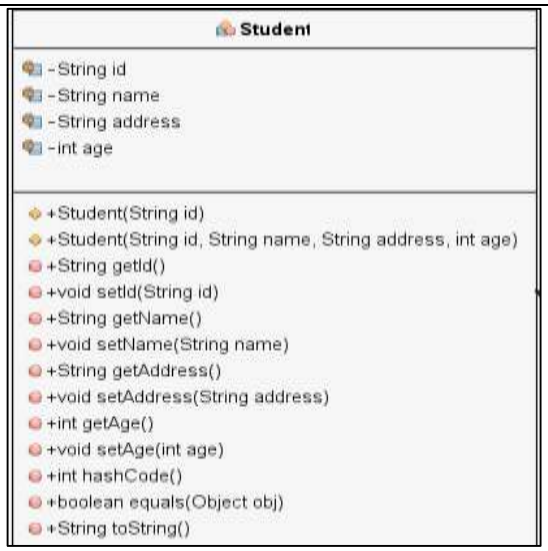
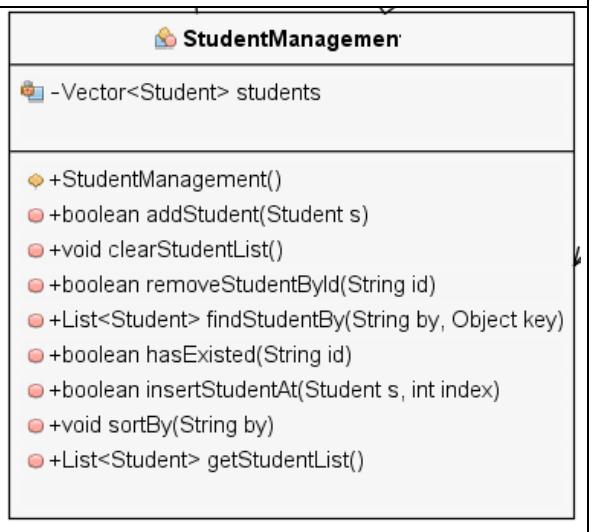
Bài 7: BÀI TẬP MINH HỌA THAO TÁC VỚI VECTOR

Vector và đối tượng tự tạo.

Bài toán	Gợi ý thực hiện
Tạo một Vector lưu trữ danh sách sinh viên (mã sinh viên, tên sinh viên, địa chỉ, tuổi).	- Tạo lớp cơ sở Student - Tạo lớp StudentManagement chứa danh sách sinh viên (sử dụng vector) lưu trữ và xử lý thông tin.
Nội dung thực hiện	**StudentManagement**
1. Thêm sinh viên vào danh sách: có kiểmt ra trùng khóa.	addStudent(..)
2. Tìm kiếm sinh viên tùy theo lựa chọn: tìm theo mã, theo tên, theo tuổi.	findStudentByID(…)
3. Chèn thêm 1 sinh viên vào danh sách đã có theo vị trí chỉ định	insertStudentAt(…)
4. In danh sách sinh viên	
5. Xóa sinh viên theo mã nhập vào.	removeStudentByID(…)
6. Xóa danh sách sinh viên	clearStudentList()
7. Sắp xếp sinh viên theo tiêu chí chỉ định: mã, tên, tuổi.	sortBy(…)

Sơ đồ mô hình hóa





Để thực hiện quản lý sinh viên, xây dựng chương trình chính chứa hàm main hiển thị bảng chọn. Lớp **StudnentManagementDemo.java**:

Menu chương trình chính:

=====Student Management System ===== Select a option below: 1) Add new student 2) Remove a student 3) Find a student 4) Insert new student 5) Sort student list 6) Print student list 7) Clear student list 8) Exit	Hiển thị lặp lại cho mỗi lựa chọn trong bảng chọn chính thực hiện chương trình

