ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ

МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

Төслийн тайлан бичиг

Өгөгдлийн бүтэц ба алгоритм (F.CSM203)

2025-2026 оны хичээлийн жилийн намар

Лабораторийн цаг: 2-2

Хичээл заасан багш: ..... Д. Батмөнх /F.CS21/

Хийсэн оюутан 1: Э.Халиунаа/B242270097/

Хийсэн оюутан 2: ..... Б. Энхжин /B241910034/

Улаанбаатар хот

2025 он

Агуулга

1. Оршил

Бие даалтын сэдэв “Оюутны дүн бүртгэл” гэх сэдэвтэй байсан ба энэхүү бие даалтын хүрээнд шугаман жагсаалтын (Linear List) ойлголтыг бодит жишээн дээр хэрэгжүүлж хийсэн. Энэ бие даалтын гол зарчим нь оюутны хичээл, мэргэжил болон шалгалтын дүнгийн мэдээллийн файл ашиглан уншиж, боловсруулж, жагсаалтын бүтэц ашиглан удирдахад тулгуурласан.

Энэхүү систем нь Subject, Major, Student, Lessons, Registration зэрэг class-уудаас бүрдэх ба гол зорилго нь оюутнуудын дүнгийн мэдээллийг хадгалах, дундаж голч дүн (GPA)-г тооцоолох, “F” үнэлгээтэй хичээлүүдийн тооноос хамаарч хасагдах оюутнуудыг тодорхойлох, мөн хичээл болон мэргэжил тус бүрээр оюутны дүнг харуулах юм.

2. Системийн бүтэц

2.1 Системийн үндсэн функцүүд

Мэдээллийн жагсаалтууд

- Хичээлүүдийн жагсаалтыг харуулах – Бүх бүртгэгдсэн хичээлүүдийн мэдээллийг харах.
- Мэргэжлүүдийн жагсаалтыг харуулах – Бүх мэргэжлүүдийн мэдээллийг харах.

Тооцооллын функцүүд

- Нийт оюутны дундаж GPA тооцоолох – Бүх оюутнуудын голч дүнг тооцож харуулах.

Шүүлт хайлтын функцүүд

- Гурваас дээш “F” үнэлгээтэй оюутнуудыг харуулах – Эрстэлтэй оюутнуудыг тодорхойлох.
- Хичээл бүрээр дүнгийн жагсаалт харуулах – Хичээл тус бүрийн оюутнуудын дүнг харах.
- Мэргэжил бүрээр дүнгийн жагсаалт харуулах – Мэргэжил тус бүрийн оюутнуудын GPA харах.

2.2 Оролтын файлууд

Систем нь дараах гурван текст файлаас өгөгдлийг уншиж ажиллана.

SubjectData.txt

- Хичээлийн_код/Хичээлийн_нэр/Кредит
- Жишээ: F.CSA100/Компьютерийн ухааны үндэс/3.0

MajorData.txt

- Мэргэжлийн_код/Мэргэжлийн_нэр
- Жишээ: B24190/ Компьютерийн ухаан

ExamData.txt

- Оюутны_код/Хичээлийн_код/Оноо
- Жишээ: B241910001/F.ITA100/98

3. Архитектур болон дизайн

3.1 UML диаграмм (PlantUML)

@startuml

' UML for provided Java code (BD1.Main and inner classes)

```
package "bd1.app" {
  class Main {
    + main(args: String[]): void
  }
  class Registration {
    - studentList: MyArrayLinearList
    - subjectList: MyArrayLinearList
    - majorList: MyArrayLinearList
    + Registration()
    + Read(): void
    + GetLessons(code: String): MyChain
    + PrintAllHicheel(): void
    + PrintAllMergejil(): void
    + PrintAllGPA(): void
    + PrintAll3F(): void
    + PrintAllDun(): void
    + PrintAllDun2(): void
  }
```

