Message Methods:

```
createMessage(text: String, user: User, forum: Forum?): Message
    SET msg ← new Message()
    SET msg.text ← text
    SET msg.user ← user
    SET msg.forum ← forum
    SET msg.timestamp ← now()
    IF forum ≠ null THEN
        ADD msg to forum.messageList
    ENDIF
    RETURN msg

sendMessage(): Boolean
    IF forum ≠ null THEN
        ADD this to forum.messageList
        RETURN true
    ELSE
        RETURN false
    ENDIF

displayMessage(): void
    PRINT "[", timestamp, "] ", user.name, ": ", text

editMessage(newText: String): Boolean
    IF user is the original author THEN
        SET text ← newText
        RETURN true
    ELSE
        RETURN false
    ENDIF

deleteMessage(): Boolean
    IF forum ≠ null AND this in forum.messageList THEN
        REMOVE this from forum.messageList
        RETURN true
    ELSE
        RETURN false
    ENDIF

addReaction(type: String, by: User): void
    IF react contains key type THEN
        IF by not in react[type] THEN
            ADD by to react[type]
        ENDIF
    ELSE
        SET react[type] ← [by]
    ENDIF
```

```
displayReactions(): void
   FOR EACH (type, users) IN react
      PRINT type, ": ", [u.name FOR u IN users]
   ENDFOR

removeReaction(type: String, by: User): void
   IF react contains key type AND by in react[type] THEN
      REMOVE by from react[type]
      IF react[type] is empty THEN
         REMOVE key type from react
      ENDIF
   ENDIF

displayReply(): void
   FOR EACH replyMsg IN reply
      CALL replyMsg.displayMessage()
   ENDFOR
```

Course Methods:

```
createCourse(name: String, courseID: String, sectionNumber: Int, professor: User, startDate: Date):
Course
   SET course ← new Course()
   SET course.name ← name
   SET course.courseID ← courseID
   SET course.sectionNumber ← sectionNumber
   SET course.professor ← professor
   SET course.startDate ← startDate
   INITIALIZE course.memberList as empty list
   RETURN course

displayCourse(): void
   PRINT "Course: ", name, " (", courseID, " Section ", sectionNumber, ")"
   PRINT "Professor: ", professor.name
   PRINT "Start Date: ", startDate

editCourse(newName: String?, newSectionNumber: Int?, newProfessor: User?): void
   IF newName ≠ null THEN SET name ← newName ENDIF
   IF newSectionNumber ≠ null THEN SET sectionNumber ← newSectionNumber ENDIF
   IF newProfessor ≠ null THEN SET professor ← newProfessor ENDIF

deleteCourse(): Boolean
   // assume workspace or system context holds all courses
   IF this in System.courses THEN
      REMOVE this from System.courses
      RETURN true
   ELSE
      RETURN false
   ENDIF
```

```
addMember(user: User): void
   IF user not in memberList THEN
      ADD user to memberList
      ADD this to user.courseList
   ENDIF

displayMembers(): List<User>
   FOR EACH u IN memberList
      PRINT u.name, " (", u.userType, ")"
   ENDFOR
   RETURN memberList

updateMember(user: User, newRole: String): void
   IF user in memberList THEN
      // e.g. TA → change role in user.preference or profile
      SET user.preference["role in " + courseID] ← newRole
   ENDIF

removeMember(user: User): Boolean
   IF user in memberList THEN
      REMOVE user from memberList
      REMOVE this from user.courseList
      RETURN true
   ELSE
      RETURN false
   ENDIF
```

Assignment Methods:

```
createAssignment(course: Course, name: String, description: String, dueDate: Date, assigner: User):
Assignment
   SET a ← new Assignment()
   SET a.course ← course
   SET a.name ← name
   SET a.description ← description
   SET a.dueDate ← dueDate
   SET a.assigner ← assigner
   INITIALIZE a.feedback as empty list
   SET a.completionStatus ← false
   ADD a to course.assignmentList
   RETURN a

displayAssignment(): void
   PRINT "Assignment: ", name
   PRINT "Description: ", description
   PRINT "Due: ", dueDate
   PRINT "Status: ", (completionStatus ? "Complete" : "Incomplete")
```

```
editAssignment(newDescription: String?, newDueDate: Date?): void
  IF newDescription ≠ null THEN SET description ← newDescription ENDIF
  IF newDueDate ≠ null THEN SET dueDate ← newDueDate ENDIF

deleteAssignment(): Boolean
  IF this in course.assignmentList THEN
    REMOVE this from course.assignmentList
    RETURN true
  ELSE
    RETURN false
  ENDIF

assign(to: User): void
  SET submitter ← to
  ADD this to to.assignmentList

unassign(from: User): void
  IF submitter = from THEN
    SET submitter ← null
    REMOVE this from from.assignmentList
  ENDIF

uploadText(content: String): void
  // student uploads draft or notes
  SET description ← description + "\n\n" + content

removeText(): void
  // clears any appended text
  // assume original description stored elsewhere if needed
  // for now, reset any temporary text
  // no-op or log removal

uploadFile(file: File): void
  SET attachment ← file

removeFile(fileID: String): void
  IF attachment.id = fileID THEN
    SET attachment ← null
  ENDIF

submitWork(submitter: User, content: String): void
  IF submitter = this.submitter THEN
    SET description ← content
    SET completionStatus ← false
  ENDIF

unsubmitWork(): void
  IF completionStatus = false THEN
    SET description ← ""
    ENDIF

submitGrade(grade: String): void
    SET this.grade ← grade

addFeedback(comment: String): void
    ADD comment to feedback

displayFeedback(): void
    FOR EACH c IN feedback
        PRINT "- ", c
    ENDFOR

editFeedback(index: Int, newComment: String): void
    IF index within bounds of feedback THEN
        SET feedback[index] ← newComment
    ENDIF

removeFeedback(index: Int): void
    IF index within bounds of feedback THEN
        REMOVE feedback[index]
    ENDIF

markCompleted(): void
    SET completionStatus ← true
```