```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2.3\lib\idea_rt.jar=51769:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2.3\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\Createch\IdeaProjects\untitled3\target\classes;C:\Users\Createch\.m2\repository\com\github\javaparser\javaparser-core\3.25.1\javaparser-core-3.25.1.jar org.example.ParserMainAST
Class name: StudentRegister
Class AST:
ClassOrInterfaceDeclaration: class StudentRegister {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        ArrayList<Student> studentList = new ArrayList<>();
        System.out.println("Welcome to the Student Register!");
        while (true) {
            System.out.println("Please select an option:");
            System.out.println("1. Add Student");
            System.out.println("2. View Student List");
            System.out.println("3. Delete Student by Name");
            System.out.println("4. Delete Student by ID");
            System.out.println("5. Exit");
            int choice = input.nextInt();
            // consume newline character
            input.nextLine();
            switch(choice) {
                case 1:
                    System.out.println("Enter student name:");
                    String name = input.nextLine();
                    System.out.println("Enter student ID:");
```

```
25                      String id = input.nextLine();
26                      System.out.println("Enter student
    major:");
27                      String major = input.nextLine();
28                      Student student = new Student(
   name, id, major);
29                      studentList.add(student);
30                      System.out.println("Student added
    successfully.");
31                      break;
32                  case 2:
33                      System.out.println("Student List
   :");
34                      for (Student s : studentList) {
35                          System.out.println("Name
   : " + s.getName() + ", ID: " + s.getId() + ", Major
   : " + s.getMajor());
36                      }
37                      break;
38                  case 3:
39                      System.out.println("Enter student
    name to delete:");
40                      String nameToDelete = input.
   nextLine();
41                      if (deleteStudentByName(
   studentList, nameToDelete)) {
42                          System.out.println("Student
    " + nameToDelete + " deleted successfully.");
43                      } else {
44                          System.out.println("Student
    " + nameToDelete + " not found.");
45                      }
46                      break;
47                  case 4:
48                      System.out.println("Enter student
    ID to delete:");
49                      String idToDelete = input.
   nextLine();
50                      if (deleteStudentById(studentList
   , idToDelete)) {
51                          System.out.println("Student
```

```
with ID " + idToDelete + " deleted successfully.");
                    } else {
                        System.out.println("Student
  with ID " + idToDelete + " not found.");
                    }
                    break;
                case 5:
                    System.out.println("Thank you for
   using the Student Register!");
                    return;
                default:
                    System.out.println("Invalid
  option, please try again.");
                    break;
            }
        }
    }

    public static boolean deleteStudentByName(
  ArrayList<Student> studentList, String name) {
        for (Student s : studentList) {
            if (s.getName().equalsIgnoreCase(name)) {
                studentList.remove(s);
                return true;
            }
        }
        return false;
    }

    public static boolean deleteStudentById(ArrayList
  <Student> studentList, String id) {
        for (Student s : studentList) {
            if (s.getId().equalsIgnoreCase(id)) {
                studentList.remove(s);
                return true;
            }
        }
        return false;
    }
}
  SimpleName: StudentRegister
```

```
 87   MethodDeclaration: public static void main(String
    [] args) {
 88     Scanner input = new Scanner(System.in);
 89     ArrayList<Student> studentList = new ArrayList
    <>();
 90     System.out.println("Welcome to the Student
    Register!");
 91     while (true) {
 92         System.out.println("Please select an option
    :");
 93         System.out.println("1. Add Student");
 94         System.out.println("2. View Student List");
 95         System.out.println("3. Delete Student by
    Name");
 96         System.out.println("4. Delete Student by ID
    ");
 97         System.out.println("5. Exit");
 98         int choice = input.nextInt();
 99         // consume newline character
100         input.nextLine();
101         switch(choice) {
102             case 1:
103                 System.out.println("Enter student
    name:");
104                 String name = input.nextLine();
105                 System.out.println("Enter student ID
    :");
106                 String id = input.nextLine();
107                 System.out.println("Enter student
    major:");
108                 String major = input.nextLine();
109                 Student student = new Student(name,
    id, major);
110                 studentList.add(student);
111                 System.out.println("Student added
    successfully.");
112                 break;
113             case 2:
114                 System.out.println("Student List:");
115                 for (Student s : studentList) {
116                     System.out.println("Name: " + s.
```

```
116 getName() + ", ID: " + s.getId() + ", Major: " + s.