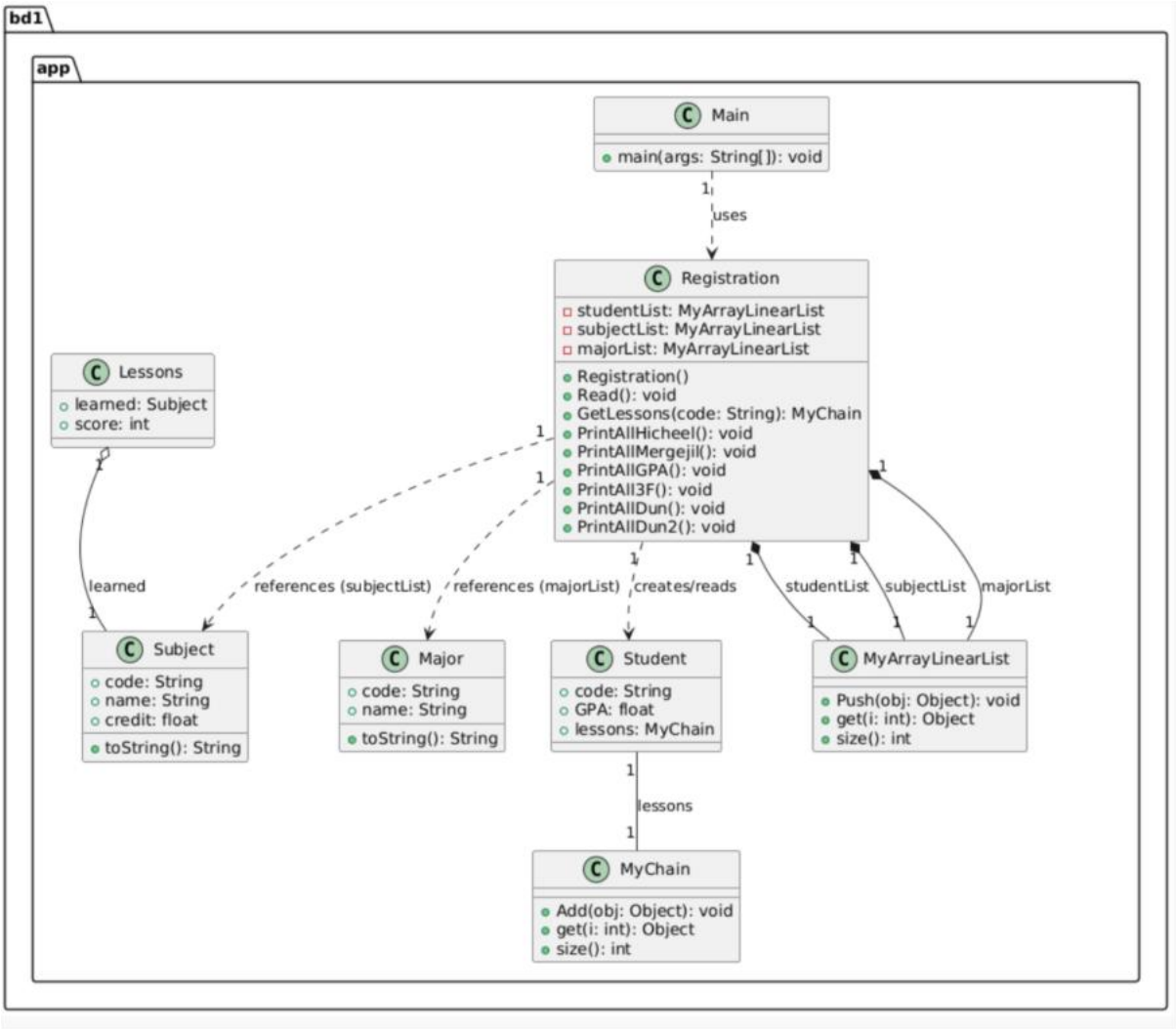
```
  class Subject {
    + code: String
    + name: String
    + credit: float
    + toString(): String
  }
  class Major {
    + code: String
    + name: String
    + toString(): String
  }
  class Student {
    + code: String
    + GPA: float
    + lessons: MyChain
  }
  class Lessons {
    + learned: Subject
    + score: int
  }

  ' External / collection classes (interfaces / placeholders)
  class MyArrayLinearList {
    + Push(obj: Object): void
    + get(i: int): Object
    + size(): int
 }


  class MyChain {
    + Add(obj: Object): void
    + get(i: int): Object
    + size(): int
  }
}
' relationships (associations)
Registration "1" *-- "1" MyArrayLinearList : studentList
Registration "1" *-- "1" MyArrayLinearList : subjectList
Registration "1" *-- "1" MyArrayLinearList : majorList

Student "1" -- "1" MyChain : lessons
Registration "1" ..> Student : creates/reads
Registration "1" ..> Subject : references (subjectList)
Registration "1" ..> Major : references (majorList)
Lessons "1" o-- "1" Subject : learned

Main "1" ..> Registration : uses
@enduml
```



