# GitHub Link:

# https://github.com/20CE027/20ce027/

# **Code** –

*# Consider an example of declaring the examination result. Design three classes: Student, Exam, and Result.*

*# The Student class has data members such as those representing rollNumber, Name, etc.*

*# Create the class Exam by inheriting Student class. The Exam class adds fields representing the marks scored in six*

*# subjects. Derive Result from the Exam class, and it has its own fields such as total\_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(subject), "Grade : ",

**self**.\_\_getGrade(**self**.exam.marks\_of\_student.get(subject)))

            print("-----------------------------------------------------------------")

            print(

                f"Total marks : {self.exam.total\_subjects \* self.exam.get\_marks\_criteria()}/{sum(self.exam.marks\_of\_student.values())}")

            print("-----------------------------------------------------------------")

            print(

                f"Overall Grade : {self.\_\_getGrade(sum(self.exam.marks\_of\_student.values())/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 –

/