Assessment (non-exam) Brief

UCL
SCHOOL OF MANAGEMENT

Module code/name	INST0004 Programming 2
Module leader name	Dr Daniel Onah
Academic year	2023/24
Term	1
Assessment title	Tutorial Sheets 1, 2,3,4,6,8
Individual/group assessment	Individual

Submission deadlines: Students should submit all work by the published deadline date and time. Students experiencing sudden or unexpected events beyond your control which impact your ability to complete assessed work by the set deadlines may request mitigation via the extenuating circumstances procedure. Students with disabilities or ongoing, long-term conditions should explore a Summary of Reasonable Adjustments.

Return and status of marked assessments: Students should expect to receive feedback within one calendar month of the submission deadline, as per UCL guidelines. The module team will update you if there are delays through unforeseen circumstances (e.g. ill health). All results when first published are provisional until confirmed by the Examination Board.

Academic Misconduct: Academic Misconduct is defined as any action or attempted action that may result in a student obtaining an unfair academic advantage. **Academic misconduct includes plagiarism, obtaining help from/sharing work with others be they individuals and/or organisations or any other form of cheating.** Refer to Academic Manual Chapter 6, Section 9: Student Academic Misconduct Procedure - 9.2 Definitions.

Referencing: You must reference and provide full citation for ALL sources used, including articles, textbooks, lecture slides and module materials. This includes any direct quotes and paraphrased text. If in doubt, reference it. If you need further guidance on referencing please see UCL's referencing tutorial for students. Failure to cite references correctly may result in your work being referred to the Academic Misconduct Panel.

Use of Artificial Intelligence (AI) Tools in your Assessment: Your module leader will explain to you if and how AI tools can be used to support your assessment. In some assessments, the use of generative AI is **not permitted** at all. In others, AI may be used in an **assistive** role which means students are permitted to use AI tools to support the development of specific skills required for the assessment as specified by the module leader. In others, the use of AI tools may be an **integral** component of the assessment; in these cases the assessment will provide an opportunity to demonstrate effective and responsible use of AI. See page 3 of this brief to check which category use of AI falls into for this assessment. Students should refer to the UCL guidance on acknowledging use of AI and referencing AI. Failure to correctly reference use of AI in assessments may result in students being reported via the Academic Misconduct procedure. Refer to the section of the UCL Assessment success guide on Engaging with AI in your education and assessment.

:

Content of this assessment brief

Section	Content
A	Core information
B	Coursework brief and requirements
C	Module learning outcomes covered in this assessment
D	Groupwork instructions (if applicable)
E	How your work is assessed
F	Additional information

Section A Core information

Submission date	On the module page
Submission time	On the module page
Assessment is marked out of:	100
% weighting of this assessment within total module mark	100%
Maximum word count/page length/duration	NA
Footnotes, appendices, tables, figures, diagrams, charts included in/excluded from word count/page length?	NA
Bibliographies, reference lists included in/excluded from word count/page length?	NA
Penalty for exceeding word count/page length	No specified maximum (and thus no penalty for exceeding)
Penalty for late submission	Standard UCL penalties apply. Students should refer to Refer to https://www.ucl.ac.uk/academic-manual/chapters/chapter-4assessment-framework-taught-programmes/section-3-moduleassessment#3.12
Submitting your assessment	Weekly at the scheduled deadline/time

Anonymity of identity. Normally, <u>all</u> submissions are anonymous unless the nature of the submission is such that anonymity is not appropriate, illustratively as in presentations or where minutes of group meetings are required as part of a group work submission	The nature of this assessment is such that anonymity is not required.

Section B Assessment Brief and Requirements

On the Tutorial sheets.

