GitHub Link - link

Code –

```
# Consider an example of declaring the examination result. Design three classes: St
udent, Exam, and Result.
# The Student class has data members such as those representing rollNumber, Name, e
tc.
# Create the class Exam by inheriting Student class. The Exam class adds fields rep
resenting the marks scored in six
# subjects. Derive Result from the Exam class, and it has its own fields such as to
tal_marks.
# Write an interactive program to model this relationship.

# 20CE034 - DEV GUNDALIA
# GitHub Repo Link - https://github.com/20CE034/PIP-II

class Student:
    # data members of the student class
    name = "DEFAULT_NAME"
    age = "18_DEFAULT"
    gender = "MALE_DEFAULT"
    roll_no = 18

    # constructor class...
    def _init_(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def set_roll_no(self, roll_no):
        self.roll_no = roll_no


class Exam(Student):

    sub_list = []
    total_subjects = 6
    marks_of_student = {}
    __marks_criteria = 100

    __gotMarks = False
    def _init_(self, name, age, gender, sub_length):
        super()._init_(name, age, gender)
        self.total_subjects = sub_length

        for i in range(self.total_subjects):
            sub_name = input(f"Enter name of the subject {i + 1} : ")
            self.sub_list.append(sub_name)

    def show_subject_list(self):
        print("The subjects are : ")
        for subject in self.sub_list:
            print(subject)

    def change_subject_list(self):
        confirmation = input(
            "Are you sure you want to change the subjects? (y/n) : ")
        if 'y' in confirmation or 'Y' in confirmation:
            print("Enter all the subject's name again!")
            for i in range(self.total_subjects):
                self.sub_list.append(input("Enter subject name : "))
            print("All subject's name changed successfully!")
        else:
            print("Operation cancelled successfully!")

        self.__gotMarks = False

    def alter_subjects(self, start, end):
        if start <= 0 or end > self.total_subjects:
            print("There is some error in arguments passed for alteration!")
        else:
            for i in range(start, end + 1):
                self.sub_list.pop(i - 1)
                self.sub_list.insert(
                    i - 1, input(f"Enter name of the subject {i} : "))

        self.__gotMarks = False

    def alter_subject(self, index):
        self.alter_subjects(index, index)

    def __getMarks(self):
        self.marks_of_student.clear()
        for subject in self.sub_list:
            marks = int(input(f"Enter marks for {subject} : "))
            if marks < 0 or (marks > self.__marks_criteria):
                print("Error in marks...entered...!")
                return
            else:
                self.marks_of_student.update({subject: marks})

        self.__gotMarks = True
        print("Mark-sheet has been updated!")

    def grab_marks(self):
        if self.__gotMarks:
            confirmation = input("""Are you sure you want to grab the marks again,
because the subjects are neither
            changed nor altered! (y/n) : """)
            if 'y' in confirmation or 'Y' in confirmation:
                self.__getMarks()
            else:
                print("Operation of getting new Mark-sheet has been cancelled!")
        else:
            self.__getMarks()

    def change_marks_criteria(self, max_marks):
        self.__marks_criteria = max_marks

    def show_marks_criteria(self):
        print(self.__marks_criteria)

    def get_marks_criteria(self):
        return self.__marks_criteria


class Result:

    exam = ""
    passing_criteria = 100
    grade_list = {"A": 90, "B": 80, "C": 60, "D": 50}

    def _init_(self, exam):
        self.exam = exam
        self.passing_criteria = (
            exam.get_marks_criteria() * exam.total_subjects) * 0.5

    def __getGrade(self, marks):
        for grades in self.grade_list.keys():
            if marks >= self.grade_list.get(grades):
                return grades
        return "Fail"

    def show_mark_sheet(self):
        if len(self.exam.marks_of_student) == self.exam.total_subjects:
            print("\n==============================================================
===")
            print(f"Mark-sheet of Student : {self.exam.name}")
            print(f"Roll no : {self.exam.roll_no}")
            print("\n--------------------------------------------------------------
---")
            for subject in self.exam.marks_of_student.keys():
                print(f"Marks in {subject} : ", self.exam.marks_of_student.get(subj
ect), "Grade : ",
                      self.__getGrade(self.exam.marks_of_student.get(subject)))
            print("----------------------------------------------------------------
-")
            print(
                f"Total marks : {self.exam.total_subjects * self.exam.get_marks_cri
teria()}/{sum(self.exam.marks_of_student.values())}")
            print("----------------------------------------------------------------
-")
            print(
                f"Overall Grade : {self.__getGrade(sum(self.exam.marks_of_student.v
alues())/self.exam.total_subjects)}")
            print("----------------------------------------------------------------
-\n")
            print("================================================================
=\n")
        else:
            print("Marks has not been updated!")


if __name__ == "_main_":
    ExamA = Exam("Student", 19, "MALE", 4)
    ResultA = Result(ExamA)
    ResultA.exam.grab_marks()
    ResultA.show_mark_sheet()
```

Output –

/