Menu phụ lựa chọn tiếp tục hay không chức năng đang thực hiện

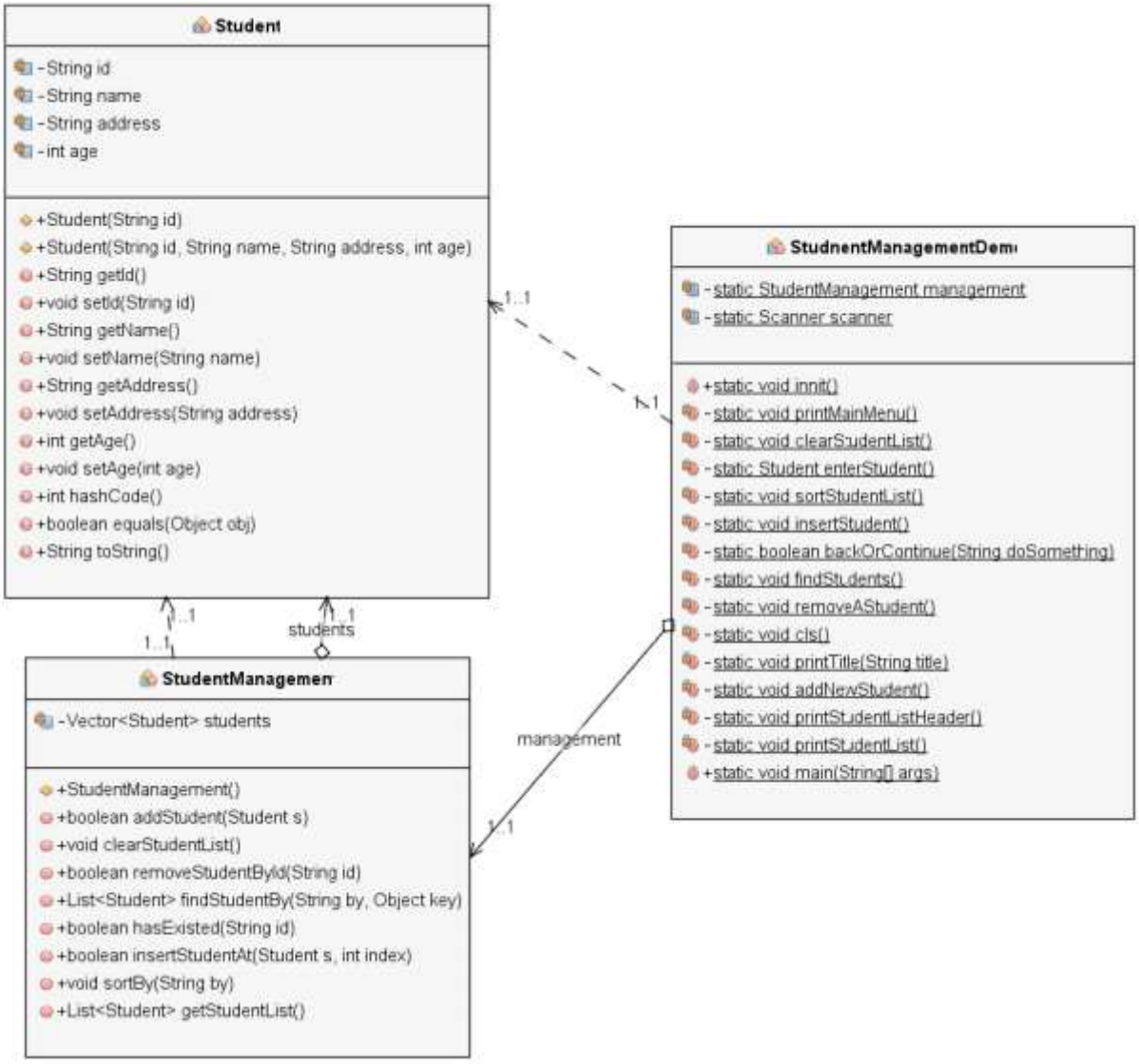
Select the option: 1) Continue …. 2) Return home	Hiển thị sau mỗi thao tác cho phép thực hiện lại lựa chọn vừa thực hiện