C: Section Module Learning Outcomes covered in this Assessment

This assessment contributes towards the achievement of the following stated module Learning Outcomes as highlighted below:

Weekly lectures for INST0004 Programming 2

Section D

: Groupwork Instructions (where relevant/appropriate)

NA

Section E

: How your work is assessed

Within each section of this assessment you may be assessed on the following aspects, as applicable and appropriate to this assessment, and should thus consider these aspects when fulfilling the requirements of each section:

- The accuracy of any calculations required.
- The strengths and quality of your overall analysis and evaluation;
- Appropriate use of relevant theoretical models, concepts and frameworks;
- The rationale and evidence that you provide in support of your arguments;
- The credibility and viability of the evidenced conclusions/recommendations/plans of action you put forward;
- Structure and coherence of your considerations and reports;
- Appropriate and relevant use of, as and where relevant and appropriate, real world examples, academic materials and referenced sources. Any references should use either the Harvard OR Vancouver referencing system (see References, Citations and Avoiding Plagiarism)
- Academic judgement regarding the blend of scope, thrust and communication of ideas, contentions, evidence, knowledge, arguments, conclusions.
- Each assessment requirement(s) has allocated marks/weightings.

Student submissions are reviewed/scrutinised by an internal assessor and are available to an External Examiner for further review/scrutiny before consideration by the relevant Examination Board.
It is not uncommon for some students to feel that their submissions deserve higher marks (irrespective of whether they actually deserve higher marks). To help you assess the relative strengths and weaknesses of your submission please refer to UCL Assessment Criteria Guidelines, located at https://www.ucl.ac.uk/teaching-learning/sites/teaching-learning/files/migratedfiles/UCL_Assessment_Criteria_Guide.pdf

The above is an important link as it specifies the criteria for attaining 85% +, 70% to 84%, 60% to 69%, 50% to 59%, 40% to 49%, below 40%.

You are strongly advised to **not** compare your mark with marks of other submissions from your student colleagues. Each submission has its own range of characteristics which differ from others in terms of breadth, scope, depth, insights, and subtleties and nuances. On the surface one submission may appear to be similar to another but invariably, digging beneath the surface reveals a range of differing characteristics.

Students who wish to request a review of a decision made by the Board of Examiners should refer to the UCL Academic Appeals Procedure, taking note of the acceptable grounds for such appeals.

Section F

Note that the purpose of this procedure is not to dispute academic judgement – it is to ensure correct application of UCL's regulations and procedures. The appeals process is evidence-based and circumstances must be supported by independent evidence.

Section G

: Additional information from module leader (as appropriate)

Work in accordance with the weekly assessment brief and the tutorial sheet questions herein.

Section H

INST0004: Programming 2
Tutorial Sheet 08
Week 08

Academic Year: 2023/24

Set by: Daniel Onah

The aim of the practical sheet is to provide you with a set of exercises to practice developing your programming skills. Writing programmes will help you develop programming skills and prepare for the Programming Test and the Moodle Quiz.

Assessed Tutorial Questions:

Some questions in the labs are marked with a [*] symbol. These questions are compulsory and you will be assessed on one or more of these in the following week's lab. For most of you, this means you will be assessed on:

- Tutorial sheet 1 in Lab 2
- Tutorial sheet 2 in Lab 3
- Tutorial sheet 3 in Lab 4
- Tutorial sheet 4 in Lab 5
- Tutorial sheet 6 in Lab 7, and
- Tutorial sheet 8 in Lab 9

Please look at all the questions spend some time thinking carefully about them, before asking for help. If you are still stuck:

- attend the live practical & booster sessions and ask me, or the TAs, for help
- or post a question on the Moodle discussion board.

Once you have completed the assessed questions you should upload the specified files to the Assessment tab on Moodle under the appropriate submission link. You can submit at the specified submission time on Moodle.

Published September 3rd 2022

Section I

In preparation for your 1-to-1 viva, you should be able to clearly indicate which constructs such as classes, functions, variables used in your program is of your own work and which classes or features where from other sources (e.g., from textbooks, online sources etc.).

Please refer to Read about scheduling and live sessions and Read about tutorial based assessment, submissions and vivas for additional information on the procedures of being assessed.

