HACETTEPE UNIVERSITY

COMPUTER ENGINEERING

SOFTWARE ENGINEERING LABORATORY

BBM487

Group 17

Members

ID	Name and Surname
21200894	Gismat Kazımlı
21228602	Hiledin Özer
20902958	Rinat Davidov
21228046	Burak Aykanat

Document Version: Coding Standart

Date: 25.04.2017

Table of contents

1. Naming Convention Standards

We used Camel Case standard for our naming standard.

1.1. Naming Variables

It is necessary to write the name of the variable that the variable represents so that it can be understood as what a variable represents. Variable names can be written in abbreviations, such as when the names are too long, but these abbreviations need to keep the variable understandable. We have declared our variables according to these rules and adopted as a standard.

An example variable name which is written as a Camel Case standard "String bookText"

1.2. Naming Constants

Constants mean the variable which does not change in any steps of code. These are generally implemented using **static final** keywords. For example static final String USER.

1.3. Naming Collection

Collection is defining arrays and array list in our code. Each array and array list named in a way that indicates the stored data.

For example: Arraylist<Book> bookList = new Arraylist<>();

1.4. Naming Components

We have a few components and so we created our standard to specify them. Our main component which has most component is User interface. And we named them all to their purpose and what are they doing clearly. For example to see the down of the page we have mouseDown button name.

2. File Organization

We write our code using Java. And We know that Java has to two file extension as *.java which is source code and *.class which is Java byte code that is compiled by JVM. Every class identified like public class DatabaseHandler. And in addition every class name start with an uppercase and defines the main goal of the class like Database handler that is specifying connections between software and database. In our software

- Each class has a separate file, so each class is public.
- We kept each class less than 500 line.
- Each class was created in separate packages according to its own type. Such interfaces in the interface package, objects are in the object package.

3. Comment standards

Our every source files start with c-styles command lines and these block lines are followed

```
/*
 * ProjectName.java
 *
 *  Created on: dd/mm/yyyy
 *      Author: *
 */
```

And to create understandable methods and readable software each methods has a definition according to its own properties like following;

```
/**
*
* @param email
* @param password
* @return
*/
public Librarian checkLogin(String email, String password){

//operations

}
```

This is the format to create Javadoc format actually and we used it so may use Javadoc to understand and read our software better.

These two kind of command are the beginning comments block lines that are using before starting and kind of a class or method. So we used another types of command line and these are

End of Line Comments that is creating command lines by using //

We use this type to commenting the different lines of code. For example;

```
int id;  // user unique id
String fullname; // user name
String email; // user email
```

Coding conventions

Our coding conventions contains filenames such as package and class name, declarations for each conventions, statement for each line of code, naming conventions that has standards. Our software contains many variable, constant, collection and component in its code. So we have standards for each of them to create a readable and understandable software.

White space

We have **blank lines** in our code to improve readability. But you should know that these blank line are not between any opening and closing parenthesis pair. We mostly used blank lines after a line which is finished with comma. By doing this, each part of code can be more legible. In additional we have white spaces to create a better understandable software. And this white space mostly used between operation conditions, initializing values, after opening and before closing parenthesis if there is variable between them. For example;

```
If( a == b ){

x += x + y;

y =  (2 * y ) + ( 2 * a);

}
```