Q4. *(This exercise is based on a previous mock question by by Dr. W.K. Chan / CS2312 [2013-2014 Sem A])*

Study the 4 given classes `Course`, `AddController`, `DropController`, `Programme`, and the `main()` method for testing. This program maintains a list of courses that should be taken in the programme. When a course is added in the programme, all the pre-requisite courses of this course (in preRequisites) must be added into the programme.

```
public class Course
{
  private String name;
  private List<Course> preRequisites;

  public Course(String cName) {
    name = cName;
    preRequisites = new ArrayList<Course>();
  }

  @Override
  public String toString() {return name;}

  public void addPreReq(Course p) {
    if (!preRequisites.contains(p))
        preRequisites.add(p);
  }

  public List<Course> getPreRequisites() {
    return this.preRequisites;
  }
}
```

```
public class AddController
{
  private List<Course> courseList;

  public AddController(List<Course> list) {
    courseList = list;
  }

  public void printCourses() {
    System.out.print("Course List: ");
    for(Course c : courseList) {
      System.out.printf("[%s] ",c);
    }
    System.out.println();
  }

  public void process(Course c) {
    /* PLACE YOUR CODE HERE */
  }
}
```

```
public class DropController
{
  private List<Course> courseList;

  public DropController(List<Course> list) {
    courseList = list;
  }

  public void printCourses() {
    System.out.print("Course List: ");
    for(Course c : courseList) {
      System.out.printf("[%s] ",c);
    }
    System.out.println();
  }

  public void process(Course c) {
    /* PLACE YOUR CODE HERE */
  }
}
```

```
public class Programme
{
  private List<Course> courseList;
  private AddController addAgent;
  private DropController dropAgent;

  public Programme() {
    courseList = new ArrayList<Course>();
    addAgent = new AddController(courseList);
    dropAgent = new DropController(courseList);
  }

  public void add(Course c) {
    addAgent.process(c);
    addAgent.printCourses();
  }

  public void drop(Course c) {
    dropAgent.process(c);
    dropAgent.printCourses();
  }
}
```

```
public class Main
{
  public static void main(String[] args)
  {
    Course CS1102 = new Course("CS1102");
    Course CS2311 = new Course("CS2311");
    CS2311.addPreReq(CS1102);
    Course CS2312 = new Course("CS2312");
    CS2312.addPreReq(CS2311);
    Course CS2115 = new Course("CS2115");
    Course CS3103 = new Course("CS3103");
    CS3103.addPreReq(CS2115);
    CS3103.addPreReq(CS2312);
    Course CS3334 = new Course("CS3334");
    CS3334.addPreReq(CS2311);

    Programme programme = new Programme();
    programme.add(CS3334);
    programme.drop(CS2312);
    programme.drop(CS2311);
    programme.drop(CS3334);
  }
}
```

Step 1.

Complete the `process` method in `AddController`.

```
public void process(Course c) {
    /* PLACE YOUR CODE HERE */

    Approach:
    If c is already in the courseList, ignore and return
    For each course p in the preRequisite list of the course c
        process p to add p and p's prerequisites to the courseList
    Add the course c to the courseList
}
```

Step 2.

Complete the process method in DropController, as given below:

```
public void process(Course c) {
    /* PLACE YOUR CODE HERE */
    At the moment we simply remove the unwanted course (We will apply further checking in Step 5):
    courseList.remove(c);
}
```

Step 3.

Test the program. Study the output. The expected result is shown below.

```
public class Main {

  public static void main(String[] args)
  {
    Course CS1102 = new Course("CS1102");
    Course CS2311 = new Course("CS2311");
    Course CS2312 = new Course("CS2312");
    Course CS2115 = new Course("CS2115");
    Course CS3103 = new Course("CS3103");
    Course CS3334 = new Course("CS3334");
    CS2311.addPreReq(CS1102);
    CS2312.addPreReq(CS2311);
    CS3103.addPreReq(CS2115);
    CS3103.addPreReq(CS2312);
    CS3334.addPreReq(CS2311);

    Programme programme = new Programme();
    programme.add(CS3334);
    programme.drop(CS2312); //not exist in the programme.
    programme.drop(CS2311);
    programme.drop(CS3334);
  }
}
```

```
Output:
Course List: [CS1102] [CS2311] [CS3334]
Course List: [CS1102] [CS2311] [CS3334]
Course List: [CS1102] [CS3334]
Course List: [CS1102]
```

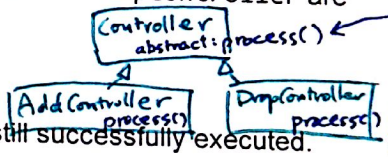
Step 4.