A reminder on how to compile a programme

This is just a quick reminder of how to compile a simple Python program.

1) Make sure you organise your files into folders for each of the weeks:
   a. You should keep your files on the `N:` drive. I suggest creating a folder called `INST0002` with subfolders `week01, week02` etc. for each of the tutorial files.
2) You should open the Command Prompt Shell and navigate to the current directory where your Python files are located, for example, `N:/INST0002/week01`. To do this, use the cd command. For example:

   ```
   cd  N:/INST0002/week01
   ```

   To change the drive from C: to N: simply type in `N:` in the command prompt shell.

   Speak to a TA or a peer student if you need help with using the N drive or the Command Prompt shell.

3) Once in the working directory, you can execute/run the files without providing the path to the file:

   ```
   python hello_world.py
   ```

   (where python is the name of the interpreter that will trigger the Runtime Environment and execute specific file)

Section J

Exercise 1[*] Tuberculosis Treatment Regimen

Learning Objectives: [A] input() function [B] method [C] class [D] object [E] recursive method [F] return statement [G] inheritance [H] loops and conditional statements.

MEDICAL CASE STUDY: Algorithm and Pseudocode

This case study example is about a patient undergoing treatment for Tuberculosis (TB). In order for the doctors to decide the appropriate treatment regimen and drugs for treating the patient, they have to perform a few clinical trials. The first treatment regimen was to provide the patient with three different drugs to be taken at different time intervals (for a week). The medical team would need to observe the side effects from the drugs. The common symptoms of liver inflammation associated with this treatment regimen are: 1) Nausea or vomiting, 2) Abdominal pain. Some of the drugs regimen could be associated with the liver inflammation issue. In this task, you are required to prescribe drugs that are safe to the patients that would not cause any liver inflammation issues.

At the initial stage of administering the treatment regimen, the patient is given four common drugs that are used for treating TB.

1. Rifinah
2. Pyridoxine
3. Pyrazinamide
4. Ethambutol

It is well known that **Pyrazinamide** is associated with liver inflammation. This drug produces high Alanine Transaminase (ALT), which is an enzyme found in the liver. This is then release into the blood stream in excess and makes the liver to work or operate very hard to control the flow of the enzymes.

Using a menu option, present the common symptoms to be selected by the patients. Common symptoms from the treatment regimen as a result of the four drugs would be:

- Itching and skin rashes
- Feeling nauseous
- stomach-ache
- Nausea or vomiting >> *symptom of liver inflammation*
- Abdominal pain >> *symptom of liver inflammation*

Your task is to write an algorithmic program using two classes:

1) **DrugRegimen** for prescribing the drug regimen and

2) **TuberculosisTreatment** that would take in these symptoms as input from the menu options. If patient's symptoms are in these categories, provide the associated treatment regimen.

Section K

DrugRegimen class

The **DrugRegimen** class should have a method called *drugRegimen()* that takes two parameter lists **self** and a list of **menu** items. Define a conditional statement to check whether the selected symptom options are associated to liver inflammation.

TuberculosisTreatment class

The **TuberculosisTreatment** class should inherit the method and features from the **DrugRegimen** class (think carefully how to do this … see lecture 8 for a guide). Within the class declare an empty list called '**menu**'.

Define the following methods within the class:

a) menuOption() method: This method should take only the **self** parameter and continuously display the following menu options:

   i. Itching and skin rashes
   ii. Feeling nauseous
   iii. Stomach-ache
   iv. Nausea or vomiting
   v. Abdominal pain
   vi. Exit

   The user should be able to select as many symptoms as possible. If a symptom is selected display that symptom (see the sample output below). Use conditional statement constructors for this to check and display the selected symptoms. Otherwise, display "**Symptom NOT in the main menu option**" and called the **displayOptions()** method.

