



SWIM-Suite

A Swim Meet and Swimmer Management Software

Prototype and Testing Report

Submitted on: November 4th, 2010

Team Members (in alphabetical order):

AbhishekBindiganavile

Robert Greenberg

SedatOzer

Jay Takle

1. Project Repository:

Google code repository: <http://code.google.com/p/swimsuite/>
SVN: <http://code.google.com/p/swimsuite/source/browse/#svn>

2. Subsystem Services:

In our prototype we have implemented three subsystems, viz. WebInterface, UserManagement and SwimDatabase. Following is the detailed object design and specifications for our subsystem interfaces implemented in the first prototype.

2.1 Subsystem WebInterface

Subsystem webinterface provides an interface between the user and other subsystems in the application logic layer.

Class Summary	
formField	This class consists of a text field provided to the user to input his/her text.
login	This class is a base class which describes the template for all the buttons.
displayProfile	This class handles the display functions required to display contents on the screen.

Subsystem webinterface Description

Subsystem webinterface provides an interface between the user and the other subsystems in the lower (application logic layer). It contains the classes FormFields, Buttons and Display. FormFields class provides a text field where the user can input his/her profile details. Buttons class is the base class which performs the respective functionality when the corresponding button is clicked. Display class provides a UI for displaying content as requested on the webpage.

webinterface

Class formField

java.lang.Object

webinterface.formField

```
public class formField
extends java.lang.Object
```

This class consists of a text field for the user to input his/her profile details. The text field takes an input of type String. The input from the user is the parameter sent to the webinterface subsystem. It acts as the interface between the user and the application.

Version: 1.0

Constructor Summary

`formField()`

Method Summary

`void`

`createForm()`

Generates the input text field that is displayed when the user starts the application

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

formField

`Public formField()`

Method Detail

createForm

`public void createForm()`

throws `IOException`

Generates the input text field when the user starts the application

Parameters:

`form_name` - a `String` storing the name of the user

`form_password` - a `String` storing the password of the user

Throws:

`IOException` - if the input format is incorrect

webinterface

Class login

java.lang.Object

webinterface.login

public class login

extends java.lang.Object

This class belongs to the base class which describes the template for all the buttons. It is inherited by the subclass buttons each of which provide a different service to the user. It belongs to the input terminal subsystem which is the interface between the user and the application

Version: 1.0

Constructor Summary

login()

Method Summary

void

getParameter()

This function accepts the input parameters from the user.

void

setAttributes()

This function puts the parameters in variables which can be sent to some other class.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

login

Public **login**()

Method Detail

submit

public void getParameter()

This function accepts the input parameters from the user.

public void setAttributes()

This function puts the parameters in variables which can be sent to some other class.

webinterface

Class Display

java.lang.Object

View

webinterface.Display

```
public class Display
```

```
extends Display
```

This class handles the display functions required to display the contents on the screen. This class extends the base class Display.

Version: 1.0

Constructor Summary

Display()

Method Summary

void	setAttributes() This function puts the parameters in variables which can be sent to some other class.
void	getAttributes() This function accepts the parameters in variables which have been sent from some other class.
void	getRequestDispatcher() This function redirects a page.

Constructor Detail

Display

Public **Display**()

Method Detail

displayPage

```
public void setAttributes()
```

This function puts the parameters in variables which can be sent to some other class

```
public void getAttributes()
```

This function accepts the parameters in variables which have been sent from some other class

```
public void getRequestDispatcher()
```

This function redirects a page.

2.2 Subsystem UserManagement

Subsystem UserManagement handles the control in the application logic layer related to different functionalities of the system pertaining to different users

Class Summary	
SwimmerApplication	This class manages the different functions and the decision logic to delegate control to the different subsystems pertaining to the swimmer.

Subsystem UserManagement Description

Subsystem UserManagement handles the control in the application logic layer related to different functionalities of the system pertaining to different users. It consists of classes OrganizerApplication and SwimmerApplication. The OrganizerApplication class handles control to the different functions that the organizer is authorized to perform, i.e the control is delegated to the respective subsystems that the organizer has access to. The SwimmerApplication class handles control to the different functions that the swimmer is allowed to perform, i.e, the control is delegated to the respective subsystems that the swimmer has access to.

