**Course registration for next summer**

**Student id**

**Full name**

**Functional Requirements:**

* Add new students to a section
* Add teacher to a section
* Students in a section should be 10
* View history
* File handling

**Non Functional requirements:**

* The system should be fast
* System should be error free
* System should be tested

Now we will discuss the structure of the classes here so

1. Person

|  |  |
| --- | --- |
| Attributes | Data Types |
| Name | String |
| ID | Integer |
| Gender | String |

Now for the Functions in the class person are as fallows

* Functions

|  |  |  |
| --- | --- | --- |
| Functions | Arguments | Return type |
| Getname | Void | String |
| Getid | Void | Int |
| Getgender | Void | String |
| Setname | String | Null |
| Setid | Integer | Null |
| Setgender | String | Null |

1. teacher

|  |  |
| --- | --- |
| Attributes | Data Types |
| Designation | String |
| Pay | Int |

Now for the Functions in the class are as fallows

* Functions

|  |  |  |
| --- | --- | --- |
| Functions | Arguments | Return type |
| Getdesignation | Void | String |
| Setdesignation | String | Null |
| getpay | Void | Int |
| setpay | Int | Null |

1. student

|  |  |
| --- | --- |
| Attributes | Data Types |
| department | String |
| Rollno | Int |

Now for the Functions in the class are as fallows

* Functions

|  |  |  |
| --- | --- | --- |
| Functions | Arguments | Return type |
| Getdepartment | Void | String |
| Setdepartment | String | Null |
| Getrollno | Void | Int |
| Setrollno | Int | Null |

1. section

|  |  |
| --- | --- |
| Attributes | Data Types |
| Teacher\_ | Teacher |
| Students | List |
| No\_of\_students | Int |

Now for the Functions in the class are as fallows

* Functions

|  |  |  |
| --- | --- | --- |
| Functions | Arguments | Return type |
| add\_student | Void | Void |
| add\_teacher | Void | Void |

1. class\_

|  |  |
| --- | --- |
| Attributes | Data Types |
| section | List |
| Course\_code | String |
| No | Int |

Now for the Functions in the class are as fallows

* Functions

|  |  |  |
| --- | --- | --- |
| Functions | Arguments | Return type |
| addclass | Void | Void |

1. file\_handling

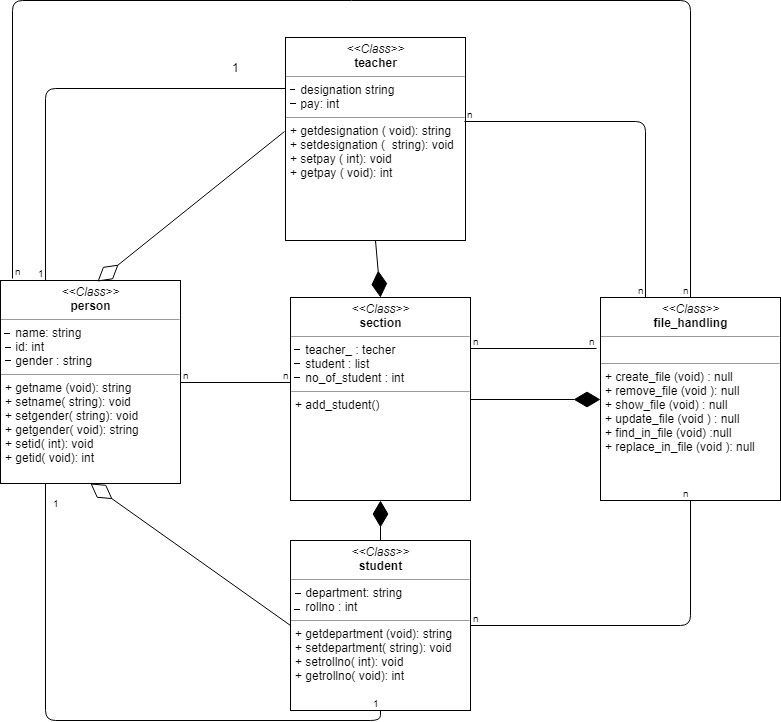
Now for the Functions in the class are as fallows