    getMajor());
117                 }
118                 break;
119             case 3:
120                 System.out.println("Enter student
    name to delete:");
121                 String nameToDelete = input.nextLine
    ();
122                 if (deleteStudentByName(studentList
    , nameToDelete)) {
123                     System.out.println("Student " +
    nameToDelete + " deleted successfully.");
124                 } else {
125                     System.out.println("Student " +
    nameToDelete + " not found.");
126                 }
127                 break;
128             case 4:
129                 System.out.println("Enter student ID
     to delete:");
130                 String idToDelete = input.nextLine
    ();
131                 if (deleteStudentById(studentList,
    idToDelete)) {
132                     System.out.println("Student with
     ID " + idToDelete + " deleted successfully.");
133                 } else {
134                     System.out.println("Student with
     ID " + idToDelete + " not found.");
135                 }
136                 break;
137             case 5:
138                 System.out.println("Thank you for
    using the Student Register!");
139                 return;
140             default:
141                 System.out.println("Invalid option,
    please try again.");
142                 break;
143         }
```

```
144     }
145 }
146     Modifier: public
147     Modifier: static
148     SimpleName: main
149     Parameter: String[] args
150       ArrayType: String[]
151         ClassOrInterfaceType: String
152           SimpleName: String
153       SimpleName: args
154     VoidType: void
155     BlockStmt: {
156     Scanner input = new Scanner(System.in);
157     ArrayList<Student> studentList = new ArrayList
    <>();
158     System.out.println("Welcome to the Student
    Register!");
159     while (true) {
160         System.out.println("Please select an option
    :");
161         System.out.println("1. Add Student");
162         System.out.println("2. View Student List");
163         System.out.println("3. Delete Student by
    Name");
164         System.out.println("4. Delete Student by ID
    ");
165         System.out.println("5. Exit");
166         int choice = input.nextInt();
167         // consume newline character
168         input.nextLine();
169         switch(choice) {
170             case 1:
171                 System.out.println("Enter student
    name:");
172                 String name = input.nextLine();
173                 System.out.println("Enter student ID
    :");
174                 String id = input.nextLine();
175                 System.out.println("Enter student
    major:");
176                 String major = input.nextLine();
```

```
177                 Student student = new Student(name,
    id, major);
178                 studentList.add(student);
179                 System.out.println("Student added
    successfully.");
180                 break;
181             case 2:
182                 System.out.println("Student List:");
183                 for (Student s : studentList) {
184                     System.out.println("Name: " + s.
    getName() + ", ID: " + s.getId() + ", Major: " + s.
    getMajor());
185                 }
186                 break;
187             case 3:
188                 System.out.println("Enter student
    name to delete:");
189                 String nameToDelete = input.nextLine
    ();
190                 if (deleteStudentByName(studentList
    , nameToDelete)) {
191                     System.out.println("Student " +
    nameToDelete + " deleted successfully.");
192                 } else {
193                     System.out.println("Student " +
    nameToDelete + " not found.");
194                 }
195                 break;
196             case 4:
197                 System.out.println("Enter student ID
     to delete:");
198                 String idToDelete = input.nextLine
    ();
199                 if (deleteStudentById(studentList,
    idToDelete)) {
200                     System.out.println("Student with
     ID " + idToDelete + " deleted successfully.");
201                 } else {
202                     System.out.println("Student with
     ID " + idToDelete + " not found.");
203                 }
```

```
                break;
            case 5:
                System.out.println("Thank you for
    using the Student Register!");
                return;
            default:
                System.out.println("Invalid option,
    please try again.");
                break;
        }
    }
}
      ExpressionStmt: Scanner input = new Scanner(
    System.in);
        VariableDeclarationExpr: Scanner input = new
     Scanner(System.in)
          VariableDeclarator: input = new Scanner(
    System.in)
            ClassOrInterfaceType: Scanner
              SimpleName: Scanner
            SimpleName: input
            ObjectCreationExpr: new Scanner(System.
    in)
              ClassOrInterfaceType: Scanner
                SimpleName: Scanner
              FieldAccessExpr: System.in
                NameExpr: System
                  SimpleName: System
                SimpleName: in
      ExpressionStmt: ArrayList<Student> studentList
     = new ArrayList<>();
        VariableDeclarationExpr: ArrayList<Student>
    studentList = new ArrayList<>()
          VariableDeclarator: studentList = new
    ArrayList<>()
            ClassOrInterfaceType: ArrayList<Student>
              SimpleName: ArrayList
              ClassOrInterfaceType: Student
                SimpleName: Student
            SimpleName: studentList
            ObjectCreationExpr: new ArrayList<>()
```

```
236                ClassOrInterfaceType: ArrayList<>
237                  SimpleName: ArrayList
238       ExpressionStmt: System.out.println("Welcome to
     the Student Register!");
239         MethodCallExpr: System.out.println("Welcome
    to the Student Register!")
240           FieldAccessExpr: System.out
241             NameExpr: System
242               SimpleName: System
243             SimpleName: out
244           SimpleName: println
245           StringLiteralExpr: "Welcome to the Student
     Register!"
246       WhileStmt: while (true) {
247     System.out.println("Please select an option:");
248     System.out.println("1. Add Student");
249     System.out.println("2. View Student List");
250     System.out.println("3. Delete Student by Name");
251     System.out.println("4. Delete Student by ID");
252     System.out.println("5. Exit");
253     int choice = input.nextInt();
254     // consume newline character
255     input.nextLine();
256     switch(choice) {
257         case 1:
258             System.out.println("Enter student name
    :");
259             String name = input.nextLine();
260             System.out.println("Enter student ID:");
261             String id = input.nextLine();
262             System.out.println("Enter student major
    :");
263             String major = input.nextLine();
264             Student student = new Student(name, id,
    major);
265             studentList.add(student);
266             System.out.println("Student added
    successfully.");
267             break;
268         case 2:
269             System.out.println("Student List:");
```

```
270             for (Student s : studentList) {
271                 System.out.println("Name: " + s.
    getName() + ", ID: " + s.getId() + ", Major: " + s.
