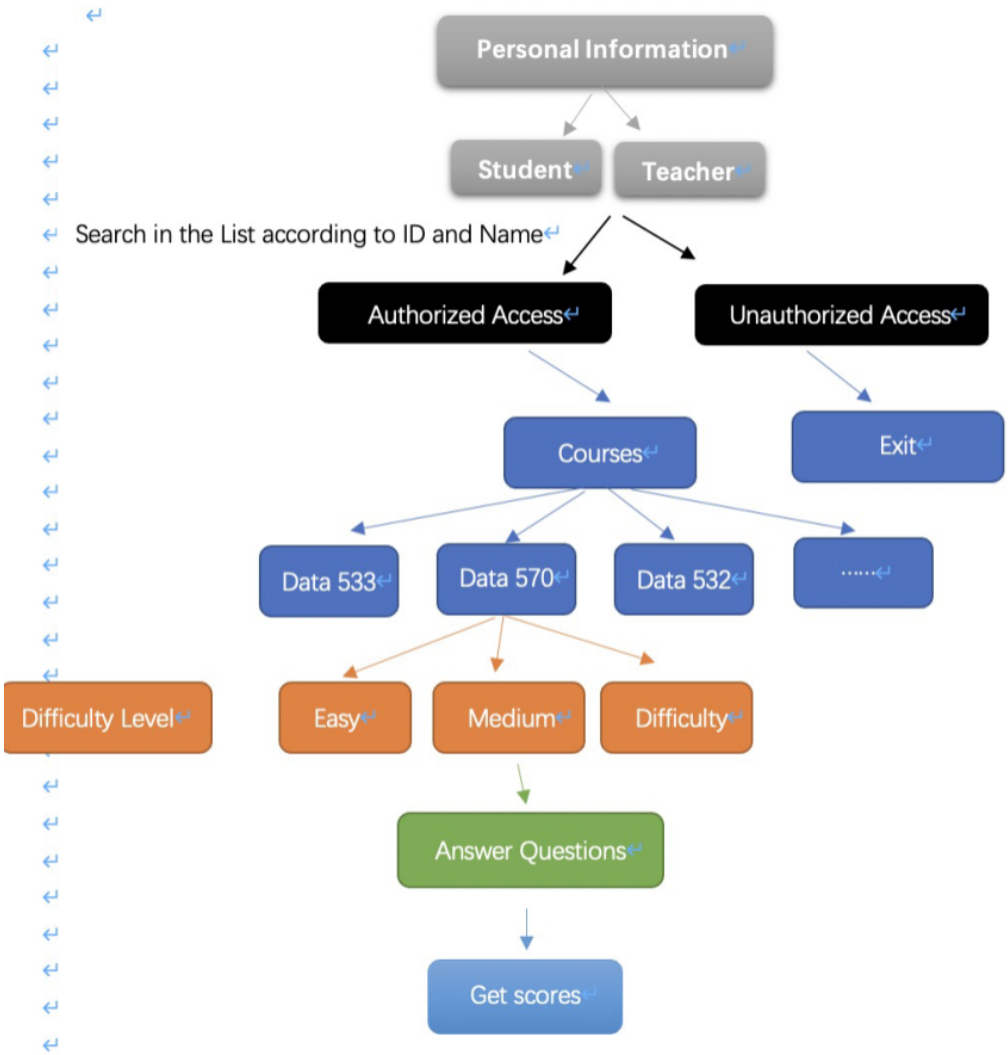
Lili Tang, Justin Chan

1. **Motivation:**

We want to create a quizzing apps for the authoized student of The University of British Columbia's Master of Data Science program to test their understanding on the material before taking the actual exam. This application will ask user for the subject(Data533, DATA570, DATA532…) and difficulty (easy, normal, hard, all(default)) that they want to be quized on. And validated their answer by giving feedback to them after every question. Finally the quiz apps will calculate a total marks of the quiz after finishing all the question inorder for them to keep track on their understanding on a specific topic. The application also allows authoized teacher to access the quiz to see what answer is the student being ask and for error checking purpose.

