LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: Abhishek Pandey

Group: C9

London Met ID: 20048809

College ID: NP01CP4S210239

Assignment Due Date: 20th August

Assignment Submission Date: 16th August

Table of Contents

Table of Figures

Table of Tables

1. CLASS DIAGRAM

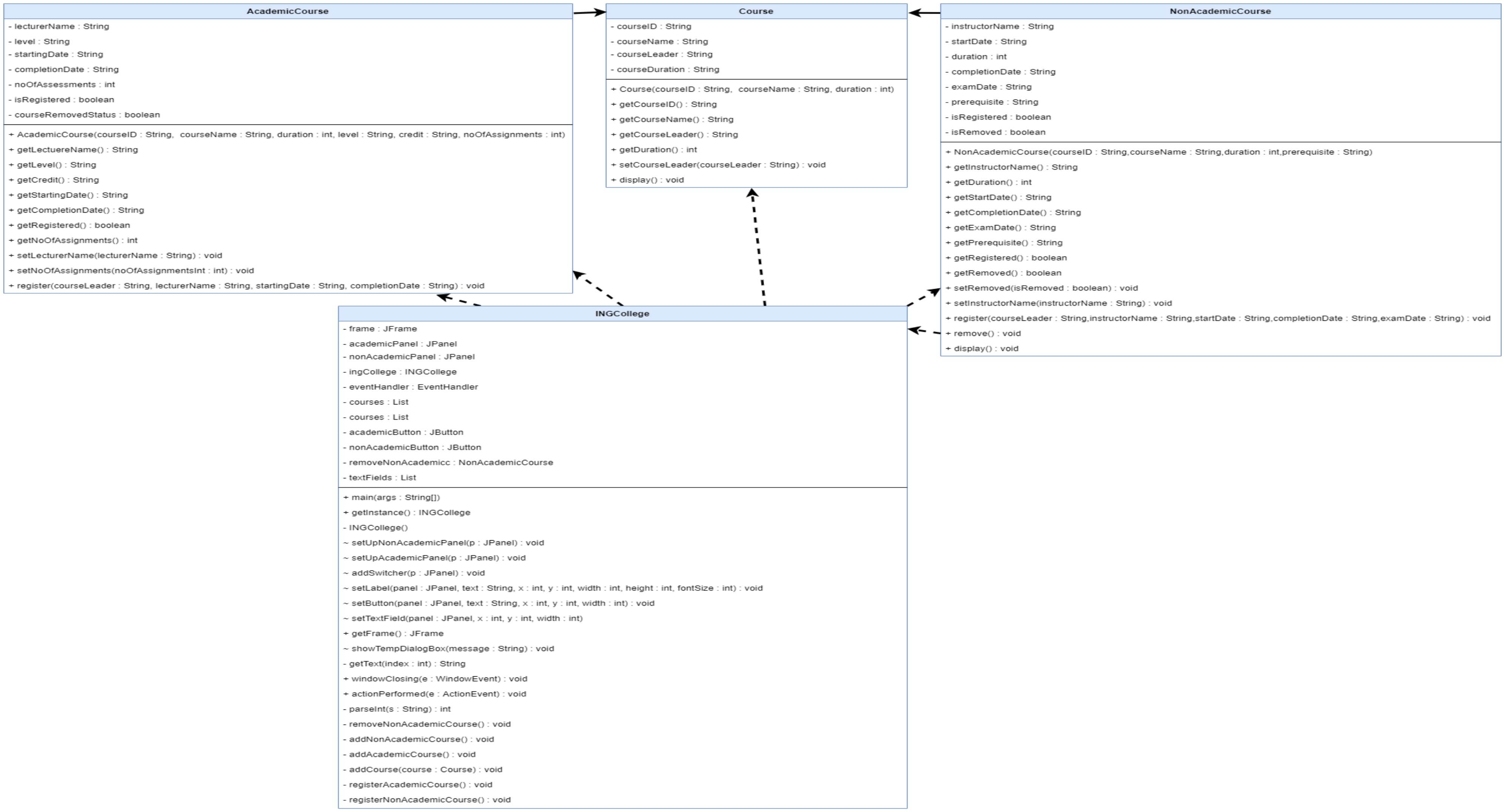

AcademicCourse
- lecturerName : String
- level : String
- startingDate : String
- completionDate : String
- noOfAssessments : int
- isRegistered : boolean
- courseRemovedStatus : boolean
+ AcademicCourse(courseID : String, courseName : String, duration : int, level : String, credit : String, noOfAssignments : int)
+ getLectuereName() : String
+ getLevel() : String
+ getCredit() : String
+ getStartingDate() : String
+ getCompletionDate() : String
+ getRegistered() : boolean
+ getNoOfAssignments() : int
+ setLecturerName(lecturerName : String) : void
+ setNoOfAssignments(noOfAssignmentsInt : int) : void
+ register(courseLeader : String, lecturerName : String, startingDate : String, completionDate : String) : void
Course
- courseID : String
- courseName : String
- courseLeader : String
- courseDuration : String
+ Course(courseID : String, courseName : String, duration : int)
+ getCourseID() : String
+ getCourseName() : String
+ getCourseLeader() : String
+ getDuration() : int
+ setCourseLeader(courseLeader : String) : void
+ display() : void
NonAcademicCourse
- instructorName : String
- startDate : String
- duration : int
- completionDate : String
- examDate : String
- prerequisite : String
- isRegistered : boolean
- isRemoved : boolean
+ NonAcademicCourse(courseID : String,courseName : String,duration : int,prerequisite : String)
+ getInstructorName() : String
+ getDuration() : int
+ getStartDate() : String
+ getCompletionDate() : String
+ getExamDate() : String
+ getPrerequisite() : String
+ getRegistered() : boolean
+ getRemoved() : boolean
+ setRemoved(isRemoved : boolean) : void
+ setInstructorName(instructorName : String) : void
+ register(courseLeader : String,instructorName : String,startDate : String,completionDate : String,examDate : String) : void
+ remove() : void
+ display() : void
INGCollege
- frame : JFrame
- academicPanel : JPanel
- nonAcademicPanel : JPanel
- ingCollege : INGCollege
- eventHandler : EventHandler
- courses : List
- courses : List
- academicButton : JButton
- nonAcademicButton : JButton
- removeNonAcademicc : NonAcademicCourse
- textFields : List
+ main(args : String[])
+ getInstance() : INGCollege
- INGCollege()
~ setUpNonAcademicPanel(p : JPanel) : void
~ setUpAcademicPanel(p : JPanel) : void
~ addSwitcher(p : JPanel) : void
~ setLabel(panel : JPanel, text : String, x : int, y : int, width : int, height : int, fontSize : int) : void
~ setButton(panel : JPanel, text : String, x : int, y : int, width : int) : void
~ setTextField(panel : JPanel, x : int, y : int, width : int)
+ getFrame() : JFrame
~ showTempDialogBox(message : String) : void
- getText(index : int) : String
+ windowClosing(e : WindowEvent) : void
+ actionPerformed(e : ActionEvent) : void
- parseInt(s : String) : int
- removeNonAcademicCourse() : void
- addNonAcademicCourse() : void
- addAcademicCourse() : void
- addCourse(course : Course) : void
- registerAcademicCourse() : void
- registerNonAcademicCourse() : void

Figure 1 - Class Diagram

2. PESUDO CODE

Course Class

```
CREATE CLASS Course

    DECLARE private instance variables courseID, courseName, courseLeader of type String

    DECLARE private instance variable duration of type int

    CREATE CONSTRUCTOR Course(TAKE parameters courseID, courseName of String type AND duration of int type)
        ASSIGN value of parameter courseID to instance variable courseID of the class

        ASSIGN value of parameter coursenamme to instance variable coursename of the class

        ASSIGN value of parameter duration to instance variable duration of the class

        INITIALIZE courseLeader to empty string
    END CONSTRUCTOR
```

```
DEFINE METHOD getCourseID
        RETURN courseID
END METHOD

DEFINE METHOD getName
        RETURN courseName
END METHOD

DEFINE METHOD getCourseLeader
        RETURN courseLeader
END METHOD

DEFINE METHOD getDuration
        RETURN duration
END METHOD

DEFINE METHOD setCourseLeader(TAKE parameter courseLeader of type
String)
        ASSIGN value of courseLeader to instance variable courseLeader of the
        class
END METHOD
```

```
        DEFINE METHOD display
                PRINT "courseID: " AND value of courseID variable
                PRINT "courseName: " AND value of courseName variable
                PRINT "courseDuration: " AND value of courseDuration variable

                IF courseLeader variable does not have an empty value ""
                        PRINT "courseLeader: " AND value of courseLeader variable
                END IF
        END METHOD

END CLASS
```

AcademicCourse Class

```
CREATE CLASS AcademicCourse that EXTENDS Course class
        DECLARE private instance variables lecturerName, level, credit, startingDate,
        completionDate of type String

        DECLARE private variable noOfAssessments of type int

        DECLARE private variables isRegistered, courseRemovedStatus of type boolean
```

```
CREATE CONSTRUCTOR AcademicCourse(TAKE parameters courseID,
courseName, level of type String, duration, noOfAssessments, credit of type
int)
          CALL Course(PASS arguments courseID, courseName, duration) from
          superclass Course
          INITIALIZE lecturerName to empty String ""
          INITIALIZE startingDate to empty String ""
          INITIALIZE completionDate to empty String ""
          SET value isRegistered to false
END CONSTRUCTOR


DEFINE METHOD getLecturerName
          RETURN lecturerName
END METHOD


DEFINE METHOD getLevel
          RETURN level
END METHOD


DEFINE METHOD getCredit
          RETURN credit
END METHOD
```

```
DEFINE METHOD getStartingDate
        RETURN startingDate
END METHOD

DEFINE METHOD getCompletionDate
        RETURN completionDate
END METHOD

DEFINE METHOD getRegistered
        RETURN isRegistered
END METHOD

DEFINE METHOD getNoOfAssessments
        RETURN noOfAssessments
END METHOD

DEFINE METHOD setLecturerName(TAKE parameter lecturerName of type
String)
    ASSIGN value of lecturerName to instance variable lecturerName of the class
END METHOD

DEFINE METHOD setNoOfAssessments(TAKE parameters noOfAssessments of
type int)
    ASSIGN value of noOfAssessments to instance variable
    noOfAssessments of the class
END METHOD
```

```
DEFINE METHOD register(TAKE parameters courseLeader, lecturerName,
startingDate, completionDate of type String)
        IF isRegistered
                CALL showTempDialogBox(PASS “Printing..” as argument) from
                INGCollege class
                PRINT "Instructor name: " AND lecturerName
                PRINT "Starting date: " AND startingDate
                PRINT "Completion date: " AND completionDate
        ELSE
                ASSIGN value of lecturerName to instance variable lecturerName of
                the class

                ASSIGN value of startingDate to instance variable startingDate of the
                class

                ASSIGN value of completionDate to instance variable completionDate
                of the class

                CALL setCourseLeader(PASS argument courseLeader) from
                superclass

                SET value of isRegisreted to true

                SET value of courseRemovedStatus to false

                CALL showTempDialogBox(PASS “Registering..” as argument) from
                INGCollege class
```

```
            END IF
        END METHOD

        DEFINE METHOD display
            CALL display method from superclass
            IF isRegistered
                PRINT "Lecturer name: " AND lecturerName
                PRINT "Lebel: " AND level
                PRINT "Starting date: " AND startingDate
                PRINT "Completion date: " AND completionDate
                PRINT "Number of Assessments: " AND noOfAssessments
            END IF
        END METHOD

END CLASS
```

NonAcademicCourse Class

```
CREATE CLASS NonAcademicCourse that EXTENDS Course class

    DECLARE private instance variables instructorName, startDate,
    completionDate, examDate, prerequisite of type String

    Declare private instance variable duration of type int

    Declare private instance variables isRegistered, isRemoved of type boolean

    CREATE CONSTRUCTOR NonAcademicCourse(TAKE parameters courseID,
    courseName, prerequisite of type String, duration of type int)

        CALL Course(PASS arguments courseID, courseName, duration)

        ASSIGN value of parameter prerequisite to instance variable prerequisite of
        the class

        INITIALIZE startDate, completionDate, examDate to empty String ""

        INITIALIZE isRegistered to False
    END CONSTRUCTOR
```

```
DEFINE METHOD getInstructorName
        RETURN instructorName
END METHOD


DEFINE METHOD getDuration
        RETURN duration
END METHOD


DEFINE METHOD getStartDate
        RETURN startDate
END METHOD


DEFINE METHOD getCompletionDate
        RETURN completionDate
END METHOD


DEFINE METHOD getExamDate
        RETURN examDate
END METHOD


DEFINE METHOD getPrerequisite
        RETURN prerequisite
END METHOD
```

```
DEFINE METHOD getRegistered
        RETURN isRegistered
END METHOD

DEFINE METHOD getRemoved
        RETURN isRemoved
END METHOD

DEFINE METHOD setRemomved(TAKE parameter isRemoved of type boolean)
    ASSIGN value of isRemoved to instance variable isRemoved of the class
END METHOD

DEFINE METHOD setInstructorName(TAKE parameter instructorName of type
String)
        IF not isRegitered
           ASSIGN value of instructorName to instance variable instructorName of the
           class
        ELSE
           SHOW ERROR DIALOG "It is not possible to change instructor name since
           non academic course has already been registered"
        END IF
END METHOD
```

```
DEFINE METHOD register(TAKE parameters courseLeader, instructorName,
startDate, completionDate, examDate of type String)
    IF not isRegestered
        CALL setInstuctorName PASS instructorName as argument

        ASSIGN value of startDate to instance variable startDate of the
        class

        ASSIGN value of completionDate to instance variable completionDate
        of the class

        ASSIGN value of examDate to instance variable examDate of the
        class
        SET value of isRegistered to true
        CALL showTempDialogBox(PASS “Registering..” as argument) from
        INGCollege class
    ELSE
        SHOW ERROR DIALOG "The course has already been registered"
    END IF
END METHOD
```

```
        DEFINE METHOD remove
            IF isRemoved
                SHOW ERROR DIALOG "The course has already been removed"
            ELSE
                CALL from super class setCourseLeader PASS empty String "" as
                argument
                SET value of startDate to empty String ""
                SET value of completionDate to empty String ""
                SET value of examDate to empty String ""
                CALL showTempDialogBox(PASS "Removing.." as argument) from
                INGCollege class
                SET value of isRegestered to false
                SET value of isRemoved to true
            END IF
        END METHOD

END CLASS
```

INGCollege Class

```
CREATE CLASS INGCollege

    DECLARE private instance variable frame of type JFrame

    DECLARE private instance variables academicPanel and nonAcademicPanel of type JPanel

    DECLARE private instance constant eventHandler of type EventHandler and INITIALIZE it to new
    instance of EventHandler class

    DECLARE private constant List course that carries objects of Course class and its    sub classes
    and INITIALIZE it to new instance of ArrayList class

    DECLARE variable academicButton and nonAcademicButton of type JButton

    DECLARE constant textFields of type ArrayList that stores JTextField objects

    DECLARE variable ingCollege of type INGCollege
```

```
DEFINE MAIN METHOD

    INITIALIZE ingCollege to new instance of INGCollege class

END MAIN METHOD



CREATE CONSTRUCTOR INGCollege

    CREATE LOCAL CLASS Panel

        DEFINE METHOD setUpNonAcademicPanel(TAKE parameter p of type JPanel)

            CALL addSwitcher(PASS p as argument)



            CALL setLabel(title="Non Academic Course", bounds=250,0,800,100, font size = 30) for
            panel p

            CALL setLabel(title="CourseID", bounds=20,90,70,20, font size = 15) for panel p

            CALL setLabel(title="Course Name", bounds=20,140,125,20, font size = 15) for panel p

            CALL setLabel(title="Instructor Name", bounds=400,90,125,20, font size = 15) for panel
            p

            CALL setLabel(title="Duration", bounds=400,140,130,20, font size = 15) for panel p

            CALL setLabel(title="prerequisite", bounds=20,195,125,20, font size = 15) for panel p
```

