Statement and Confirmation of Own Work

A signed copy of this form must be submitted with every assignment.
If the statement is missing your work may not be marked.

Student Declaration

I confirm the following details:

Candidate Name:	SHIHAB SAJID MIRZA
Candidate ID Number:	202100255
Qualification:	NCC LEVEL 4 DIPLOMA IN COMPUTING (L4DC)
Unit:	DDOOCP (With C#)
Centre:	ZCAS UNIVERSITY, LUSAKA ZAMBIA.

I have read and understood both NCC Education's *Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.

I confirm that this is my own work and that I have not colluded or plagiarised any part of it.

Candidate Signature:	
Date:	26/10/2021

CONTENTS-

CODEBOX SURVEY SYSTEM.

Class Diagram Design Justification.

User classes:

To start off, a users class is made with two other classes that inherit from it, namely: Administrators and Consumers.

Now the users class itself has 3 properties

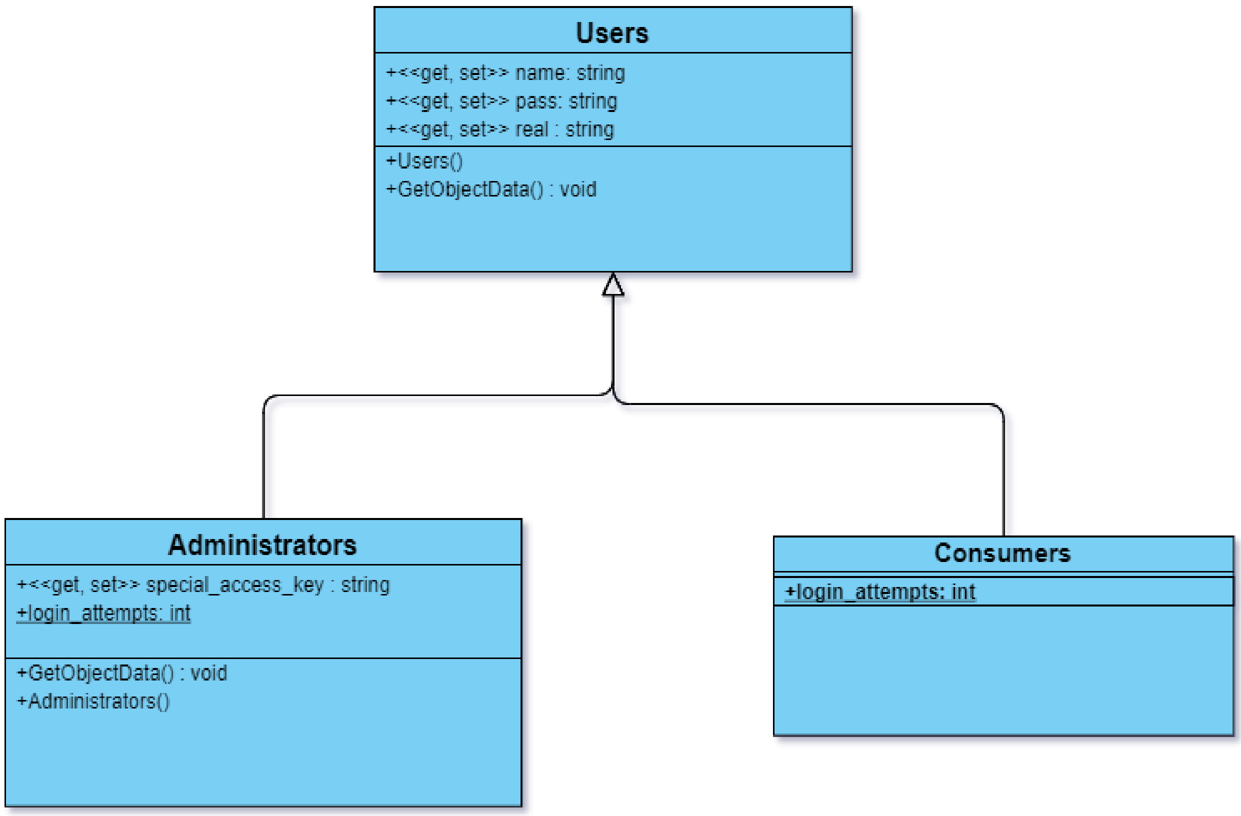
- Username
- Real name
- And password

The admin class has an extra property called “special access key” to add more security when admins try to log in.

The consumers and admin class also has login attempts as a static field which decreases as a user fails to enter the correct credentials and a cool down is started as soon as a user runs out of login attempts. An admin has 3 login attempts and a consumer has 6 login attempts.

It should be noted that all these users also get serialized and deserialized often to a file called “system_user.dat”. this file is extremely crucial to the system, as it stores all of the user objects.

This is done by using a polymorphic list, this is the list that gets serialized and deserialized. Below is a class diagram about the users class and the two other classes that inherit.



Survey/Question Answer/Survey answer classes:

To make surveys, we will need a question class that will hold two main properties that will be constructed:

- Question type (uses an enum to specify if question is yes no, Likert or scale).
- Question (the question itself).

There will also be 5 other properties that will hold the options for Likert type questions, but these won't be used in the constructor since not all questions have to be Likert type, but we can set those 5 options if the questions are Likert type. There will also be a static field here to keep track of question count.