    getMajor());
272             }
273             break;
274         case 3:
275             System.out.println("Enter student name
    to delete:");
276             String nameToDelete = input.nextLine();
277             if (deleteStudentByName(studentList,
    nameToDelete)) {
278                 System.out.println("Student " +
    nameToDelete + " deleted successfully.");
279             } else {
280                 System.out.println("Student " +
    nameToDelete + " not found.");
281             }
282             break;
283         case 4:
284             System.out.println("Enter student ID to
    delete:");
285             String idToDelete = input.nextLine();
286             if (deleteStudentById(studentList,
    idToDelete)) {
287                 System.out.println("Student with ID
     " + idToDelete + " deleted successfully.");
288             } else {
289                 System.out.println("Student with ID
     " + idToDelete + " not found.");
290             }
291             break;
292         case 5:
293             System.out.println("Thank you for using
    the Student Register!");
294             return;
295         default:
296             System.out.println("Invalid option,
    please try again.");
297             break;
298     }
```

```
299 }
300         BooleanLiteralExpr: true
301         BlockStmt: {
302     System.out.println("Please select an option:");
303     System.out.println("1. Add Student");
304     System.out.println("2. View Student List");
305     System.out.println("3. Delete Student by Name");
306     System.out.println("4. Delete Student by ID");
307     System.out.println("5. Exit");
308     int choice = input.nextInt();
309     // consume newline character
310     input.nextLine();
311     switch(choice) {
312         case 1:
313             System.out.println("Enter student name
    :");
314             String name = input.nextLine();
315             System.out.println("Enter student ID:");
316             String id = input.nextLine();
317             System.out.println("Enter student major
    :");
318             String major = input.nextLine();
319             Student student = new Student(name, id,
    major);
320             studentList.add(student);
321             System.out.println("Student added
    successfully.");
322             break;
323         case 2:
324             System.out.println("Student List:");
325             for (Student s : studentList) {
326                 System.out.println("Name: " + s.
    getName() + ", ID: " + s.getId() + ", Major: " + s.
    getMajor());
327             }
328             break;
329         case 3:
330             System.out.println("Enter student name
    to delete:");
331             String nameToDelete = input.nextLine();
332             if (deleteStudentByName(studentList,
```

```
332 nameToDelete)) {
333                 System.out.println("Student " +
    nameToDelete + " deleted successfully.");
334             } else {
335                 System.out.println("Student " +
    nameToDelete + " not found.");
336             }
337             break;
338         case 4:
339             System.out.println("Enter student ID to
    delete:");
340             String idToDelete = input.nextLine();
341             if (deleteStudentById(studentList,
    idToDelete)) {
342                 System.out.println("Student with ID
     " + idToDelete + " deleted successfully.");
343             } else {
344                 System.out.println("Student with ID
     " + idToDelete + " not found.");
345             }
346             break;
347         case 5:
348             System.out.println("Thank you for using
    the Student Register!");
349             return;
350         default:
351             System.out.println("Invalid option,
    please try again.");
352             break;
353     }
354 }
355           ExpressionStmt: System.out.println("Please
     select an option:");
356             MethodCallExpr: System.out.println("
    Please select an option:")
357               FieldAccessExpr: System.out
358                 NameExpr: System
359                   SimpleName: System
360                 SimpleName: out
361               SimpleName: println
362               StringLiteralExpr: "Please select an
```

```
362 option:"
363           ExpressionStmt: System.out.println("1. Add
     Student");
364             MethodCallExpr: System.out.println("1.
    Add Student")
365               FieldAccessExpr: System.out
366                 NameExpr: System
367                   SimpleName: System
368                 SimpleName: out
369               SimpleName: println
370               StringLiteralExpr: "1. Add Student"
371           ExpressionStmt: System.out.println("2.
    View Student List");
372             MethodCallExpr: System.out.println("2.
    View Student List")
373               FieldAccessExpr: System.out
374                 NameExpr: System
375                   SimpleName: System
376                 SimpleName: out
377               SimpleName: println
378               StringLiteralExpr: "2. View Student
    List"
379           ExpressionStmt: System.out.println("3.
    Delete Student by Name");
380             MethodCallExpr: System.out.println("3.
    Delete Student by Name")
381               FieldAccessExpr: System.out
382                 NameExpr: System
383                   SimpleName: System
384                 SimpleName: out
385               SimpleName: println
386               StringLiteralExpr: "3. Delete Student
    by Name"
387           ExpressionStmt: System.out.println("4.
    Delete Student by ID");
388             MethodCallExpr: System.out.println("4.
    Delete Student by ID")
389               FieldAccessExpr: System.out
390                 NameExpr: System
391                   SimpleName: System
392                 SimpleName: out
```

```
393               SimpleName: println
394               StringLiteralExpr: "4. Delete Student
    by ID"
395           ExpressionStmt: System.out.println("5.
    Exit");