Các nhiệm vụ thực hiện trong bảng chọn chính

Select your option: 1 **Add new student**	Phương thức : **addNewStudent()** **Thực hiện:** Thêm 1 sinh viên vào danh sách
Select your option: 2 **Remove a student**	Phương thức : **removeAStudent()** **Thực hiện:** Xóa sinh viên khỏi danh sách
Select your option: 3 **Find a student**	Phương thức : **findStudents()** **Thực hiện: Tìm sinh viên theo yêu cầu**
Select your option: 4 **Insert new student**	Phương thức : **insertStudent**() **Thực hiện:** chèn thêm sinh viên vào vị trí
Select your option: 5 **Sort student list**	Phương thức : **sortStudentList**() **Thực hiện:** sắp xếp danh sách
Select your option: 6 **Print student list**	Phương thức : **printStudentList**(); **Thực hiện**: in danh sách
Select your option: 7 **Clear student list**	Phương thức : **clearStudentList**(); **Thực hiện**: Xóa danh sách
Select your option: 8 **Exit**	Phương thức : **System.exit(0)**; **Thực hiện**: Kết thúc chương trình

Mô hình hóa lớp chứa hàm main.

Sơ đồ tổng thể thực hiện bài toán.



Kết quả thực hiện chương trình:

====Student Management System==== Select a option below: 1) Add new student 2) Remove a student 3) Find a student 4) Insert new student 5) Sort student list 6) Print student list 7) Clear student list 8) Exit	Menu chính Chọn các số 1,2,3,4,5,6,7,8 từ bàn phím sẽ gọi các nhiệm vụ tương ứng như mô tả trê.
Lựa chọn 1: =====Add new student===== Enter the information for new student: - Student id(must be unique):001 - Student name:Nguyen Van A - Student address:Ha noi - Student age:20 New student successfully added!	Hiển thị nội dung nhập thông tin sinh viên
Select the option: 1) Continue adding new student 2) Return home 1 Enter the information for new student: - Student id(must be unique):002 - Student name:Nguyen Van Ha - Student address:Son Tay - Student age:22 New student successfully added! Select the option: 1) Continue adding new student 2) Return home 2	Hiển thị bảng chọn phụ cho phép tiếp tục thao tác thêm hay thực hiện kết thúc. - Nếu chọn 1, chương trình tiếp tục cho nhập thông tin sv 2 - Nếu chọn 2 chương trình cho phép trở lại menu chính.

Lựa chọn 6: in thông tin

Select your option: 6 ==========Student List========== index id name age address	Hiển thị kết quả theo danh sách

0 001 Nguyen Van A 20 Ha noi 1 002 Nguyen Van Ha 22 Son Tay	
Select the option: 1) Continue printing student list 2) Return home 2	Hiển thị bảng chọn phụ cho phép tiếp tục thao tác thêm hay thực hiện kết thúc

Lựa chọn 3:

Select your option: 3 ====Find Students==== Select the field to find with: 1) By id 2) By name 3) By age 1 Keyword: 001 Total founds: 1 index id name age address 0 001 Nguyen Van A 20 Ha noi
Select the option: 1) Continue finding 2) Return home 2

Lựa chọn 4

Select your option: 4 ======Insert Student====== Enter the index(0-1): 1 Enter the information for new student: - Student id(must be unique):003 - Student name:Truong Thi May - Student address:Ha Nam - Student age:19 Student successfully inserted!
Select the option: 1) Continue inserting student 2) Return home 2

Gọi lại lựa chọn 6 để kiểm tra kết quả

Lựa chọn 5

Select your option: 5 ==========Sort Student List========== Select the field to sort by: 1) By id 2) By name 3) By age 2 Student list successfully sorted! Select the option: 1) Continue sorting student list 2) Return home 2
Gọi lựa chọn 6 để kiểm tra kết quả sắp xếp

