PRE DEFINED Students in File

- **Name- Kanishk Jain**
  Registration ID- 1
  Password -hello
  EmailID- kanishk112001@gmail.com
  Contact no- 8178653882
  Course- OOPS
  Marks-[89,90,87]

  Name- Ayush
  Registration ID- 2
  Password -ayush1234
  EmailID- ayush@gmail.com
  Contact no- 46513654658
  Course- OOPS
  Marks-[54,87,48]

- **Name- Rohit**
  Registration ID- 3
  Password -rohit1234
  EmailID- rohit@gmail.com
  Contact no- 84655565795
  Course- OOPS
  Marks-[12,54,65]

PRE -DEFINED TEACHER(Hardcoded in Code)

- **Name-Amit**
  Registration-ID-100
  Password-helloamit
  EmailID- amitdua@pilani.bits-pilani.ac.in
  Course- OOPS
  Contact No.-1113334445

PRE DEFINED COURSE(Reading From File)

- CourseName- OOPS
  Topics-Polymorphism, Inheritance,Encapsulation

  CourseName-JAVA
  Topics-Object Oriented Language, Oracle, Similar to C++

CLASS **User**

- `User(String name, Integer regId,Course course, String emailId, String password, String contactNo)`

  It is the constructor function for creating student

- `public boolean Login(Integer regId, String password)`

  It returns the boolean logic after passing registration Id and password

- `public User updateProfile()`

  It returns the updated User object after I/O operations in the definition of function

- `public void printDetails()`

  Prints the details of users

- `public Course returnCourse()`

  Returns the Course object of the value assigned to the user

CLASS **Student**

- `Student(String name, Integer regId,Course course, String emailId, String password, String contactNo)`
- `public String getName()`
- `public void setName(String name)`

  Getter and Setter function for Name

- `public ArrayList<Integer> getMarks()`
- `public void setMarks(ArrayList<Integer> arrMarks)`

  Getter and Setter function for marks

- `public int getRegID()`
- `public String getCourse()`

CLASS **Admin**

- `Admin(String name, Integer regId, Course course, String emailId, String password, String contactNo)`
- `public String getTId()    //getter function fot TId`
- `public String getName()   // getter function for Name of admin`
- `public List<Student> updateStudent(List<Student> studentList, Integer regId)`

  This functions updates the studentList by storing updated student information in place of old student
- `public ArrayList<Student> removeStudent(Student student, ArrayList<Student> arrstu)`

  Removes the given student from the list of students
- `public String courseName()`

CLASS **DriverCode**

- `public void welcome(String username)  // this function simply welcomes the guest`
- `public Student search(List<Student> studentList, Integer regId)`

  This function searches a particular student of registration Id from list and returns that student

CLASS ClearScreen

- `public static void clearScreen()`

  It simply clears the terminal window to make output much tidier

CLASS **AdminService**

- `AdminService(Admin teacher)        // constructor`
- `public void viewDetails(Admin teacher)  //prints detail of teacher`
- `public void viewDetails(Student student)      //prints details of Student`
- `public void viewMarks(Student student)  // views the marks of studdent`
- `public ArrayList<Integer> updateMarks(ArrayList<Integer> arrMarks)`

  This function iterates over the arrayList of marks using list iterator and updates the value of marks by removing previous score and updating it with the new value.

- `public Course updateCourse(Course course)`

Returns the modified course where either new course is created if not present else it rewrites the topics of th course using constructor and modifyCourse method of Course.

- `public void viewEnrolledStudent(List<Student> studentList,Course course)`

Prints out the name of students who are in the course by iterating over the studentList.

- `public void viewCourse()`

Prints the name of course which is associated the the teacher

CLASS **StudentService**

- `StudentService(Student student)  //constructor`
- `public void viewDetails(Student student)  //prints out the details of student`
- `public void viewMarks(Student student) //Prints out the marks of student`
- `public void viewCourse()  //Prints the course associated to this student`
- `public void viewEnrolledStudent(List<Student> studentList,Course course)`

Prints out all the students name from studentList who are associated with this Course.

CLASS **Course**

- `Course()  //Constructor`
- `Course(String courseName)  //Constructor`
- `Course(String courseName, ArrayList<Topics> topics) //Constructor`
- `String getCourseName() //returns the courseNmae`
- `public void getTopics()  //Prints out all the topics of the course`
- `public ArrayList<Topics> modifyCourse()`

This function first prints out all the topics of course and then asks the user to either empty the topic content or modify the topic from scratch. It will keep asking users to modify the topic name until the user does not enter No. We can add various topics in the course using this function.

CLASS **Topics**

- `Topics(String topicName)  //constructor`
- `public void viewAllTopics(ArrayList <Topics>`

Prints all the topics of the Course

- `public String getName()  //Returns the topicName of give Topic`
- `public void changeTopic(String topic)  //Changes the TopicName of Given topic`

CLASS **Lms**

- The main function of the class works with the help of nested switch, while,for and if loops until the user does not wish to exit the program
- It allows users to enter the program either as Student or Teacher who are already stored if the data file.
- Information about the teacher is hard coded into the program.
- Each user can perform multiple actions as per its access.
- All users must verify their credentials by entering valid registration Id and corresponding password.
- Students mainly have viewing access and updating his profile access.
- Teacher has access to viewing as well as modifying the marks, details of course, removing course, removing student and various other privileges.
- To find the correct student with regId from Studentlist(List) a separate search function is defined.

NOTE(Prerequisite):

- The code works on **JavaSE-16** .
- Only use **Windows** for running this code.
- If possible use **vs-code** for evaluation as it is optimised for it.
- While using the terminal make sure it covers more than **50%** of the screen to take advantage of the clear screen option implemented for formatting the output.
- Hardcoded Student and Course data are provided in their respective files.
- I*f data files are corrupted due to compression simply create new data files with the same name and formats and delete the previous files. Uncomment the line*

427,428, 430,431 *during initial run and then after simply comment those lines.Now you can add students, courses to the file using Student and Teacher function respectively. Populate the marks of student by teacher*.