396             MethodCallExpr: System.out.println("5.
    Exit")
397               FieldAccessExpr: System.out
398                 NameExpr: System
399                   SimpleName: System
400                 SimpleName: out
401               SimpleName: println
402               StringLiteralExpr: "5. Exit"
403           ExpressionStmt: int choice = input.nextInt
    ();
404             VariableDeclarationExpr: int choice =
    input.nextInt()
405               VariableDeclarator: choice = input.
    nextInt()
406                 PrimitiveType: int
407                 SimpleName: choice
408                 MethodCallExpr: input.nextInt()
409                   NameExpr: input
410                     SimpleName: input
411                   SimpleName: nextInt
412           ExpressionStmt: // consume newline
    character
413 input.nextLine();
414             MethodCallExpr: input.nextLine()
415               NameExpr: input
416                 SimpleName: input
417               SimpleName: nextLine
418           SwitchStmt: switch(choice) {
419     case 1:
420         System.out.println("Enter student name:");
421         String name = input.nextLine();
422         System.out.println("Enter student ID:");
423         String id = input.nextLine();
424         System.out.println("Enter student major:");
425         String major = input.nextLine();
426         Student student = new Student(name, id,
```

```
426 major);
427         studentList.add(student);
428         System.out.println("Student added
    successfully.");
429         break;
430     case 2:
431         System.out.println("Student List:");
432         for (Student s : studentList) {
433             System.out.println("Name: " + s.getName
    () + ", ID: " + s.getId() + ", Major: " + s.getMajor
    ());
434         }
435         break;
436     case 3:
437         System.out.println("Enter student name to
    delete:");
438         String nameToDelete = input.nextLine();
439         if (deleteStudentByName(studentList,
    nameToDelete)) {
440             System.out.println("Student " +
    nameToDelete + " deleted successfully.");
441         } else {
442             System.out.println("Student " +
    nameToDelete + " not found.");
443         }
444         break;
445     case 4:
446         System.out.println("Enter student ID to
    delete:");
447         String idToDelete = input.nextLine();
448         if (deleteStudentById(studentList,
    idToDelete)) {
449             System.out.println("Student with ID " +
    idToDelete + " deleted successfully.");
450         } else {
451             System.out.println("Student with ID " +
    idToDelete + " not found.");
452         }
453         break;
454     case 5:
455         System.out.println("Thank you for using the
```

```
455 Student Register!");
456         return;
457     default:
458         System.out.println("Invalid option, please
    try again.");
459         break;
460 }
461             NameExpr: choice
462               SimpleName: choice
463             SwitchEntry: case 1:
464     System.out.println("Enter student name:");
465     String name = input.nextLine();
466     System.out.println("Enter student ID:");
467     String id = input.nextLine();
468     System.out.println("Enter student major:");
469     String major = input.nextLine();
470     Student student = new Student(name, id, major);
471     studentList.add(student);
472     System.out.println("Student added successfully
    .");
473     break;
474
475               IntegerLiteralExpr: 1
476               ExpressionStmt: System.out.println("
    Enter student name:");
477                 MethodCallExpr: System.out.println("
    Enter student name:")
478                   FieldAccessExpr: System.out
479                     NameExpr: System
480                       SimpleName: System
481                     SimpleName: out
482                   SimpleName: println
483                   StringLiteralExpr: "Enter student
    name:"
484               ExpressionStmt: String name = input.
    nextLine();
485                 VariableDeclarationExpr: String name
     = input.nextLine()
486                   VariableDeclarator: name = input.
    nextLine()
487                     ClassOrInterfaceType: String
```

```
488                       SimpleName: String
489                     SimpleName: name
490                     MethodCallExpr: input.nextLine()
491                       NameExpr: input
492                         SimpleName: input
493                       SimpleName: nextLine
494               ExpressionStmt: System.out.println("
    Enter student ID:");
495                 MethodCallExpr: System.out.println("
    Enter student ID:")
496                   FieldAccessExpr: System.out
497                     NameExpr: System
498                       SimpleName: System
499                     SimpleName: out
500                   SimpleName: println
501                   StringLiteralExpr: "Enter student
    ID:"
502               ExpressionStmt: String id = input.
    nextLine();
503                 VariableDeclarationExpr: String id
     = input.nextLine()
504                   VariableDeclarator: id = input.
    nextLine()
505                     ClassOrInterfaceType: String
506                       SimpleName: String
507                     SimpleName: id
508                     MethodCallExpr: input.nextLine()
509                       NameExpr: input
510                         SimpleName: input
511                       SimpleName: nextLine
512               ExpressionStmt: System.out.println("
    Enter student major:");
513                 MethodCallExpr: System.out.println("
    Enter student major:")
514                   FieldAccessExpr: System.out
515                     NameExpr: System
516                       SimpleName: System
517                     SimpleName: out
518                   SimpleName: println
519                   StringLiteralExpr: "Enter student
    major:"
```

```
520                 ExpressionStmt: String major = input.
    nextLine();
521                   VariableDeclarationExpr: String
    major = input.nextLine()
522                     VariableDeclarator: major = input.
    nextLine()
523                       ClassOrInterfaceType: String
524                         SimpleName: String
525                       SimpleName: major
526                       MethodCallExpr: input.nextLine()
527                         NameExpr: input
528                           SimpleName: input
529                         SimpleName: nextLine
530                 ExpressionStmt: Student student = new
    Student(name, id, major);
531                   VariableDeclarationExpr: Student
    student = new Student(name, id, major)
532                     VariableDeclarator: student = new
    Student(name, id, major)
533                       ClassOrInterfaceType: Student
534                         SimpleName: Student
535                       SimpleName: student
536                       ObjectCreationExpr: new Student(
    name, id, major)
537                         ClassOrInterfaceType: Student
538                           SimpleName: Student
539                         NameExpr: name
540                           SimpleName: name
541                         NameExpr: id
542                           SimpleName: id
543                         NameExpr: major
544                           SimpleName: major
545                 ExpressionStmt: studentList.add(
    student);
546                   MethodCallExpr: studentList.add(
    student)
547                     NameExpr: studentList
548                       SimpleName: studentList
549                     SimpleName: add
550                     NameExpr: student
551                       SimpleName: student
```

```
552                ExpressionStmt: System.out.println("
    Student added successfully.");
553                  MethodCallExpr: System.out.println("
    Student added successfully.")