CALL setLabel(**title**="Duration", **bounds**=400,290,130,20, **font size** = 15) for panel *p*

CALL setTextField(**x**=180, **y**=85, **width**=170) for panel *p*

CALL setTextField(**x**=180, **y**=135, **width**=170) for panel *p*

CALL setTextField(**x**=535, **y**=85, **width**=160) for panel *p*

CALL setTextField(**x**=535, **y**=135, **width**=160) for panel *p*

CALL setTextField(**x**=180, **y**=190, **width**=170) for panel *p*

CALL setTextField(**x**=180, **y**=240, **width**=170) for panel *p*

CALL setTextField(**x**=535, **y**=185, **width**=160) for panel *p*

CALL setTextField(**x**=535, **y**=285, **width**=160) for panel *p*

CALL setTextField(**x**=535, **y**=235, **width**=160) for panel *p*

CALL setButton(**title**="Remove", **x**=290, **y**=330, **height**=130) for panel *p*

CALL setButton(**title**="Add Non Academic Course", **x**=5, **y**=380, **height**=250) for panel *p*

CALL setButton(**title**="Display Non Academic Course", **x**=5, **y**=330, **height**=250) for panel *p*

```
    CALL setButton(title="Clear", x=465, y=330, height=260) for panel p

    CALL setButton(title="Register Non Academic Course", x=465, y=380,
    height=260) for panel p

END METHOD



DEFINE METHOD setUpAcademicPanel(TAKE parameter p of type JPanel)

    CALL addSwitcher(PASS p as argument)



    CALL setLabel(title=Academic Course", bounds=250, 0, 800, 100, font size=30) for
    panel p

    CALL setLabel(title="CourseID:", bounds=20, 90, 70, 20, font size=15) for panel p

    CALL setLabel(title="Duration:", bounds=20, 140, 70, 20, font size=15) for panel p

    CALL setLabel(title="Completion Date:", bounds=390, 290, 130, 20, font size=15) for
    panel p

    CALL setLabel(title="Level:", bounds=400, 190, 70, 20, font size=15) for panel p

    CALL setLabel(title="Course Name:", bounds=400, 90, 125, 20, font size=15) for panel
    p
```

CALL setLabel(**title**="No. of Assessments:", **bounds**=370, 140, 185, 20, **font size**=15) for panel *p*

CALL setLabel(**title**="Start Date:", **bounds**=400, 240, 125, 20, **font size**=15) for panel *p*

CALL setLabel(**title**="Credit:", **bounds**=20, 240, 125, 20, **font size**=15) for panel *p*

CALL setLabel(**title**="Lecturer Name:", **bounds**=20, 190, 125, 20, **font size**=15) for panel *p*

CALL setLabel(**title**="Course Leader:", **bounds**=20, 290, 125, 20, **font size**=15) for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=180, **y**=85, **width**=170) for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=535, **y**=85, **width**=160 for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=180, **y**=135, **width**=170) for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=535, **y**=190, **width**=160) for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=535, **y**=285, **width**=160) for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=535, **y**=235, **width**=160) for panel *p*

CALL setTextField(**PASS** argument *p*, **x**=180, **y**=235, **width**=170) for panel *p*

```
    CALL setTextField(PASS argument p, x=535, y=135, width=160) for panel p

    CALL setTextField(PASS argument p, x=180, y=190, width=170) for panel p

    CALL setTextField(PASS argument p, x=180, y=235, width=170) for panel p



    CALL setButton(title="Add Academic Course", x=5, y=380, height=250) for panel p

    CALL setButton(title="Display Academic Course", x=5, y=330, height=250) for panel p

    CALL setButton(title="Clear", x=465, y=330, height=260) for panel p

    CALL setButton(title="Register Academic Course",x=465,y=380, height=260) for panel p

END METHOD



DEFINE METHOD addSwitcher(TAKE panel of type JPanel as parameter)

    INITIALIZE academicButton to new instance of JButton PASS "Academic" as  argument

    INITIALIZE nonAcademicButton to new instance of JButton PASS "Non Academic" as
    argument

    SET blue colored border in panel

    SET size of panel 740x500
    **SET** blue background for academicButton

    **SET** grey background for nonAcademicButton

    **SET** academicButton **bounds** = 350, 5, 100, 20

    **SET** nonAcademicButton **bounds** = 450, 5, 135, 20

    **ADD** action listener for academicButton and nonAcademicButton

    **ADD** academicButton and nonAcademicButton to *panel*

 **END METHOD**


**DEFINE METHOD** setLabel(**TAKE** *panel* of type JPanel, *text* of type String, *x*, *y*, *width*,
*height*, *fontSize* of type int as parameter)

    **CREATE** object of JLabel(PASS *text* as argument)

    **SET bounds** = *x*, *y*, *width*, *height* from parameter

    **ASSIGN** value of *fontSize* from parameter to font size of JLabel

    **IF** *text* contains "Academic" **SET** blue foreground color

    **ELSE** text cotains "Which"

        **SET** red foreground color
    END IF

    ADD this JLabel to panel

 END METHOD


 DEFINE METHOD setButton(TAKE panel of type JPanel, text of type String, x,    y, width of
 type int as parameter)

    CREATE object of JButton(PASS text as argument)

    SET bounds = x, y, width from parameter, 30

    ADD action listener

    ADD button JButton to panel

 END METHOD


 DEFINE METHOD setTextField(TAKE panel of type JPanel, x, y, width of type int as
 parameter)

    CREATE object of JTextField

    SET bounds = x, y, width from parameter, 25

    ADD action listener
```

```
        ADD this JTextField to panel

      END METHOD

    END LOCAL CLASS



    INITIALIZE frame variable to new instance of JFrame class(PASS "Course    Registration" as
argument)

    INITIALIZE frame variable to new instance of JFrame class(PASS "Course    Registration" as
    argument)

    INITIALIZE nonAcademicPanel variable to new instance of JPanel class PASS null as
    argument)

    CALL setUpAcademicPanel(PASS academicPanel as argument)

    CALL setUpNonAcademicPanel(PASS nonAcademicPanel as argument)



    SET size of frame 740x500

    ADD academicPanel to frame

    SET blue colored border in nonAcademicPanel

    SET size of nonAcademicPanel 740x500
```

```
        ADD windowListener to frame PASS eventHandler as argument

        MAKE frame visible

    END CONSTRUCTOR



    DEFINE METHOD getFrame

        RETURN data of frame variable

    END METHOD



DEFINE method showTempDialogBox(TAKE message of type String as parameter){

        CREATE message dialog with message from parameter as text

        CREATE constant w and set its value as 150

        CREATE constant h and set its value as 30

        SET bounds(x = X axis of frame, width of frame/2)-(w/2),

        y = X axis of frame, height of frame-h, width = w, height = h);

        SET modality type to modeless
        MAKE dialog visible

        TRY

            SLEEP for 0.5 seconds

        END TRY

          MAKE frame invisible

    END METHOD



    CREATE INNER CLASS EventHandler that extends WindowAdapter and implements
ActionListener

        DEFINE METHOD getText(TAKE index of type int as parameter)

            RETURN text of index-th object from textFields

        END METHOD

        DEFINE METHOD windowClosing(TAKE e of type WindowEvent as parameter)

            SHOW message dialog "Thank you for trying"

            TERMINATE the program

        END METHOD
```

```
DEFINE METHOD actionPerformed(TAKE e of type ActionEvent as parameter)
    CASE OF "getActionCommand of e"
        "Clear" : SET text of all elements of textFields ArrayList to empty String ""
        "Add Academic Course" : CALL METHOD addAcademicCourse
        "Add Non Academic Course" : CALL METHOD addNonAcademicCourse
        "Display Academic Courses" :
            FOR all elements e in courses ArrayList
                IF e is AcademicCourse
                    CAST e to AcademicCourse and store in nac
                    CALL METHOD display from nac
                END IF
            END FOR
        END
```

```
"Display Non Academic Courses" :
    FOR all elements e in courses ArrayList
        IF e is NonAcademicCourse
            CAST e to NonAcademicCourse and store in nac
            CALL METHOD display from nac
        END IF
    END FOR
END
"Remove" : CALL METHOD removeNonAcademicCourse
"Register Academic Course" : CALL METHOD registerAcademicCourse
"Register Non Academic Course" : CALL METHOD registerNonAcademicCourse
"Academic" :
  HIDE nonAcademicPanel from frame
  SHOW academicPanel to frame
END
```

```
        "Non Academic" :
           HIDE academicPanel from frame
           SHOW nonAcademicPanel to frame
        END
     END CASE
  END METHOD


  DEFINE METHOD showParseError(TAKE e of type Exception as parameter)
     DECLARE variable log and INITIALIZE by message of e
       IF e is NumberFormatException
            PREPEND "Please input valid integer" to log
       END IF
  END METHOD


  DECLARE variable removeNonAcademic of type NonAcademicCourse
```

```
DEFINE METHOD removeNonAcademicCourse

   DECLARE constant courseID of type String

   CALL getText(PASS 10 as argument) and store result in courseID



     FOR all elements c in courses ArrayList

          IF c NonAcademicCourse AND courseID is equal to courseID variable

                  CAST c to NonAcademicCourse and ASSIGN the value to
                  removeNonAcademic variable

                  REMOVE c from courses ArrayList

          END IF

     END FOR

   IF removeNonAcademic does not have null value

      CALL remove method from removeNonAcademic object



      ELSE CALL showTempDialogBox(PASS "Not added yet") as argument

   END IF
END METHOD


DEFINE METHOD addAcademicCourse

     TRY

          DECLARE constant courseID of type String

          CALL getText(PASS 0 as argument) and STORE result in courseID

          DECLARE constant courseName of type String

          CALL getText(PASS 1 as argument) and STORE result in courseName

          DECLARE constant duration of type int

          CALL getText(PASS 2 as argument) and STORE result in duration

          DECLARE constant level of type String

          CALL getText(PASS 3 as argument) and STORE result in level

          DECLARE constant credit of type int

          CALL parseInt(PASS getText(PASS 6 as argument) as argument) and STORE result
          in credit

          DECLARE constant noOfAssessments of type int
```

```
            CALL parseInt(PASS getText(PASS 7 as argument) as argument) and STORE result
            in noOfAssessments)

            DECLARE variable course AND INITIALIZE it by new instance of
            AcademicCourse(PASS courseID, courseName, duration, level, credit,
            noOfAssessments as arguments)

            add course to courses ArrayList

        CATCH Exception as e

            CALL showParseError(PASS e as argument)

        END CATCH

      END TRY

  END METHOD

  DEFINE METHOD addNonAcademicCourse

      TRY

            DECLARE constant courseID of type String

            CALL getText(PASS 10 as argument) and STORE result in courseID
            DECLARE constant courseName of type String

            CALL getText(PASS 11 as argument) and STORE result in courseName

            DECLARE constant duration of type int

            CALL getText(PASS 13 as argument) and STORE result in duration

            DECLARE prerequisite level of type String

            CALL getText(PASS 14 as argument) and STORE result in prerequisite



            DECLARE variable course AND INITIALIZE it by new instance of

            AcademicCourse(PASS courseID, courseName, duration, prerequisite as

            arguments)

            ADD course to courses ArrayList

      CATCH Exception as e

            CALL showParseError(PASS e as argument)

      END CATCH



      END TRY
```

```
END METHOD



DEFINE METHOD addCourse(TAKE course of type Course as parameter)

    FOR all elements e in textFields

        IF text of e is blank

            SHOW message dialog "Please make sure you have filled all values"

        END IF

    END FOR



  DECLARE variable text of type String

  IF course is Academic Course

     THEN ASSIGN value of getText(PASS 0 as argument) to text

     ELSE ASSIGN value of getText(PASS 10 as argument) to text

  END IF

 FOR all elements c in courses ArrayList

     IF course id of c object is equal to text
```

```
            SHOW WARNING MESSAGE DIALOG "The course has already been added"

            REMOVE c object from courses ArrayList

            END METHOD

         END IF

     END FOR



    ADD course to courses ArrayList

    CALL showTempDialogBox(PASS "Adding.." as argument)

END METHOD



DEFINE METHOD registerAcademicCourse

   DECLARE constant courseLeader of type String

   CALL getText(PASS 9 as argument) and STORE result in courseLeader

   DECLARE constant lecturerName of type String

   CALL getText(PASS 8 as argument) and STORE result in lecturerName

   DECLARE constant startingDate of type String
```

```
    CALL getText(PASS 5 as argument) and STORE result in startingDate

    DECLARE constant completionDate of type String

    CALL getText(PASS 4 as argument) and STORE result in completionDate

      FOR all elements c in courses ArrayList

          IF c is AcademicCourse AND its getText(PASS 0 as argument) is equal to
          getCourseID of c object

                DECLARE variable ac of type AcademicCourse, CAST c to AcademicCourse
                and store result in ac

                CALL register method of ac object(PASS courseLeader, lecturerName,
                startingDate, completionDate as arguments)

          END IF

      END FOR

END METHOD


DEFINE METHOD registerNonAcademicCourse

    DECLARE constant courseLeader of type String

    CALL getText(PASS 15 as argument) and STORE result in courseLeader
```

```
        DECLARE constant courseName and startingDate of type String

        CALL getText(PASS 11 as argument) and STORE result in courseName

        CALL getText(PASS 16 as argument) and STORE result in startingDate

        DECLARE constant completionDate of type String

        CALL getText(PASS 18 as argument) and STORE result in completionDate

         FOR all elements c in courses ArrayList

            IF c is NonAcademicCourse AND getText(PASS 10 as argument) is equal to
            getCourseID of c object

                DECLARE variable nac of type NonAcademicCourse, CAST c to
                AcademicCourse and store result in nac

                CALL register method of nac object(PASS courseLeader, courseName,
                startingDate, completionDate as arguments)

            END IF

         END FOR

      END METHOD

   END INNER CLASS

END CLASS
```

3. DESCCRIPTION ABOUT METHODS

Course class

- **getCourseID()**

  Returns the value stored in private variable courseID which is of type String from Course class.
- **getCourseName()**

  Returns the value stored in private variable courseName which is of type String from course class.
- **getCourseLeader()**

  Returns the value stored in private variable courseLeader which is of type String from Course class.
- **getDuration()**

  Returns the value stored in private variable duration of type int from Course class.
- **setCourseLeader()**

  Takes a String as parameter and assigns the value stored in parameter to private variable courseLeader of Course class.

- **display()**

  Displays the value of courseID, courseName, duration variables of that class. If courseLeader variable of type String has a value, it displays value stored in that variable to the console too.

AcademicCourse class

- **getLecturerName()**

  Returns the value stored in private variable lecturerName which is of type String from AcademicCourse class.

- **getLevel()**

  Returns the value stored in private variable level which is of type String from AcademicCourse class.

- **getCredit()**

  Method returns the value stored in private variable credit which is of type String from AcademicCourse class.

- **getStartingDate()**

  Returns the value stored in private variable startingDate which is of type String from AcademicCourse class.

