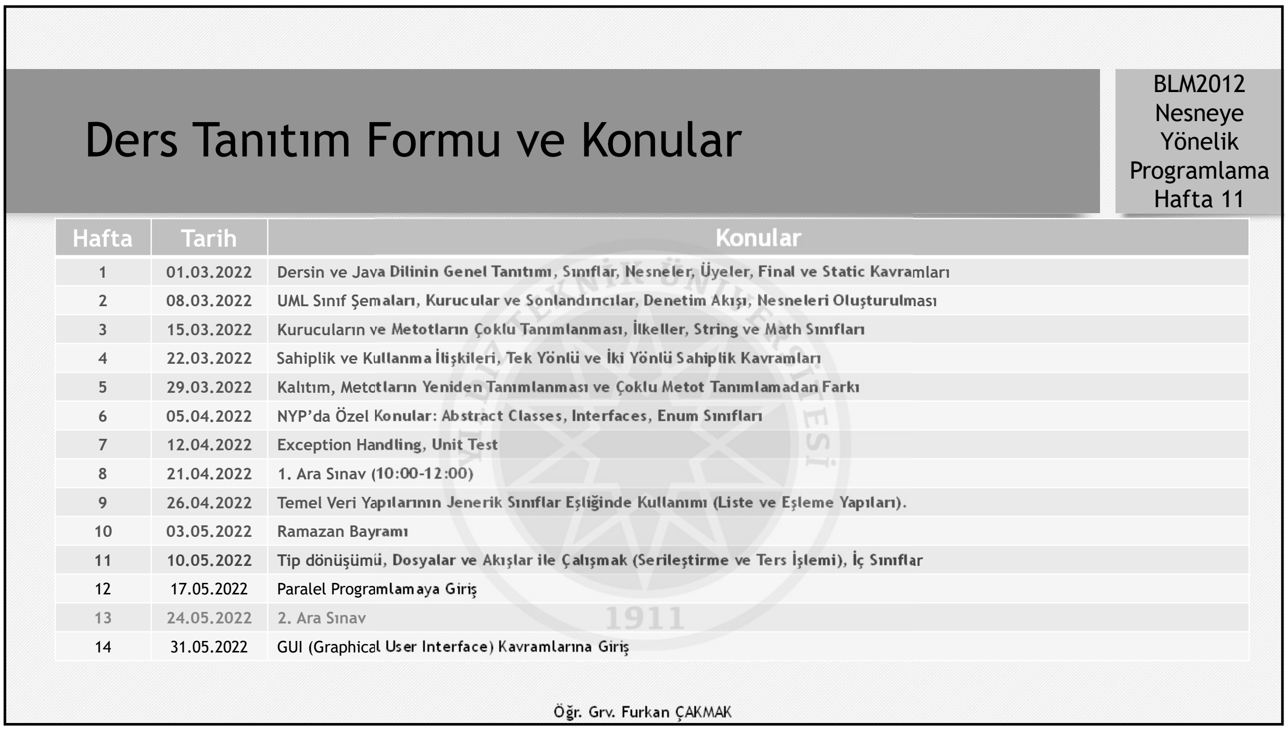


Nesneye Yönelik Programlama BLM2012

Öğr. Grv. Furkan ÇAKMAK



Ders Tanıtım Formu ve Konular

BLM2012 Nesneye Yönelik Programlama Hafta 11

Hafta	Tarih	Konular
1	01.03.2022	Dersin ve Java Dilinin Genel Tanıtımı, Sınıflar, Nesneler, Üyeler, Final ve Static Kavramları
2	08.03.2022	UML Sınıf Şemaları, Kurucular ve Sonlandırıcılar, Denetim Akışı, Nesneleri Oluşturulması
3	15.03.2022	Kurucuların ve Metotların Çoklu Tanımlanması, İlkeller, String ve Math Sınıfları
4	22.03.2022	Sahiplik ve Kullanma İlişkileri, Tek Yönlü ve İki Yönlü Sahiplik Kavramları
5	29.03.2022	Kalıtım, Metotların Yeniden Tanımlanması ve Çoklu Metot Tanımlamadan Farkı
6	05.04.2022	NYP'da Özel Konular: Abstract Classes, Interfaces, Enum Sınıfları
7	12.04.2022	Exception Handling, Unit Test
8	21.04.2022	1. Ara Sınav (10:00-12:00)
9	26.04.2022	Temel Veri Yapılarının Jenerik Sınıflar Eşliğinde Kullanımı (Liste ve Eşleme Yapıları).
10	03.05.2022	Ramazan Bayramı
11	10.05.2022	Tip dönüşümü, Dosyalar ve Akışlar ile Çalışmak (Serileştirme ve Ters İşlemi), İç Sınıflar
12	17.05.2022	Paralel Programlamaya Giriş
13	24.05.2022	2. Ara Sınav
14	31.05.2022	GUI (Graphical User Interface) Kavramlarına Giriş