554                    FieldAccessExpr: System.out
555                      NameExpr: System
556                        SimpleName: System
557                      SimpleName: out
558                    SimpleName: println
559                    StringLiteralExpr: "Student added
    successfully."
560                BreakStmt: break;
561              SwitchEntry: case 2:
562    System.out.println("Student List:");
563    for (Student s : studentList) {
564        System.out.println("Name: " + s.getName
    () + ", ID: " + s.getId() + ", Major: " + s.getMajor
    ());
565    }
566    break;
567
568                IntegerLiteralExpr: 2
569                ExpressionStmt: System.out.println("
    Student List:");
570                  MethodCallExpr: System.out.println("
    Student List:")
571                    FieldAccessExpr: System.out
572                      NameExpr: System
573                        SimpleName: System
574                      SimpleName: out
575                    SimpleName: println
576                    StringLiteralExpr: "Student List:"
577                ForEachStmt: for (Student s :
    studentList) {
578    System.out.println("Name: " + s.getName() + ",
    ID: " + s.getId() + ", Major: " + s.getMajor());
579 }
580                  VariableDeclarationExpr: Student s
581                    VariableDeclarator: s
582                      ClassOrInterfaceType: Student
583                        SimpleName: Student
```

```
584                       SimpleName: s
585                   NameExpr: studentList
586                     SimpleName: studentList
587                   BlockStmt: {
588     System.out.println("Name: " + s.getName() + ",
    ID: " + s.getId() + ", Major: " + s.getMajor());
589 }
590                     ExpressionStmt: System.out.println
    ("Name: " + s.getName() + ", ID: " + s.getId() + ",
    Major: " + s.getMajor());
591                       MethodCallExpr: System.out.
    println("Name: " + s.getName() + ", ID: " + s.getId
    () + ", Major: " + s.getMajor())
592                         FieldAccessExpr: System.out
593                           NameExpr: System
594                             SimpleName: System
595                           SimpleName: out
596                         SimpleName: println
597                         BinaryExpr: "Name: " + s.
    getName() + ", ID: " + s.getId() + ", Major: " + s.
    getMajor()
598                           BinaryExpr: "Name: " + s.
    getName() + ", ID: " + s.getId() + ", Major: "
599                             BinaryExpr: "Name: " + s.
    getName() + ", ID: " + s.getId()
600                               BinaryExpr: "Name: " + s
    .getName() + ", ID: "
601                                 BinaryExpr: "Name
    : " + s.getName()
602                                   StringLiteralExpr: "
    Name: "
603                                   MethodCallExpr: s.
    getName()
604                                     NameExpr: s
605                                       SimpleName: s
606                                     SimpleName:
    getName
607                                 StringLiteralExpr: ",
    ID: "
608                               MethodCallExpr: s.getId
    ()
```

```
609                                 NameExpr: s
610                                   SimpleName: s
611                                 SimpleName: getId
612                               StringLiteralExpr: ",
    Major: "
613                             MethodCallExpr: s.getMajor()
614                               NameExpr: s
615                                 SimpleName: s
616                               SimpleName: getMajor
617                 BreakStmt: break;
618               SwitchEntry: case 3:
619     System.out.println("Enter student name to delete
    :");
620     String nameToDelete = input.nextLine();
621     if (deleteStudentByName(studentList,
    nameToDelete)) {
622         System.out.println("Student " + nameToDelete
     + " deleted successfully.");
623     } else {
624         System.out.println("Student " + nameToDelete
     + " not found.");
625     }
626     break;
627
628                 IntegerLiteralExpr: 3
629                 ExpressionStmt: System.out.println("
    Enter student name to delete:");
630                   MethodCallExpr: System.out.println("
    Enter student name to delete:")
631                     FieldAccessExpr: System.out
632                       NameExpr: System
633                         SimpleName: System
634                       SimpleName: out
635                     SimpleName: println
636                     StringLiteralExpr: "Enter student
    name to delete:"
637                 ExpressionStmt: String nameToDelete =
    input.nextLine();
638                   VariableDeclarationExpr: String
    nameToDelete = input.nextLine()
639                     VariableDeclarator: nameToDelete
```

```
639  = input.nextLine()
640                      ClassOrInterfaceType: String
641                        SimpleName: String
642                      SimpleName: nameToDelete
643                      MethodCallExpr: input.nextLine()
644                        NameExpr: input
645                          SimpleName: input
646                        SimpleName: nextLine
647                IfStmt: if (deleteStudentByName(
    studentList, nameToDelete)) {
648      System.out.println("Student " + nameToDelete
     + " deleted successfully.");
649  } else {
650      System.out.println("Student " + nameToDelete
     + " not found.");
651  }
652                  MethodCallExpr: deleteStudentByName(
    studentList, nameToDelete)
653                    SimpleName: deleteStudentByName
654                    NameExpr: studentList
655                      SimpleName: studentList
656                    NameExpr: nameToDelete
657                      SimpleName: nameToDelete
658                  BlockStmt: {
659      System.out.println("Student " + nameToDelete
     + " deleted successfully.");
660  }
661                    ExpressionStmt: System.out.println
    ("Student " + nameToDelete + " deleted successfully
    .");
662                      MethodCallExpr: System.out.
    println("Student " + nameToDelete + " deleted
    successfully.")