GỢI Ý MÃ HÓA BÀI TOÁN

Xây dựng lớp **Student.java**

```
public class Student{
    private String id;
    private String name;
    private String address;
    private int age;
```

```
/**
* hàm tạo
* @param id  tạo đối tượng chỉ ID có giá trị
*/
public Student(String id) {
      this.id = id;
      }
```

```
/**
* tạo đối tượng có đầy đủ tham số
* @param id: truyền cho mã sinh viên
* @param name: truyền cho tên
* @param address: truyền cho địa chỉ
* @param age : truyền cho tuổi
*/
public Student(String id, String name, String address, int age) {
        this(id);
        this.name = name;
        this.address = address;
```

```
        this.age = age;
}
```

```
public String getId() {
        return id;
    }
public void setId(String id) {
        this.id = id;
    }
public String getName() {
        return name;
    }
public void setName(String name) {
        this.name = name;
    }
public String getAddress() {
        return address;
    }
public void setAddress(String address) {
        this.address = address;
    }
public int getAge() {
        return age;
    }
public void setAge(int age) {
        this.age = age;
    }
```

```
/**
* xác định mã hashcode của đối tượng xác định theo id
* @return
*/
@Override
public int hashCode() {
      return id.hashCode();
    }
```

```
/**
* xác thuộc tính phân biệt sinh viên theo ID
* @param obj
* @return
*/
@Override
public boolean equals(Object obj) {
        if(super.equals(obj))    return true;
        if(!(obj instanceof Student)) return false;
        Student other=(Student)obj;
        return id.equals(other.id);
}
```

```
/**
* @return giá trị kết xuất thông tin đối tượng
*/
@Override
public String toString() {
      return
       String.format("%15s%15s%15s%15s",id,name,address,age);
     }
}
```

StudentManagement.java

```
public class StudentManagement {
```

```
private Vector<Student> students;
```

```
/**
* hàm tạo danh sách khởi tạo tập hợp
*/
public StudentManagement() {
      students = new Vector<>();
}
```

```
/**
* thêm sinh viên vào danh sách
* @param s
* @return true nếu thêm được và ngược lại.
*/
public boolean addStudent(Student s) {
      if (students.contains(s))
            return false;
      students.addElement(s);
      return true;
}
```

```
/**
* xóa toàn bộ danh sách.
*/
public void clearStudentList() {
            students.clear();
     }
```

```
/**
* @param id: mã sinh viên cần xóa
* @return true nếu xóa thành công và ngược lại
*/
public boolean removeStudentById(String id) {
          return students.remove(new Student(id));
     }
```

```
/**
* tìm kiếm sinh viên có
* @param by: tiêu chí tìm kiếm. nếu by=id -> tìm theo ID
* by=name thì tiêu chí tìm kiếm theo tên,
by= age-> tìm kiếmtheo tuổi.
* @param key gía trị cần tìm. Vì chưa rõ kiểu dữ liệu nên đặt kiểu
chung Object
* @return Danh sách các phần tử có tiêu chi tìm phù hợp
*/
public List<Student> findStudentBy(String by,Object key){
    List<Student> results=new Vector<>();
    for (Student s : students) {
        if((by.equals("id")&&s.getId().equals(key))||
           (by.equals("name")&&          s.getName().equals(key))||
           (by.equals("age")&&s.getAge()==(int)key)) {
           results.add(s);
         }
    }
    return results;
}
```

```
/**
* @param id: mã sinh viên cần kiểm tra
* @return true: nếu kiểm tra mã sinh viên đã tồn tại trong danh sách
*/
public boolean hasExisted(String id) {
        return students.contains(new Student(id));
    }
```

```
/**
*Chèn Student s vào vị trí index
* @param s sinh viên cần chèn
* @param index- vị trí chèn
* @return true nếu chèn được
*/
public boolean insertStudentAt(Student s, int index) {
     if (students.contains(s) || index < 0 || index >=
     students.size())
            return false;
            students.insertElementAt(s, index);
            return true;
    }
```

```
/**
* @param by : sắp xếp danh sách theo tiêu chí lựa chọn:
* theo tên, theo mã hay theo tuổi
*/
public void sortBy(String by) {
     //tạo biến so sánh theo tiêu chí mã
```

```
      Comparator<Student> byIdComparator= new
      Comparator<Student>() {
      @Override
      public int compare(Student o1, Student o2) {
            return o1.getId().compareTo(o2.getId());
      }
      };
     //tạo biến so sánh theo tiêu chí tên
     Comparator<Student> byNameComparator=
            new Comparator<Student>() {
           @Override
           public int compare(Student o1, Student o2) {
            return o1.getName().compareTo(o2.getName());
             }
        };
        //tạo biến so sánh theo tiêu chí tuổi
      Comparator<Student> byAgeComparator=
                  new Comparator<Student>() {
             @Override
             public int compare(Student o1, Student o2) {
                 return
      Integer.compare(o1.getAge(),o2.getAge());
             }
        };
        //lựa chọn tiêu chí sắp và thực hiện lựa chọn tương ứng
        switch (by) {
        case "id":
            students.sort(byIdComparator);
            break;
        case "name":
            students.sort(byNameComparator);
            break;
        case "age":
            students.sort(byAgeComparator);
            break;
        default:
            throw new UnsupportedOperationException("Sorting by %s
field unsupported!"+(by));
        }
    }
```

```
/**
 * @return Danh sách sv không cho phép sửa đổi
 */
  public List<Student> getStudentList(){
        return Collections.unmodifiableList(students);
    }
```

```
}//end of class
```