Öğr. Grv. Furkan ÇAKMAK

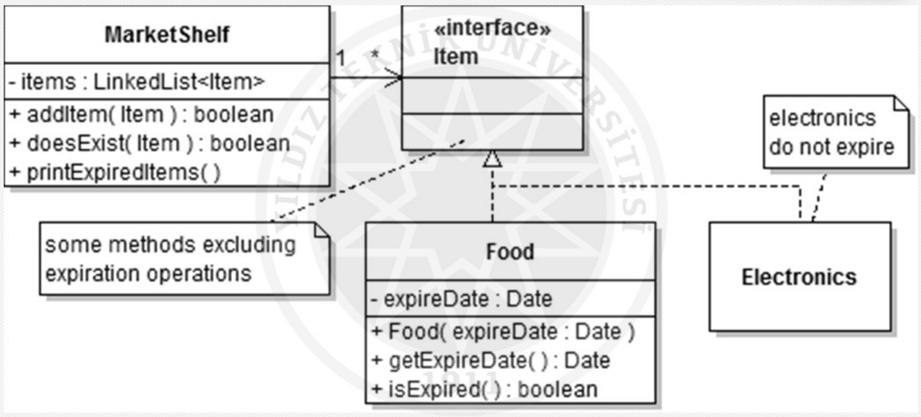
TYPECASTING

BLM2012 Nesneye Yönelik Programlama Hafta 11

- An instance of a sub class can be used wherever an instance of its super class is expected.
  - Type-Safe Operation
- We can convert a specific object to a more general one without loosing any information.
- You can make a manual cast from one type to another, according to the following rules:
  - From the interface to the class of the object
  - From the super class to the sub class

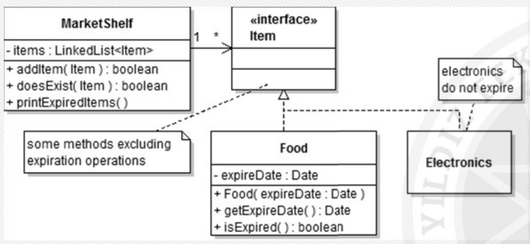
TYPECASTING - EXAMPLE

BLM2012 Nesneye Yönelik Programlama Hafta 11


MarketShelf
- items : LinkedList<Item>
+ addItem(Item) : boolean
+ doesExist(Item) : boolean
+ printExpiredItems( )
1
*
«interface»
Item
some methods excluding expiration operations
Food
- expireDate : Date
+ Food(expireDate : Date)
+ getExpireDate( ) : Date
+ isExpired( ) : boolean
electronics do not expire
Electronics

TYPECASTING – EXAMPLE (CON’T)