663                        FieldAccessExpr: System.out
664                          NameExpr: System
665                            SimpleName: System
666                          SimpleName: out
667                        SimpleName: println
668                        BinaryExpr: "Student " +
    nameToDelete + " deleted successfully."
669                          BinaryExpr: "Student " +
```

```
669 nameToDelete
670                             StringLiteralExpr: "
    Student "
671                             NameExpr: nameToDelete
672                               SimpleName: nameToDelete
673                           StringLiteralExpr: " deleted
     successfully."
674                 BlockStmt: {
675     System.out.println("Student " + nameToDelete
     + " not found.");
676 }
677                   ExpressionStmt: System.out.println
    ("Student " + nameToDelete + " not found.");
678                     MethodCallExpr: System.out.
    println("Student " + nameToDelete + " not found.")
679                       FieldAccessExpr: System.out
680                         NameExpr: System
681                           SimpleName: System
682                         SimpleName: out
683                       SimpleName: println
684                       BinaryExpr: "Student " +
    nameToDelete + " not found."
685                         BinaryExpr: "Student " +
    nameToDelete
686                           StringLiteralExpr: "
    Student "
687                           NameExpr: nameToDelete
688                             SimpleName: nameToDelete
689                           StringLiteralExpr: " not
    found."
690               BreakStmt: break;
691             SwitchEntry: case 4:
692     System.out.println("Enter student ID to delete
    :");
693     String idToDelete = input.nextLine();
694     if (deleteStudentById(studentList, idToDelete
    )) {
695         System.out.println("Student with ID " +
    idToDelete + " deleted successfully.");
696     } else {
697         System.out.println("Student with ID " +
```

```
697 idToDelete + " not found.");
698     }
699     break;
700 
701                 IntegerLiteralExpr: 4
702                 ExpressionStmt: System.out.println("
    Enter student ID to delete:");
703                   MethodCallExpr: System.out.println("
    Enter student ID to delete:")
704                     FieldAccessExpr: System.out
705                       NameExpr: System
706                         SimpleName: System
707                       SimpleName: out
708                     SimpleName: println
709                     StringLiteralExpr: "Enter student
    ID to delete:"
710                 ExpressionStmt: String idToDelete =
    input.nextLine();
711                   VariableDeclarationExpr: String
    idToDelete = input.nextLine()
712                     VariableDeclarator: idToDelete =
    input.nextLine()
713                       ClassOrInterfaceType: String
714                         SimpleName: String
715                       SimpleName: idToDelete
716                       MethodCallExpr: input.nextLine()
717                         NameExpr: input
718                           SimpleName: input
719                         SimpleName: nextLine
720                 IfStmt: if (deleteStudentById(
    studentList, idToDelete)) {
721     System.out.println("Student with ID " +
    idToDelete + " deleted successfully.");
722 } else {
723     System.out.println("Student with ID " +
    idToDelete + " not found.");
724 }
725                   MethodCallExpr: deleteStudentById(
    studentList, idToDelete)
726                     SimpleName: deleteStudentById
727                     NameExpr: studentList
```

```
728                       SimpleName: studentList
729                     NameExpr: idToDelete
730                       SimpleName: idToDelete
731                   BlockStmt: {
732     System.out.println("Student with ID " +
    idToDelete + " deleted successfully.");
733 }
734                     ExpressionStmt: System.out.println
    ("Student with ID " + idToDelete + " deleted
    successfully.");
735                       MethodCallExpr: System.out.
    println("Student with ID " + idToDelete + " deleted
    successfully.")
736                         FieldAccessExpr: System.out
737                           NameExpr: System
738                             SimpleName: System
739                           SimpleName: out
740                         SimpleName: println
741                         BinaryExpr: "Student with ID
     " + idToDelete + " deleted successfully."
742                           BinaryExpr: "Student with ID
     " + idToDelete
743                             StringLiteralExpr: "
    Student with ID "
744                             NameExpr: idToDelete
745                               SimpleName: idToDelete
746                           StringLiteralExpr: " deleted
     successfully."
747                   BlockStmt: {
748     System.out.println("Student with ID " +
    idToDelete + " not found.");
749 }
750                     ExpressionStmt: System.out.println
    ("Student with ID " + idToDelete + " not found.");
751                       MethodCallExpr: System.out.
    println("Student with ID " + idToDelete + " not
    found.")
752                         FieldAccessExpr: System.out
753                           NameExpr: System
754                             SimpleName: System
755                           SimpleName: out
```

```
756                         SimpleName: println
757                         BinaryExpr: "Student with ID
     " + idToDelete + " not found."
758                           BinaryExpr: "Student with ID
     " + idToDelete
759                             StringLiteralExpr: "
    Student with ID "
760                             NameExpr: idToDelete
761                               SimpleName: idToDelete
762                           StringLiteralExpr: " not
    found."
763                 BreakStmt: break;
764               SwitchEntry: case 5:
765     System.out.println("Thank you for using the
    Student Register!");
766     return;
767
768                 IntegerLiteralExpr: 5
769                 ExpressionStmt: System.out.println("
    Thank you for using the Student Register!");
770                   MethodCallExpr: System.out.println("
    Thank you for using the Student Register!")