Lớp chứa hàm main vận hành bài toán **StudnentManagementDemo.java**

```
public class StudnentManagementDemo {
    private static StudentManagement management;
    private static Scanner scanner;
/**
* khởi gán các giá trị tổng thể.
*/
public static void innit() {
        management = new StudentManagement();
        scanner = new Scanner(System.in);
    }
/**
* Bảng chọn chính chương trình.
*/
private static void printMainMenu() {
        printTitle("Student Management System");
        System.out.println("Select a option below:");
        System.out.println("1)\tAdd new student");
        System.out.println("2)\tRemove a student");
        System.out.println("3)\tFind a student");
        System.out.println("4)\tInsert new student");
        System.out.println("5)\tSort student list");
        System.out.println("6)\tPrint student list");
        System.out.println("7)\tClear student list");
        System.out.println("8)\tExit");
    }
/**
 * xóa sạch danh sách sinh viên.
 */
private static void clearStudentList() {
    management.clearStudentList();
    System.out.println("Student list successfully cleared!");
}
/**
 * thêm sinh viên vào danh sách.
 * @return sinh viên được thêm vào nếu thông tin mã không bị
trùng.
 */
private static Student enterStudent() {
```

```
    System.out.println("Enter the information for new student:");
    System.out.print("- Student id(must be unique):");
    String id = scanner.nextLine();
    if (management.hasExisted(id))
            return null;
    System.out.print("- Student name:");
    String name = scanner.nextLine();
    System.out.print("- Student address:");
    String address = scanner.nextLine();
    System.out.print("- Student age:");
    int age = scanner.nextInt();
    scanner.nextLine();
    return new Student(id, name, address, age);
}
```

```
/**
 * sắp xếp danh sách sinh viên, cho phép lựa chọn 2 tùy biến
 * 1. sắp theo id
 * 2. sắp theo tên
 * 3. sắp theo tuổi.
 */
private  static void sortStudentList() {
    boolean run = true;
    while (run) {
        printTitle("Sort Student List");
        System.out.println("Select the field to sort by:");
        System.out.println("1)\tBy id");
        System.out.println("2)\tBy name");
        System.out.println("3)\tBy age");
        //nhập lựa chọn
        int opt = scanner.nextInt();
          //bỏ qua ký tự trắng cuối nếu tiếp sau nhập xâu
        scanner.nextLine();;
        if (opt < 1 || opt > 3) {
            System.out.println("Invalid option");
            //bỏ qua lần này và tiếp theo với lần lặp mới
            continue;
        } else {
            management.sortBy(opt == 1 ? "id" : opt == 2 ? "name" : "age");
            System.out.println("Student list successfully sorted!");
```

```
        }
        //hỏi lại diễn biến công việc
        run = backOrContinue("sorting student list");
    }
}
```

```
/**
 * thêm sinh viên vào danh sách tại vị trí index được chỉ ra.
 * Quá trình dừng lại nếu không muốn thực hiện tiếp
 */
private static void insertStudent() {
    boolean run = true;
    while (run) {
        printTitle("Insert Student");
        System.out.printf("Enter the index(0-%d%n):
",management.getStudentList().size() - 1);
        int index = scanner.nextInt();
        scanner.nextLine();
        //sinh viên sẽ được chèn vào