- **getCompletionDate()**

  Returns the value stored in private variable completionDate which is of type String from AcademicCourse class.

- **getRegistered()**

  Returns the value stored in private variable isRegistered which is of type boolean from AcademicCourse class.

- **getNoOfAssessments()**

  Returns the value stored in private variable noOfAssessments which is of type int from AcademicCourse class.

- **setLecturerName(String)**

  Takes a String as parameter and assigns the value stored in parameter to private variable lecturerName of AcademicCourse class.

- **setNoOfAssessments(int)**

  Takes a int as parameter and assigns the value stored in parameter to private variable noOfAssessments of Course class.

- **register(String, String, String, String)**

  Takes courseLeader, lecturerName, startingDate, completionDate of type String as parameters. If the AcademicCourse has already been registered, it displays the value stored in instructorName, startingDate, completionDate from class to the console. Else, it sets the values of respective variables as the values in its parameter, sets isRegistered to true and courseRemovedStatus variable to false.

- **display()**

  Calls the display method from its superclass(Course class). If the AcademicCourse has already been registered, also displays values stored in lecturerName, level, startingDate, completionDate, noOfAssessments to the console with appropriate message.

NonAcademicCourse class

- **getInstructorName()**

  Returns the value stored in private variable instructorName which is of type String from NonAcademicCourse class.

- **getDuration()**

  Returns the value stored in private variable duration which is of type int from NonAcademicCourse class.

- **getStartDate()**

Returns the value stored in private variable startDate which is of type String from NonAcademicCourse class.

- **getCompletionDate()**

  Returns the value stored in private variable startingDate which is of type String from NonAcademicCourse class.

- **getExamDate()**

  Returns the value stored in private variable examDate which is of type String from NonAcademicCourse class.

- **getPrerequisite()**

  Returns the value stored in private variable prerequisite which is of type String from NonAcademicCourse class.

- **getRemoved()**

  Returns the value stored in private variable isRemoved which is of type boolean from NonAcademicCourse class.

- **setRemoved()**

  Takes a boolean as parameter and assigns the value stored in parameter to private variable isRemoved of NonAcademicCourse class.

- **setInstructorName()**

  Takes a String as parameter and if the NonAcademicCourse has not been registered yet, assigns the value stored in parameter to private variable instructorName of NonAcademicCourse class. Else, displays suitable message to the console indicating that NonAcademicCourse has already been registered and it is not possible to change instructorName.

- **Register(String, String, String, String)**

  Takes courseLeader, instructorName, startDate, completionDate, examDate of type String as parameters and if the NonAcademicCourse has not been registered yet, calls setInstructorName method, assigns the value stored in parameter to private variables startDate, completionDate, examDaate of NonAcademicCourse class and also changes the value of isRegistered to true. Else, displays suitable message to the console indicating that NonAcademicCourse has already been registered.

- **remove()**

  Displays that the course has already been removed if the value stored in isRemoved variable is true. Else, calls setCourseLeader method with empty String “” as argument from superclass(Course class), sets the values of startDate, completionDate, examDate to empty String “” and also changes value of isRegistered to false and isRemoved variable to true.

- **display()**

  Calls method to display values of getCourseID, getCourseName, getDuration from its superclass (Course class). If the NonAcademicCourse has already been registered, it also displays the values of instructorName, startDate, completionDate, examDate.

INGCollege class

- **main(String[] args)**

  Assigns new instance of INGCollege class to ingCollege variable and initializes the program

- **getInstance()**

  Returns current instance of INGCollege class.

- **setUpNonAcademicPanel(JPanel)**

  Sets up style and components in non-academic panel.

- **setUpAcademicPanel(JPanel)**

  Sets up style and components in academic panel.

- **addSwitcher(JPanel)**

  Adds buttons to switch from/to academic and non-academic panels along with a JLabel for question.

- **setLabel(JPanel, String, int, int, int, int, int)**

  Sets up JLabel as per parameters provided and adds to required JPanel.

- **setButton(JPanel, String, int, int, int)**

  Sets up JButton as per parameters provided and adds to required JPanel.

- **setTextField(JPanel, int, int, int)**

  Sets up JTextField as per parameters provided and adds to required JTextField.

- **getFrame()**

  Returns the current JFrame instance.

- **showTempDialogBox(String)**

  Shows a message dialog with text from parameters that closes itself after 0.5 seconds.

- **getText(int)**

  Returns text from text fields as per the provided index.

- **windowClosing(WindowEvent)**

  Shows a message dialog when the program is closed, then terminates the program.

- **actionPerformed(ActionEvent)**

  Sets actions for different components when they are pressed

  - **Add Academic Course**

    The input values of Course ID, Course Name, Duration, Level, Credit and Number of Assessments are used to create a new object of type NonAcademicCourse which is added to an array list of Course class.

  - **Add Non Academic Course**

The input values of Course ID, Course Name, Duration, and Prerequisites are used to create a new object of type NonAcademicCourse which is added to an array list of Course class.

- **Register Academic Course**

  The input value of Course ID is compared to the existing Course ID, and if valid Course ID has been entered, it is used to register the academic course from the list. The register method from AcademicCourse class is called by casting Course object to AcademicCourse if possible.

- **Register Non Academic Course**

  The input value of Course ID is compared to the existing Course ID, and if valid Course ID has been entered, it is used to register the academic course from the list. The register method from NonAcademicCourse is called by casting Course object to NonAcademicCourse if possible.

- **Remove**

  The input value of the Course ID is compared to the existing Course ID in the list. If a valid value has been entered, it is used to remove the specified non-academic course from array list of Course. The remove method from NonAcademicCourse class is called here by casting Course object to NonAcademicCourse if possible.

- **Display Academic Course**

  The information related to AcademicCourse class is displayed if the object has been created and stored in array list of Course class. The display method from AcademicCourse class is called here by casting Course object to AcadeicCourse if possible.

- **Display Non Academic Course**

  The information related to NonAcademicCourse class is displayed if the object has been created and stored in array list of Course class. The display method from

NonAcademicCourse class is called here by casting Course object to AcademicCourse if possible.

- **Clear**

  The inputted values of all text fields are cleared.

- **showParseError(Exception)**

  Shows error message dialog to not disrupt the flow of program when an exception occurs while parsing String to int and creating object.

- **removeNonAcademicCourse()**

  Creates an object of NonAcademicCourse class and calls addCourse method if course id does not already exist.

- **addAcademicCourse()**

  Creates an object of AcademicCourse class and calls addCourse method if course id does not already exist.

- **addCourse(Course)**

  Checks if course id of subclasses of Course is equal to inputted course id. If course id already exists, shows appropriate message, else adds the object to ArrayList.

- **registerAcademicCourse()**

  Calls register method from AcademicCourse class if a valid course id already exists in ArrayList.

- **registerNonAcademicCourse()**

  Calls register method from AcademicCourse class if a valid course id already exists in ArrayList.

4. Tests

Test 1

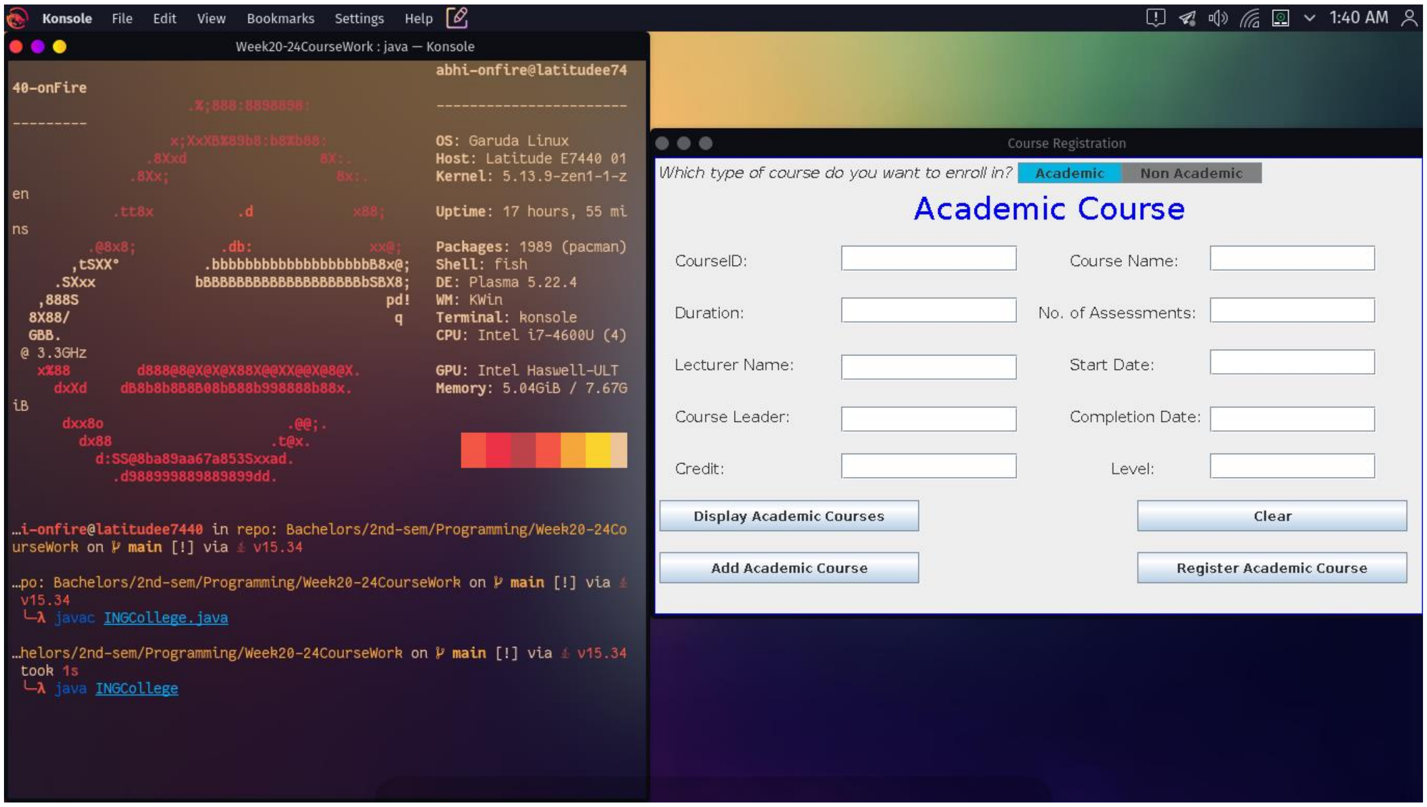
Checking if program can be compiled and run using command prompt

Testing

Table 1 - Test 1

Objective	To check if program can be compiled and run using
Action	• javac INGCollege.java(with its absolute path) in cmd to compile the class • java INGCollege to run the compiled program(.class file)
Expected Result	A Graphical User Interface should appear on screen
Actual Result	A Graphical User Interface appeared on screen
Conclusion	The test was successful

Evidence



```
                                                    abhi-onfire@latitudee74
40-onFire
                      .%;888:8898898:               -----------------------
---------
                   x;XxXBX89b8:b8Xb88:              OS: Garuda Linux
                .8Xxd                 8X:.          Host: Latitude E7440 01
              .8Xx;                    8x:.         Kernel: 5.13.9-zen1-1-z
en
            .tt8x          .d            x88;       Uptime: 17 hours, 55 mi
ns
          .@8x8;          .db:              xx@;    Packages: 1989 (pacman)
        ,tSXX°          .bbbbbbbbbbbbbbbbbbbbB8x@;  Shell: fish
      .SXxx            bBBBBBBBBBBBBBBBBBBBBbSBX8;  DE: Plasma 5.22.4
    ,888S                                     pd!   WM: KWin
   8X88/                                       q    Terminal: konsole
   GBB.                                             CPU: Intel i7-4600U (4)
 @ 3.3GHz
   x%88        d888@8@X@X@X88X@@XX@@X@8@X.          GPU: Intel Haswell-ULT
     dxXd    dB8b8b8B8B08bB88b998888b88x.           Memory: 5.04GiB / 7.67G
iB
      dxx8o                       .@@;.
        dx88                    .t@x.
          d:SS@8ba89aa67a853Sxxad.
            .d988999889889899dd.


…i-onfire@latitudee7440 in repo: Bachelors/2nd-sem/Programming/Week20-24Co
urseWork on  main [!] via  v15.34

…po: Bachelors/2nd-sem/Programming/Week20-24CourseWork on  main [!] via 
 v15.34
 └─λ javac INGCollege.java

…helors/2nd-sem/Programming/Week20-24CourseWork on  main [!] via  v15.34
 took 1s
 └─λ java INGCollege
```

Figure 2 – Running the program using Command Prompt

Test 2

Table 2 - Test 2

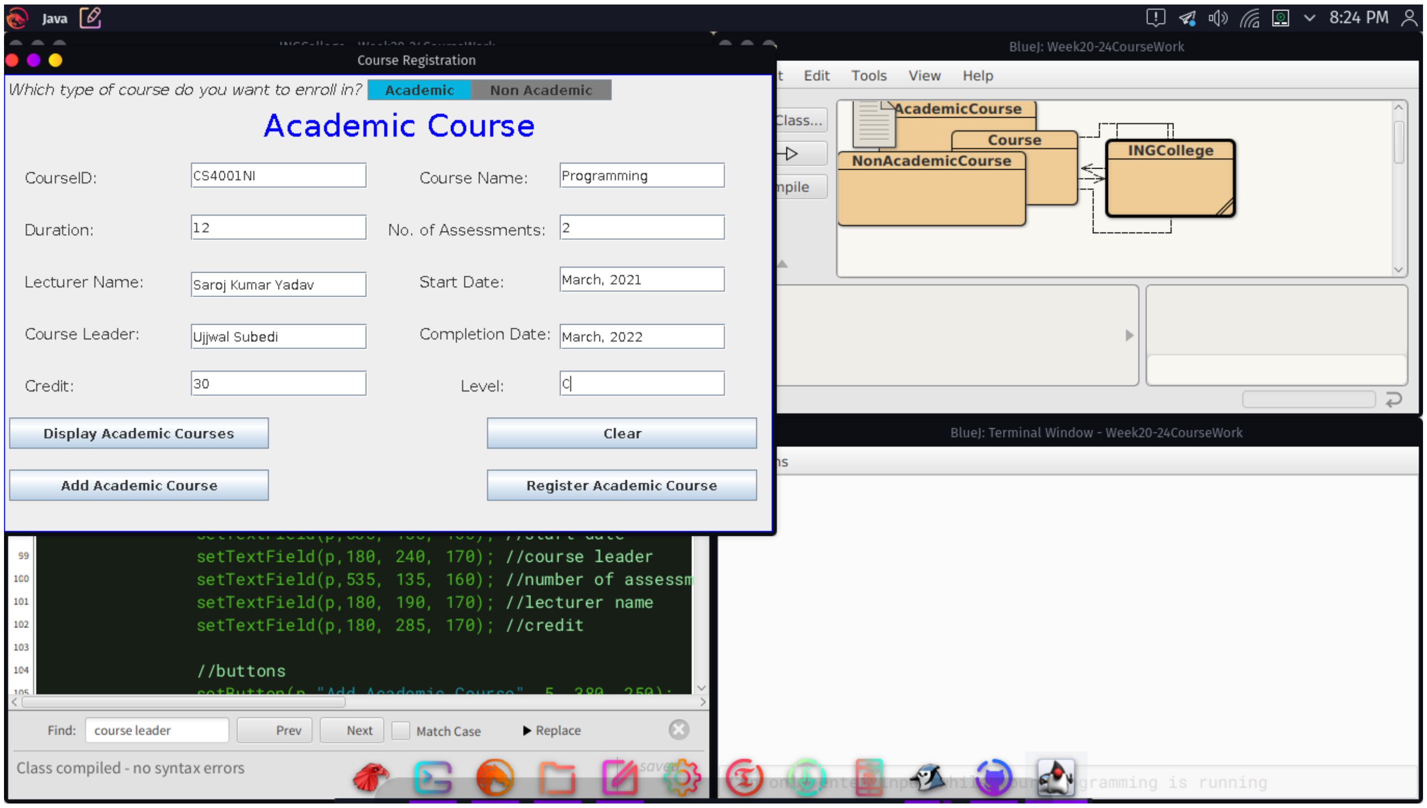
Objective	• To check if adding, registering academic course and adding, registering, removing non-academic course works
Action	• All text fields were filled with relevant data • Display Academic Courses button was pressed • Add Academic Course button was pressed • Display Academic Courses button was pressed again • Register Academic Course button was pressed • Display Academic Button was pressed • Bluej terminal was cleared • Panel was switched to that of non academic course registration • Display Non Academic Courses button was pressed • Add Non Academic Courses button was pressed • Display Non Academic Courses button was pressed again • Register Non Academic Course button was pressed • Display Non Academic Courses button was pressed again • Bluej terminal was cleared • Remove button was pressed • Display Non Academic Courses button was pressed again

