Technical University of Denmark
DTU Informatics
Assoc. Prof. H. Baumeister

Spring 2013
Prog. Assignment Nr. 3
February 26, 2013

02161: Software Engineering 1

3.1 EclEmma

- Install the Java code coverage tool EclEmma (`www.eclemma.org`) in Eclipse. Installation instructions can be found on the homepage of EclEmma `http://eclemma.org/installation.html`.
- Run EclEmma with the tests and your implementation from previous week. What is your code coverage?

3.2 Library Application (cont.): Logoff

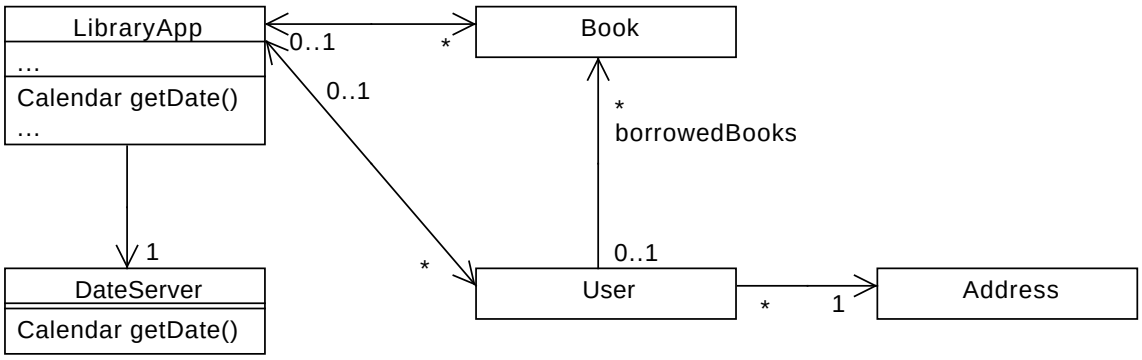
- The administrator can logoff the library application.
- Tasks
  - Download library03.zip
  - Copy your source files developed in the last weeks into the src directory
  - Develop test-driven the logoff functionality by repeating creating tests for usage scenarios and then the implementation for these scenarios. Please note that you should put your tests into the source directory `student_tests`. This will later allow you to copy your tests into the next version of the library project provided by me.
  - Run EclEmma to see the code coverage you have achieved

3.3 Library Application (cont.): Manage users

- The administrator can unregister a user from the library. The administrator can unregister a user only, if the user does not have any borrowed books.
- Tasks
  - Develop test-driven the unregister functionality by repeating creating tests for usage scenarios and then the implementation for these scenarios
  - Make sure that you cover all the important scenarios: e.g. that the administrator is logged in before performing the unregistration and what happens if he is not and what happens if the user still has borrowed books.
  - Run EclEmma to see the code coverage you have achieved

3.4 Library Application (cont:): Overdue books

- Implement the scenario that the user is not allowed to borrow a book if he has an overdue book. An exception should be thrown in this case
- Tasks
  - Implement the tests in the class TestDateServer
  - Implement one by one the tests in the class TestOverdueBook
  - Refactor your solution, such that the method signature of the boorowBook operation in class User is `public void borrowBook(Book book) throws BorrowException`
  - Run EclEmma to see the code coverage you have achieved
- The intended design for the solution is as follows
  - Book and User each should have a reference to a LibraryApp to be able to get the current date from the DateServer.



Tip on implementing access to the date server

- To implement checking if a book is overdue, the class Book and maybe also the class User (depending on your implementation), need access to an object of class LibraryApp that has a reference to the date server. This is shown in the above class diagram by arrows between class LibraryApp and Book and LibraryApp and User respectively. An instance of class LibraryApp has references to books (stored in the field books of type List<Book>) and an instance of class Book / User has a field libApp of type LibraryApp. That is,

```
public class LibraryApp {
  private DateServer dateServer = new DateServer();
  private List<Book> books = new ArrayList<Book>();
  private List<User> users = new ArrayList<User>();
  ...
  public Calendar getDate() { return dateServer.getDate(); }
  void setDateServer(DateServer ds) { dateServer = ds; }
  ...
}
```

```
public class Book {
  private String signature;
  private String title;
  private String author;
  private LibraryApp libApp;
  ...
  void setLibraryApp(LibraryApp libApp) {
    this.libApp = libApp;
  }
  ...
}
```

- Class field libApp in class User can be implemented in a similar way as in class Book.
- One question remains: when to set the field in class Book and User, resp.? The smartest way to do this is, when a book or a user is added to the library. In this case, the instance of class LibraryApp sets the link back to the LibraryApp with each book that is added to the library. This allows one to make sure, that whenever a book is in the library (i.e. is contained in the field books of LibraryApp), its libApp field points back to the instance of the LibraryApp it is part in.

```
public void addBook(Book book) {
  ...
  books.add(book);
  book.setLibraryApp(this);
  ...
}
```

- Again, the same applies for class User. Here the libApp field in class User is set, when the user is registered with the LibraryApp.

3.5 Library Application (cont:): Sending mail reminders regarding overdue books

Implement, using test-driven development, the function sendEMailReminder(). When called, this function should send out an e-mail to all the users who have overdue books with the following subject: "Overdue book(s)" and the following contents: "You have <n> overdue books", where <n> is replaced with the number of overdue books the user has.

Similar to the class DateServer, the class LibraryApp has a field containing the MailService to be used (this allows one to later substitute the real MailService by a mock mail service).

For the purpose of this exercise, the MailService class has a method send(String email, String subject, String text), that simply prints out e-mail address, subject and text on the console. A possible implementation of the MailService class is:

```
public class MailService {

  public void send(String email, String subject, String text) {
    System.out.println("Mail sent to "+email+". Subject: \""+subject+"\", text: \""+text+"\"");
  }

}
```

- Tasks
  - Implement the function sendEMailReminder test-driven.

  - Use mock objects to test, that the method sendEMailReminder tries to send the correct e-mail messages using the MailService, i.e., that the method send(email,subject,text) is sent to the MailService with the correct arguments.
  - Create at least test cases for the following scenarios
    * One user has one overdue book and then sendEMailReminder is called
    * One user has two overdue books and then sendEMailReminder is called
    * Two users have each one overdue book and then sendEMailReminder is called
    * Two users have borrowed books, but they are not overdue and then sendEMailReminder is called

- Here is documentation on the mock object framework Mockito `http://docs.mockito.googlecode.com/hg/latest/org/mockito/Mockito.html`