771                     FieldAccessExpr: System.out
772                       NameExpr: System
773                         SimpleName: System
774                       SimpleName: out
775                     SimpleName: println
776                     StringLiteralExpr: "Thank you for
    using the Student Register!"
777                 ReturnStmt: return;
778               SwitchEntry: default:
779     System.out.println("Invalid option, please try
    again.");
780     break;
781
782                 ExpressionStmt: System.out.println("
    Invalid option, please try again.");
783                   MethodCallExpr: System.out.println("
    Invalid option, please try again.")
784                     FieldAccessExpr: System.out
785                       NameExpr: System
```

```
786                         SimpleName: System
787                       SimpleName: out
788                     SimpleName: println
789                     StringLiteralExpr: "Invalid option
    , please try again."
790                 BreakStmt: break;
791   MethodDeclaration: public static boolean
    deleteStudentByName(ArrayList<Student> studentList,
    String name) {
792     for (Student s : studentList) {
793         if (s.getName().equalsIgnoreCase(name)) {
794             studentList.remove(s);
795             return true;
796         }
797     }
798     return false;
799 }
800     Modifier: public
801     Modifier: static
802     SimpleName: deleteStudentByName
803     Parameter: ArrayList<Student> studentList
804       ClassOrInterfaceType: ArrayList<Student>
805         SimpleName: ArrayList
806         ClassOrInterfaceType: Student
807           SimpleName: Student
808       SimpleName: studentList
809     Parameter: String name
810       ClassOrInterfaceType: String
811         SimpleName: String
812       SimpleName: name
813     PrimitiveType: boolean
814     BlockStmt: {
815     for (Student s : studentList) {
816         if (s.getName().equalsIgnoreCase(name)) {
817             studentList.remove(s);
818             return true;
819         }
820     }
821     return false;
822 }
823       ForEachStmt: for (Student s : studentList) {
```

```
824     if (s.getName().equalsIgnoreCase(name)) {
825         studentList.remove(s);
826         return true;
827     }
828 }
829         VariableDeclarationExpr: Student s
830           VariableDeclarator: s
831             ClassOrInterfaceType: Student
832               SimpleName: Student
833             SimpleName: s
834         NameExpr: studentList
835           SimpleName: studentList
836         BlockStmt: {
837     if (s.getName().equalsIgnoreCase(name)) {
838         studentList.remove(s);
839         return true;
840     }
841 }
842           IfStmt: if (s.getName().equalsIgnoreCase(
    name)) {
843     studentList.remove(s);
844     return true;
845 }
846             MethodCallExpr: s.getName().
    equalsIgnoreCase(name)
847               MethodCallExpr: s.getName()
848                 NameExpr: s
849                   SimpleName: s
850                 SimpleName: getName
851               SimpleName: equalsIgnoreCase
852               NameExpr: name
853                 SimpleName: name
854             BlockStmt: {
855     studentList.remove(s);
856     return true;
857 }
858               ExpressionStmt: studentList.remove(s);
859                 MethodCallExpr: studentList.remove(s
    )
860                   NameExpr: studentList
861                     SimpleName: studentList
```

```
862                     SimpleName: remove
863                     NameExpr: s
864                       SimpleName: s
865                 ReturnStmt: return true;
866                   BooleanLiteralExpr: true
867       ReturnStmt: return false;
868         BooleanLiteralExpr: false
869   MethodDeclaration: public static boolean
    deleteStudentById(ArrayList<Student> studentList,
    String id) {
870     for (Student s : studentList) {
871         if (s.getId().equalsIgnoreCase(id)) {
872             studentList.remove(s);
873             return true;
874         }
875     }
876     return false;
877 }
878     Modifier: public
879     Modifier: static
880     SimpleName: deleteStudentById
881     Parameter: ArrayList<Student> studentList
882       ClassOrInterfaceType: ArrayList<Student>
883         SimpleName: ArrayList
884         ClassOrInterfaceType: Student
885           SimpleName: Student
886       SimpleName: studentList
887     Parameter: String id
888       ClassOrInterfaceType: String
889         SimpleName: String
890       SimpleName: id
891     PrimitiveType: boolean
892     BlockStmt: {
893     for (Student s : studentList) {
894         if (s.getId().equalsIgnoreCase(id)) {
895             studentList.remove(s);
896             return true;
897         }
898     }
899     return false;
900 }
```

```
901       ForEachStmt: for (Student s : studentList) {
902     if (s.getId().equalsIgnoreCase(id)) {
903         studentList.remove(s);
904         return true;
905     }
906 }
907         VariableDeclarationExpr: Student s
908           VariableDeclarator: s
909             ClassOrInterfaceType: Student
910               SimpleName: Student
911             SimpleName: s
912         NameExpr: studentList
913           SimpleName: studentList
914         BlockStmt: {
915     if (s.getId().equalsIgnoreCase(id)) {
916         studentList.remove(s);
917         return true;
918     }
919 }
920           IfStmt: if (s.getId().equalsIgnoreCase(id
    )) {
921     studentList.remove(s);
922     return true;