Expected Result	**For Academic Course Registration** • Nothing should be displayed when pressing Display Academic Courses button initially • After pressing Add Academic Courses button and pressing Display Academic Courses button, Course ID, Course Name and Duration details should be shown in the terminal • After pressing Register Academic Courses button and pressing Display Academic Courses button, Course ID, Course Name, Duration, Course Leader, Lecturer Name, Level, Staring Date, Completion Date, Number of Assessments details should be shown in the terminal **For Non Academic Course Registration** • Nothing should be displayed when pressing Display Non Academic Courses button initially • After pressing Add Non Academic Courses button and pressing Display Academic Course button, Course ID, Course Name and Duration details should be shown in the terminal • After pressing Register Non Academic Courses button and pressing Display Non Academic Course button, Course ID, Course Name, Duration, Instructor Name, Staring Date, Completion Date, Exam Date details should be shown in the terminal

	• After clearing the terminal, pressing Remove button and pressing Display Non Academic Courses button, nothing should be displayed in the terminal
Actual Result	**For Academic Course Registration** • Nothing was displayed when pressing Display Academic Courses button initially • After pressing Add Academic Courses button and pressing Display Academic Courses button, Course ID, Course Name and Duration details was shown in the terminal • After pressing Register Academic Courses button and pressing Display Academic Courses button, Course ID, Course Name, Duration, Course Leader, Lecturer Name, Level, Staring Date, Completion Date, Number of Assessments details was shown in the terminal **For Non Academic Course Registration** • Nothing was displayed when pressing Display Non Academic Courses button initially • After pressing Add Non Academic Courses button and pressing Display Academic Course button, Course ID, Course Name and Duration details was shown in the terminal

	• After pressing Register Non Academic Courses button and pressing Display Non Academic Course button, Course ID, Course Name, Duration, Instructor Name, Staring Date, Completion Date, Exam Date details was shown in the terminal • After clearing the terminal, pressing Remove button and pressing Display Non Academic Courses button, nothing was displayed in the terminal
Conclusion	The test was successful

Evidence



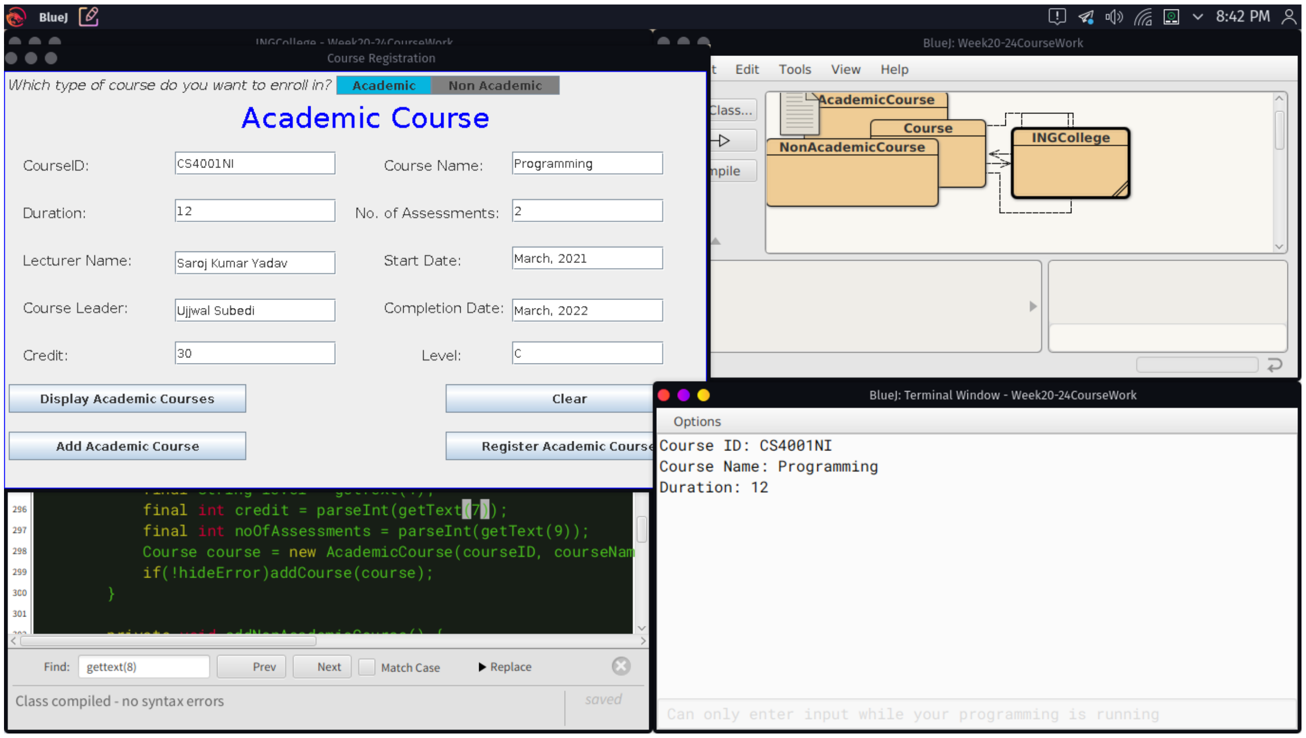
CourseID: CS4001NI
Course Name: Programming
Duration: 12
No. of Assessments: 2
Lecturer Name: Saroj Kumar Yadav
Start Date: March, 2021
Course Leader: Ujjwal Subedi
Completion Date: March, 2022
Credit: 30
Level: C

```
        setTextField(p,180, 240, 170); //course leader
        setTextField(p,535, 135, 160); //number of assessm
        setTextField(p,180, 190, 170); //lecturer name
        setTextField(p,180, 285, 170); //credit

        //buttons
```

Find: course leader | Prev | Next | Match Case | Replace

Class compiled - no syntax errors

Figure 3 - Academic Course before adding



Which type of course do you want to enroll in? Academic Non Academic

Academic Course

CourseID:	CS4001NI	Course Name:	Programming
Duration:	12	No. of Assessments:	2
Lecturer Name:	Saroj Kumar Yadav	Start Date:	March, 2021
Course Leader:	Ujjwal Subedi	Completion Date:	March, 2022
Credit:	30	Level:	C

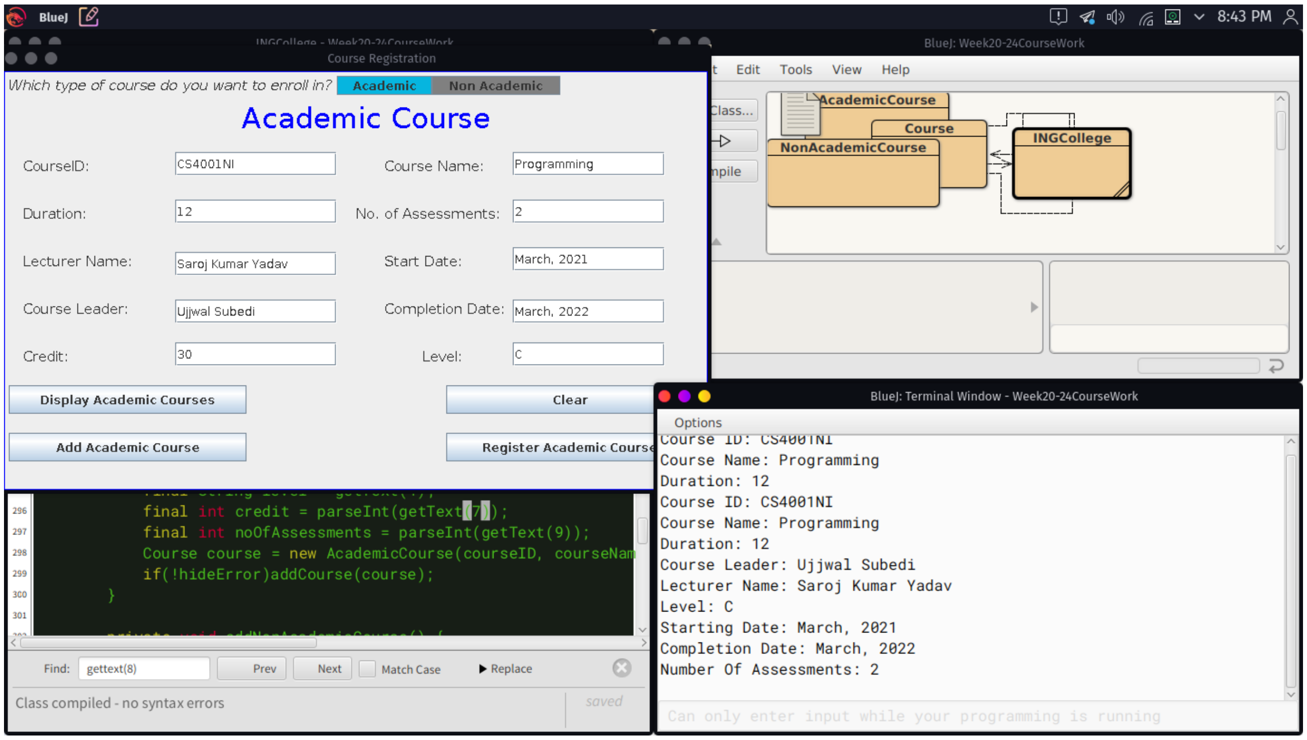
```
        final int credit = parseInt(getText(7));
        final int noOfAssessments = parseInt(getText(9));
        Course course = new AcademicCourse(courseID, courseNam
        if(!hideError)addCourse(course);
    }
```

Class compiled - no syntax errors

```
Course ID: CS4001NI
Course Name: Programming
Duration: 12
```

Can only enter input while your programming is running

Figure 4 - Academic Course after adding



Which type of course do you want to enroll in? Academic Non Academic

Academic Course

CourseID:	CS4001NI	Course Name:	Programming
Duration:	12	No. of Assessments:	2
Lecturer Name:	Saroj Kumar Yadav	Start Date:	March, 2021
Course Leader:	Ujjwal Subedi	Completion Date:	March, 2022
Credit:	30	Level:	C

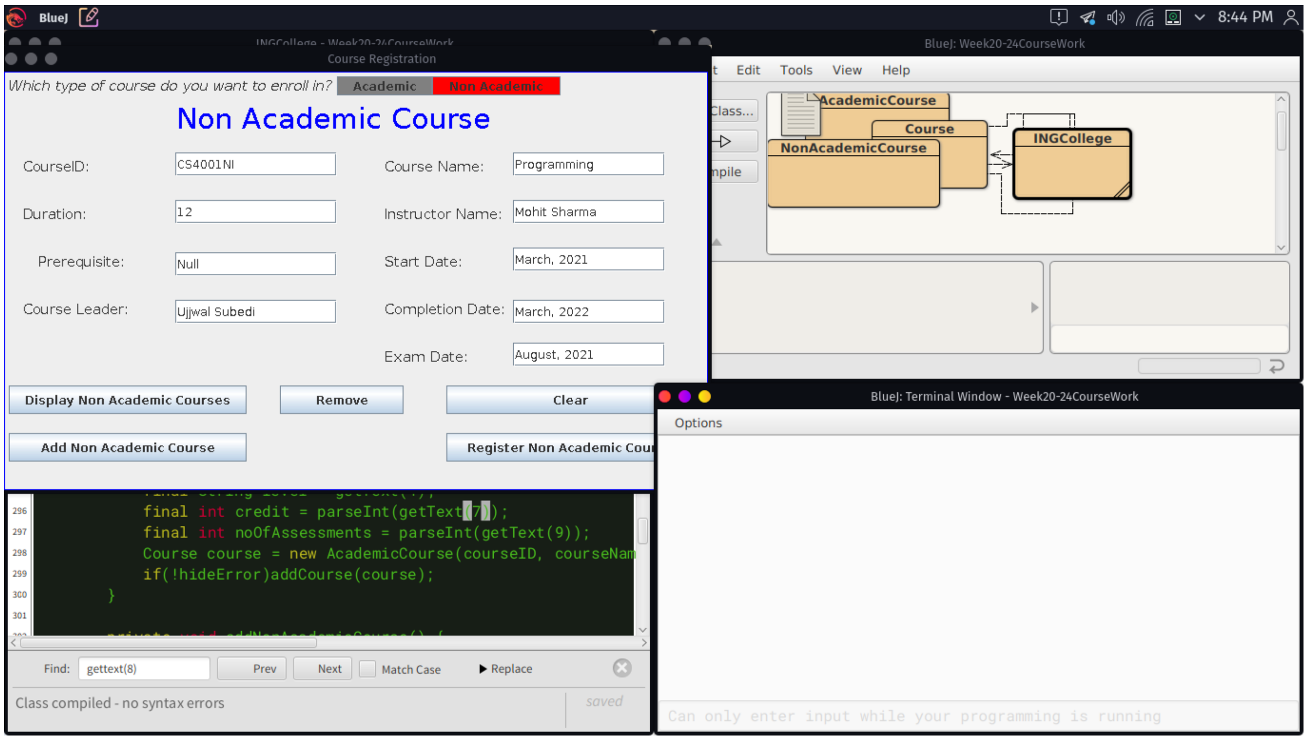
```
        final int credit = parseInt(getText(7));
        final int noOfAssessments = parseInt(getText(9));
        Course course = new AcademicCourse(courseID, courseNam
        if(!hideError)addCourse(course);
    }
```

Class compiled - no syntax errors

```
Course ID: CS4001NI
Course Name: Programming
Duration: 12
Course ID: CS4001NI
Course Name: Programming
Duration: 12
Course Leader: Ujjwal Subedi
Lecturer Name: Saroj Kumar Yadav
Level: C
Starting Date: March, 2021
Completion Date: March, 2022
Number Of Assessments: 2
```