BLM2012 Nesneye Yönelik Programlama Hafta 11

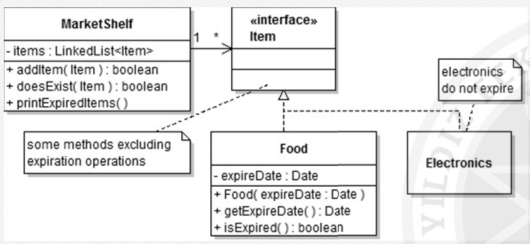

MarketShelf
- items : LinkedList<Item>
+ addItem(Item) : boolean
+ doesExist(Item) : boolean
+ printExpiredItems( )
1
*
«interface»
Item
electronics do not expire
some methods excluding expiration operations
Food
- expireDate : Date
+ Food(expireDate : Date)
+ getExpireDate( ) : Date
+ isExpired( ) : boolean
Electronics

```
package mypkg;
import java.util.*;
public class MarketShelf {
    private LinkedList<Item> items;
    public MarketShelf() {
        items = new LinkedList<Item>();
    }
    public boolean doesExist( Item anItem ) {
        for( Item item : items )
            if( item == anItem )
                return true;
        return false;
    }
    public boolean addItem( Item anItem ) {
        if( doesExist(anItem) )
            return false;
        items.add(anItem);
        return true;
    }
```

```
public void printExpiredItems( ) {
    boolean hasExpiredItem = false;
    System.out.println("Expired item(s): ");
    for( Item item : items ) {
        if( item instanceof Food )
            if( ((Food)item).isExpired() ) {
                hasExpiredItem = true;
                System.out.println(item);
            }
    }
    if( hasExpiredItem == false )
        System.out.println("All items are fresh!");
}
```

TYPECASTING – EXAMPLE (CON’T)

BLM2012 Nesneye Yönelik Programlama Hafta 11

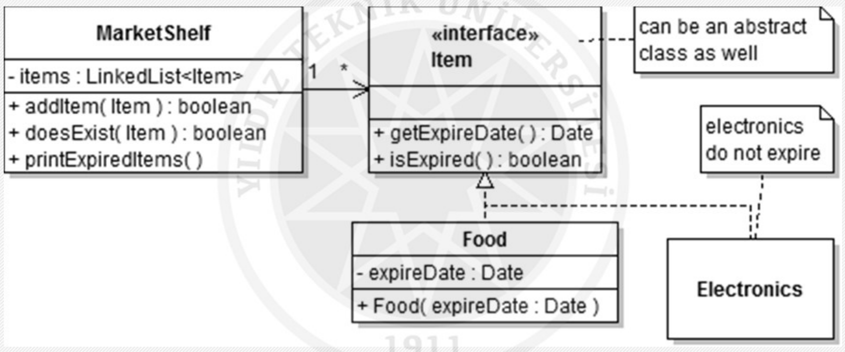

MarketShelf
- items : LinkedList<Item>
+ addItem(Item) : boolean
+ doesExist(Item) : boolean
+ printExpiredItems( )
1
*
«interface»
Item
electronics do not expire
some methods excluding expiration operations
Food
- expireDate : Date
+ Food(expireDate : Date)
+ getExpireDate( ) : Date
+ isExpired( ) : boolean
Electronics

```
public static void main( String[] args ) {
    MarketShelf shelf = new MarketShelf();
    Calendar cal = Calendar.getInstance();
    cal.add(Calendar.DAY_OF_MONTH,1);
    Date future = cal.getTime();
    cal.set(Calendar.YEAR, 2010);
    cal.set(Calendar.MONTH, 0); //0: January
    cal.set(Calendar.DAY_OF_MONTH, 1);
    Date past = cal.getTime();
    shelf.addItem( new Food(past) );
    shelf.addItem( new Food(future) );
    shelf.addItem( new Electronics() );
    shelf.checkForExpiration();
}
```

```
package mypkg;
import java.util.Date;
public class Food implements Item {
    private Date expireDate;
    public Food( Date expireDate ) { this.expireDate = expireDate; }
    public Date getExpireDate() { return expireDate; }
    public boolean isExpired() {
        Date today = new Date();
        if( today.after(expireDate) )
            return true;
        else
            return false;
    }
    public String toString() {
        return "Food expiring at " + expireDate;
    }
}
```

TYPECASTING – EXAMPLE (CON'T)

BLM2012 Nesneye Yönelik Programlama Hafta 11


MarketShelf
- items : LinkedList<Item>
+ addItem(Item) : boolean
+ doesExist(Item) : boolean
+ printExpiredItems( )
1
*
«interface» Item
+ getExpireDate( ) : Date
+ isExpired( ) : boolean
can be an abstract class as well
electronics do not expire
Food
- expireDate : Date
+ Food(expireDate : Date)
Electronics

WORKING WITH FILES

BLM2012 Nesneye Yönelik Programlama Hafta 11

- RELATED EXCEPTIONS
  - java.io.IOException: Represents I/O exceptions in general.
  - java.io.EOFException extends IOException: Indicates that the end of file or stream has been reached unexpectedly.
  - java.io.FileNotFoundException extends IOException: Indicates that the requested file cannot be found in the given path.
  - java.lang.SecurityException extends java.lang.RuntimeException: Indicates that the requested operation cannot be executed due to security constraints.
- File operations are separated into two main groups in Java:
  - File management: Operations such as creating, renaming, deleting files and folders.
  - I/O operations.