        Student s = enterStudent();
        if (s == null) {
            System.out.println("Student id has existed.  Try
again!");
          //bỏ qua lần lặp hiện thời
            continue;
        } else {
            //chèn s vào vị trí index
            management.insertStudentAt(s, index);
            System.out.println("Student successfully
inserted!");
        }
        //hỏi lại diễn biến công việc
        run = backOrContinue("inserting student");
    }
}
```

```
/**
 * @param doSomething: 1 nếu tiếp tục, 2 sẽ dừng lại trở về
trang chính
 * @return true nếu tiếp tục và ngược lại
 */
private static boolean backOrContinue(String doSomething) {
    int opt=0;//biến nhận lựa chọn
    System.out.println("Select the option:");
```

```
    System.out.println("1) Continue " + doSomething);
    System.out.println("2) Return home");
    //quá trình thực hiện sẽ dừng lại khi chọn 2.
    while(true) {
        opt = scanner.nextInt();
        scanner.nextLine();
        if (opt == 1)
            return true;
        else if (opt == 2)
            return false;
        else
            System.out.println("Invalid option!");
    }
}
```

```
/**
* tìm sinh viên theo lựa chọn:
* 1 theo id; 2 theo name; 3 theo age.
* hàm thực hiện in toàn bộ thông tin tìm được theo tiêu chí
được lựa chọn.
* Quá trình dừng lại nếu không muốn thực hiện tiếp
*/
private static void findStudents() {
    boolean run = true;//biến nhận lựa chọn điều khiển lặp
    while (run) {
        cls();//xóa sạch cửa sổ lệnh.
        printTitle("Find Students");
        //Hiển thị menu phụ cho chọn tiêu chí tìm kiếm
        System.out.println("Select the field to find with:");
        System.out.println("1)\tBy id");
        System.out.println("2)\tBy name");
        System.out.println("3)\tBy age");
        int opt = scanner.nextInt();
        scanner.nextLine();//bỏ qua dòng nhập hiện tại
        List<Student> results = null;
        System.out.print("Keyword: ");
        if (opt == 1 || opt == 2) {
           String key = scanner.nextLine();
           results = management.findStudentBy(opt == 1 ? "id"
: "name", key);
        } else if (opt == 3) {
           int age = scanner.nextInt();
           scanner.nextLine();
```

```
            results = management.findStudentBy("age", age);
        } else {
            System.out.println("Invalid option!");
            continue;//bỏ qua lần lặp hiện thời nếu lựa chọn
không đúng
        }
        System.out.println("Total founds: " + results.size());
        printStudentListHeader();
        for (int i = 0; i < results.size(); i++) {
            Student s = results.get(i);
            System.out.printf("%5d%15s%15s%15s%15s%n",
                i,   s.getId(),   s.getName(),   s.getAge(),
s.getAddress());
        }
        //hỏi lại diễn biến công việc
        run = backOrContinue("finding");
    }
}
```

```
/**
* Xóa sinh viên trong danh sách theo mã
* Quá trình dừng lại nếu không muốn thực hiện tiếp
*/
private static void removeAStudent() {
    printTitle("Remove A Student");
    boolean run = true;
    while (run) {
        System.out.print("Enter student id: ");
        String id = scanner.nextLine();
        //gọi   phương   thức   xóa   được   viết   trong
SutudentManagement
        if (management.removeStudentById(id)) {