4. Классын зохион байгуулалт

Subject (Хичээл)

Зорилго: Хичээлийн лавлах мэдээллийг агуулах.
Атрибутууд:

- code: String — Хичээлийн код (жишээ: F.ITA100)
- name: String — Хичээлийн нэр
- credit: float — Кредит хэмжээ (жишээ: 3.0)

Методууд:

- toString(): String — Текст дэлгэцэд харуулах зориулалттай.

  Тайлбар: Subject нь credit талбараараа GPA тооцоололд ашиглагдана. Файлаас унших үед code/name/credit форматаар гарч ирнэ.

Major (Мэргэжил)

Зорилго: Мэргэжлийн код, нэр зэргийг агуулна.
Атрибутууд:

- code: String — Мэргэжлийн код (жишээ: B241940)
- name: String — Мэргэжлийн нэр

Методууд:

- toString(): String

  Тайлбар: Major код нь оюутны кодын эхний хэсэгтэй таарч тухайн оюутан ямар мэргэжилд харьяалагдахыг тогтоохоор ашиглагддаг.

Student (Оюутан)

Зорилго: Оюутны код, GPA болон авсан хичээлүүдийг агуулах.
Атрибутууд:

- code: String — Оюутны код (жишээ: B241940001)
- GPA: float — Тооцоологдсон голч дүн .
- lessons: MyChain – Тухайн оюутны авсан Lessons объектуудын жагсаалт.

  Тайлбар: lessons нь тухайн оюутан бүрийн оноо болон тухайн хичээлийн credit-ийг агуулсан байх ёстой.

Lessons (Lesson бүртгэл)

Зорилго: Нэг оюутны нэг хичээлийн оноо, мөн тухайн хичээлийн metadata-г агуулах.
Атрибутууд:

- learned: Subject — Тухайн авсан хичээл (Subject объект)
- score: int — Авсан оноо (0-100)
  Тайлбар: Нэрийг Lessons гэж ашигласан тул олон объект авч хэвшинэ.

Registration

Зорилго: Файлаас өгөгдөл унших, domain объектуудыг үүсгэх, тайлан хэвлэх гол класс.
Атрибутууд:

- studentList: MyArrayLinearList — Бүртгэлтэй оюутнууд
- subjectList: MyArrayLinearList — Бүх Subject-ууд
- majorList: MyArrayLinearList — Бүх Major-ууд

Гол функцүүд:

- Read() — SubjectData.txt, MajorData.txt,
  Exam.txt файлуудаас унших ба объектууд үүсгэх.
- GetLessons(String code) – Тухайн оюутны Exam.txt-ээс бүх lesson-ийг авч MyChain-ийг буцаах.
- PrintAllHicheel() – Бүх хичээлийг хэвлэх
- PrintAllMergejil() – Бүх мэргэжлийг хэвлэх

- PrintAllGPA() — Оюутнуудын GPA-г хэвлэх
- PrintAll3F() — Гурваас дээш “F” авсан оюутнуудын жагсаалт
- PrintAllDun() — Хичээл бүрээр оюутнуудын оноог хэвлэх
- PrintAllDun2() – Мэргэжил бүрээр оюутнуудын код ба GPA-г хэвлэх

  Тайлбар: Read() функц нь Exam.txt-ийг уншихдаа оюутны lessons-ийг GetLessons()-аар үүсгэн, GPA-г тухайн оюутанд тооцож studentList-д нэмнэ.

MyArrayLinearList, MyChain

Зорилго: Төслийн өөрийн бүтэцтэй массив/жагсаалт.