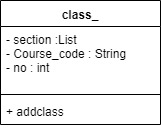
* Functions

|  |  |  |
| --- | --- | --- |
| Functions | Arguments | Return type |
| create\_file | Void | Null |
| remove\_file | Void | Null |
| show\_file | Void | Null |
| update\_file | Void | Null |
| find\_in\_file | Void | Null |
| replace\_in\_file | void | Null |

**Class Diagram :**

The class diagram of the code is given below





**Assumptions :**

Here the assumptions made are as fallows

* Data about teacher is already filled
* Data regarding the student and department is already added
* fee calculation is according to system time
* There is some admin who is managing the system

**Pseudo code:**

import os

class person:

name=''

id=0

gender=''

def \_\_init\_\_(self):

name=''

id=0

gender=''

pass

def getname(self):

return self.name

def getid(self):

return self.id

def getgender(self):

return self.gender

def setname(self,a):

self.name=a

def setid(self,a):

self.id=a

def setgender(self,a):

self.gender=a

class teacher(person):

designation=''

pay=0

def getpay(self):

return pay

def setpay(self,a):

self.pay=a

def \_\_init\_\_(self):

designation=''

shift=''

def getdesignation(self):

return self.designation

def setdesignation(self,a):

self.designation=a

def setid(self, a):

id=a

def setname(self,a):

name=a

class student (person):

rollno = 0

department = ''

def \_\_init\_\_(self):

rollno = 0

department = ''

def getrollno(self):

return self.rollno

def getdepartment(self):

return self.department

def setrollno(self,a):

self.rollno=a

def setid(self, a):

id=a

def setname(self,a):

name=a

def setdepartment(self,a):

self.department=a

class section:

teacher\_ = teacher()

students=[]

no\_of\_students=0

def \_\_init\_\_(self):

pass

def add\_student(self):

if self.no\_of\_students < 10 :

a=str(input("Enter the name "))

b=str(input("Enter the rollno "))

c=str(input("Enter the department "))

stu=student()

stu.setrollno(b)

stu.setdepartment(c)

self.students.append(stu)

file = open('Student.txt', "a")

leng =len(self.students)

aa="The name of the student is "+ a + '\n'

bb="The rollno of the student is "+ b+ '\n'

cc="The department of the student is "+ c+ '\n'

file.write("Student no " + str(leng) +'\n')

file.write(aa)

file.write(bb)

file.write(cc)

file.write('\n')

file.write('\n')

file.close()

def add\_teacher(self):

a=str(input("Enter the name "))

b=str(input("Enter the designation "))

c=str(input("Enter the ID "))

teacher1=teacher()

teacher1.setdesignation(b)

teacher1.setid(c)

teacher1.setname(a)

self.teacher\_=teacher1

file = open('Teacher.txt', "a")

#leng =len(self.teacher\_)

aa="The name of the Teacher is "+ a + '\n'

bb="The designation of the teacher is "+ b+ '\n'

cc="The id of the teacher is "+ c+ '\n'

file.write("The teacher for this course is " +'\n')

file.write(aa)

file.write(bb)

file.write(cc)

file.write('\n')

file.write('\n')

file.close()

class class\_():

section=[]

no=3

name=''

course\_code=''

def \_\_init\_\_(self):

pass

def addclass(self):

a=str(input("Enter the name of the course "))

b=str(input("Enter the course id "))

c=str(input("Enter the credit hours "))

file = open('Course.txt', "a")

aa="The name of the course is "+ a + '\n'

bb="The Course code is "+ b+ '\n'

cc="The Credit hours are "+ c+ '\n'

file.write(aa)

file.write(bb)

file.write(cc)

file.write('\n')

class file\_handling():

def \_\_init\_\_():

pass

def create\_file():

file\_name = str(input ("please enter the name of the file to be created "))

file\_name = file\_name + '.txt'

f = open(file\_name,"w+")

def remove\_file():

file\_name = str(input("please enter the name of the file to be deleted "))

file\_name = file\_name + '.txt'

if os.path.isfile('./' + file\_name):

os.remove(file\_name)

print("File Removed!")