2. **Quiz App Frame (Introduction):**



3. Packages and Functions:

3.1 Sub-Package 1

This package will include the personal information input and output. Also, we will divide it into two categories, student and teacher.

3.1.1 Sub-Package 1 Module 1

In Module 1, we will set functions about **information-input**, such as name, ID. If a student, users need to input their major and degree; if a teacher, users need to input their research direction and teaching length.

Functions

```
#information-input


def group():
        g=input("Are you a student or a teacher?")
        return g

def name():
        n=input("What's your name?")
        return n

def ID():
        I=input("What's your ID? ")
        return I

# a student
def major():
        n=input("What's your major? ")
        return n
def degree():
        n=input("Are you a undergraduate or postgraduate?")
        return n
# a teacher
def research():
        n=input("what is your research direction?")
        return n

def length():
        n=input("How many years since be a tearcher?")
        return n
```

3.1.2 Sub-Package 1 Module 2

In Module 2, we will set functions about **information-output**, we use the **inheritance** in this module. The Group will show the common information: name and ID. Inside the Group, there are Student (show additional information about major and degree) and Teacher (show additional information about research and length)

Functions

```
#infromation-output

class Group:
    def __init__(self, name, ID):
        self.name = name
        self.ID = ID

    def display(self):
        print('Name:{} ID:{} '.format(self.name, self.ID))


class Student(Group):
    def __init__(self, name, ID, major, degree):
        Group.__init__(self, name, ID)
        self.major=major
        self.degree=degree

    def display(self):
        Group.display(self)
        print('Major: {} Degree: {}'.format(self.major, self.degree))

class Teacher(Group):
    def __init__(self, name, ID, research,length):
        Group.__init__(self, name, ID)
        self.research=research
        self.length=length

    def display(self):
        Group.display(self)
        print('Research: {} Length: {}'.format(self.research, self.length))
```

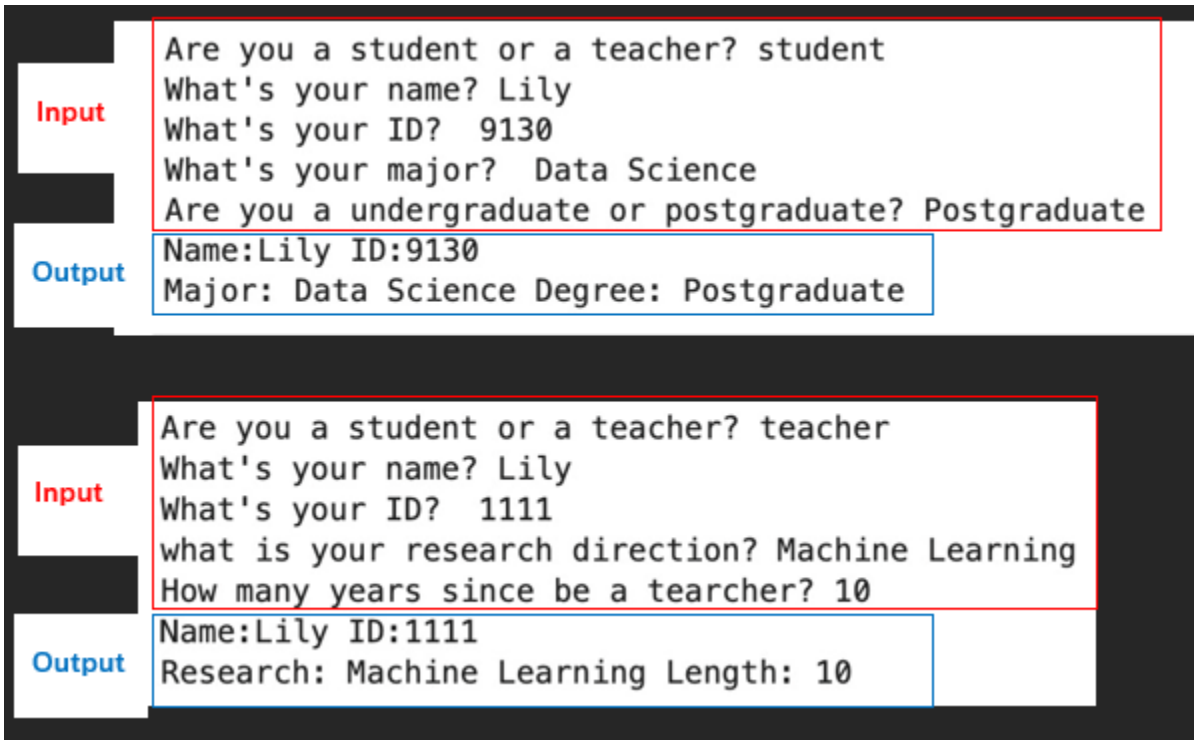
Finally, we need to conclude whether it is a student or a teacher and input and output different type information.