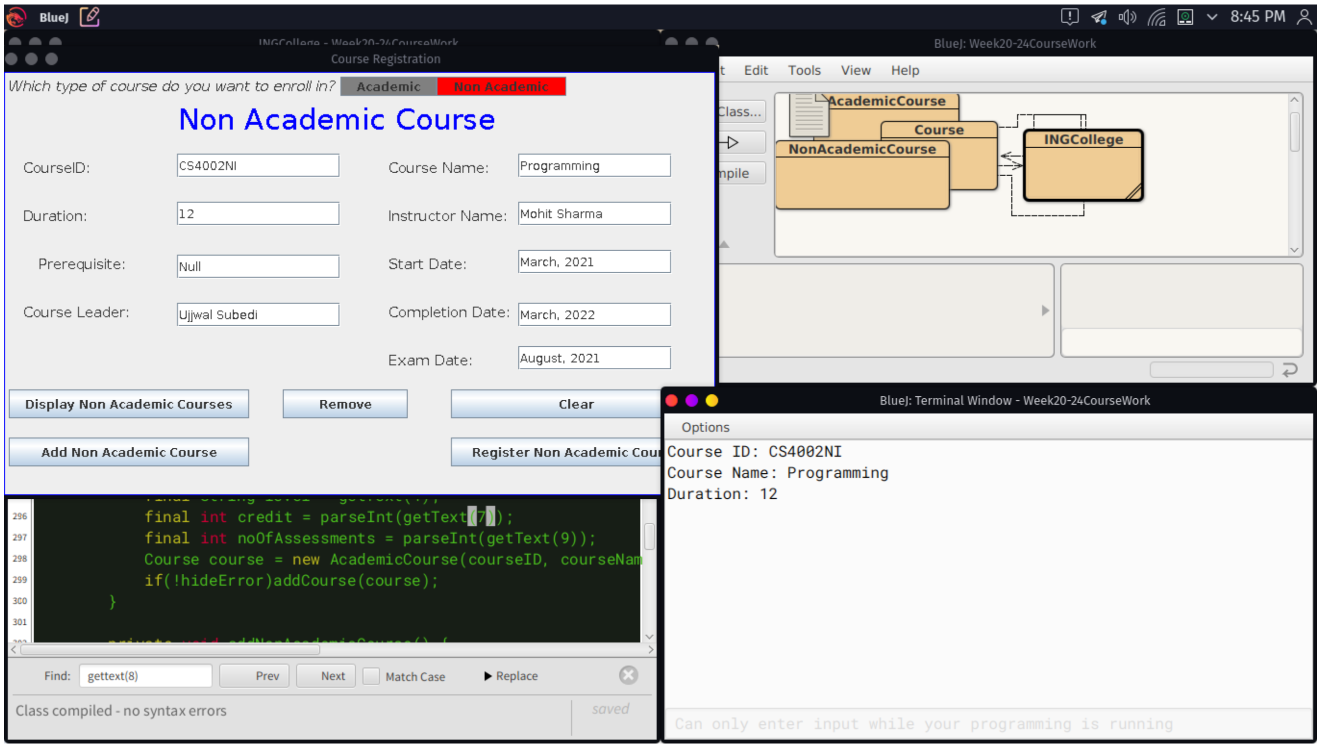
Can only enter input while your programming is running

Figure 5 - Academic Course after registering



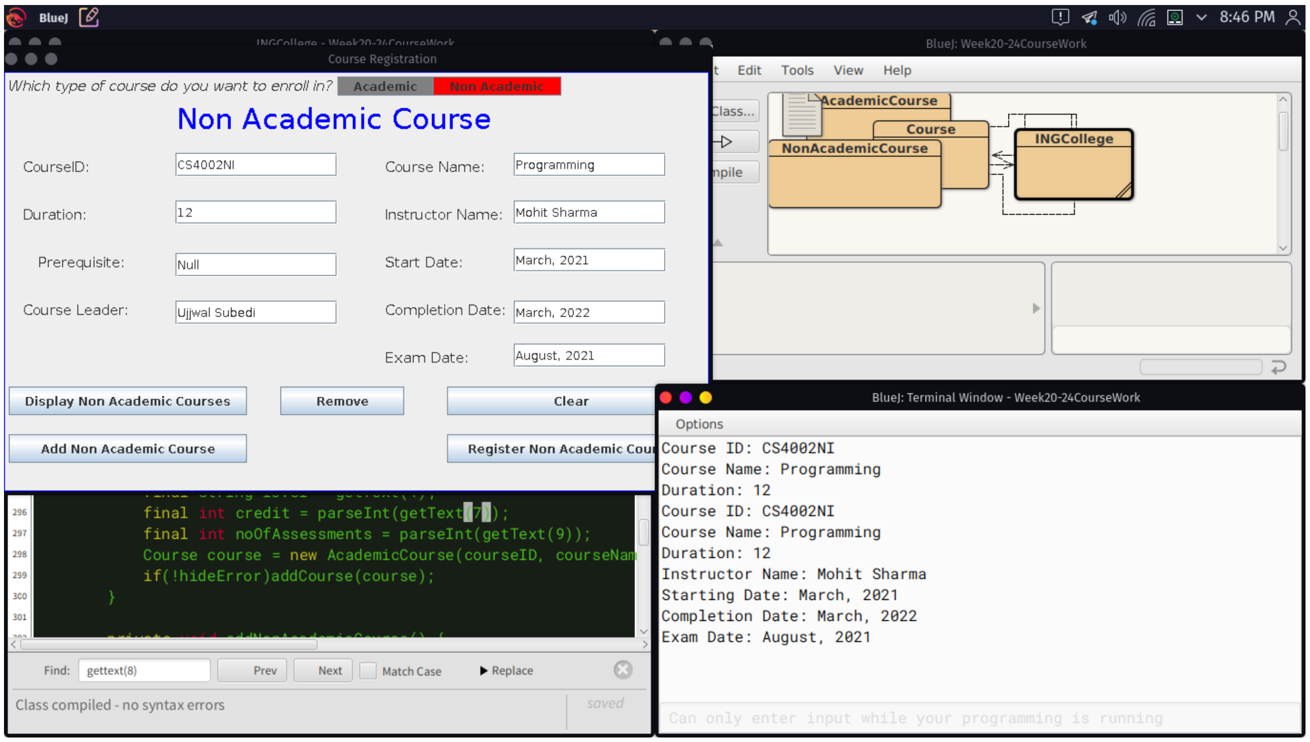
```
        final int credit = parseInt(getText(7));
        final int noOfAssessments = parseInt(getText(9));
        Course course = new AcademicCourse(courseID, courseNam
        if(!hideError)addCourse(course);
    }
```

Figure 6 - Non Academic Course before adding



```
Course ID: CS4002NI
Course Name: Programming
Duration: 12
```

Figure 7 - Non Academic Course after adding



Which type of course do you want to enroll in?

Non Academic Course

CourseID:	CS4002NI	Course Name:	Programming
Duration:	12	Instructor Name:	Mohit Sharma
Prerequisite:	Null	Start Date:	March, 2021
Course Leader:	Ujjwal Subedi	Completion Date:	March, 2022
		Exam Date:	August, 2021

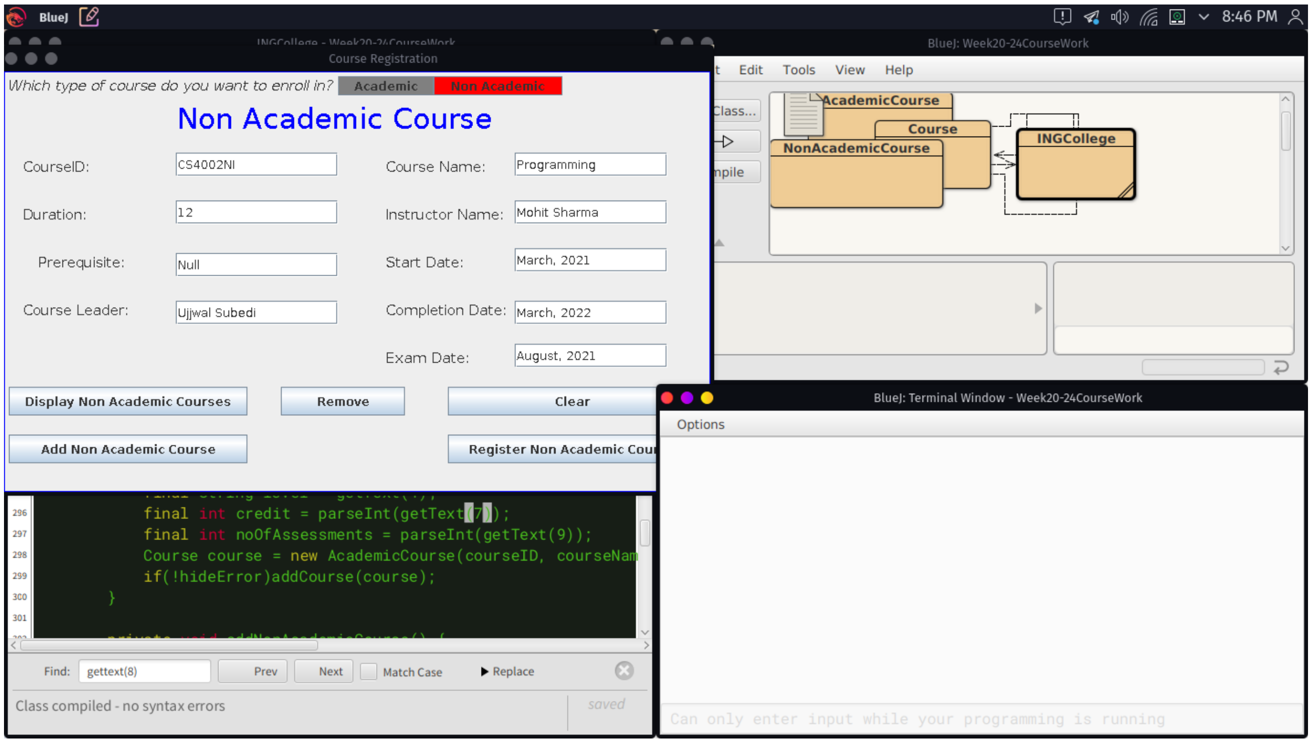
```
        final int credit = parseInt(getText(7));
        final int noOfAssessments = parseInt(getText(9));
        Course course = new AcademicCourse(courseID, courseNam
        if(!hideError)addCourse(course);
    }
```

Class compiled - no syntax errors

```
Course ID: CS4002NI
Course Name: Programming
Duration: 12
Course ID: CS4002NI
Course Name: Programming
Duration: 12
Instructor Name: Mohit Sharma
Starting Date: March, 2021
Completion Date: March, 2022
Exam Date: August, 2021
```

Can only enter input while your programming is running

Figure 8 - Non Academic Course after registering



Which type of course do you want to enroll in?

Non Academic Course

CourseID:	CS4002NI	Course Name:	Programming
Duration:	12	Instructor Name:	Mohit Sharma
Prerequisite:	Null	Start Date:	March, 2021
Course Leader:	Ujjwal Subedi	Completion Date:	March, 2022
		Exam Date:	August, 2021

```
        final int credit = parseInt(getText(7));
        final int noOfAssessments = parseInt(getText(9));
        Course course = new AcademicCourse(courseID, courseNam
        if(!hideError)addCourse(course);
    }
```

Class compiled - no syntax errors

Can only enter input while your programming is running

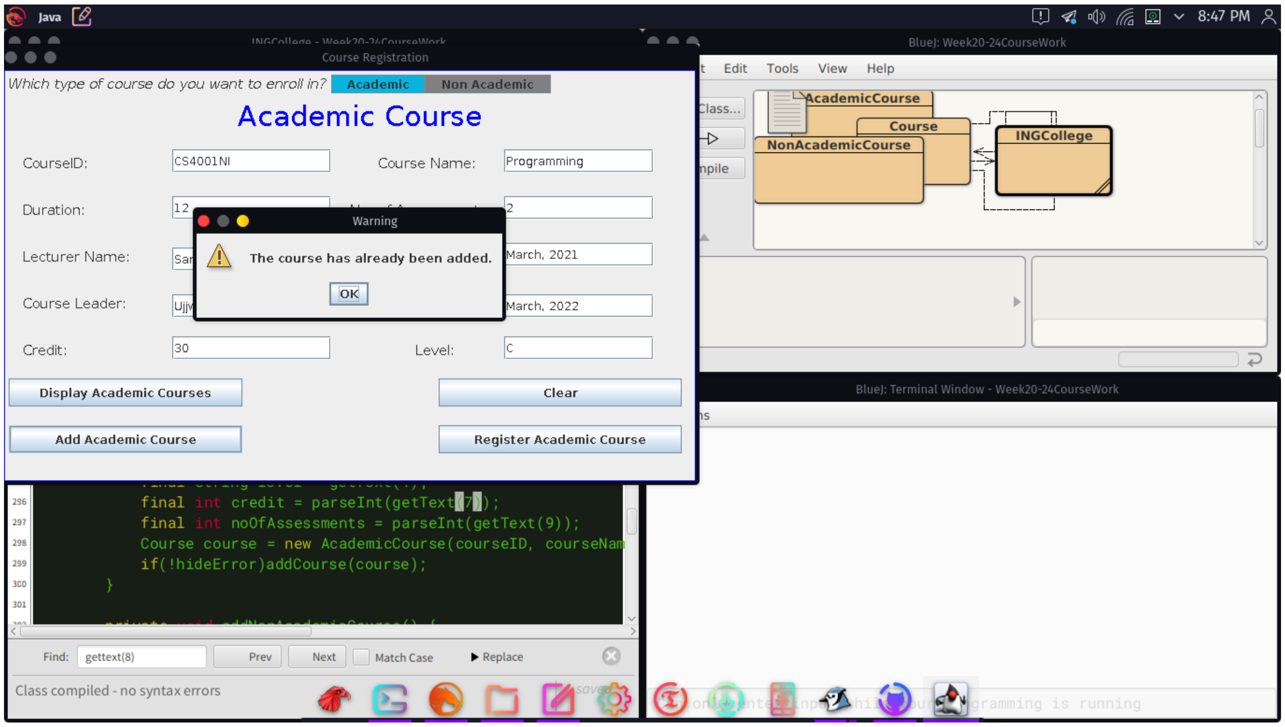
Figure 9 - Non Academic Course after removing

Test 3

Table 3 - Test 3

Objective	To test if appropriate dialog boxes appear when • Trying to add duplicate course id • Trying to register already registered code • Trying to remove non-academic course which is already removed
Action	• Details were filled in text fields • Add Academic Button was pressed twice with same data • Register Academic Button was pressed twice with same data • Panel was changed to that of non-academic course registration • Add Non Academic Course button was pressed twice with same data • Register Non Academic Course button was pressed twice with same data • Remove button was pressed twice with same data
Expected Result	Expected dialog boxes should appear when pressing buttons for the second time with duplicate data
Actual Result	Expected dialog boxes appeared when pressing buttons for the second time with duplicate data
Conclusion	The test was successful

Evidence



Academic Course

CourseID:	CS4001NI	Course Name:	Programming
Duration:	12		2
Lecturer Name:	Sar		March, 2021
Course Leader:	Ujjv		March, 2022
Credit:	30	Level:	C

Warning

The course has already been added.