UserManagement

Class SwimmerApplication

java.lang.Object

UserManagement.SwimmerApplication

```
public class SwimmerApplication
```

```
extends java.lang.Object
```

This class manages the control handling pertaining to the swimmer and transfers control to the respective subsystems depending on the functionality requested by the swimmer.

Version: 1.0

Constructor Summary	
SwimmerApplication()	

Method Summary	
void	createProfile(string ID) This function stores the form details pertaining to the id that is taken as an input parameter, into the database server.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail**SwimmerApplication**

Public **SwimmerApplication()**

Method Detail**createSwimmerProfile**

public void **createProfile(string ID)**

This function submits a query to the database server to enter the profile details pertaining to the ID that is input as a parameter, into the database

Parameters:

ID - is a string data type that specifies the identity of the user(Swimmer).

2.3 Subsystem SwimDatabase

Subsystem SwimDatabase handles all the data in the system.

Class Summary

connect	This class provides the interface between the Server/Storage layer and the application logic layer.
---------	---

Subsystem SwimDatabase Description

Subsystem SwimDatabase provides an interface between the Server/Storage Layer and the Application Logic layer of the system. It contains the class connect which maintains a list of all the data present in the system in the form of database tables. It accepts queries from the UserManagement subsystem in this prototype and returns relevant information to it.

SwimDatabase

Class database

java.lang.Object

SwimDatabase.connect

public class connect

extends java.lang.Object

This class provides the interface between the Storage/Server layer and the application logic layer.

Version: 1.0

Constructor Summary

`connect()`

Method Summary

<code>string</code>	<code>executeupdate()</code> This function accepts a string as an input parameter, the string could pertain to swimmer. It then puts the string into the table requested by the functions.
<code>string</code>	<code>executequery()</code> This function accepts a string as an input parameter, the string pertains to swimmer in this prototype. It then displays all the strings related to this parameter in the table requested by the functions.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

connect

`public connect()`

3 OCL specifications:

3.1 Textual OCL Notation of the UserManagement subsystem

CreateProfile Class

context. CreateProfile.inv:

FirstName<=50 and LastName<=50 and SwimmerId<=10.

context.CreateProfile:: getParameter("FirstName") **Pre:**
FirstName->Empty()

context.CreateProfile:: getParameter("LastName") **Pre:**
LastName->Empty()

context.CreateProfile:: getParameter("SwimmerId") **Pre:**
SwimmerId->Empty()

context.CreateProfile:: getParameter("DateOfBirth") **Pre:**
DateOfBirth-> Empty()

context.CreateProfile:: getParameter("FirstName") **Pre:**
FirstName-> Empty()

context.CreateProfile:: getParameter("LastName") **Pre:**
LastName-> Empty()

context.CreateProfile:: getParameter("SwimmerId") **Post:**
SwimmerId-> NotNull ()

context.CreateProfile:: getParameter("LastName") **Post:**
LastName-> NotNull ()

context.CreateProfile:: getParameter("SwimmerId") **Post:**
SwimmerId-> NotNull ()

context.CreateProfile:: getParameter("DateOfBirth") **Post:**
DateOfBirth->NotNull ()

context.CreateProfile:: getParameter("Gender") **Post:**
Gender->NotNull ()

context.CreateProfile:: getParameter("Club") **Post:**
SwimmerId->NotNull ()
CreateProfile.response()

3.2 Textual OCL Notation of the UserManagement subsystem

Login Class

context.Login inv:

FirstName<=50 and LastName<=50 and SwimmerId<=10.

context.Login::getParameter("FirstName") **Pre:**

FirstName->Empty()

context.Login::getParameter("FirstName") **Pre:**

FirstName->Empty()

context.Login::getParameter("FirstName") **Pre:**

FirstName->Empty()

!submit()

context.Login::submit() **Post:**

displayprofile()

3.3 OCL Specification UserManagement Subsystem.

The main motive of the UserManagement subsystem is to manage the different types of users who interact with the system and perform operations such as creating profile information. Once the control is transferred from the WebInterface subsystem, the UserManagement subsystem, depending upon the key such as the unique ID, the required message page is displayed to the swimmer and the organizer respectively.

The OCL specifications helped the coding of prototype a great deal as it gives us the understanding of the invariant, pre and post conditions of member functions and variables. Further because of the OCL specifications some unwanted operations which would cause the code to work illogically for some input values is avoided (Such as invalid swimmer id's). OCL specifications lead to improved code efficiency.