**คำชี้แจง ให้เขียน method เรียงตามลำดับข้อย่อย คำนวณคะแนนเฉพาะ method ที่ทำงานได้สมบูรณ์**

**1)** จงเขียนโปรแกรมเพื่อสร้างคลาส **Course** และคลาส **RegisStudent** ตามโครงสร้างในแผนภาพ UML นักศึกษาสามารถเพิ่มเติมความสัมพันธ์ interface, class, method หรือ attribute ได้ตามความจำเป็นและเหมาะสม ***แต่ห้ามแก้ไขโครงสร้างที่กำหนดให้*** ดังนี้

คลาส **RegisStudent** ประกอบด้วย

* **attributes** รหัสนักศึกษา (**studentId**) และชื่อนักศึกษา (**studentName**)
* **equals** method เปรียบเทียบความเท่ากันของนักศึกษา 2 คนโดยพิจารณาจากรหัสนักศึกษา

คลาส **Course** ประกอบด้วย

* **attributes** ชื่อวิชา (**courseName**) และ จำนวนนักศึกษาที่ลงทะเบียน **(numberOfStudents)**
* **toString** จะทำการแสดงชื่อวิชา และ นักศึกษาที่ลงทะเบียนทั้งหมดในวิชา
* **1.1\_\_\_\_\_\_\_\_\_\_\_ (2 คะแนน) Constructor** รับชื่อวิชาใหม่และจำนวนนักศึกษาที่เปิดรับให้ลงทะเบียนในวิชานั้นได้ จากนั้นสร้าง array เท่ากับจำนวนนักศึกษาที่เปิดรับ
* **1.2\_\_\_\_\_\_\_\_\_\_\_ (3 คะแนน) addStudent** รับพารามิเตอร์เป็นนักศึกษาที่ต้องการลงทะเบียน จากนั้นตรวจสอบว่านักศึกษาทำการลงทะเบียนวิชานั้นซ้ำหรือไม่ ถ้านักศึกษาลงทะเบียนแล้ว ไม่ต้องทำการเพิ่มใน array และให้ return ค่า -2 กรณีถ้าไม่พบซ้ำ ทำการเพิ่มเข้าไปใน array students และเพิ่ม numberOfStudents ขึ้นหนึ่ง เพื่อว่าเราจะได้ไม่รับนักศึกษาที่ลงทะเบียนเกินจากจำนวนรับนักศึกษาที่ระบุไว้ตอนเปิดวิชาใหม่ หลังจากทำงานเสร็จสิ้นจะ return ตำแหน่ง index ที่เพิ่มนักศึกษาสำเร็จ -1 ถ้าไม่สามารถเพิ่มนักศึกษาได้เนื่องจากลงทะเบียนเกินจำนวนรับ
* **1.3\_\_\_\_\_\_\_\_\_\_\_ (5 คะแนน) dropStudent** รับพารามิเตอร์เป็นนักศึกษาที่ต้องการถอนรายวิชาและลบข้อมูลนักศึกษาออกจาก array students จากนั้นให้ทำการเลื่อนข้อมูลนักศึกษาคนต่อไปใน array ให้เรียงชิดติดกันเหมือนเดิม และลดจำนวนนักศึกษา numberOfStudents ลงหนึ่ง หลังจากทำงานเสร็จสิ้นจะ return true ถ้าลบสำเร็จและ false ถ้าลบไม่สำเร็จ ไม่พบข้อมูลหรือยังไม่มีนักศึกษาลงทะเบียน

**numberOfStudents=4**  **numberOfStudents=3**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Before*** | Student1 | Student2 | Student3 | Student4 |  | ***After*** | Student1 | Student3 | Student4 | Student4 |

**Course**

-courseName: String

-numberOfStudents: int

+Course(courseName:String, numOfStudentCanRegis: int)

+getCourseName(): String

+addStudent(std:RegisStudent): int

+dropStudent(std:RegisStudent): boolean

+getStudents(): RegisStudent[]

+getNumberOfStudents(): int

+toString():String

**RegisStudent**

-studentId: int

-studentName: String

+RegisStudent(stdId:int, stdName:String)

+getStudentId(): int

+getStudentName(): String

+equals(Object o):boolean

+toString(): String

**คำชี้แจง ให้เขียน method เรียงตามลำดับข้อย่อย คำนวณคะแนนเฉพาะ method ที่ทำงานได้สมบูรณ์**

**2)** จงเขียนโปรแกรมเพื่อสร้างคลาสCard (ไพ่) และคลาส Deck(สำรับ)และกำหนดความสัมพันธ์ของคลาสตามแผนภาพ Class Diagram นักศึกษาสามารถเพิ่มเติมความสัมพันธ์ interface, class, method หรือ attribute ได้ตามความจำเป็นและเหมาะสม ***แต่ห้ามแก้ไขโครงสร้างที่กำหนดให้*** ดังนี้