The data field courseList and the method printCourses() in AddController and DropController are the same. Remove the duplicated code.

i. Figure out the abstract class Controller.

ii. Put courseList and printCourses() into Controller.

iii. Revise AddController and DropController to make the program still successfully executed.



Step 5.

ArrayList is used in the constructors of Course and Programme. Change them to use LinkedList. (You need to change to import java.util.LinkedList.)
You should find that it quickly works without other amendments in other places of the whole program.

Apply the @Override annotation where appropriate (Self-check: You should do so for toString of Course, and for the process methods in AddController and DropController.)

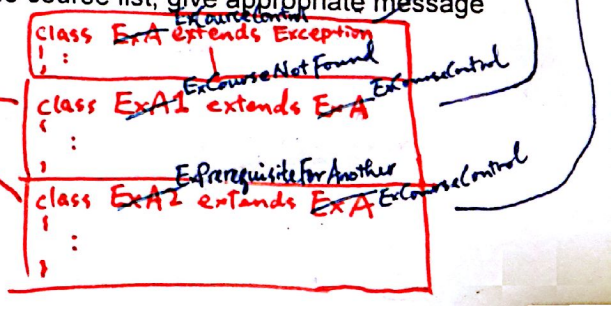
Submit your work to PASS.

Step 6.

Revise the code for course dropping (DropController and other classes, where needed) such that the following cases are handled as exceptions:

- If the course to drop is not in the course list, give appropriate message
- If the course to drop is a pre-requisite of another course in the course list, give appropriate message

```
Revised Output:
Course List: [CS1102] [CS2311] [CS3334]
Cannot drop CS2312 (Course doesn't exist in the list)
Cannot drop CS2311 (Required for CS3334)
Course List: [CS1102] [CS2311]
```



Submit your work to PASS.

Answer

```
public class ExCourseControl extends Exception {
    public ExCourseControl() { super("Course Controller Exception!"); }
    public ExCourseControl(String message)    { super(message); }
}
```

```
public class ExCourseNotFound extends ExCourseControl {
    public ExCourseNotFound() { super("Course not found!"); }
    public ExCourseNotFound(String message)   { super(message); }
}
```

```
public class ExPrerequisiteForAnother extends ExCourseControl {
    public ExPrerequisiteForAnother() { super("Required for another course!"); }
    public ExPrerequisiteForAnother(String message)    { super(message); }
}
```

```
public class Programme {

    private List<Course> courseList;
    private AddController addAgent;
    private DropController dropAgent;

    public Programme() {
        courseList = new LinkedList<Course>();
        addAgent = new AddController(courseList);
        dropAgent = new DropController(courseList);
    }

    public void add(Course c) {
        addAgent.process(c);
        addAgent.printCourses();
    }

    public void drop(Course c) {
        try {
            dropAgent.process(c);
            dropAgent.printCourses();
        } catch (ExPrerequisiteForAnother e) {
            System.out.println(e.getMessage());
        } catch (ExCourseNotFound e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
public abstract class Controller{

    private List<Course> courseList;

    public Controller(List<Course> list) { courseList = list;}

    protected List<Course> getCourseList() { return courseList;}


    public void printCourses() {
        System.out.print("Course List: ");
        for(Course c : courseList) {
            System.out.printf("[%s] ",c);
        }
        System.out.println();
    }

    public abstract void process(Course c) throws ExCourseControl;
}
```

```
public class DropController extends Controller {

    public DropController(List<Course> list) {super(list); }

    @Override
    public void process(Course c) throws ExPrerequisiteForAnother, ExCourseNotFound {
        if (!getCourseList().contains(c))
            throw new ExCourseNotFound("Cannot drop "+c+" (Course doesn't exist in the list)");

        for (Course other: getCourseList())
            if (other.getPreRequisites().contains(c))
                throw new ExPrerequisiteForAnother("Cannot drop "+c+" (Required for "+other+")");
        getCourseList().remove(c);
    }
}
```

```
public class AddController extends Controller{

    public AddController(List<Course> list) {super(list); }

    @Override
    public void process(Course c) {
        if (getCourseList().contains(c)) return;
        for (Course p: c.getPreRequisites())
            process(p);
        getCourseList().add(c);
    }
}
```