else:

print ('file does not exist')

def show\_file():

file\_name = str(input("please enter the name of the file to be shown "))

file\_name = file\_name + '.txt'

if os.path.isfile('./' + file\_name):

file = open(file\_name, "r")

for line in file:

print(line)

else:

print('file not found ')

def update\_file():

file\_name = str(input("please enter the name of the file to be updated "))

file\_name = file\_name + '.txt'

choice = str(input ("please enter 'a' for append and 'o' for overwrite "))

if choice =='a':

msg = ''

if os.path.isfile('./' + file\_name):

file = open(file\_name, "a")

while True:

msg = str(input ("please enter the line of message to be updated to the file or 'e' to end the writing "))

if msg != 'e':

msg="\n" + msg

file.write(msg)

else:

break

file.close()

else:

print ('file does not exist')

if choice =='o':

if os.path.isfile('./' + file\_name):

file = open(file\_name, "w")

while True:

msg = str(input ("please enter the line of message to be updated to the file or 'e' to end the writing "))

if msg != 'e':

file.write(msg + "\n")

else:

break

file.close()

else:

print ('file does not exist')

def find\_in\_file():

file\_name = str(input("please enter the name of the file to do the search in "))

file\_name = file\_name + '.txt'

if os.path.isfile('./' + file\_name):

input\_ = str(input("please enter the string you want to find "))

file = open(file\_name, "r")

for i, line in enumerate(file,1):

num=line.find(input\_)

if num==0:

print('string found on line number ' + str(i))

else:

print ('File not found ')

def replace\_in\_file():

file\_name = str(input("please enter the name of the file to do the replacing "))

file\_name = file\_name + '.txt'

if os.path.isfile('./' + file\_name):

input\_ = str(input("please enter the string you want to replace "))

input1\_ = str(input("please enter the string you want to replace with "))

file = open(file\_name,"r")

i=0

file\_dump=''

for line in file:

file\_dump += line.replace(input\_,

input1\_)

file = open(file\_name, "w")

file.write(file\_dump)

file.close()

else:

print ('file not found')

**Test Cases:**

For the test cases we implemented a file management system and there are three test cases for this code

**Test\_for\_courses:**

First is test\_for\_courses it simply takes input for courses saves them in the list and do the file writing for courses in course.txt this test case takes input from the user and saves the name of the course . the course of the code and total number of credit hours of the course

**Test\_for\_students:**

Second test is called the test for student It takes input from the user on the terminal and stores that information in an object than this object is store in a list and then it is written on the file It writes the information like student no the name of the student , the roll no of the student and the department of the student

**Test\_for\_teachers**

This test is used to test the adding of the teacher as it is adding right or not first of all the object is created for the teacher class then the credentials are entered and set for the class after that it stored in a teacher variable as the problem statement states that only one teacher can teach one section so after storing the information it is written to the file it writes the name of the teacher after that the designation of the teacher and then the id of the teacher .

**Conclusion :**

In conclusion I must say that it was an informative and interesting task I learned how to do the proper inheritance with different scenarios I also learnt a lot about student management systems In the end I will also say that it was not that easy of a task I had to do some research before starting to gather requirements to make it better it can be made better with the Gui Implementation of Hospital management system in tkinter or other Graphical library so that I can be more interactive to the people

**Reflection**:

**Student 1:**

**Documentation**: Contributed in collection of functional and nonfunctional requirements for the best quality project

**Code**: contributed in developing project logic and find the loop holes in previous versions of the code

**Quality**: Contributed in the quality of the input, output and submission of the project

**Student 2:**

**Documentation:** Contributed in making of UML diagrams

**Code**: Contributed in making the code object oriented and error free and to see if it is meeting the required out puts.

**Quality:** worked on the quality of the report and code if it is according to the requirements or not

**Student 3:**

**Documentation:** Gathered and compiledthe report and added the pseudo code and double check the if there are any errors Added assumption, conclusion and reflection part as I was leading the team

**Code:** Compiled and tested the code if its error free and added my part of file handling and checked the hierarchy of the code

**Quality:** Assured that everything is in order and submissions was according to the requirement project logic was fine and input output statements were according to the requirements