923 }
924             MethodCallExpr: s.getId().
    equalsIgnoreCase(id)
925               MethodCallExpr: s.getId()
926                 NameExpr: s
927                   SimpleName: s
928                 SimpleName: getId
929               SimpleName: equalsIgnoreCase
930               NameExpr: id
931                 SimpleName: id
932             BlockStmt: {
933     studentList.remove(s);
934     return true;
935 }
936               ExpressionStmt: studentList.remove(s);
937                 MethodCallExpr: studentList.remove(s
    )
938                   NameExpr: studentList
```

```
939                       SimpleName: studentList
940                     SimpleName: remove
941                     NameExpr: s
942                       SimpleName: s
943                 ReturnStmt: return true;
944                   BooleanLiteralExpr: true
945         ReturnStmt: return false;
946           BooleanLiteralExpr: false
947
948 Class name: Student
949 Class AST:
950 ClassOrInterfaceDeclaration: class Student {
951
952     private String name;
953
954     private String id;
955
956     private String major;
957
958     public Student(String name, String id, String
    major) {
959         this.name = name;
960         this.id = id;
961         this.major = major;
962     }
963
964     public String getName() {
965         return name;
966     }
967
968     public String getId() {
969         return id;
970     }
971
972     public String getMajor() {
973         return major;
974     }
975 }
976   SimpleName: Student
977   FieldDeclaration: private String name;
978     Modifier: private
```

```
 979     VariableDeclarator: name
 980       ClassOrInterfaceType: String
 981         SimpleName: String
 982       SimpleName: name
 983   FieldDeclaration: private String id;
 984     Modifier: private
 985     VariableDeclarator: id
 986       ClassOrInterfaceType: String
 987         SimpleName: String
 988       SimpleName: id
 989   FieldDeclaration: private String major;
 990     Modifier: private
 991     VariableDeclarator: major
 992       ClassOrInterfaceType: String
 993         SimpleName: String
 994       SimpleName: major
 995   ConstructorDeclaration: public Student(String
     name, String id, String major) {
 996     this.name = name;
 997     this.id = id;
 998     this.major = major;
 999 }
1000     Modifier: public
1001     SimpleName: Student
1002     Parameter: String name
1003       ClassOrInterfaceType: String
1004         SimpleName: String
1005       SimpleName: name
1006     Parameter: String id
1007       ClassOrInterfaceType: String
1008         SimpleName: String
1009       SimpleName: id
1010     Parameter: String major
1011       ClassOrInterfaceType: String
1012         SimpleName: String
1013       SimpleName: major
1014     BlockStmt: {
1015     this.name = name;
1016     this.id = id;
1017     this.major = major;
1018 }
```

```
1019       ExpressionStmt: this.name = name;
1020         AssignExpr: this.name = name
1021           FieldAccessExpr: this.name
1022             ThisExpr: this
1023             SimpleName: name
1024           NameExpr: name
1025             SimpleName: name
1026       ExpressionStmt: this.id = id;
1027         AssignExpr: this.id = id
1028           FieldAccessExpr: this.id
1029             ThisExpr: this
1030             SimpleName: id
1031           NameExpr: id
1032             SimpleName: id
1033       ExpressionStmt: this.major = major;
1034         AssignExpr: this.major = major
1035           FieldAccessExpr: this.major
1036             ThisExpr: this
1037             SimpleName: major
1038           NameExpr: major
1039             SimpleName: major
1040   MethodDeclaration: public String getName() {
1041     return name;
1042 }
1043     Modifier: public
1044     SimpleName: getName
1045     ClassOrInterfaceType: String
1046       SimpleName: String
1047     BlockStmt: {
1048     return name;
1049 }
1050       ReturnStmt: return name;
1051         NameExpr: name
1052           SimpleName: name
1053   MethodDeclaration: public String getId() {
1054     return id;
1055 }
1056     Modifier: public
1057     SimpleName: getId
1058     ClassOrInterfaceType: String
1059       SimpleName: String
```

```
    BlockStmt: {
    return id;
}
      ReturnStmt: return id;
        NameExpr: id
          SimpleName: id
  MethodDeclaration: public String getMajor() {
    return major;
}
    Modifier: public
    SimpleName: getMajor
    ClassOrInterfaceType: String
      SimpleName: String
    BlockStmt: {
    return major;
}
      ReturnStmt: return major;
        NameExpr: major
          SimpleName: major

No class AST found for StudentRegister
No class AST found for Student
Total number of tokens: 2158
CASE: 5
FALSE: 2
PACKAGE: 1
DOT: 68
RETURN: 8
TRUE: 3
STATIC: 3
STRING_LITERAL: 27
RBRACKET: 1
NEW: 3
_DEFAULT: 1
IF: 4
INT: 1
FOR: 3
LBRACE: 20
PRIVATE: 3
SWITCH: 1
ELSE: 2
```

```
1101 CLASS: 2
1102 GT: 4
1103 VOID: 1
1104 LT: 4
1105 PLUS: 13
1106 EOF: 1
1107 UNIX_EOL: 118
1108 SPACE: 1410
1109 LBRACKET: 1
1110 THIS: 3
1111 BOOLEAN: 2
1112 RPAREN: 59
1113 ASSIGN: 12
1114 WHILE: 1
1115 SEMICOLON: 56
1116 COMMA: 8
1117 COLON: 9
1118 IDENTIFIER: 199
1119 IMPORT: 2
1120 PUBLIC: 7
1121 LPAREN: 59
1122 BREAK: 5
1123 RBRACE: 20
1124 SINGLE_LINE_COMMENT: 1
1125 INTEGER_LITERAL: 5
1126
1127 Process finished with exit code 0
1128
```