FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	CSC2B10 **COMPUTER SCIENCE 2B**
CAMPUS	AUCKLAND PARK CAMPUS (APK)
PRACTICAL TEST	A

DATE: 2015-09-22 **SESSION:** 14h00 - 17h00

ASSESOR(S): **MR A MAGANLAL**

MODERATOR: **DR DT VAN DER HAAR**

DURATION: 180 MINUTES **MARKS:** 100

Instructions

- Work on the T: drive.
- Save every 5 minutes.
- Submit every 10 minutes.
- The folder on the T: drive as well as the submitted zip file must be named as follows:

  `SURNAME_INITIALS_STUDENTNUMBER_CSC2B10_2015_PTA`

Make sure to save and submit your work regularly.

The University of Johannesburg has tasked you to develop a networked client application and server application. The application is a downsized social media application which makes use of the POSTIT protocol. The POSTIT protocol allows clients to register and login. Each registered client is assigned a folder where their social media posts will be stored. Only clients which are registered and logged in can post. Posts consist of a text file and an image. POSTIT runs on port 2015.

The following request commands are available in POSTIT:

- `REG <Name> <Password>` Provide a name and password for client registration, e.g. `REG John p455w0rd`. The server will allocate a folder for this user for posts. The folder will be created in the `server` folder in the `data` folder. If the client is already registered then an error must be returned, otherwise a success message is returned if successful.
- `LOGIN <Name> <Password>` Provide a name and password for client login, e.g.

  `LOGIN John p455w0rd`. The server will allow the user to post once logged in. If the client doesn't exist or the password is incorrect then an error must be returned, otherwise a success message is returned when successful.
- `POST <Title> <Message> <Image>` Creates a new post in the client's folder. This command will create a text file which will store the `<Message>` and an image file. Both the text file and image file will be named `<Title>`. If the user is not logged in then an error must be returned.
- `LOGOFF` Log a client off.

The following responses are used in POSTIT:

- `#OK <Message>` Successful command with `<Message>` providing a helpful message from the server.
- `&ERR <Message>` Unsuccessful command with `<Message>` providing a reason that the command did not work.

Complete the **`POSTITServer`** class. This class is responsible for binding to the POSTIT port to listen for clients. The server must be able to handle multiple clients. Any clients which connect are handled by the **`POSTITHandler`**.

Complete the **`POSTITHandler`** class. This class is responsible for handling commands which are received from a client. This class is also responsible for handling client registration and login. Registered users are stored in a text file called `users.txt`.

Complete the **POSTITClient** and **POSTITClientFrame** classes. The **POSTITClient** class will processes any communication using POSTIT. The **POSTITClientFrame** class will have buttons for each command.

- When the client clicks on the `REG` button the client must be asked for a name and password.
- When the client clicks on the `LOGIN` button the client must be asked for a name and password.
- When the client clicks on the `POST` button the client must be asked for a title, message and must be able to choose an image file to send.

Any errors from the server must be displayed to the client.

Reminder

Your submission must follow the naming convention.

`SURNAME_INITIALS_STUDENTNUMBER_CSC2B10_2015_PTA`

Your submission must include the following folders:

- `bin` - Should be empty at submission but will contain runnable binaries when your submission is compiled.
- `docs` - Generated JavaDoc and any additional documentation files.
- `src` - Contains all relevant source code. Source code must be places in relevant sub-packages!
- `data` - Contains sub-folders for client and server where transferred files are saved.

Student #										**PC #**				
Surname										**Initials**				
ID Number														

Before final submission, read through and tick in the last column.		
1.	The full and final solution that I intend to submit has been uploaded to the correct locations as specified by the invigilators.	
2.	A zip file containing the full and final solution listed in point 1 has been uploaded to Eve.	
3.	I have **personally confirmed** the version of the full and final solution that has been saved to the backup media is the **correct copy** of the solution in point 1.	
Signature		

Official use only					
EVE		**CD**		**USB**	
Assistant signature			**Assistant initials**		

Official use only					
Marker signature			**Marker initials**		
Test total	100	**Mark**		**Moderation**	

See mark sheet on next page.

NB

Submissions which **do not compile** will be capped at 40%

Execution marks are awarded for a correctly functioning application and not for having some related code.

Failure to save the solution to the correct locations will mean that the Academy will not be able to mark the submission and you will forfeit marks as a result.

Mark sheet

1. **`POSTITServer`**
   (a) Create **`ServerSocket`**. ____ out of [02]
   (b) Accept client and pass to **`POSTITHandler`**. ____ out of [03]
   (c) Multi-threaded client handling. ____ out of [05]
2. **`POSTITHandler`**
   (a) Handle `REG`. ____ out of [05]
   (b) Handle `LOGIN`. ____ out of [05]
   (c) Handle `POST`.
      i. Process parameters. ____ out of [03]
      ii. Save `<Message>`. ____ out of [05]
      iii. Save `<Image>`. ____ out of [05]
   (d) Handle `LOGOFF`. ____ out of [02]
3. **`POSTITClient`**
   (a) Connect to **`POSTITServer`**. ____ out of [02]
   (b) Setup streams. ____ out of [02]
   (c) Send text commands. ____ out of [02]
   (d) Send image data. ____ out of [02]
   (e) Process responses. ____ out of [02]
4. **`POSTITClientFrame`**
   (a) GUI Layout. ____ out of [02]
   (b) `REG` button and listener. ____ out of [02]
   (c) `LOGIN` button and listener. ____ out of [02]
   (d) `POST` button and listener. ____ out of [02]
   (e) `LOGOFF` button and listener. ____ out of [02]
5. Coding convention (structure, layout, OO design) ____ out of [05]
6. Commenting (normal and JavaDoc commenting). ____ out of [05]
7. Correct execution
   (a) Show **`POSTITClientFrame`**. ____ out of [05]
   (b) Sending requests and responses. ____ out of [10]
   (c) Successfully sending and saving images. ____ out of [10]
   (d) Handle exceptions. ____ out of [05]
   (e) Connection cleanup. ____ out of [05]