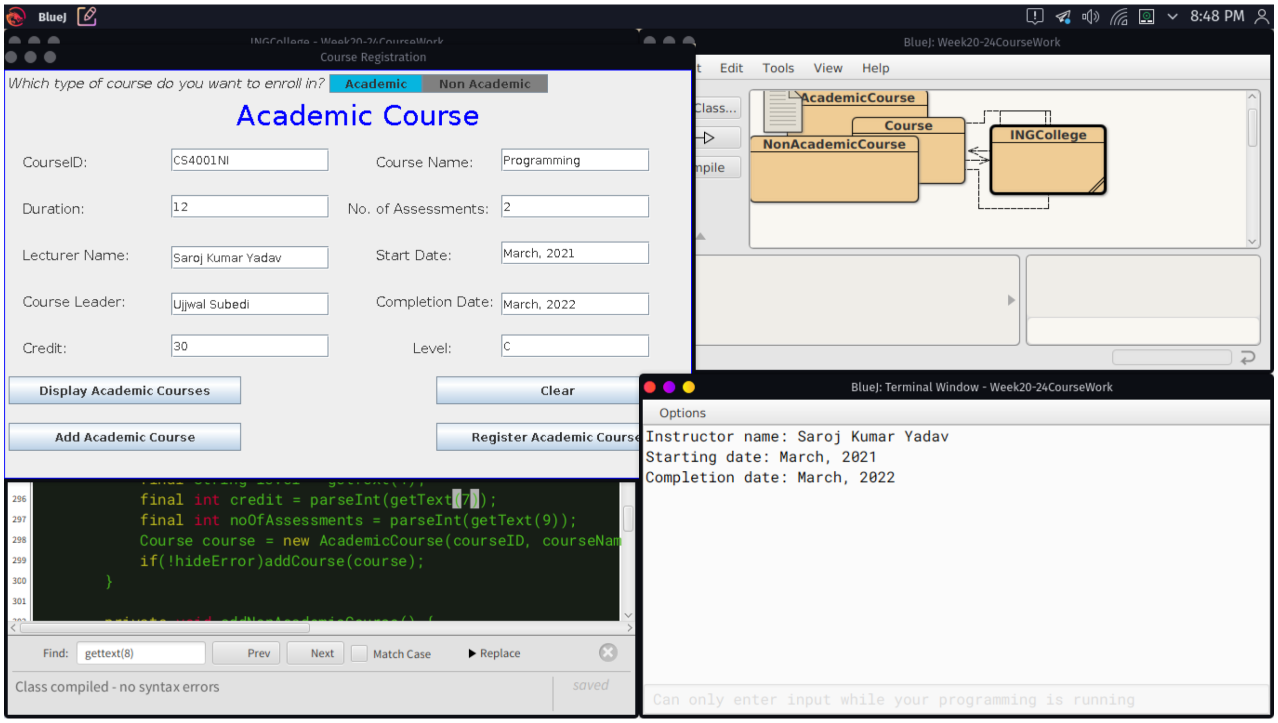
OK

```
            final int credit = parseInt(getText(7));
            final int noOfAssessments = parseInt(getText(9));
            Course course = new AcademicCourse(courseID, courseNam
            if(!hideError)addCourse(course);
        }
```

Find: gettext(8)

Class compiled - no syntax errors

Figure 10 - When trying to add duplicate AcademicCourse



Academic Course

CourseID:	CS4001NI	Course Name:	Programming
Duration:	12	No. of Assessments:	2
Lecturer Name:	Saroj Kumar Yadav	Start Date:	March, 2021
Course Leader:	Ujjwal Subedi	Completion Date:	March, 2022
Credit:	30	Level:	C

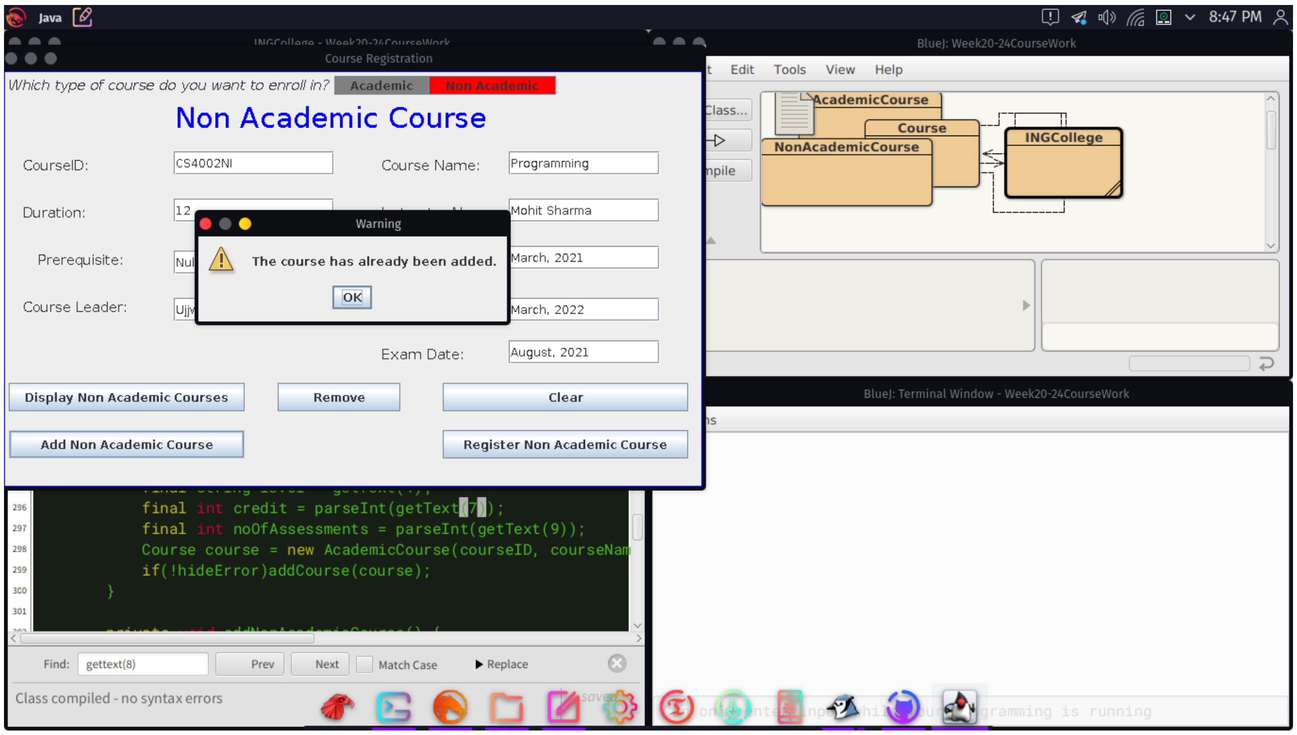
```
            final int credit = parseInt(getText(7));
            final int noOfAssessments = parseInt(getText(9));
            Course course = new AcademicCourse(courseID, courseNam
            if(!hideError)addCourse(course);
        }
```

Find: gettext(8)

Class compiled - no syntax errors

```
Instructor name: Saroj Kumar Yadav
Starting date: March, 2021
Completion date: March, 2022
```

Figure 11 - When trying to register duplicate AcademicCourse



Course Registration

Which type of course do you want to enroll in? Academic Non Academic

Non Academic Course

CourseID:	CS4002NI	Course Name:	Programming
Duration:	12		Mohit Sharma
Prerequisite:	Nul		March, 2021
Course Leader:	Ujjv		March, 2022
		Exam Date:	August, 2021

Warning

The course has already been added.

OK

Display Non Academic Courses | Remove | Clear

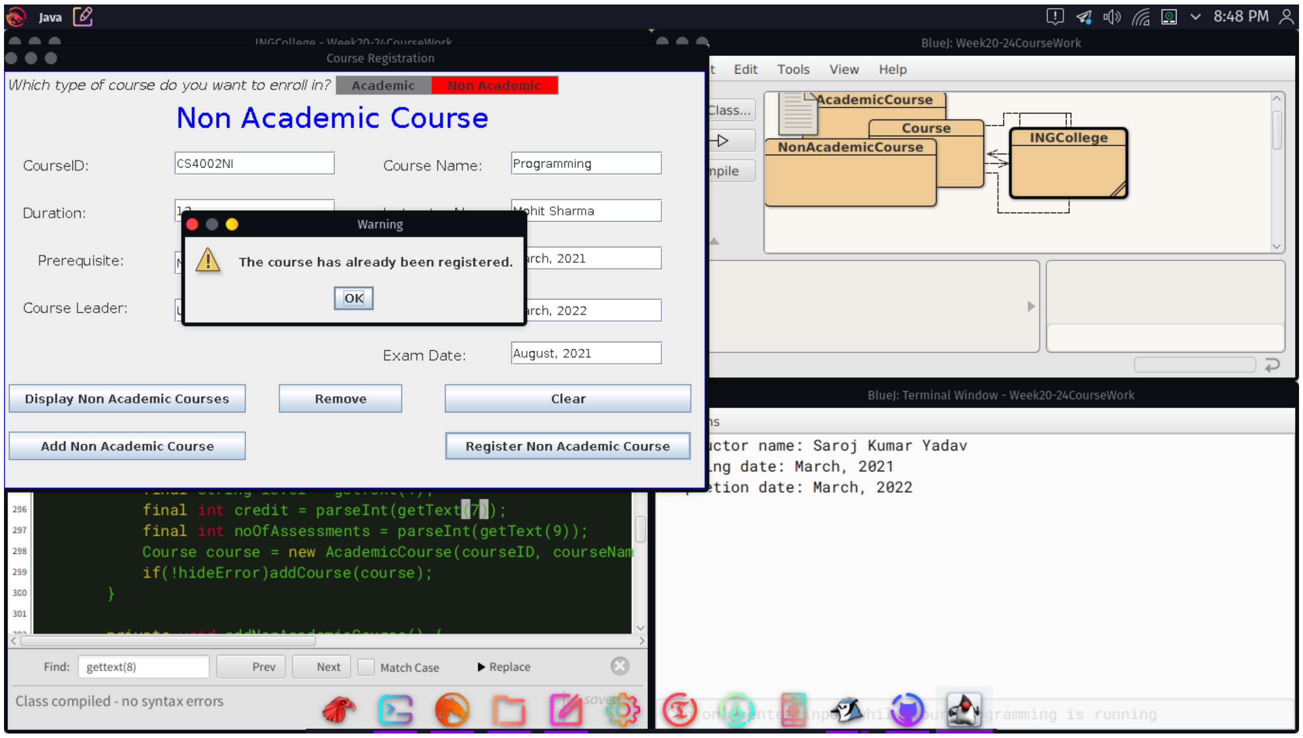
Add Non Academic Course | Register Non Academic Course

```
            final int credit = parseInt(getText(7));
            final int noOfAssessments = parseInt(getText(9));
            Course course = new AcademicCourse(courseID, courseNam
            if(!hideError)addCourse(course);
        }
```

gettext(8)

Class compiled - no syntax errors

Figure 12 - When trying to add duplicate NonAcademicCourse



Course Registration

Which type of course do you want to enroll in? Academic Non Academic

Non Academic Course

CourseID:	CS4002NI	Course Name:	Programming
Duration:	12		Mohit Sharma
Prerequisite:	N		rch, 2021
Course Leader:	U		rch, 2022
		Exam Date:	August, 2021

Warning

The course has already been registered.

OK

Display Non Academic Courses | Remove | Clear

Add Non Academic Course | Register Non Academic Course

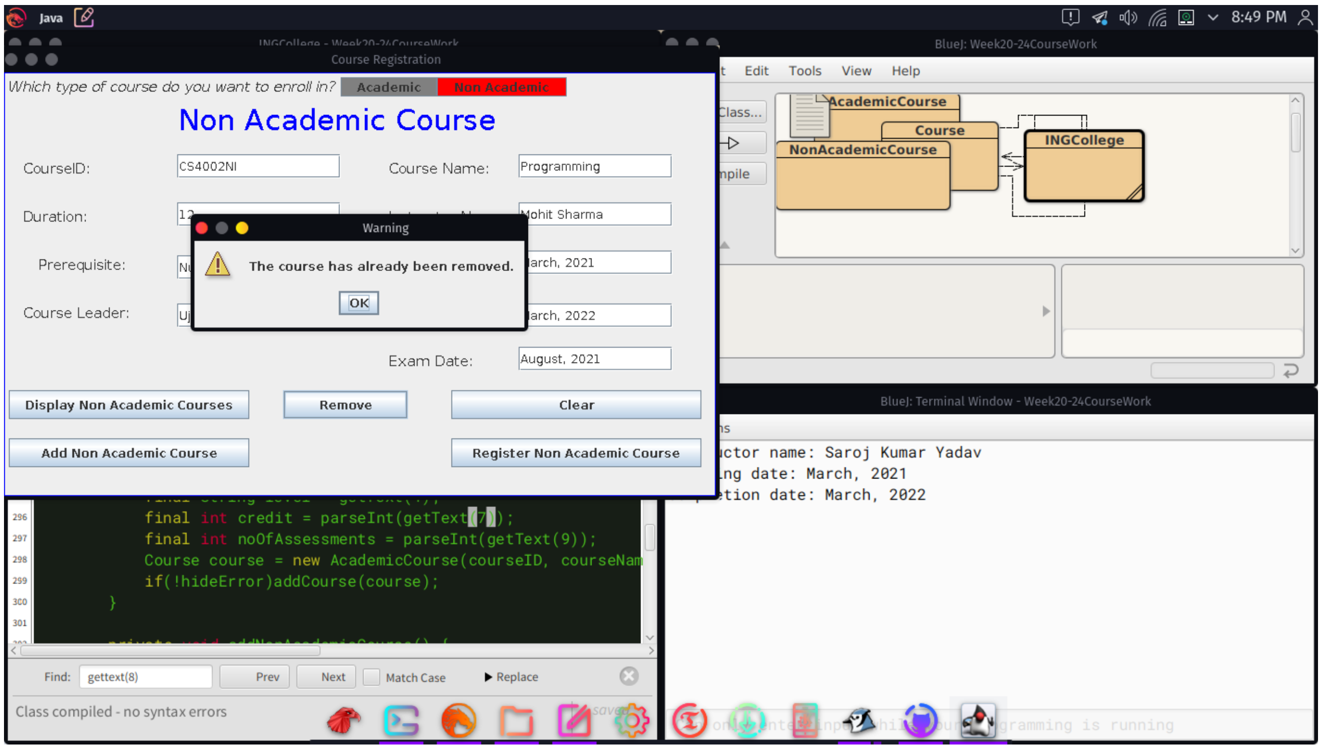
```
            final int credit = parseInt(getText(7));
            final int noOfAssessments = parseInt(getText(9));
            Course course = new AcademicCourse(courseID, courseNam
            if(!hideError)addCourse(course);
        }
```

gettext(8)

Class compiled - no syntax errors

```
uctor name: Saroj Kumar Yadav
ing date: March, 2021
etion date: March, 2022
```

Figure 13 - When trying to register duplicate NonAcademicCourse



Which type of course do you want to enroll in?

