

Gebze Technical University  
Computer Engineering

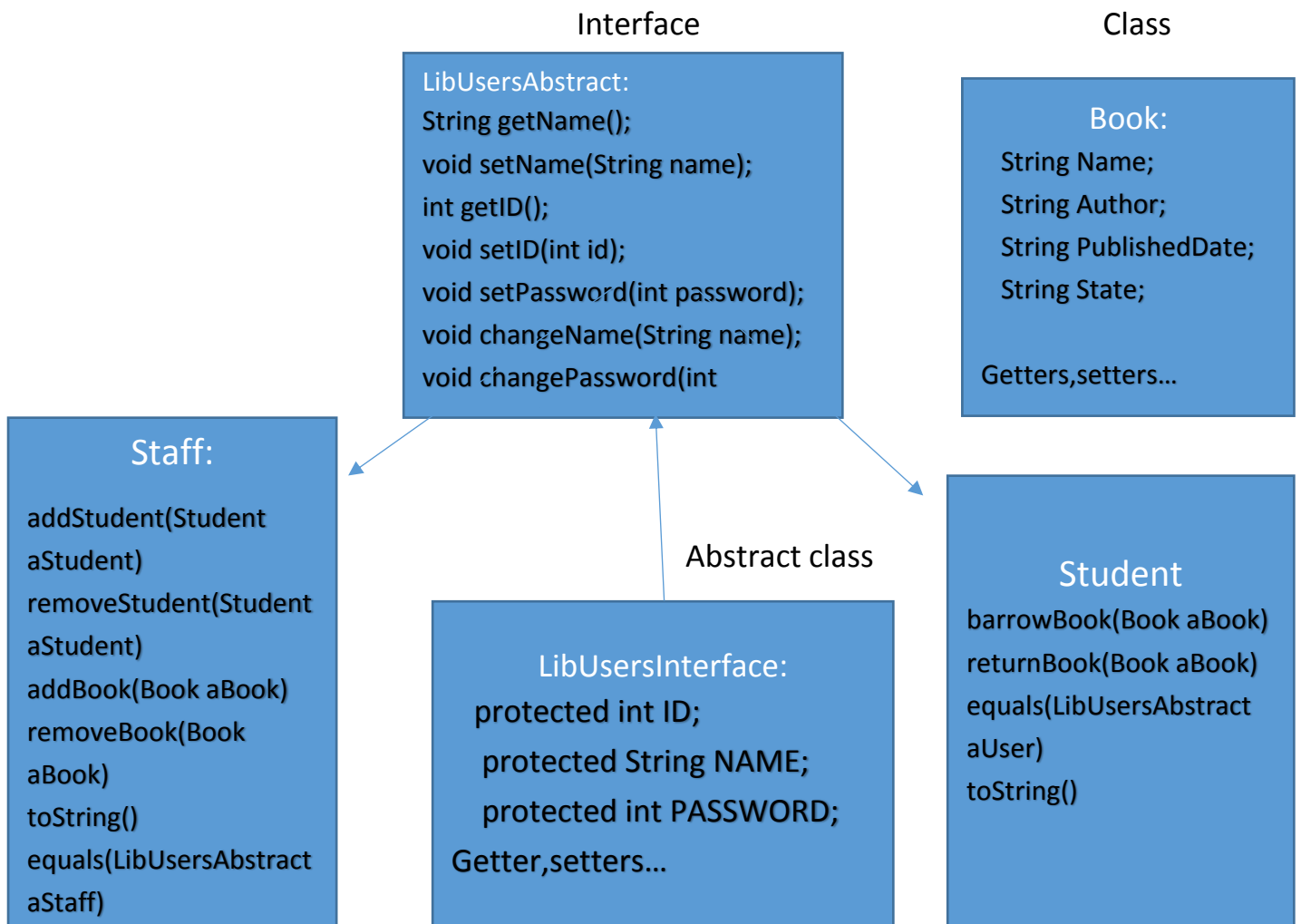
CSE 222  
2017 Spring

HOMEWORK 01 REPORT

GULZADA IISAEVA  
131044085

- Ödevde istenildiği üzere bir kütüphane sistemi tasarlandı. Kullanacak olduğumuz interface ve abstract class'ların yapıları tasarlandı. Tasarım aşamasında gerekli requirementlar belirlenerek classlar arasındaki hiyerarşi kuruldu.
- Proje Moodle'da verildiği CSE222-VM İntelijde yapıldı
- Object Oriented'ın temel prensiplerine ve clean code standartlarına uyulmaya dikkat edildi.
- Grup dataları bir arada tutmak için sadece **ArrayList** yapısı kullanılmıştır.
- **Javadoc** kullanılarak class içerisinde yer alan methodların implementasyonları gerçekleştirilmiştir.
- **Exception Handling** gerekli methodlarda uygulanmıştır.

