ปฏิบัติการนี้เป็นการศึกษาคลาส `HashMap` (https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html) ของจาวาที่เก็บคู่ของ key และ value ใน hash table เพื่อให้ใช้ key เพื่อไปดึง value จาก hash table ได้อย่างรวดเร็ว และนำ `HashMap` ไปประยุกต์ใช้งาน โดยต้องใช้ method `get`, `put`, `containsKey`, `containsValue` และสร้าง `hashCode` ของ `Object` ที่จะใช้เป็น key ใน `hashMap` ให้เหมาะสม

จาก https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html#hashCode-- ระบุว่า

public int hashCode()

Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by HashMap.

The general contract of hashCode is:

- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified. This integer need not remain consistent from one execution of an application to another execution of the same application.
- If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.
- It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hash tables.

As much as is reasonably practical, the hashCode method defined by class Object does return distinct integers for distinct objects. (This is typically implemented by converting the internal address of the object into an integer, but this implementation technique is not required by the Java™ programming language.)

โจทย์ปฏิบัติการ ข้อ 1

จาก https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)#Weighted_graph กราฟแบบมีน้ำหนัก (weighted graph) ประกอบด้วย **vertex** หรือ **node** โดยมี **edge** เชื่อมต่อ vertex 2 จุด โดยแต่ละ edge มีน้ำหนัก (weight) ของมัน

กำหนด calss Vertex และ class Edge ดังนี้

```
public class Edge {
    private Vertex st, end;
    public Edge(Vertex b, Vertex e) {
        st = b;
        end = e;
    }
    public int getSource() {
        return st.getName();
    }
    public int getDest() {
        return end.getName();
    }
}
```

```
public class Vertex {
    private int name;
    public Vertex(int name) {
        this.name = name;
    }
    public int getName() {
        return name;
    }
    public String toString() {
        return ""+name;
    }
}
```

ให้ใช้คลาสทั้งสองนี้เพื่อสร้างคลาส `Graph` ที่เก็บ vertex เป็น array ของ `Vertex` และเก็บ edge เป็น array ของ `Edge` โดยที่มี `hashMap` เก็บน้ำหนักของ edge (`Edge` เป็น key และน้ำหนักของ Edge เป็น value) คลาสนี้มี method ต่อไปนี้

- constructor ที่รับจำนวน vertex และจำนวน edge ในกราฟมาแล้วสร้าง weighted graph โดยสุ่ม edge และน้ำหนักของ edge (1-4)

- constructor ที่รับ adjacency matrix ของกราฟ (คือ array 2 มิติที่เก็บน้ำหนักของ edge ในกราฟ ค่าที่ตำแหน่ง `[a, b]` ใน array นี้เป็นน้ำหนักของ edge ระหว่าง vertex ที่ตำแหน่ง `a` และตำแหน่ง `b`) แล้วมากราฟที่มี edge ที่มีน้ำหนักตามที่กำหนดด้วย adjacency matrix
- `weight(Edge e)` ที่ตรวจสอบว่ามี edge `e` ในกราฟนี้หรือไม่ ถ้ามี ให้คืนค่าเป็นน้ำหนักของ edge `e` ถ้าไม่มี ให้คืนค่าเป็น `0`
- `toMatrix()` ที่คืนค่าเป็น adjacency matrix ของกราฟ (adjacency matrix ของกราฟเป็น array 2 มิติที่เก็บน้ำหนักของ edge ในกราฟ ค่าที่ตำแหน่ง `[a, b]` ใน array นี้เป็นน้ำหนักของ edge ระหว่าง vertex ที่ตำแหน่ง `a` และตำแหน่ง `b`)

เขียน method `main` ที่สร้าง weighted graph 2 กราฟที่มีจำนวน vertex เท่ากัน (โดยสุ่มจำนวนเต็ม 5-10)

กราฟที่ 1 สร้างโดย constructor ที่รับ adjacency matrix ดังนั้นให้สร้าง adjacency matrix โดยสุ่มว่าจะมี edge จาก vertex ใดไปยัง vertex ใดบ้าง และสุ่มว่า edge มีน้ำหนักเท่าไร (1-5) แล้วนำไปสร้าง `Graph`

