คลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList` ที่เรียนมาเป็นตัวอย่างการสร้างคลาสสำหรับโครงสร้างข้อมูลพื้นฐานที่คล้ายกับ `ArrayList<E>` และ `LinkedList<E>` ในจาวา แบบฝึกหัดนี้เป็นการฝึก

- เปรียบเทียบคลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList`,
- สร้าง method เพิ่มในคลาส `ArrayList`, `SinglyLinkedList`, `LinkedList` และ
- นำคลาส `LinkedCollection` ไปสร้างโครงสร้างข้อมูลใหม่

แบบฝึกหัดบางส่วนมาจาก “โครงสร้างข้อมูล : ฉบับวาจาจาวา” (https://www.cp.eng.chula.ac.th/books/ds-vjjv/) โดย รศ.ดร.สมชาย ประสิทธิ์จูตระกูล

เปรียบเทียบคลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList`

เขียนโปรแกรมจับเวลาเพื่อเปรียบเทียบ running time สำหรับ operation ต่างๆ สำหรับคลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList` โดยหาเวลาทำงานเฉลี่ย (average running time) จากการทำงานกับค่าที่สุ่มขึ้นมา 10000 ครั้ง สำหรับ List ที่มีขนาดต่างกัน (เช่น 10000, 100000) เพื่อนำมาวาดกราฟเปรียบเทียบ running time ของ แต่ละ operation บน List ขนาดต่างๆ กัน operation ที่จะเปรียบเทียบ คือ method **`add`**, method **`get`**, method **`set`**, method **`remove`**, method **`contains`**

เขียน methods สำหรับคลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList`

เขียน method ต่อไปนี้เพิ่มสำหรับคลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList`

- `Object getFirst()` ที่คืนข้อมูลตัวแรกใน List ที่เป็น implicit parameter
- `Object getLast()` ที่คืนข้อมูลตัวสุดท้ายใน List ที่เป็น implicit parameter
- `void removeFirst()` ที่ลบข้อมูลตัวแรกใน List ที่เป็น implicit parameter
- `void removeLast()` ที่ลบข้อมูลตัวสุดท้ายใน List ที่เป็น implicit parameter
- `void removeRange(int from, int to)` ที่ลบข้อมูลตัวที่ `from` ไปถึงตัวที่ `to-1` ออกจาก List ที่เป็น implicit parameter
- `int indexOf(Object e)` ที่คืนเลขลำดับใน List ที่พบ object `e` เป็นตัวแรก
- `int lastIndexOf(Object e)` ที่คืนเลขลำดับใน List ที่พบ object `e` เป็นตัวสุดท้าย
- `void reverse()` ที่เรียงค่าใน List ย้อนกลับทาง
- `void cutPaste(LinkedList a, int i, int j, int k))` ที่ตัดโหนดที่ลำดับที่ `i` ถึง `j` ของ `a` ไปแทรกใน list ที่เป็น implicit parameter ที่ลำดับที่ `k` เป็นต้นไป

และสร้าง test class ที่ใช้คลาส `ArrayList`, `SinglyLinkedList` และ `LinkedList` เพื่อทดสอบ method ที่สร้างมา

เขียน method เพิ่มสำหรับคลาส `Polynomial`

เขียน method ต่อไปนี้เพิ่มสำหรับคลาส `Polynomial`

- `Polynomail add(Polynomial p)` ที่คืนผลบวกของ `Polynomial p` กับ `Polynomial` ที่เป็น implicit parameter
- `Polynomail multiply(Polynomial p)` ที่คืนผลคูณของ `Polynomial p` กับ `Polynomial` ที่เป็น implicit parameter
- `double eval(double c)` ที่คืนค่าของ polynomial function เมื่อ `x` มีค่าเป็น `c`
- `int order()` ที่คืนเลขชี้กำลังสูงสุดของ polynomial function
- `void removeLast()` ที่ลบข้อมูลตัวสุดท้ายใน List ที่เป็น implicit parameter

สร้างคลาส `Polynomial` ใหม่

เขียนคลาส `Polynomial` ใหม่ โดย extends จากคลาส `LinkedList`

เขียน method เพิ่มสำหรับคลาส `SparseMatrix`

สำหรับคลาส **`SparseMatrix`** ที่เรียนมา เขียน method **`power(int c)`** ที่คืนค่าเป็น matrix นั้นคูณกัน **`c`** ครั้ง แต่ให้ใช้วิธีที่ไม่ใช้เวลามากเท่าการคูณ **`c`** ครั้ง (แนะนำ: $x^8 = ((x^2)^2)^2$ และ $x^7 = ((x^2)^2)(x^2)(x)$)

สร้าง interface `Vector` และ interface `Matrix`

คลาส **`SparseVector`** และคลาส **`SparseMatrix`** ที่เรียนมามี method ที่รับพารามิเตอร์ในคลาส **`SparseVector`** และ **`SparseMatrix`** เท่านั้น ให้

- สร้าง interface **`Vector`** และ interface **`Matrix`** แล้ว
- สร้างคลาส **`DenseVector`** ที่ implements **`Vector`** และคลาส **`DenseMatrix`** ที่ implements **`Matrix`** โดยให้ method ในคลาสทั้งสองรับพารามิเตอร์ที่เป็น dense หรือ sparse vector/matrix ก็ได้
- แก้คลาส **`SparseVector`** ให้ implements **`Vector`** และคลาส **`SparseMatrix`** ให้ implements **`Matrix`** และให้ method ในคลาสทั้งสองรับพารามิเตอร์ที่เป็น dense หรือ sparse vector/matrix ก็ได้

จากนั้นเขียนคลาส **`Main`** เพื่อทดสอบคลาสทั้งหมดที่สร้างมา