   Outside the loop construct, call the **menuList()** method and **drugRegimen()** method from the **DrugRegimen** class appropriately.

b) **menuList():** This method should display the selected menu options stored in the menu list.
c) **displayOptions()** method: This is a recursive method that takes only the **self** parameter. The method should display the following message if an invalid option is entered " ***Invalid option! Would you like to see the menu options again? (Y/N)***". If the answer is 'Y', display the menu options by invoking the ***menuOption()*** method accordingly. If the answer is 'N', display "***Goodbye and Good Luck!!"***. You can either exit the program or display the menu list item. But if, otherwise, call the ***displayOptions()*** method recursively.

Create Instance of the class

Create an instance of the **TuberculosisTreatment** class and invoke the ***menuOption()*** method.

Section L

Pseudocode:

- If signs are:
    a. itching, rash and stomach-ache,
    b. then recommend the following complete treatment regimen: **Rifinah**, **Pyridoxine**, **Ethambutol**, and **Pyrazinamide** for 6 months.
- If signs are:
    a. Nausea or vomiting, Abdominal pain,
    b. then recommend the following treatment regimen: **Rifinah**, **Pyridoxine**, and **Ethambutol** for 9 months.

Hint: You are required to write this program using you knowledge of classes, objects, and inheritance (see lecture 8). **Note** that this TB treatment regimen is for specific individual patient treatment on a case-by-case basis.

The output of the programme should be:

```
danny$ python3 tuberculosis_drug_regimen_ex01.py
*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: 1
Symptom: Itching and skin rashes


*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
```

Section M

```
Please select a symptom: 2
Symptom: Feeling nauseous

*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: 3
Symptom: Stomach-ache

*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: 6
You are exiting. Thank you!
The selected menu(s) are/is:['1', '2', '3']
Prescribed Drug Regimen (6 Months):
1. Rifinah
2. Pyridoxine
3. Pyrazinamide
4. Ethambutol
```

Section N

```
danny$ python3 tuberculosis_drug_regimen_ex01.py
*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: 3
Symptom: Stomach-ache

*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: 4
Symptom: Abdominal pain

*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
```

Section O

```
Please select a symptom: 6
You are exiting. Thank you!
The selected menu are:  ['3', '4']
Prescribed Drug regimen (9 Months):
1. Rifinah
2. Pyridoxine
3. Ethambutol

danny$ python3 tuberculosis_drug_regimen_ex01.py
*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: Blank input
Symptom NOT in main menu option!

Invalid option! Would you like to see the main menu again? (Y/N): y
*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: k
```

```
Symptom NOT in main menu option!
Invalid option! Would you like to see the main menu again? (Y/N): y


*** Please select an option ***
1: Itching and skin rashes
2: Feeling nauseous
3: Stomach-ache
4: Nausea or vomiting
5: Abdominal pain
6: Exit
Please select a symptom: 9
Symptom NOT in main menu option!
Invalid option! Would you like to see the main menu again? (Y/N): n
Goodbye!
The selected menu(s) are/is: []
```

Exercise 2 [*] Exclusive Members Club

Learning Objectives: [A] use a for & while loop; [B] use of Python random; [C] Primitive types and relational operations [D] conditional statements – if, elif, else, break & continue [E] functions.

Case Study:

This case study is based on the policy of an exclusive members club. Exclusive club are no fun space unless you are a registered member of the club. The club is exclusive to only a selected and elite few. The membership is unique to specific group and access to the club exclusive network can only be given to this group. The club has a policy of introducing new members to the club by providing three guests with free access to the exclusive club space daily. This practice allows the club executives to indoctrinate new members into the organisation and the club activities.

Section Q

Similar to every real-life organisation, each registered member has to enter a **username** and a **password**. Members of the exclusive club has to enter their username and password, otherwise, the member won't be able to login and access the club page. With a successful login, each member is personally greeted. Each member is randomly allocated a security level as the login token to access the club page and network system. The guests are given the lowest level of security from 1 to 3. The guest's security levels are also randomly generated and allocated to each guest for the day. The idea and reason for the security level is to provide limited or specific resource accessibility to the guest users.