Functions

```
if  group()=="student":
    v1 = Student(name(), ID(), major(),degree());
    groups= [v1]
    for group in groups:
        group.display()

else:
    v1 = Teacher(name(), ID(), research(),length());
    groups= [v1]
    for group in groups:
        group.display()
```

Examples:



3.2 Sub-Package 2

This sub-package is all about processing question database and contain all the necessary checking function for the question and answer when user input in the quiz apps. This Sub-Package include 3 Module, Checking(the brain of the quiz apps), Question(for processing database question) and Course(appending defaulty to the database question)

3.2.1 Sub-Package 2 Module 1

This Module contain ```class Checking: ```, which is the brain of the quizzing apps, it contain all the necessary checking function such as:

1. **isEmpty(self) -** checking if there is more question to the question set. If there is, run function next_question() to prompt the user with new question and ask for their input.
2. **next_question(self) -** prompt the user with new question and ask for their input and validate user answer using check_answer().
3. **check_answer(self, user_answer, correct_answer)** - checking if the user input matched the database answer and if not prompt the user with the correct anser.
4. **check_score(self)** - checking user score.

```
def __init__(self, questions):
    self.quesBank = questions
    # self.usr_input = usr_input
    self.quesNum = 0
    self.score = 0
    self.ez_course_q = []
    self.hard_course_q = []
    self.normal_course_q = []

def next_question(self):
    current_question = self.quesBank[self.quesNum]

    self.quesNum += 1
    usr_input = input("Question " + str(self.quesNum)+ ". " + str(current_question.text))
    self.check_answer(usr_input.lower(), current_question.answer)

def isEmpty(self):
    return self.quesNum < len(self.quesBank)

def check_answer(self, user_answer, correct_answer):
    if user_answer == correct_answer:
        print("Correct!!! \n \n \n")
        self.score += 1
    else:
        print("Wrong   :( \n \n \n")
        print("The correct answer is: " + str(correct_answer)+ "\n")

def check_score(self):
    print("************************************ ")
    print("************************************ \n")
    print("Final score: " + str(self.score) + " out of " + str(self.quesNum) + ". \n")
    print("************************************ ")
    print("************************************ \n")
```

3.2.2 Sub-Package 2 Module 2

This Module contain ```class Question(Checking):``` which is inherited from Checking class, this class only have one function:

1. **__init__ (self, q_text, q_answer, q_diff)** - process database's question and allocate them into question, anser and difficulty.

```
def __init__(self, q_text, q_answer, q_diff):
    self.text = q_text
    self.answer = q_answer
    self.diff = q_diff
```

3.2.3 Sub-Package 2 Module 3

This Module contain ```Course(Question,Checking):``` which is inherited from Checking class and Question class, this module are for appending database's questions, answer into separate array with different difficulty for the other class to access based on user inputs in the future. this class only have 4 function:

1. **append_text(self,question) -** for appending database's question in to its own question array
2. **append_answer(self,question) -** for appending database's answer in to its own answer array
3. **append_difficulty(self,question) -** for appending database's difficultyin to its own difficulty array
4. **append_courseq(self) -** for appending database's question, answer according to the difficulty into 4 different question set array for user to access it after knowing the user difficulty that he/she want to be quiz on.

```
def __init__(self,question_data,input_diff):
    self.course_q = []
    self.ez_course_q = []
    self.hard_course_q = []
    self.normal_course_q = []
    self.question_data = question_data
    self.inputdiff = input_diff
    self.append_courseq()

    if self.inputdiff == "easy":
            Checking.__init__(self,self.ez_course_q)
    elif self.inputdiff == "normal":
            Checking.__init__(self,self.normal_course_q)
    elif self.inputdiff == "hard":
            Checking.__init__(self,self.hard_course_q)
    else:
            Checking.__init__(self,self.course_q)


def append_text(self,question):
    self.question_text = question["question"]

def append_answer(self,question):
    self.question_answer = question["correct_answer"]

def append_difficulty(self,question):
    self.question_difficulty = question["difficulty"]


def append_courseq(self):
    for question in self.question_data:
        self.append_text(question)
        self.append_answer(question)
        self.append_difficulty(question)
        new_question = Question(self.question_text, self.question_answer,self.question_difficulty)

        self.course_q.append(new_question)

        if self.question_difficulty == "easy":
            new_question = Question(self.question_text, self.question_answer,self.question_difficulty)
            self.ez_course_q.append(new_question)
        elif self.question_difficulty == "normal":
            new_question = Question(self.question_text, self.question_answer,self.question_difficulty)
            self.normal_course_q.append(new_question)
        elif self.question_difficulty == "hard":
            new_question = Question(self.question_text, self.question_answer,self.question_difficulty)
            self.hard_course_q.append(new_question)
```

Main() :

```
data533_q= []
data541_q= []

while True:
    usr_input = input("What do you want to be Quiz on? ")
    print("\n\n")


    if usr_input != '':
        input_diff = input("What difficulty? ")
        if usr_input == "data533":
            data553 = Course([
                                        {
                                            "question": "Which of the following represents a template or blueprint that defi
                                            "correct_answer": "A",
                                            "difficulty": "easy"
                                        },
                                        {
                                            "question": "How many of the following statements are TRUE?(Answer in Letter) \n
                                            "correct_answer": "3",
                                            "difficulty": "hard"
                                        }
                                    ],
                                    input_diff)

            print("************************************************")
            print("*             Welcome to DATA533.              *")
            print("* There is " + str(data553.quesNum) + " questions in total in DATA533.  *")
            print("************************************************")
            while data553.isEmpty():
                data553.next_question()

            data553.check_score()


        elif usr_input == "data541":
            data541 = Course([
                                        {
                                            "question": "How many of the following statements are TRUE?(Answer in Number) \n
                                            "correct_answer": "3",
                                            "difficulty": "easy"
                                        },
                                        {
                                            "question": "Which of the following command creates a copy of a remote repositor
                                            "correct_answer": "c",
                                            "difficulty": "noraml"
                                        },
                                        {
                                            "question": " How many of the following statements are TRUE?(Answer in Number) \
                                            "correct_answer": "3",
                                            "difficulty": "easy"
                                        }
                                    ],

                                    input_diff)


            print("************************************************")
            print("*             Welcome to DATA541.              *")
            print("* There is " + str(data541.quesNum) + " questions in total in DATA541.  *")
            print("************************************************")
            while data541.isEmpty():
                data541.next_question()

            data541.check_score()


        break #maybe Loop back and ask if use want to quiz or continoue

    else:
        print('User input is empty. Please Enter the course number that you would like to be quiz on.')
```

Example:

```
What do you want to be Quiz on?  data541



What difficulty?  easy
*******************************************
*             Welcome to DATA541.           *
* There is 0 questions in total in DATA541.  *
*******************************************
Question 1. How many of the following statements are TRUE?(Answer in Number)
1) Git is a is distributed version control system
2) Git is designed to support parallel development
3) Git is a web-based repository Service
4) Git doesn’t require a server
 3
Correct!!!



Question 2.  How many of the following statements are TRUE?(Answer in Number)
1) git branch NewBranch creates a new branch
2) git branch NewBranch does not check out to the NewBranch
3) git branch is tightly integrated with the git checkout and git merge commands
4) git branch -d NewBranch can be used to force delete the branch called NewBranch
 2
Wrong   :(



The correct answer is: 3

************************************
************************************

Final score: 1 out of 2.

************************************
************************************
```