Онцлог: Push, get, size зэрэг үндсэн үйлдлүүдтэй. MyChain нь Add, get, size-тай

5. Алгоритм

5.1 Файл унших

Файлын форматууд

- SubjectData.txt each line: F.ITA100/Medeelliin tehnologiin undes/3.0
- MajorData.txt each line: B241940/Medeelliin technology
- Exam.txt each line: B241940001/F.ITA100/92

Жагсаалтын сонголт ба шалтгаан

- subjectList — MyArrayLinearList ашигласан: file-д хийгдсэн дарааллыг хадгалах, дараа лавлаж харахдаа subjectList-ээр дамжин нэг бүрчлэн олох.
- studentList – Оюутнуудын бүрэн мэдээллийг хадгална. GetLessons() нь Exam.txt-ээс тухайн оюутны lessons-ийг бүрдүүлж lessons талбарт онооно.

Файлыг унших (псевдо/жишээ үйлдэл)

1. SubjectData.txt-г нэг мөр уншаад line.split(“/”) → code, name, credit → Subject үүсгэн subjectList.Push(s)
2. MajorData.txt-г уншаад Major үүсгэн majorList.Push(m)
3. Exam.txt-г уншаад мөр бүрийг values = line.split(“/”) → studentId = values[0]… → GetLessons(studentId) дуудах → Student үүсгэн lessons-аар дүүргэн studentList-д нэмэх

Код

```
java.io.BufferedReader input = new java.io.BufferedReader
        (new java.io.FileReader("./SubjectData.txt"));
String line = input.readLine();
while(line != null) {
    String[] values = line.split("/");
      System.out.println(values[0]);
    line = input.readLine();
    Subject s = new Subject();
    s.code = values[0];
    s.name = values[1];
    s.credit = Float.parseFloat(values[2]);
    subjectList.Push(s);
}
```

Алдаа ба хамгаалалт

- Файлын мөрөнд хоосон мөр, буруу форматын мөр байж болно. parts.lenght шалгана.
- Subject олдохгүй бол GetLessons()-д lesson-ийг үүсгэх боломжгүй тул тухайн мөрийг subjectList-д нэмэхгүйгээр алгасаж байна.

5.2 GPA тооцоолох

Тооцоо

averageScore < 60 - GPA = 0.0
60 <= x < 65 - GPA = 1.0
65 <= x < 70 - GPA = 1.3
70 <= x < 73 - GPA = 1.7
73 <= x < 77 - GPA = 2.0
77 <= x < 83 - GPA = 2.3
83 <= x < 87 - GPA = 2.7
87 <= x < 90 - GPA = 3.3
90 <= x < 95 - GPA = 3.7
95 <= x - GPA = 4.0

6. Хэрэглэх заавар

6.1 Программ ажиллуулах

1. **Программыг ажиллуулах команд:**

javac BD1/Main.java
java BD1.Main

2. **Terminal дээр дараах цэс гарна:**

1 - Хичээлүүд
2 - Мэргэжлүүд
3 - Оюутны GPA
4 - 3-аас дээш “F” авсан оюутнууд
5 - Хичээл бүрийн дүн
6 - Мэргэжил бүрийн дүн
0 – Гарах

6.2 Цэсийн товч тайлбар

№	Үйлдэл
1	Хичээлүүдийн жагсаалтыг харуулна
2	Мэргэжлийн жагсаалтыг харуулна
3	Бүх оюутны GPA-г тооцож гаргана
4	3-аас дээш “F” авсан оюутнуудыг харуулна
5	Хичээл бүрийн дүнг харуулна
6	Мэргэжил бүрийн GPA-г харуулна
0	Программыг зогсооно

6.3 Жишээ

Таны сонголт: 3
B241940001 → GPA = 3.3
B241940102 → GPA = 1.7

7. UNIT TEST

7.1 Unit test-ийн зорилго

Unit test нь системийн тус тусдаа модулиуд, ангилал, эсвэл функцүүдийг тус тусад нь шалгаж, тэдгээрийн зөв ажиллагааг баталгаажуулахад оршино. Энэхүү туршилтаар өгөгдлийн жагсаалт хоосон биш эсэх, GPA-ийн утгууд зөв мужид байгаа эсэх мөн “F” үнэлгээтэй оюутнуудыг илрүүлэх зэрэг гол шалгалтуудыг хийв.