Description

The program should be developed using two classes **Members** and **SecurityPass**.

Members Class

The members class should inherit the **SecurityPass** class. The class should display a class document describing the program. The document string should be "Exclusive *Club Network 5* Registered Club Members Only*." (see how this is displayed in the output section). Define a constructor that initialises two attributes ***username*** and ***password***. The constructor should also invoke the **checkLogin()** method of the **SecurityPass** class with the appropriate arguments.

Prompt users to enter their **username** and **password** and stored these using appropriate attribute names. Create a class instance with these attributes passed-in as arguments.

SecurityPass Class

This is the helper class that would conduct the login and security accessibility permission. This class would allow the checking of login members to ascertain whether they are registered and are members of the exclusive network club. In this class import a random module for generating the security level numbers to be allocated to both the guests and club members at login stage.

Create an empty dictionary or set to store the details of new members to the club.

checkLogin () Method

Create a authentication method called **checkLogin ()** that checks whether the username and password is empty. If this is the case should display a message "Username *and Password cannot be empty*." and prompt the user whether they would like to continue "*Do you wish to try again (yes/no)*". If response is yes, provide the user with the login details again. Otherwise, display "*Goodbye! Do visit our club again!!*" and exit the program.

The method should be able to identify whether the user is a member of the exclusive club from their login information. If the user is not a member, provide them with the opportunity to register "*Do you want to register? Enter (Yes or No)*". If the response is **yes**, then add the login details to the ***empty dictionary*** and this username can be used throughout the program execution with new **security level** each time they login (see the output section for a guide). Then the membership database should be updated automatically after each successful registration. Otherwise, display "*Goodbye! Please do visit our club again!!*" and exit the program.

Section R

Club Members

If the login user is a member of the club, they are allocated randomly generated security clearance level between 4 – 200. The security clearance level for members would allow them to access all the resources of the club.

Guest Users

The program should be able to identify whether a user is a guest member. Ideally, the club rule is to provide some login access to only three guest members. If the login user is identified as a **guest** or **Guest**, then they are registered and provided their allocated randomly generated security clearance level (between 1 – 3).

Important Security Information: Whenever the registered members or guests login, their security clearance level should change (i.e. new randomly generated security number should be produced and allocated to the user).

Hint: The program should continuously request for user login details. Define a sentinel control that would stop the loop and allow the program to terminate when the string word **finish** is entered in the username or a string **0** is entered in the password. This control can be added in any of the classes.

The output of the programme may look as follows:

```
danny$ python3 Members_ex02.py
Exclusive Club Network 5*
        Club Members Only


To exit the program enter (finish or 0)
Enter Username: Danny
Enter Password: Welcome1
Login failed. You're not a member of the exclusive club.


Do you want to register (Yes or No): yes
The user database is now updated.
Welcome, Danny, you are now a member of the exclusive club!
Your security number is:  5
```

Section S

To exit the program enter (finish or 0)

Enter username: Danny

Enter password: Welcome1

Welcome, Danny, you are now a member of the exclusive club!

Your security number is: 30

To exit the program enter (finish or 0)

Enter Username: Guest

Enter Password: guest1

Login failed. You're not a member of the exclusive club.

Do you want to register (Yes or No): Y

The user database is now updated.

Welcome, Guest, you are now a guest member of the exclusive club!

Your guest security number is: 1

To exit the program enter (finish or 0)

Enter Username: Mike

Enter Password: hello54

Login failed. You're not a member of the exclusive club.

Do you want to register (Yes or No): No

Section T

To exit the program enter (finish or 0)
Enter Username: Mike
Enter Password: hello54
Login failed. You're not a member of the exclusive club.