Non Academic Course

CourseID: CS4002NI
Course Name: Programming
Duration: 12
Instructor Name: Mohit Sharma
Prerequisite: Nu
Start Date: March, 2021
Course Leader: Uj
Completion Date: March, 2022
Exam Date: August, 2021

The course has already been removed.

```
        final int credit = parseInt(getText(7));
        final int noOfAssessments = parseInt(getText(9));
        Course course = new AcademicCourse(courseID, courseNam
        if(!hideError)addCourse(course);
    }
```

```
uctor name: Saroj Kumar Yadav
ing date: March, 2021
etion date: March, 2022
```

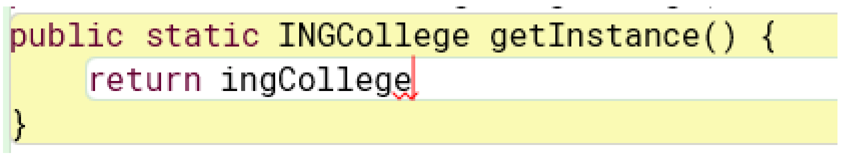
Class compiled - no syntax errors

Figure 14 - When trying to remove same NonAcademicCourse more than once

5. ERROR DETECTION AND CORRECTION

Syntax error

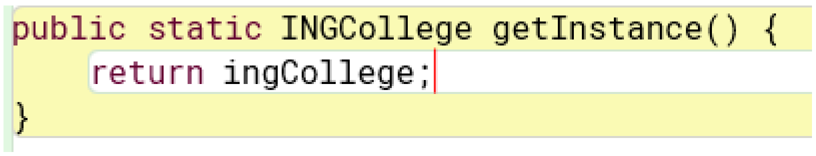
While making GUI in java, I kept missing semi colons to terminate a line.



```
public static INGCollege getInstance() {
    return ingCollege
}
```

Figure 15 - Syntax error detection

I fixed it by simply adding a semi colon at the end

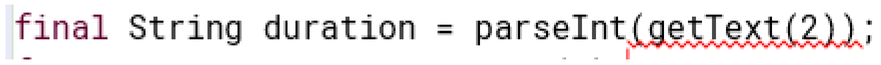


```
public static INGCollege getInstance() {
    return ingCollege;
}
```

Figure 16 - Syntax error correction

Semantic error

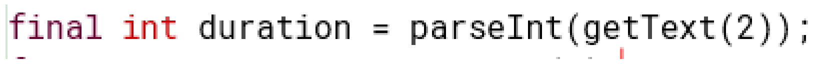
While writing GUI in java, I faced a semantic error where I parsed String to int but wrote String as return type of constant



```
final String duration = parseInt(getText(2));
```

Figure 17 - Semantic error detection

I fixed it by making the constant return int

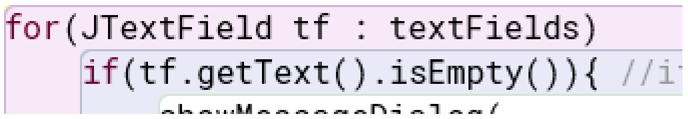


```
final int duration = parseInt(getText(2));
```

Figure 18 - Semantic error correction

Logical error

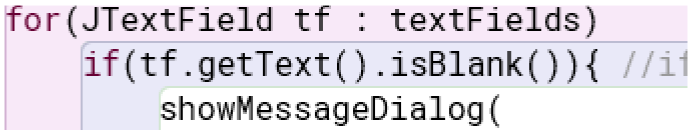
During the process of writing code for assignment and testing it, I noticed that we could just pass a character with only spaces and bugs would be introduced, preventing the code to run as intended.



```
for(JTextField tf : textFields)
    if(tf.getText().isEmpty()){ //i
```

Figure 19 - Logical error detection

I fixed it by using isBlank() method which strips down wide spaces before checking if String is empty



```
for(JTextField tf : textFields)
    if(tf.getText().isBlank()){ //i
        showMessageDialog(
```

Figure 20 - Logical error correction

6. CONCLUSION

The program has been created to implement a real-world scenario using the concepts of object oriented programming. It consists of 4 classes: Course, AcademicCourse, NonAcademicCourse and INGCollege classes. Course class is the superclass and AcademicCourse and NonAcademicCourse are its subclasses with additional attributes, property which also override a method ‘display()’ from their superclass. The inheritance relation is hierarchical. A Graphical User Interface has been created in INGCollege class, which is linked to other classes via dependency relation.

The INGCollege classes has 2 panels: academic and non-academic course registration and text fields for entering Course ID, Course Name, Duration, Course Leader, Lecturer Name, Level, Credit, Start Date, Completion Date, Number of Assessments, Instructor Name, Exam Date, Prerequisite as per the panel. The academic course registration panel has buttons to add, register and display courses which uses ArrayList data structure to store objects. The non-academic course registration panel has buttons to add, register, remove and display courses which uses ArrayList data structure to store objects. There is a single generic ArrayList which stores objects of both AcademicCourse and NonAcademicCourse classes, which are subclasses of the Course class. The concept of inheritance and object casting has been used to store and access attributes and properties of desired classes.

Doing the assignment was quite an enjoyable process. I got opportunity to learn various things from the assignment, especially things related to building a Graphical User Interface, managing its various properties, configuring the Graphical User Interface as per our needs, showing dialog boxes, making and switching multiple panels inside a single frame, setting action listeners to make the Graphical User Interface interactive. I also learned about ArrayList data structure which is a dynamic-sized collection, where objects/data are stored sequentially. Moreover, I learned the concept of inner, static, local and anonymous classes and interfaces, handling exceptions to not disrupt the flow of our program when an unexpected event occurs. Throughout the course, I used most of what I have learned and understood in academics.

I encountered some difficulties such as some variables getting null values during inspection in the process. Some given tasks were hard to understand initially. There were some bugs with IDE (Integrated Development Environment) bluej requiring me to roll back to a previous JDK version in that tool. I had to use JDK 15 instead of 16.

I received great support, suggestions and advises from teachers which helped me overcome those difficulties. When I was confused carrying out assigned task, I wrote to the teacher in email who suggested me to break down the task which was a great advise and it helped me understand and code out the task.

In general, the course work helped me learn practical usage of many concepts in java and I am grateful for this to the college as well as teachers and friends.

7. APPENDIX

Course class

```
import static java.lang.System.out


public class Course{
    private String courseID, courseName, courseLeader;
    private int duration;
    public Course(String courseID, String courseName, int duration){
        this.courseID=courseID;//setting value of variable in field equal to the value of respective fields
in parameter
        this.courseName=courseName;
        this.duration=duration;
        courseLeader="";
    }
    public String getCourseID(){
        return courseID;
    }


    public String getCourseName(){
        return courseName;
    }
    public String getCourseLeader(){
        return courseLeader;
    }

    public int getDuration(){
        return duration;
    }

    public void setCourseLeader(String courseLeader){
        this.courseLeader=courseLeader;
    }

    public void display(){
        out.println("Course ID: "+courseID);
        out.println("Course Name: "+courseName);
        out.println("Duration: "+duration);
        if(!courseLeader.trim().isEmpty()){
            out.println("Course Leader: "+courseLeader);
        }
    }
}
```

AcademicCourse class

```
import static java.lang.System.out;.out multiple times in this class

public class AcademicCourse extends Course{
    private String lecturerName, level, startingDate, completionDate;
    private int noOfAssessments, credit;
    private boolean isRegistered;
    private boolean courseRemovedStatus;
    public AcademicCourse(String courseID, String courseName, int duration, String level, int credit,
int noOfAssessments){
        super(courseID, courseName, duration);
        lecturerName="";
        startingDate="";
        completionDate="";
        this.noOfAssessments = noOfAssessments;
        this.level = level;
        isRegistered=false
    }
    public String getLecturerName(){
        return lecturerName;
    }
```

```
public String getLevel(){
    return level;
}

public int getCredit(){
    return credit;
}

public String getStartingDate(){
    return startingDate;
}

public String getCompletionDate(){
    return completionDate;
}

public boolean getRegistered(){
    return isRegistered;
}

public int getnoOfAssessments(){
    return noOfAssessments;
}
```

```
    public void setLecturerName(String lecturerName){
        this.lecturerName=lecturerName;
    }

    public void setnoOfAssessments(int noOfAssessments){
        this.noOfAssessments=noOfAssessments;
    }

    public void register(String courseLeader, String lecturerName, String startingDate, String
completionDate){
        if(isRegistered){
            INGCollege.getInstance().showTempDialogBox("Printing..");
            out.println("Instructor name: "+this.lecturerName);
            out.println("Starting date: "+this.startingDate);
            out.println("Completion date: "+this.completionDate);
        }else{
            this.lecturerName=lecturerName;
            this.startingDate=startingDate;
            this.completionDate=completionDate;
            super.setCourseLeader(courseLeader);
            isRegistered=true;
            courseRemovedStatus=false;
            INGCollege.getInstance().showTempDialogBox("Registering..");
        }
    }

    public void display(){
        super.display();
        if(isRegistered){
            out.println("Lecturer Name: "+lecturerName);
            out.println("Level: "+level);
            out.println("Starting Date: "+startingDate);
            out.println("Completion Date: "+completionDate);
            out.println("Number Of Assessments: "+noOfAssessments);
        }
    }
}
```

NonAcademicCourse class

```
import static java.lang.System.out;
import static javax.swing.JOptionPane.showMessageDialog;
import static javax.swing.JOptionPane.WARNING_MESSAGE;
```

```
public class NonAcademicCourse extends Course{
    private String instructorName, startDate, completionDate, examDate, prerequisite;
    private int duration;
    private boolean isRegistered, isRemoved;

    public NonAcademicCourse(String courseID, String courseName, int duration, String
prerequisite){
        super(courseID, courseName, duration);
        this.prerequisite=prerequisite;
        startDate="";
        completionDate="";
        examDate="";
        isRegistered=false;
    }

    public String getInstructorName(){
        return instructorName;
    }

    public int getDuration(){
        return duration;
    }
```

```
public String getStartDate(){
    return startDate;
}

public String getCompletionDate(){
    return completionDate;
}

public String getExamDate(){
    return examDate;
}

public String getPrerequisite(){
    return prerequisite;
}

public boolean getRegistered(){
    return isRegistered;
}

public boolean getRemoved(){
    return isRemoved;
}
```

```
public void setRemoved(boolean isRemoved){
    this.isRemoved=isRemoved;
}

public void setInstructorName(String instructorName){
    if(!isRegistered){
        this.instructorName=instructorName;
    }else{
        showMessageDialog(
            INGCollege.getInstance().getFrame(),"It is not possible to change instructor name since
            non academic course has already been registered",
            "Warning",WARNING_MESSAGE
        );
    }
}

public void register(String courseLeader, String instructorName, String startDate, String
completionDate, String examDate){
    if(!isRegistered){
        setInstructorName(instructorName);
        this.startDate=startDate;
        this.completionDate=completionDate;
        this.examDate=examDate;
        isRegistered=true;
    }else{
        showMessageDialog(
            INGCollege.getInstance().getFrame(),"The course has already been registered.",
            "Warning",WARNING_MESSAGE
        );
    }
}

public void remove(){
    if(isRemoved){
        showMessageDialog(
            INGCollege.getInstance().getFrame(),"The course has already been removed.",
            "Warning",WARNING_MESSAGE
        );
    }else{
        super.setCourseLeader("");
        startDate="";
        completionDate="";
        examDate="";
        INGCollege.getInstance().showTempDialogBox("Removing…");
        isRegistered=false;
        isRemoved=true;
    }
}
```

```
    public void display(){
        out.println("Course ID: "+super.getCourseID());
        out.println("Course Name: "+super.getCourseName());
        out.println("Duration: "+super.getDuration());
        if(isRegistered){
            out.println("Instructor Name: "+instructorName);
            out.println("Starting Date: "+startDate);
            out.println("Completion Date: "+completionDate);
            out.println("Exam Date: "+examDate);
        }
    }
}
```

INGCollege class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.ArrayList;
import java.util.List;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.event.DocumentListener;

import javax.swing.event.DocumentEvent;

import static javax.swing.JOptionPane.*;

import static java.lang.Integer.parseInt;


class INGCollege {

    private final JFrame frame;


    private final JPanel academicPanel, nonAcademicPanel;


    private final EventHandler eventHandler = new EventHandler();


    private final List<Course> courses = new ArrayList<>();

    private final List<JTextField> textFields = new ArrayList<>();


    private JButton academicButton, nonAcademicButton;


    public static void main(String[] args) {

        ingCollege = new INGCollege();

    }

    private static INGCollege ingCollege;

    public static INGCollege getInstance() {

        return ingCollege;

    }
```

```
private INGCollege() {
    class Panel {
        void setUpNonAcademicPanel(JPanel p) {
            addSwitcher(p);


            setLabel(p,"Non Academic Course", 181, 0, 800, 100, 30);
            setLabel(p,"CourseID:", 20, 90, 70, 20, 15);
            setLabel(p,"Instructor Name:", 400, 140, 130, 20, 15);
            setLabel(p,"Course Name:", 400, 90, 125, 20, 15);
            setLabel(p,"Duration:", 20, 140, 125, 20, 15);
            setLabel(p,"Prerequisite:", 36, 190, 125, 20, 15);
            setLabel(p,"Course Leader:", 20, 240, 125, 20, 15);
            setLabel(p,"Start Date:", 400, 190, 130, 20, 15);
            setLabel(p,"Completion Date:", 400, 240, 130, 20, 15);
            setLabel(p,"Exam Date:", 400, 290, 130, 20, 15);


            setTextField(p,180, 85, 170);
            setTextField(p,180, 135, 170);
            setTextField(p,535, 85, 160);
            setTextField(p,535, 135, 160);
            setTextField(p,180, 190, 170);
            setTextField(p,180, 240, 170);
            setTextField(p,535, 185, 160);
            setTextField(p,535, 285, 160);
            setTextField(p,535, 240, 160);

            setButton(p,"Remove", 290, 330, 130);
            setButton(p,"Add Non Academic Course", 5, 380, 250);
            setButton(p,"Display Non Academic Courses", 5, 330, 250);
            setButton(p,"Clear", 465, 330, 260);
            setButton(p,"Register Non Academic Course", 465, 380, 260);

            p.setBorder(BorderFactory.createLineBorder(Color.BLUE));
            p.setSize(740,440);
            p.setVisible(false);
        }

        void setUpAcademicPanel(JPanel p) {
            addSwitcher(p);

            setLabel(p,"Academic Course", 250, 0, 800, 100, 30);
            setLabel(p,"CourseID:", 20, 90, 70, 20, 15);
            setLabel(p,"Duration:", 20, 140, 70, 20, 15);
            setLabel(p,"Completion Date:", 400, 240, 130, 20, 15);
            setLabel(p,"Level:", 440, 290, 130, 20, 15);
            setLabel(p,"Course Name:", 400, 90, 125, 20, 15);
            setLabel(p,"No. of Assessments:", 370, 140, 185, 20, 15);
            setLabel(p,"Start Date:", 400, 190, 130, 20, 15);
            setLabel(p,"Credit:", 20, 290, 125, 20, 15);
            setLabel(p,"Lecturer Name:", 20, 190, 125, 20, 15);
            setLabel(p,"Course Leader:", 20, 240, 125, 20, 15);

            setTextField(p,180, 85, 170);
            setTextField(p,535, 85, 160);
            setTextField(p,180, 135, 170);
            setTextField(p,535, 185, 160);
            setTextField(p, 535, 285, 160);
            setTextField(p,535, 240, 160);
            setTextField(p,180, 240, 170);
            setTextField(p,180, 285, 170);
            setTextField(p,180, 190, 170);
            setTextField(p,535, 135, 160);

            setButton(p,"Add Academic Course", 5, 380, 250);
            setButton(p,"Display Academic Courses", 5, 330, 250);
            setButton(p,"Clear", 465, 330, 260);
            setButton(p,"Register Academic Course", 465, 380, 260);

            p.setBorder(BorderFactory.createLineBorder(Color.BLUE));
            p.setSize(740,440);
        }
```

```
void addSwitcher(JPanel panel) {
    academicButton = new JButton("Academic") {
        {
            setBackground(new Color(6,181,223));
            setBounds(350,5,100,20);
            addActionListener(eventHandler);
            panel.add(this);
        }
    };

    nonAcademicButton = new JButton("Non Academic") {
        {
            setBackground(Color.GRAY);
            setBounds(450,5,135,20);
            addActionListener(eventHandler);
            panel.add(this);
        }
    };
    setLabel(panel, "Which type of course do you want to enroll in?",5,0,350,30,15);
}
```