7.2 Ашигласан технологи

- Test Framework: JUnit 5
- Хэл: Java
- Орчин: IntelliJ IDEA / Eclipse

7.3 Unit test-н код

```
package BD1;

import static org.junit.jupiter.api.Assertions.*;

import Lab2.MyChain;
import org.junit.jupiter.api.*;

public class MainTest {

    Registration reg;

    @BeforeEach
    void setUp() {
        reg = new Registration(); // өгөгдлийг уншина
    }

    @Test
    @DisplayName("1. Hicheeluudiin jagsaalt hooson bish eseh")
    void testSubjectListNotEmpty() {
        assertTrue(reg.subjectList.size() > 0, "Subject list must not be empty");
    }

    @Test
    @DisplayName("2. Mergejliin jagsaalt hooson bish eseh")
    void testMajorListNotEmpty() {
        assertTrue(reg.majorList.size() > 0, "Major list must not be empty");
    }



   @Test
   @DisplayName("3. Oyutnii jagsaalt hooson bish eseh")
   void testStudentListNotEmpty() {
       assertTrue(reg.studentList.size() > 0, "Student list must not be empty");
   }

   @Test
   @DisplayName("4. GPA tootsoolol 0-4-iin hoorond baigaa eseh")
   void testGPARange() {
       for (int i = 0; i < reg.studentList.size(); i++) {
           float gpa = ((Student) reg.studentList.get(i)).GPA;
           assertTrue(gpa >= 0 && gpa <= 4.0, "GPA must be between 0.0 and 4.0");
       }
   }

   @Test
   @DisplayName("5. F unelgeetei 3-aas deesh hicheeltei oyutan baigaa eseh")
   void testHas3FStudents() {
       boolean found = false;
       for (int i = 0; i < reg.studentList.size(); i++) {
           int f = 0;
           Student s = (Student) reg.studentList.get(i);
           for (int j = 0; j < s.lessons.size(); j++) {
               Lessons l = (Lessons) s.lessons.get(j);
               if (l.score < 60) f++;
           }
           if (f >= 3) {
               found = true;
               break;
           }
       }


       assertTrue(found, "At least one student should have 3 or more F grades");
   }

   @Test
   @DisplayName("6. Hicheeltei holbootoi dungiin ugugdul butsaah eseh")
   void testGetLessons() {
       if (reg.studentList.size() > 0) {
           String studentCode = ((Student) reg.studentList.get(0)).code;
           MyChain lessons = reg.GetLessons(studentCode);
           assertNotNull(lessons, "Lessons must not be null");
           assertTrue(lessons.size() > 0, "Student must have at least one lesson");
       }
   }
}
```

7.4 Туршилтын үр дүн

№	Тестийн нэр	Тайлбар	Үр дүн
1	Хичээлүүдийн жагсаалт хоосон биш эсэх	subjectList-ийн хэмжээ > 0	Passed
2	Мэргэжлийн жагсаалт хоосон биш эсэх	majorList-ийн хэмжээ > 0	Passed
3	Оюутны жагсаалт хоосон биш эсэх	studentList-ийн хэмжээ > 0	Passed
4	GPA тооцоолол зөв мужид байгаа эсэх	$0.0 \leq GPA \leq 4.0$	Passed
5	$F \geq 3$ хичээлтэй оюутан байгаа эсэх	Үнэлгээ < 60 байх 3+ хичээл	Passed
6	Оюутны хичээл буцааж байгаа эсэх	GetLessons() хоосон биш	Passed

8. Дүгнэлт

Энэхүү бие даалтын хүрээнд “Оюутны дүн бүртгэл” системийг шугаман жагсаалтын (Linear List) бүтэц ашиглан хийж гүйцэтгэсэн. Систем нь оюутны дүнгийн мэдээллийг файлаас унших, боловсруулах, хадгалах, мөн GPA-г тооцоолох, 3-аас дээш “F” авсан оюутнуудыг тодорхойлох, хичээл болон мэргэжил тус бүрээр дүнг харах боломжтой.