Do you want to register (Yes or No): y
The user database is now updated.
Welcome, Mike, you are now a member of the exclusive club!
Your security number is: 179

To exit the program enter (finish or 0)
Enter username: Danny
Enter password: 0
You are exiting the program!
Press the enter key to exit the program.
Thank you for visiting!

danny$ python3 Members_ex02.py
Exclusive Club Network 5*
        Registered Club Members Only.
To exit the program enter (finish or 0)
Enter username: finish
You are exiting the program!
Press the enter key to exit the program.
Thank you for visiting!

Section U

Exercise 3[*] Card Game

Learning Objectives: [A] import libraries and modules, print() function [B] concept of inheritance [C] constructor [D] method [E] return statement [F] lists [G] classes and objects [H] loops and conditional statement.

This problem set is similar to the game of cards program. In this task, randomly generated card **values** and **suits** would be drawn automatically. This card program should be developed by defining classes representing a **card**, a **deck,** and a **dealer**.

Card class

Create a card class to model the playing of cards. The class should define a constructor that takes two main parament lists **value** and **suit** in addition to the **self** parameter. The constructor should initialise these attributes in its body. The class should define another string method (def __str__) that returns the **value** and the **suit** attributes.

Deck Class

The Deck class inherits the features of the Card class. This class defines a constructor that initialises an empty list of cards. Declare a construct using a loop to iterate over a list of numbers in the range of 14 (i.e., the range ideally should be from 1 – 13 inclusive). This loop should also include the following list of card types: ['Jack', 'Queen', 'King', 'Ace']. Randomly generate the value from the loop list and append this to the empty list. You would need to ensure that you populate the empty list with items before the game or program would work. Now create a ***nested loop*** in the length of 4. Use a conditional statement to assign the loop variable with the following cards: "Hearts", "Clubs", "Diamond", "Spades". After each execution, ensure the program **shuffle** the deck of cards.

Shuffle method

Define a ***shuffle()*** method within the Deck class that takes only a **self** parameter list. This method should return the shuffled cards randomly.

Deal method

Define a ***deal()*** method within the Deck class. This method should check whether the deck is empty. If the deck is not empty, it should return the card. Otherwise, this should display a message that "*Deck is empty*.", and return a **none** value.

Hint: *Remember to import the random module*.

Dealer Class

The Dealer class should inherit the features of the Deck class. This class should import the Deck and Card class modules. It should initialise a **deck** attribute within its constructor and call the constructor of the Deck class accordingly (see lecture 8 for a guide).

Section V

getDealCard method

Define a method called ***getDealCard()*** within the Dealer class. This method should return the **deal** card from the Deck class (i.e., invoke the **deal()** method created within the Deck class).

isEmpty method

Define a method isEmpty() within the Dealer class, that checks whether the list of cards is empty or whether the length of the card is equal to zero. If this is the case, return the value, otherwise return the cards.

Instances of classes

First create an instance of the Deck class and then create an instance of the Dealer class then pass-in as an argument the instance of the Deck class into the Dealer class (*i.e., ideally, create a dealer instance with the deck object*). The next step is to **deal** a card from the deck using the instances of the Dealer class.

Finally, check whether the dealt from deck is successful. If this is True display the result.

You program should display the following statement:

```
danny$ python3 Dealer_ex03.py
Deck is dealt from.
The dealer dealt: Ace of Clubs

danny$ python3 Dealer_ex03.py
Deck is dealt from.
The dealer dealt: 6 of Hearts

danny$ python3 Dealer_ex03.py
Deck is dealt from.
The dealer dealt: 10 of Clubs

danny$ python3 Dealer_ex03.py
Deck is dealt from.
The dealer dealt: Jack of Hearts
```

Section W

Exercise 4 [*] Character Counting using inheritance.

Learning Objectives: [A] class; [B] object; [C] method; [D] for and while loop [E] conditional statement [F] import classes and inheritance [G] return statement.