            System.out.println("Student          successfully
removed!");
        } else {
            System.out.println("Student not found!");
        }
        //hỏi lại diễn biến công việc
        run = backOrContinue("removing student");
    }
}
```

```
/**
 * xóa cửa sổ lệnh và làm sạch bộ đệm
```

```
 */
private static void cls() {
    System.out.print("\033[H\033[2J");
    System.out.flush();
}
```

```
private static void printTitle(String title) {
    System.out.println("====" + title + "====");
}
```

```
/**
 * thêm danh sách sinh viên.
 * Quá trình dừng lại nếu không muốn thực hiện tiếp
 */
private static void addNewStudent() {
    cls();
    printTitle("Add new student");
    boolean run = true;
    do {
        Student s = enterStudent();
        if (s == null) {
                System.out.println("Student  id  has  existed.
Try again!");
                continue;
        }
        management.addStudent(s);
        System.out.println("New student successfully added!");
        //hỏi lại diễn biến công việc
        run = backOrContinue("adding new student");
    } while (run);
}
```

```
private static void printStudentListHeader() {
    System.out.printf("%5s%15s%15s%15s%15s%n", "index", "id",
"name", "age", "address");
}
```

```
/**
 * In danh sách sinh viên.
 * Quá trình dừng lại nếu không muốn thực hiện tiếp
 */
private static void printStudentList() {
    printTitle("Student List");//in tiêu đề
    boolean run = true;
    while (run) {
        cls();
```

```
        printStudentListHeader();//in thông tin tiêu đề
        for (int i = 0; i < management.getStudentList().size();
i++) {
         Student s = management.getStudentList().get(i);
         System.out.printf("%5d%15s%15s%15s%15s%n",        i,
s.getId(), s.getName(), s.getAge(), s.getAddress());
        }
        //hỏi lại diễn biến công việc
        run = backOrContinue("printing student list");
    }
}
```

```
/**
 * thực thi các nhiệm vụ chính.
 * Cho hiển thị bảng chọn chính và triệu gọi các
 * @param args
 */
public static void main(String[] args) {
    innit();//gọi khởi tạo giá trị tổng thể
    int opt;//biến nhận lựa chọn trong menu chính
    do {
        cls();//xóa cửa sổ lệnh
        printMainMenu();//in menu chính
        System.out.print("Select your option: ");
        opt = scanner.nextInt();//nhập lựa chọn
        scanner.nextLine();//bỏ qua dòng hiện tại
        switch (opt) {
        case 1:
                addNewStudent();//rẽ nahnh lựa chọn 1
                break;
        case 2:
                removeAStudent(); //rẽ nahnh lựa chọn 2
                break;
        case 3:
                findStudents();//rẽ nahnh lựa chọn 3
                break;
        case 4:
                insertStudent();//rẽ nahnh lựa chọn 4
                break;
        case 5:
                sortStudentList(); //rẽ nahnh lựa chọn 5
                break;
        case 6:
```

```
                printStudentList(); //rẽ nahnh lựa chọn 6
                break;
        case 7:
                clearStudentList(); //rẽ nahnh lựa chọn 7
                break;
        case 8:
                System.exit(0);//kết thúc chương trình
        default:
                System.out.println("Invalid option");
                break;
        }
    } while (opt != 8);
}//end of while
}//end of class
```