Системийн үндсэн хэсгүүдэд Subject, Major, Student, Lessons, Registration зэрэг классууд багтсан ба өгөгдлийн бүтцийн зөв зохион байгуулалт хийснээр код илүү ойлгомжтой болж байна.

9. Хавсралт (Бүтэн код)

```
package BD1;
import com.studyopedia.*;
import Lab2.*;
import java.util.*;

public class Main {
    public static void main(String[] args){
        Registration r = new Registration();
        Scanner sc = new Scanner (System.in);
        int a = -1;
        while(a !=0){
            System.out.println("1-Niit hicheeluuldiin jagsaaltiig haruulah");
            System.out.println("2-Niit mergejluudiin jagsaaltiig haruulah");
            System.out.println("3-Niit oyutnii dundaj golch dung haruulah");
            System.out.println("4-Guravaas deesh hicheeld “F” unelgee avsan hasagdah oyutnii jagsaalt");
            System.out.println("5-Hicheel bureer oyutnuudiin dungiin jagsaaltiig haruulah");
            System.out.println("6-Mergejil bureer oyutnuudiin dungiin jagsaaltiig haruulah");
            System.out.println("0-exit");

            System.out.println("Tanii songolt:");
            a = sc.nextInt();
            switch(a){
                case 1:
                    r.PrintAllHicheel();
                    break;
                case 2:
                    r.PrintAllMergejil();
                    break;
                case 3:
                    r.PrintAllGPA();
                    break;
                case 4:
                    r.PrintAll3F();
                    break;
                case 5:
                    r.PrintAllDun();
                    break;
                case 6:
                    r.PrintAllDun2();
                    break;
                case 0:
                    break;

            }
        }
    }

}
```

```
 class Subject{
    public String code;
    public String name;
    public float credit;

    public String toString(){
        return code + ","+ name + "," + credit + ".";
    }
}

 class Major{
    public String code;
    public String name;
    public String toString(){
        return code + "," + name + ",";
    }
}

 class Student{
    public String code;
    public float GPA;
    public MyChain lessons;
}

 class Lessons{
    public Subject learned;
    public int score;
}
 class Registration{
    public MyArrayLinearList studentList;
    public MyArrayLinearList subjectList;
    public MyArrayLinearList majorList;

    public Registration(){
        studentList = new MyArrayLinearList();
        subjectList = new MyArrayLinearList();
        majorList = new MyArrayLinearList();
        Read();
    }

    public void Read() {
        try {
            java.io.BufferedReader input = new java.io.BufferedReader
                    (new java.io.FileReader("./SubjectData.txt"));
            String line = input.readLine();
            while(line != null) {
                String[] values = line.split("/");
//                System.out.println(values[0]);
                line = input.readLine();
                Subject s = new Subject();
                s.code = values[0];
                s.name = values[1];
                s.credit = Float.parseFloat(values[2]);
                subjectList.Push(s);
            }
        }
```

```
        catch (Exception e) {
            System.out.println("File not found:1 ");
            System.exit(1);
        }

        try {
            java.io.BufferedReader input = new java.io.BufferedReader
                    (new java.io.FileReader("./MajorData.txt"));
            String line = input.readLine();
            while(line != null) {
                String[] values = line.split("/");
'/                System.out.println(values[0]);
                line = input.readLine();
                Major m = new Major();
                m.code = values[0];
                m.name = values[1];
                majorList.Push(m);
            }
        }
        catch (Exception e) {
            System.out.println("File not found:2 ");
            System.exit(1);
        }
         try {
             java.io.BufferedReader input = new java.io.BufferedReader
                     (new java.io.FileReader("./Exam.txt"));
             String line = input.readLine();
             while(line != null) {
                 String[] values = line.split("/");
//                System.out.println(values[0]);
                 line = input.readLine();
                 Student t = new Student();
                 t.code = values[0];
                 t.lessons = GetLessons(t.code);
                 float gpa = 0;
                 int total = 0;
                 for(int i = 0; i < t.lessons.size(); i++){
                     gpa += ((Lessons)t.lessons.get(i)).score *((Lessons)t.lessons.get(i)).learned.credit;
                     total += (int) ((Lessons)t.lessons.get(i)).learned.credit;