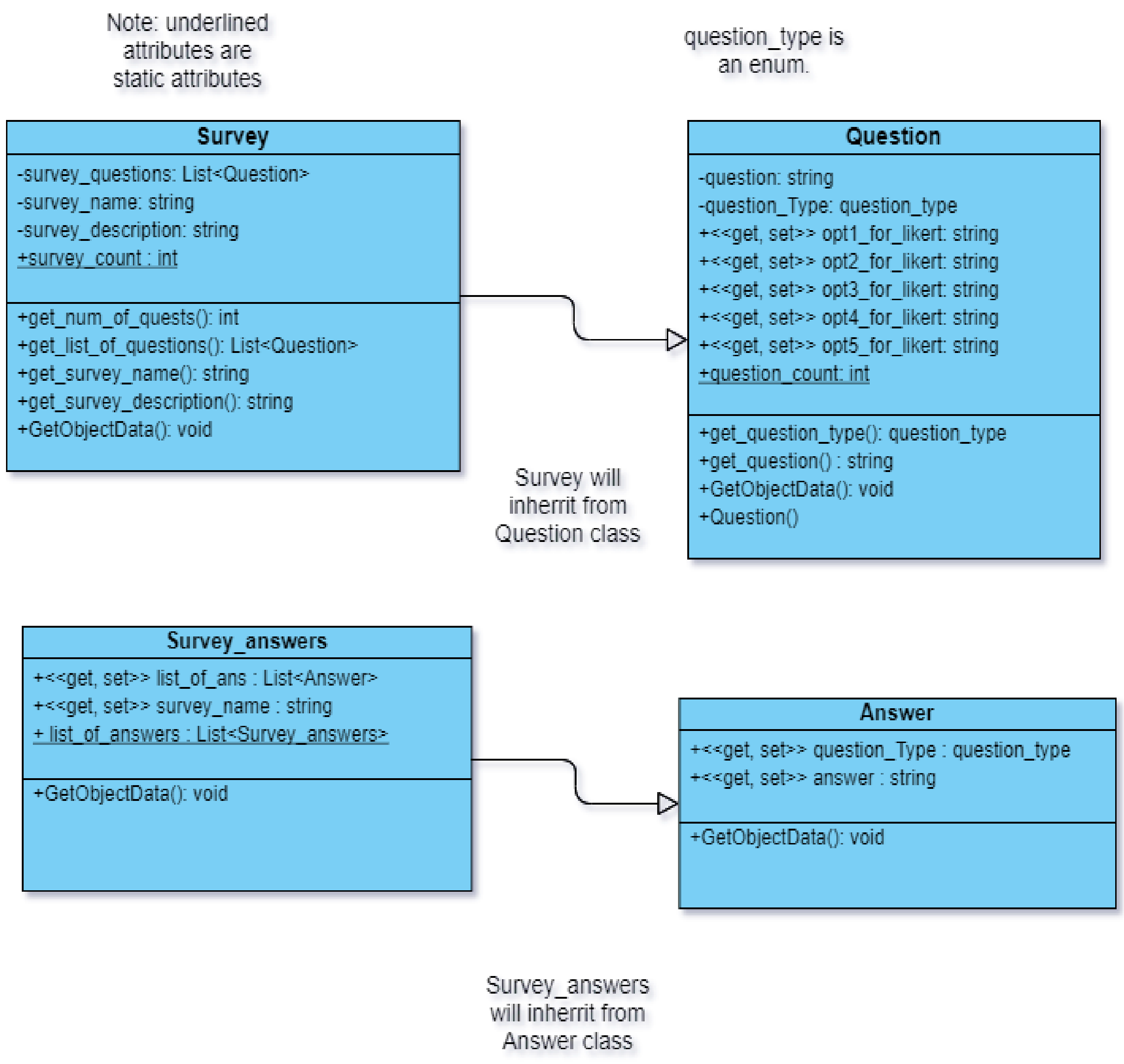
Lets now move on to the survey class. This class will have a property list of questions<Question> that will hold the specific questions and also a survey name property and a survey description property.

Once we create a survey object, we can pass it into the survey list (which is polymorphic) and then serialize it and deserialize it to a file called “Surveys.dat” when needed.

For the answer class, we will have a question type property (which uses the same enum used in the question class) , as well as an answer property which will hold the answer itself.

We will use the same technique as above and store a list of answers with a survey name in the “Survey answer” class and these Survey answer objects will have to constructed with a list of answers and a survey name (which will be the same name as the survey name used in the survey objects) too will be stored in a polymorphic list which will be serialized and deserialized when we need to. The survey and question classes will also have some useful methods that will be used to get essential info about their respective objects, methods such as get question type(), get survey name(), get list of questions() etc.

Below is a class diagram for all the classes discussed in this section.


Note: underlined attributes are static attributes
question_type is an enum.
Survey
-survey_questions: List<Question>
-survey_name: string
-survey_description: string
+survey_count : int
+get_num_of_quests(): int
+get_list_of_questions(): List<Question>
+get_survey_name(): string
+get_survey_description(): string
+GetObjectData(): void
Question
-question: string
-question_Type: question_type
+<<get, set>> opt1_for_likert: string
+<<get, set>> opt2_for_likert: string
+<<get, set>> opt3_for_likert: string
+<<get, set>> opt4_for_likert: string
+<<get, set>> opt5_for_likert: string
+question_count: int
+get_question_type(): question_type
+get_question() : string
+GetObjectData(): void
+Question()
Survey will inherrit from Question class
Survey_answers
+<<get, set>> list_of_ans : List<Answer>
+<<get, set>> survey_name : string
+ list_of_answers : List<Survey_answers>
+GetObjectData(): void
Answer
+<<get, set>> question_Type : question_type
+<<get, set>> answer : string
+GetObjectData(): void
Survey_answers will inherrit from Answer class

References:

- https://www.tutorialspoint.com/uml/uml_class_diagram.htm

Accessed on: [16th October 2021].

- https://creately.com/blog/diagrams/class-diagram-relationships/

Accessed on: [16th October 2021].

This system is powered by .NET 4.7.2