This exercise is similar to the exercise 1 character counting in Tutorial sheet 4. In this case, you would need to write the program using the knowledge of inheritance covered in lecture 8. The program should only allow access to the attributes via the getter and setter method. You should create two classes "**CharCounter**" and "**CharCounterTest**" for this program.

CharCounter class

The class should declare a global private attribute called **key**. Define a special method called **observe()** that takes a parameter called 'observe'. The method should check and count the number of characters in the sentence using the **key** attribute. The condition to be aware of is that: If the **key** attribute is equal to the **observe** parameter, then the method should increase the ***count*** by one and return the value of the count. Create a getter method **getCount(),** when invoked should return the information in relation to the count (i.e. return the count value).

CharCounterTest class

The **CharCounterTest** class utilise the **CharCounter** features by inheriting the class and its methods. The CharCounterTest class should define a special method that takes two parameter ***dotCounter*** and ***starCounter*** including the ***self*** parameter. The attributes should be initialised within the body of the special method. Define a *display()* method that takes the same two parameter lists of the special method (constructor). The display method should continuously request the user to enter the sentence with special characters 'dot' and 'star'. Then count the number of ***dots*** and ***stars*** that appears within the given sentence. Use two for-loop constructs within this display() method one for each of *dot count* and *star count* (as you did in tutorial sheet 4 exercise 1).

For-loop construct

Declare another for-loop using the range of the length of the sentence string. Store the iterated characters into a variable called "**currentChar**". Create a count variable and assign this to *currentChar* and the count of the *dot* counter attribute (i.e., count = currentChar.count(_dotCounter)). Declare an attribute *observe* and assign this to the *CharCounter* class and the **observe()** method taking as argument the **self** and **currentChar**. Lastly, declare another variable called ***count*** and assign the *getCount()* method of the **CharCounter** class. The method should have one argument called **count** and including the **self** keyword.

Class Object

Initialise a *star* counter and *dot* counter with their associated characters. Create the class object by passing the two attributes (**_starCounter** and **_dotCounter)** as arguments. Invoke the *display()* method with the class object with the same number of attribute passed-in as argument.

Section X

Hint: The program should stop requesting user input when a sentinel control value e.g., '-1' is entered.

You program should display the following statement:

```
danny$ python CharCounterTest_ex04.py
Input a line of characters: I love INST0004. It is a Programming *** module.
The . appears 2 time(s)
The * appears 3 time(s)
Input a line of characters: -1
Goodbye!
```

Exercise 5 [*] Bank Transaction

Learning Objectives: [A] Classes; [B] Objects; [C] method; [D] return statements; [E] accessor & mutator methods; [F] inheritance [G] data abstraction & encapsulation.

Imagine everyday banking transaction where customers deposit and withdraw money from the bank account. Common daily bank transaction would be withdrawing cash and depositing cash. A customer might want to perform an overdraft transaction where the bank would charge them fixed interest rate on the overdraft or loan.

In order to develop this program successfully, you would be required to create base and derived classes. In this task, you are required to create the following classes: **BankAccount**, **Deposit**, **Withdraw** and **Transaction**.

BankAccount class

The **BankAccount** class should include a constructor which initialises an account number attribute and initial balance attribute. The Bank account program should enforce information hiding (see: encapsulation and data abstraction). In order to update and gain access to the attribute information, you would need to create *setter()* and *getter()* methods that would be used for accessing the values associated to the attributes.

Deposit class

In the Deposit class, create a constructor that takes three parameter lists *accountNumber, balance, deposit* and including the *self. The class should inherit the base class i.e.,* **BankAccount** class. Use the super constructor of the BankAccount class within the constructor of the Deposit class (see lecture 8 for a guide). Create a method called **myDeposit()** that would take as parameter list *amount* and a *self*. The method should check whether the *amount* is greater than zero, if this is the case, then add the deposit amount to the initial bank account balance.