## Classes diagram:



- Tüm kullanıcılar Users **interface**'inden implement olmuşlardır. Bu bize polymorphism ve inheritance sağlamıştır. Ayrıca olarak Book classı var.
- Kütüphanenin yönetmeni (staff) ve kullanıcıları vardır. Staff ve kullanıcıların sisteme giriş için id ve şifresi vardır. Yönetici hem kullanıcıları (students) hem kitapları ekleyebilir ve silebilir. Kullanıcılar kitap ödünç alabilir ve geri verebilir.
- Ödevde istenildiği gibi csv dosyasına kayıt etme kısmını zamanımın kısıtlı olmasından dolayı yetiştiremedim.
- GitHub Link [https://github.com/gulzadaiisaeve/CSE222\\_HW1](https://github.com/gulzadaiisaeve/CSE222_HW1)
- IntelliJ de Main.java ana class seçilerek çalıştırılır.

### Program Output

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gulzada_iisaeve_131044085_hw1 > src > Main >
Project Run Main
Run Main
/usr/lib/JDK1.8.0_121/bin/java ...
/*****ADMIN *****/
Admin Name-> Gulzada IIsaeve
/*****Students*****/

ID: 1200
Name: Ali Kaya
ID: 12775
Name: Sena Turkmen
ID: 182775
Name: Ayse Demirci
/*****

*****After removing student 2*****
ID: 1200
Name: Ali Kaya
ID: 182775
Name: Ayse Demirci
/*****

/*****BOOKS *****/

State <Borrowed > and <null>

Name : Data Structures and Algorithm
Author : Elliot B.Koffman
Published date : 2009
State : null

Name : Absolute C++
Author : Walter Savitch
Published date : 2010
State : null
/*****

Compilation completed successfully in 3s 451ms (3 minutes ago)

```

The screenshot shows an IDE window titled "Iisaeva\_131044085\_hw1 - [~/gulzada\_iisaeva\_131044085\_hw1] - [gulzada\_iisaeva\_131044085\_hw1] -". The menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The project explorer on the left shows the project structure: gulzada\_iisaeva\_131044085\_hw1 > src > Main. The editor displays the code in Main.java, with line numbers 56 to 61 visible. The code is as follows:

```
56 ArrayList<Book> books=admin.getBooks();
57 System.out.println("/*****");
58 System.out.println("State <Borrowed");
59 ListIterator<Book> iter3= books.listIterator();
60 while(iter3.hasNext()){
61     System.out.print(iter3.next()+"
```

The Run window at the bottom shows the output of the program:

```
*****After removing book 1*****
Name : Absolute C++
Author : Walter Savitch
Published date : 2010
State : null
/*****
/*****After borrowing *****/
Name : Absolute C++
Author : Walter Savitch
Published date : 2010
State : Borrowed
/*****After returning *****/
Name : Absolute C++
Author : Walter Savitch
Published date : 2010
State : null
Process finished with exit code 0
```

At the bottom, a status bar indicates "Compilation completed successfully in 2s 580ms (moments ago)".

Q2) Encapsulation is possible in non-object-oriented languages. In C, for example, a structure can be declared in the public API (i.e., the header file) for a set of functions that operate on an item of data containing data members that are not accessible to clients of the API:

```
// Header file "api.h"

struct Entity;           // Opaque structure with hidden members

// API functions that operate on 'Entity' objects
extern struct Entity * open_entity(int id);
extern int process_entity(struct Entity *info);
extern void close_entity(struct Entity *info);
```

Clients call the API functions to allocate, operate on, and deallocate objects of an **opaque data type**. The contents of this type are known and accessible only to the implementation of the API functions; clients cannot directly access its contents. The source code for these functions defines the actual contents of the structure:

```
// Implementation file "api.c"

#include "api.h"

// Complete definition of the 'Entity' object
struct Entity {
    int    ent_id;           // ID number
    char   ent_name[20];     // Name
    ... and other members ...
};

// API function implementations
struct Entity * open_entity(int id)
{ ... }

int process_entity(struct Entity *info)
{ ... }

void close_entity(struct Entity *info)
{ ... }
```