WORKING WITH FILES - FILE MANAGEMENT

BLM2012 Nesneye Yönelik Programlama Hafta 11

- java.io.File
  - Files
  - Folders
- File(String fileName)
  - fileName should contain both the path and the name of the file/folder.
  - Full path vs. relative path.
- Path separator:
  - Windows uses \ (should be denoted as \\ in Strings), Unix uses /.
  - public static String File.separator
  - public static char File.separatorChar
- File(String path, String name) and File(File path, String name) constructors:

WORKING WITH FILES - FILE MANAGEMENT

BLM2012 Nesneye Yönelik Programlama Hafta 11

- Some methods of the class java.io.File:
  - boolean exists( ); tells whether the file exists or not.
  - boolean isFile( ); returns true if this File object represents a file, false otherwise, i.e. this object represents a folder.
  - File getParentFile( ); Returns the directory where this file/folder resides.
  - String getCanonicalPath( ) throws IOException; Returns the full path of the file/folder, including the file name.
  - boolean canRead( ); Can this application read form this file?
  - boolean canWrite( ); Can this application write to this file?
  - boolean createNewFile( ); Actually creates the file. Only for files!
  - boolean mkdir( ); Actually creates the folder. Only for folders!
  - boolean mkdirs( ); Actually creates the folder with all necessary parent folders. Only for folders!
  - boolean renameTo(File newName); Renames the file.
  - boolean delete( ); Deletes the file.

WORKING WITH FILES - I/O OPERATIONS

BLM2012 Nesneye Yönelik Programlama Hafta 11

- I/O OPERATIONS USING STREAMS
  - Any I/O source is represented as stream in Java
    - Files, memory, command prompt, network, etc.
  - Binary vs. Text format:
    - Binary I/O is fast and efficient, but it is not easily readable by humans.
    - Text I/O is the opposite.
  - Random vs. Sequential access:
    - Sequential access: All records are accessed from the beginning to the end
    - Random access: A particular record can be accessed directly.
    - Disk files are random access, but streams of data from a network are not.

WORKING WITH FILES - I/O OPERATIONS

BLM2012 Nesneye Yönelik Programlama Hafta 11

- Serialization – Output operations:
  - We will write entire objects to a file on disk.
  - java.io.Serializable
  - ObjectOutputStream and FileOutputStream objects are chained together for serialization.
  - About the transient keyword

WORKING WITH FILES - I/O OPERATIONS

- About the lines beginning with @ :
  - These are special commands called "annotations".
  - They work at the "meta" level, i.e. they contain "information about information".
  - They give information to the IDE, compiler, another programme, etc. about this program.
  - We have used the annotation mechanism to remove the warnings.
  - In fact, warnings must be taken into consideration.
  - Example: @SuppressWarnings("serial")

WORKING WITH FILES - I/O OPERATIONS - EXAMPLE

- Kod Gösterimi Yapılsın!

INNER CLASSES

BLM2012 Nesneye Yönelik Programlama Hafta 11

- You can code a class within a class.
  - An inner class is coded within an outer class.
- An inner class can:
  - Access all members of the outer class, including the private ones.
  - Be hidden from other classes of the same package, if defined as private.
  - It is frequently used in form of anonymous inner classes in multithreaded and GUI programming.
    - Anonymous = without a name!
- You cannot:
  - define a static method in a an inner class.

Öğr. Grv. Furkan ÇAKMAK

INNER CLASSES - EXAMPLE

BLM2012 Nesneye Yönelik Programlama Hafta 11

- Kod Gösterimi Yapılsın!

Öğr. Grv. Furkan ÇAKMAK

Sabırla Dinlediğiniz İçin Teşekkürler

BLM2012 Nesneye Yönelik Programlama Hafta 11

Teşekkürler

Öğr. Grv. Furkan Çakmak