**class** **Card** (ไพ่)

* **8 constants** ที่เก็บค่าคงที่ (SPADES, HEARTS, DIAMONDS, CLUBS, ACE, JACK, QUEEN, KING)
* **attributes** สัญลักษณ์ (suit) และเลขบนหน้าไพ่ (rank)
* **constructor** กำหนดค่าเริ่มต้นให้กับ rank และ suit attributes
* **equals** เปรียบเทียบความเท่ากันโดยพิจารณาจากสัญลักษณ์ (suit) และเลขบนหน้าไพ่ (rank)
* **toString** ทำการแสดงเลขและสัญลักษณ์ตัวแรกของไพ่

**class Deck** (สำรับไพ่)

1 สำรับ จะประกอบด้วยไพ่ทั้งหมด 52 ใบ โดยจะมีทั้งหมด 4 สัญลักษณ์ คือ โพดำ (SPADES), โพแดง (HEARTS), ข้าวหลามตัด (DIAMONDS), ดอกจิก (CLUBS) เรียงลำดับตามค่าประจำสัญลักษณ์ และแยกย่อยอีกชุดละ 13 ใบ โดยทั้งชุดจะประกอบไปด้วย ACE, 2, 3, 4, 5, 6, 7, 8, 9, 10, JACK(11), QUEEN(13), KING(13)

* **size** methodคืนค่าจำนวนไพ่ที่เหลือในสำรับ
* **deal** method ทำการแจกไพ่ใบสุดท้าย
* **isEmpty** methodเพื่อบอกสถานะว่าสำรับว่างหรือไม่ ถ้า size เท่ากับ 0 แสดงว่า empty
* **2.1\_\_\_\_\_\_\_\_\_\_\_ (2 คะแนน) constructor** เพื่อสร้างสำรับของไพ่ 52 ใบเก็บใน array และกำหนดค่าของ size เท่ากับ 52หลังจากสร้างสำรับแล้วจะได้ค่าใน array ดังนี้   
  {As, 2s, 3s, 4s, 5s, 6s, 7s, 8s, 9s, Ts, Js, Qs, Ks,  
  Ah, 2h, 3h, 4h, 5h, 6h, 7h, 8h, 9h, Th, Jh, Qh, Kh,  
   Ad, 2d, 3d, 4d, 5d, 6d, 7d, 8d, 9d, Td, Jd, Qd, Kd,  
   Ac, 2c, 3c, 4c, 5c, 6c, 7c, 8c, 9c, Tc, Jc, Qc, Kc}
* **2.2\_\_\_\_\_\_\_\_\_\_\_ (4 คะแนน) shuffle** methodให้ทำการวนสลับไพ่ทุกใบ วิธีการคือ ทำการสลับตำแหน่งของไพ่แต่ละใบ กับ ตำแหน่งที่สุ่มได้ในช่วง (0-ตำแหน่งของไพ่)และให้ทำการเรียกใช้ **swap** method ทำการสลับไพ่ตามตำแหน่ง index ต้นทางและปลายทางที่ระบุในพารามิเตอร์
* **2.3\_\_\_\_\_\_\_\_\_\_\_ (4 คะแนน) getMaxCard** ทำการหาไพ่ในสำรับที่มีค่ามากที่สุด โดยเปรียบเทียบไพ่จากค่าความต่างของค่าประจำสัญลักษณ์ (suit) ถ้าค่าประจำสัญลักษณ์ (suit) เหมือนกัน ให้คำนวนจากค่าความต่างของเลขบนหน้าไพ่ (rank) แทน เช่น (5s < Ks ) (2d > Jh) ถ้าสำรับไม่มีไพ่ ให้คืนค่า null

**Deck**

-size:int

+Deck()

+deal(): Card

+isEmpty(): boolean

+shuffle(): void

+size(): int

+swap(int sourceindex, int targetIndex): void

+getCards: Card[]

+getMaxCard: Card

+toString(): String

**Card**

+ SPADES: final int = 0;

+ HEARTS: final int = 1;

+ DIAMONDS: final int = 2;

+ CLUBS: final int = 3;

+ ACE: final int = 1;

+ JACK: final int 11;

+ QUEEN: final int 12;

+ KING: final int = 13;

- rank: int

- suit: int;

+Card(int rank, int suit)

+equals(Object that): boolean

+getRank():int

+getSuit(): int

+toString():String

+SitBankCustomer(account:long, name:String, balance:double, phone:String)

+getBalance(): double

+setBalance(balance:double): void

+getAccount(): long

+setAccount(account:long): void

+equals(Object o):boolean

+toString(): String