                 }
                 gpa /= total;

                 if(gpa < 60) {
                     t.GPA = 0f;
                 } else if (gpa < 65){
                     t.GPA = 1.0f;
                 } else if (gpa < 70) {
                     t.GPA = 1.3f;
                 } else if (gpa < 73) {
                     t.GPA = 1.7f;
                 } else if (gpa < 77) {
                     t.GPA = 2.0f;
                 } else if (gpa < 83) {
                     t.GPA = 2.3f;
                 } else if (gpa < 87) {
                     t.GPA = 2.7f;
                 } else if (gpa < 90) {
                     t.GPA = 3.3f;
                 } else if (gpa < 95) {
                     t.GPA = 3.7f;
                 } else {
                     t.GPA = 4.0f;
                 }
                  boolean isHave = false;
                  for(int i = 0; i < studentList.size(); i++){
                      if(((Student)studentList.get(i)).code.equals(t.code)){
                          isHave = true;
                          break;
                      }
                  }
                  if(!isHave) {
                      studentList.Push(t);
                  }
             }
         }
         catch (Exception e) {
             System.out.println("File not found: 3");
             System.exit(1);
         }
```

```
        }
        catch (Exception e) {
            System.out.println("File not found: 3");
            System.exit(1);
        }
    }
    public MyChain GetLessons(String code){
        MyChain c = new MyChain();
        try {
            java.io.BufferedReader input = new java.io.BufferedReader
                    (new java.io.FileReader("./Exam.txt"));
            String line = input.readLine();
            while(line != null) {
                String[] values = line.split("/");
//                System.out.println(values[0]);
                line = input.readLine();
                if(code.equals(values[0])){
                    Lessons l = new Lessons();
                    for(int j = 0; j < subjectList.size(); j++){
                        if(((Subject)subjectList.get(j)) .code.equals( values[1])){
                            l.learned = ((Subject)subjectList.get(j));
                            break;
                        }
                    }
                    l.score = Integer.parseInt(values[2]);
//                    System.out.println(Integer.parseInt(values[2]));
                    c.Add(l);
                }

            }
        }
        catch (Exception e) {
            System.out.println("File not found:4 ");
            System.exit(1);
        }
        return c;
    }
    public void PrintAllHicheel(){
        for(int i = 0; i < subjectList.size(); i++){
            System.out.println(subjectList.get(i).toString());
        }
    }

    public void PrintAllMergejil(){
        for(int i = 0; i < majorList.size(); i++){
            System.out.println(majorList.get(i));
        }
    }

    public void PrintAllGPA(){
        for(int i = 0; i < studentList.size(); i++){
            System.out.println(((Student)studentList.get(i)).GPA);
        }
    }
    public void PrintAll3F(){
        for(int i = 0; i < studentList.size(); i++){
            int f = 0;
            for(int j = 0; j < ((Student)studentList.get(i)).lessons.size(); j++){
                Lessons l = (Lessons) (((Student)studentList.get(i)).lessons.get(j));
               if(l.score < 60){
                   f++;
               }
           }
           if (f >= 3){
               System.out.println(((Student)studentList.get(i)).code);
           }
       }
   }
```

```
public void PrintAllDun(){
        for(int i = 0; i < subjectList.size(); i ++){
            System.out.println(((Subject) subjectList.get(i)).name);
            for(int j = 0; j < studentList.size(); j++){
                MyChain l = ((Student)studentList.get(j)).lessons;
                for(int k = 0; k < l.size(); k++){
                    if(((Lessons)l.get(k)).learned.code.equals(((Subject) subjectList.get(i)).code)){
                        System.out.println(((Student)studentList.get(j)).code + ", " + ((Lessons)l.get(k)).score);
                    }
                }

            }
        }
    }
    public void PrintAllDun2(){
        for(int i = 0; i < majorList.size(); i++){
            System.out.println(((Major) majorList.get(i)).name);
            for(int j = 0; j < studentList.size(); j++){
                String code = ((Student)studentList.get(j)).code;
                code = code.substring(0, 7);
                if (((Major) majorList.get(i)).code.equals(code)){
                    System.out.println(((Student)studentList.get(j)).code + ", " + ((Student)studentList.get(j)).GPA);
                }
            }
        }
    }
}
```