กราฟที่ 2 สร้างโดย constructor ที่รับจำนวน vertex และจำนวน edge ของกราฟมาแล้วสร้าง weighted graph โดยสุ่ม ให้มีจำนวน vertex เท่ากับจำนวน vertex ของกราฟ 1 และมี edge = (จำนวน vertex)^2/4

จากนั้นให้ แสดง adjacency matrix ตั้งต้นของกราฟ 1 แล้วใช้ method `toMatrix` เพื่อแปลงกราฟทั้งสองเป็น adjacency matrix แล้ว matrix ทั้ง 2 ดังตัวอย่าง

ตัวอย่างการทำงาน

```
Random adjacency matrix of graph 1
5       0       0       2       4       0       0
0       0       0       3       4       1       5
0       0       0       0       0       0       0
2       3       0       0       3       0       0
4       4       0       3       1       1       0
0       1       0       0       1       0       4
0       5       0       0       0       4       0
-----------------
Adjacency matrix created from graph 1
5       0       0       2       4       0       0
0       0       0       3       4       1       5
0       0       0       0       0       0       0
2       3       0       0       3       0       0
4       4       0       3       1       1       0
0       1       0       0       1       0       4
0       5       0       0       0       4       0
-----------------
Adjacency matrix created from graph 2
0       3       0       0       1       3       0
3       1       0       2       1       0       0
0       0       0       2       2       3       0
0       2       2       0       0       1       0
1       1       2       0       3       0       0
3       0       3       1       0       1       0
0       0       0       0       0       0       0
```

โจทย์ปฏิบัติการ ข้อ 2

สร้างคลาส `Student` ที่เก็บรหัสนิสิต ชื่อ และ นามสกุล และคลาส `CourseGrade` ที่เก็บข้อมูลของวิชาที่ลงทะเบียนเรียน ซึ่งประกอบด้วย รหัสวิชา ชื่อวิชา ภาคการศึกษา ปีการศึกษา จำนวนหน่วยกิต และเกรดที่ได้ แล้วสร้าง `HashMap` ที่เก็บข้อมูลของนิสิตคู่กับข้อมูลของวิชาที่ลงทะเบียนเรียน โดยสร้าง `hashCode` ที่เหมาะสมสำหรับคลาสที่เกี่ยวข้อง

จากนั้นในคลาส `Main` ให้เขียนโปรแกรมที่อ่านไฟล์ `register.csv` ที่เก็บข้อมูลผลการเรียนโดยแต่ละบรรทัดเป็นรหัสนิสิต ชื่อ-นามสกุล รหัสวิชา ชื่อวิชา ภาคการศึกษา ปีการศึกษา จำนวนหน่วยกิต และเกรดที่ได้ แล้วสร้าง object ในคลาส `Student` และคลาส `CourseGrade` จากนั้นเก็บเก็บข้อมูลของนิสิตคู่กับข้อมูลของวิชาที่ลงทะเบียนเรียนใน `HashMap` ที่มี `Student` เป็น key และ `ArrayList` ของ `CourseGrade` เป็น value เมื่ออ่านไฟล์ไปเก็บใน HashMap ครบแล้วให้รับรหัสนิสิตจากผู้ใช้ เพื่อพิมพ์ผลการเรียนของนิสิตคนนั้นออกมาดังตัวอย่าง

ตัวอย่างการทำงาน

```
Enter student ID
635465452
2301170  Comp Prog  2563/2  3  3.5
2301172  Comp Prog Lab  2563/1  1  2.5
2301260  Prog Tech  2564/1  4  1.5
```

```
register.csv
633456782,Tre,Hyd,2301170,Comp Prog ,1,2563,3,3.5
635465452,Max,Nim,2301170,Comp Prog ,2,2563,3,3.5
631114782,Pop,Pie,2301170,Comp Prog ,3,2563,3,3.5
633456782,Tre,Hyd,2301172,Comp Prog Lab,1,2563,1,3.0
635465452,Max,Nim,2301172,Comp Prog Lab,1,2563,1,2.5
631114782,Pop,Pie,2301172,Comp Prog Lab,2,2563,1,4
633456782,Tre,Hyd,2301260,Prog Tech,2,2563,4,2.0
635465452,Max,Nim,2301260,Prog Tech,1,2564,4,1.5
631114782,Pop,Pie,2301260,Prog Tech,2,2563,4,3.0
```