Section Y

Withdraw class

The Withdraw class should inherit the **Deposit class** and all its features. The class should have a constructor with the following parameter lists *accountNumber*, *balance*, *amount* and including the *self* parameter. Also within the body of the constructor include the *super constructor* from the ***base*** class (see lecture 8 for a guide). Initialise the attributes of the constructor of the Withdraw class accordingly. Create a method called **myWithdraw(),** this method should be able to take a single parameter list called *amount* including the *self* parameter. Define a condition to check *whether the withdraw amount is more than the balance. If the amount is more than the balance, then the program should offer the customer an opportunity for an overdraft*. Otherwise, subtract the amount from the Bank account initial balance.

Transaction class

The Transaction class should inherit the ***Withdraw*** class. The constructor in transaction class should take three parameter lists *accountNumber*, *balance*, *withdraw* and including the *self* parameter. Within the body of the constructor call the constructor of the base ***BankAccount*** class. The class should initialise the attributes accordingly. Create a *setter()* and *getter()* methods to **set** and **get** the balance. If you wish you can add additional method called *getTransactionAccount()* that returns the customer account number.

BankAccountTest class

The **BankAccountTes**t class is the testing class for the **Bank account** program. The class should import the **BankAcount**, **Deposit**, **Withdraw** and **Transaction** classes. The class should request user for the following input:

- account number
- current account balance
- cash deposit
- cash withdraw

Create the instances of the **BankAccount** , **Deposit**, **Transaction** classes with the appropriate input.

- The instance of the bank account should pass-in as arguments the *account number* and the *balance*.
- The instance of the deposit class should pass-in as arguments the *account number*, *balance,* and *deposit*.
- The instance of the transaction class should pass-in as arguments the *account number*, *balance* and *withdraw*.

Section Z

Overdraft transaction

The bank allows only a single overdraft transaction monthly. When a customer request for an overdraft transaction as a result of limited funds, the program should prompt the customer whether they want to continue the transaction. If the response is **yes**, then a message is display for the bank percent interest rate to be entered such as "*Enter bank percent interest rate/charges*". The interest rate or charges are then added to the overdraft amount. *The computed amount after the interest rate is calculated, should be added to the overdraft value.* Otherwise, if the response is **no**, then display a message such as "*Thank you for banking with us! See you again!!*"

Display the result for the current balance after overdraft if the customer request for loan, otherwise, display the current balance after a transaction is completed without overdraft.

Additional logic: Let's say the customer is allowed only one overdraft transaction per month. Think of how you could write the program to enforce this condition. That is, it should not allow any other overdraft transaction for that month if the customer already requested and completed one.

You program should display the following statement:

```
danny$ python3 BankAccountTest_ex05.py
Enter account number: 98567489
Enter account balance: 1500
Enter deposit: 500
Enter amount to withdraw: 250
Customer with account number:  98567489 has an initial bank account balance
of: £1500.0
Total balance after deposit: £2000.0
Current balance after transaction: £1750.0


danny$ python3 BankAccountTest_ex05.py
Enter account number: 75009374
Enter account balance: 250
Enter deposit: 8500
Enter amount to withdraw: 25000
```

Section AA

```
Customer with account number:  7500 has an initial bank account balance of: £250.0

Total balance after deposit: £8750.0

Withdraw amount more than balance.

Do you need overdraft? enter (yes/no): y

Current balance after transaction: £-16250.0

Enter bank percent interest rate/charges: 39.90

Current balance after '39.9%' overdraft charges: £-6483.75

Bank monthly overdraft charges plus interest: £22733.75


danny$ python3 BankAccountTest_ex05.py

Enter account number: 89746582

Enter account balance: 2390

Enter deposit: 271

Enter amount to withdraw: 3515

Customer with account number:  89746582 has an initial bank account balance of: £2390.0

Total balance after deposit: £2661.0

Withdraw amount more than balance.

Do you need overdraft? enter (yes/no): n

Thank you for banking with us! See you again!!
```