```
void setLabel(JPanel panel, String text, int x, int y, int width, int height, int fontSize) {
    new JLabel(text) {
        {
            setBounds(x, y, width, height);
            setFont(new Font(null, Font.PLAIN, fontSize));
            String txt = getText();
            if(txt.contains("Academic"))setForeground(Color.BLUE);
            else if(txt.contains("Which"))setFont(new Font(null, Font.ITALIC, fontSize));
            panel.add(this);
        }
    };
}

void setButton(JPanel panel, String text, int x, int y, int width) {
    new JButton(text) {
        {
            setBounds(x, y, width, 30);
            addActionListener(eventHandler);
            panel.add(this);
        }
    };
}
```

```
    void setTextField(JPanel panel, int x, int y, int width) {
        new JTextField() { //creating anonymous class extending JTextField with this object
            {
                setBounds(x, y, width, 25);
                textFields.add(this);
                getDocument().addDocumentListener(new DocumentListener(){
                    @Override
                    public void changedUpdate(DocumentEvent e){}

                    @Override
                    public void removeUpdate(DocumentEvent e){}

                    @Override
                    public void insertUpdate(DocumentEvent e){
                        eventHandler.removeNonAcademic = null;
                    }
                });
                panel.add(this);
            }
        };
    }
}
```

```
academicPanel = new JPanel(null);
nonAcademicPanel = new JPanel(null);

new Panel(){
   {
       setUpAcademicPanel(academicPanel);
       setUpNonAcademicPanel(nonAcademicPanel);
   }
};




frame = new JFrame("Course Registration") {
   {
       setSize(748,472);
       add(academicPanel); //show by defareult
       add(nonAcademicPanel); //hide by default
       setLocationRelativeTo(null); //center the frame by default(0,0 otherwise)
       addWindowListener(eventHandler); //when user presses X button to close program
       setVisible(true);
   }
};
```

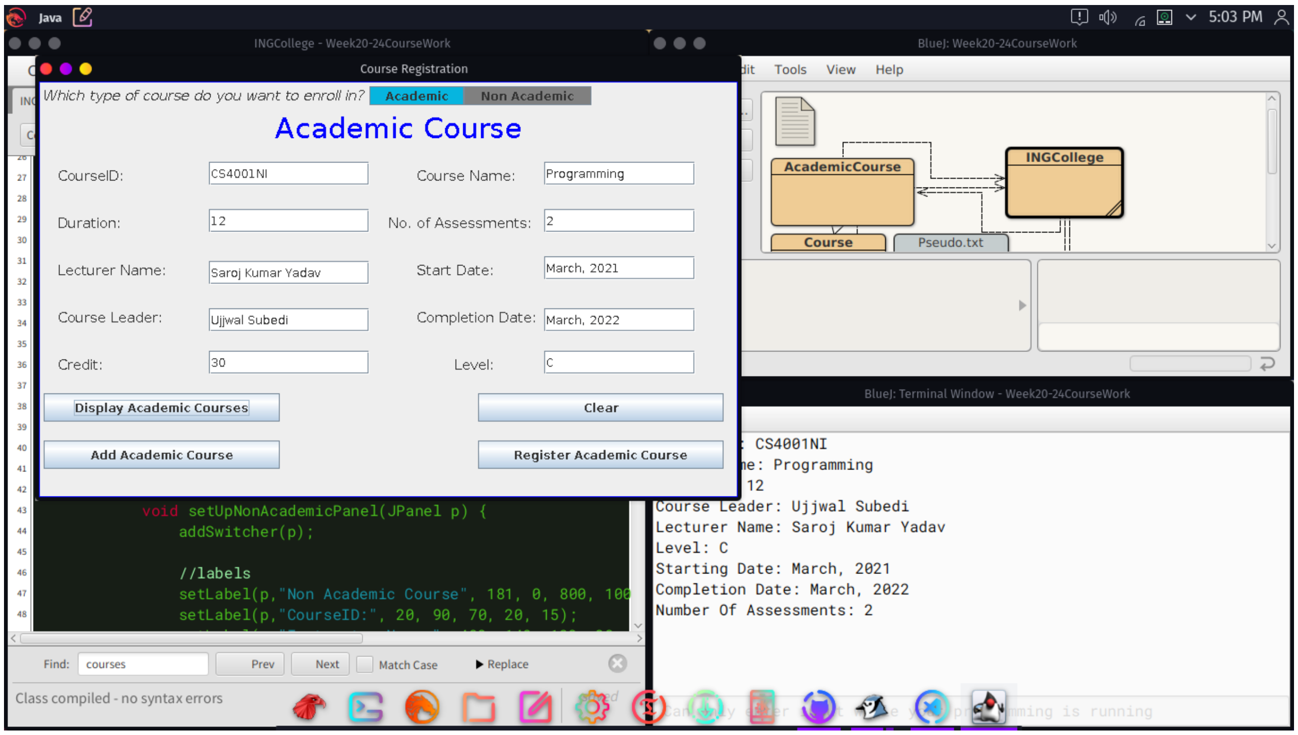
```
}


public JFrame getFrame() {
    return frame;
}


void showTempDialogBox(String message){
    JOptionPane pane = new JOptionPane();
    JDialog dialog = pane.createDialog(frame, message);


    final int w = 150;
    final int h = 30;
    dialog.setBounds(frame.getX()+(frame.getWidth()/2)-(w/2),
        frame.getY()+frame.getHeight()-h, w, h);


    dialog.setModalityType(Dialog.ModalityType.MODELESS);
    dialog.setVisible(true); //show/enable
    try{
        Thread.sleep(500);
    }finally{
        dialog.setVisible(false);
        return;
    }
}
```

```
private class EventHandler extends WindowAdapter implements ActionListener{

    private String getText(int index) {
        return textFields.get(index).getText();
    }

    @Override
    public void windowClosing(WindowEvent e) {
        showMessageDialog(frame, "Thank you for trying");
        System.exit(0);
    }



    @Override
    public void actionPerformed(ActionEvent e) {
        switch(e.getActionCommand()) {
            case "Clear"->{
                    //clear text of all textFields
                    for(JTextField t : textFields)t.setText("");
              }
            case "Add Academic Course"->addAcademicCourse();
            case "Add Non Academic Course"->addNonAcademicCourse();
            case "Display Academic Courses"->{
```

```
        for(Course c : courses)
            if(c instanceof AcademicCourse) {
                AcademicCourse ac = (AcademicCourse)c;
                ac.display();
            }
    }
case "Display Non Academic Courses"->{
        for(Course c : courses)
            if(c instanceof NonAcademicCourse) {
                NonAcademicCourse nac = (NonAcademicCourse)c;
                nac.display();
            }
    }
case "Remove"->removeNonAcademicCourse();
case "Register Academic Course"->registerAcademicCourse();
case "Register Non Academic Course"->registerNonAcademicCourse();
case "Academic"->{
        academicPanel.setVisible(true);
        nonAcademicPanel.setVisible(false);
    }
case "Non Academic"->{
        nonAcademicButton.setBackground(Color.RED);
        academicButton.setBackground(Color.GRAY);
        academicPanel.setVisible(false);
```

```
                nonAcademicPanel.setVisible(true);
            }
        }
    }

    private void showParseError(Exception e) {
        String log = e.getMessage();
        if(e instanceof NumberFormatException)
            log = "Please input valid integer\n" + log;
        showMessageDialog(frame, log, "Error", ERROR_MESSAGE);
    }

    private NonAcademicCourse removeNonAcademic;

    private void removeNonAcademicCourse() {
        final String courseID = getText(10);
        for(Course c : courses)
            if(c instanceof NonAcademicCourse && c.getCourseID().equals(courseID)) {
                removeNonAcademic = (NonAcademicCourse)c;
                courses.remove(c);
                showTempDialogBox(“Removing…”);
                break;
            }
        if(removeNonAcademic!=null)removeNonAcademic.remove();
        else showTempDialogBox(“Not added yet”);


    private void addAcademicCourse() {
        try{
            final String courseID = getText(0);
            final String courseName = getText(1);
            final int duration = parseInt(getText(2));
            final String level = getText(4);
            final int credit = parseInt(getText(7));
            final int noOfAssessments = parseInt(getText(9));
            Course course = new AcademicCourse(courseID, courseName, duration, level, credit,
noOfAssessments);
            addCourse(course);
        }catch(Exception e){
            showParseError(e);
        }
    }


    private void addNonAcademicCourse() {
        try{
            final String courseID = getText(10);
            final String courseName = getText(12);
            final int duration = parseInt(getText(11));
            final String prerequisite = getText(14);
            Course course = new NonAcademicCourse(courseID, courseName, duration,
            prerequisite);
            addCourse(course);
        }catch(Exception e){ //all exceptions
            showParseError(e);
        }
    }

    private void addCourse(Course course) {
        for(JTextField tf : textFields)
            if(tf.getText().isBlank()){
                showMessageDialog(
                    INGCollege.this.getFrame(),"Please make sure you have filled everything.",
                    "Warning",WARNING_MESSAGE
                );
                return;
            }
        String text;
        text = (course instanceof AcademicCourse) ? getText(0) : getText(10);

        for(Course c : courses)
            if(c.getCourseID().equals(text)) {
                showMessageDialog(
                    INGCollege.this.getFrame(),"The course has already been added.",
                    "Warning",WARNING_MESSAGE
                );
                return;
            }
        courses.add(course);
        showTempDialogBox("Adding..");
    }

    private void registerAcademicCourse() {
        final String courseLeader = getText(6);
        final String lecturerName = getText(8);
        final String startingDate = getText(3);
        final String completionDate = getText(5);
        for(Course c : courses)
            if(c instanceof AcademicCourse && getText(0).equals(c.getCourseID())) {
                AcademicCourse ac = (AcademicCourse)c;
                ac.register(
                    courseLeader, lecturerName, startingDate, completionDate
                );
            }
    }

    private void registerNonAcademicCourse() {
        final String courseLeader = getText(15);
```

```
        final String instructorName = getText(13);
        final String startingDate = getText(16);
        final String completionDate = getText(18);
        final String examDate = getText(17);
        for(Course c : courses)
            if(c instanceof NonAcademicCourse && getText(10).equals(c.getCourseID())) {
                NonAcademicCourse nac = (NonAcademicCourse)c;
                nac.register(
                    courseLeader, instructorName, startingDate, completionDate, examDate
                );
            }
    }
}}
```



Which type of course do you want to enroll in? Academic Non Academic

Academic Course

CourseID:	CS4001NI	Course Name:	Programming
Duration:	12	No. of Assessments:	2
Lecturer Name:	Saroj Kumar Yadav	Start Date:	March, 2021
Course Leader:	Ujjwal Subedi	Completion Date:	March, 2022
Credit:	30	Level:	C

Display Academic Courses | Clear | Add Academic Course | Register Academic Course

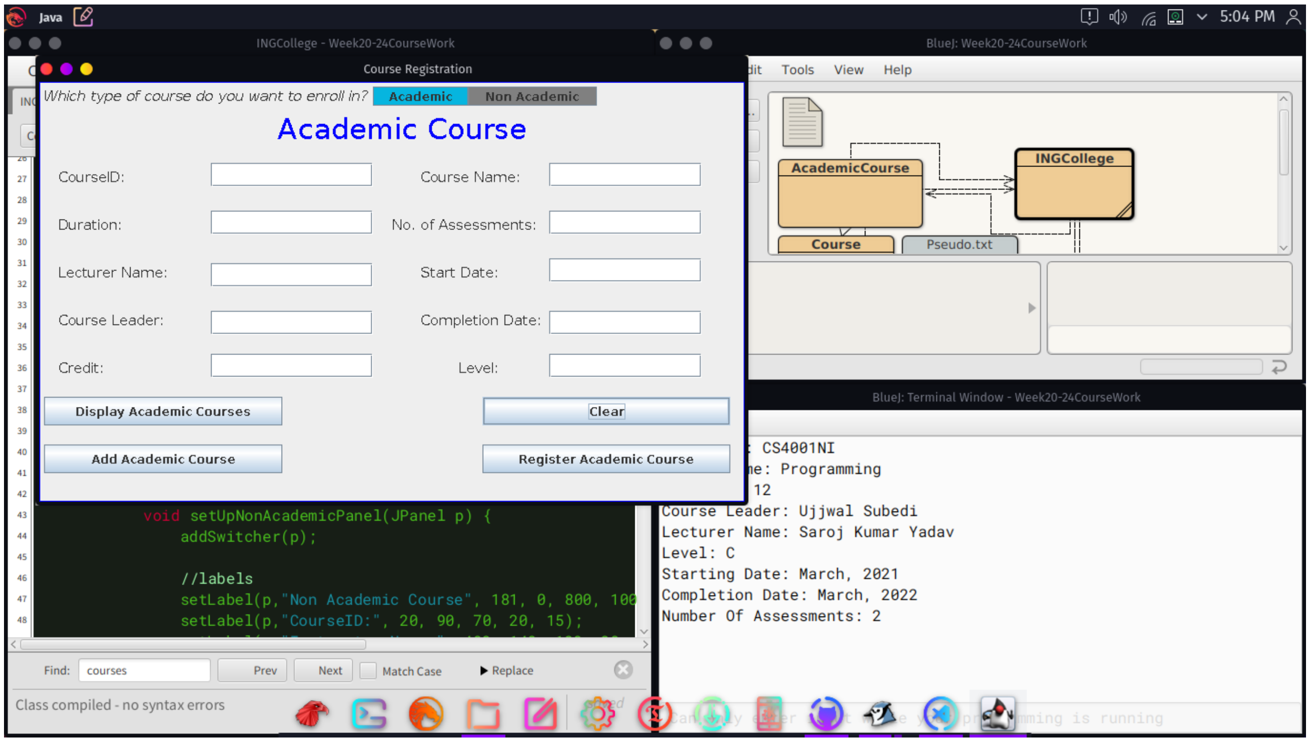
```
void setUpNonAcademicPanel(JPanel p) {
    addSwitcher(p);

    //labels
    setLabel(p,"Non Academic Course", 181, 0, 800, 100
    setLabel(p,"CourseID:", 20, 90, 70, 20, 15);
```

```
: CS4001NI
me: Programming
 12
Course Leader: Ujjwal Subedi
Lecturer Name: Saroj Kumar Yadav
Level: C
Starting Date: March, 2021
Completion Date: March, 2022
Number Of Assessments: 2
```

Find: courses

Class compiled - no syntax errors

Figure 21 - Display Button



Which type of course do you want to enroll in? Academic Non Academic

Academic Course

CourseID:		Course Name:	
Duration:		No. of Assessments:	
Lecturer Name:		Start Date:	
Course Leader:		Completion Date:	
Credit:		Level:	

Display Academic Courses | Clear | Add Academic Course | Register Academic Course

```
void setUpNonAcademicPanel(JPanel p) {
    addSwitcher(p);

    //labels
    setLabel(p,"Non Academic Course", 181, 0, 800, 100
    setLabel(p,"CourseID:", 20, 90, 70, 20, 15);
```

```
: CS4001NI
me: Programming
 12
Course Leader: Ujjwal Subedi
Lecturer Name: Saroj Kumar Yadav
Level: C
Starting Date: March, 2021
Completion Date: March, 2022
Number Of Assessments: 2
```

Find: courses

Class compiled - no syntax errors

Figure 22 - Clear Button