CPE217 – Homework 3

Homework: Linked list data structures

Homework Due Date: 8 August 2025

Patiwet Wuttisarnwattana, Ph.D.

Department of Computer Engineering

- คำชี้แจงการส่งงาน
- ให้ทุกคนเข้าโมดูล Assignment (Link สีแดง) อ่านคำอธิบายการบ้านใน PDF และโหลด Java Starter Code เพื่อทำความเข้าใจโจทย์
- หลังจากนั้นให้ทุกคนทำการบ้านพร้อมกันกับกลุ่มของตัวเอง พร้อมปรึกษากับ Core Person ในกลุ่มตัวเองว่าจะส่งคำตอบสุดท้ายว่าเป็นอะไร
- ทุกคนสามารถตรวจคำตอบของตัวเองได้ ว่าทำมาถูกต้องหรือไม่โดยใช้โมดูล Quiz (Link สีเหลือง) แต่อาจารย์จะตรวจคะแนนจาก Core Person เท่านั้น นศ ทุกคนจะได้คะแนนเท่ากันทั้งกลุ่ม แม้ว่าคุณจะส่งโค้ดใน Link สีเหลืองแตกต่างกันก็ตามที
- นศ ต้องเติมโค้ดในพื้นที่ที่กำหนดให้เท่านั้น ห้ามประกาศตัวแปรระดับคลาสเพิ่ม ห้ามสร้างฟังก์ชันอื่น ๆ หรือเรียกใช้ฟังก์ชันพิเศษเพิ่ม (ไม่แน่ใจสามารถสอบถามได้ที่อาจารย์) ให้เติมโค้ดใน Template ของอาจารย์เท่านั้น ฝ่าฝืนหักคะแนน 25% แม้ผลลัพธ์สุดท้ายจะทำงานได้ถูกต้อง
- เมื่อ Core Person ส่งคำตอบแล้ว ให้ Core Person เข้าโมดูล Assignment (Link สีแดง) และใส่รหัสของเพื่อนในกลุ่มลงในช่องคำตอบ
- TA จะตรวจคำตอบในโมดูล Quiz และนำคะแนนมาลงในโมดูล Assignment เพื่อให้ทุกคนในกลุ่มได้คะแนนเท่ากันครับ
- โค้ดของคุณต้องมี Comment เพื่ออธิบายว่าโค้ดดังที่เห็นอยู่นี้ทำงานอะไร หรือ if นี้ใช้เพื่อแยกกรณีใดออกมา กลุ่มไหนที่ไม่มีคอมเม้นต์จะถูกหักคะแนน 50% การเขียนคอมเม้นต์ไม่ต้องเขียนละเอียดยิบ เขียนเท่าที่คุณต้องการให้ผู้ตรวจทราบก็พอ

นศ ที่จะส่งคำตอบ ท่านต้องให้คำมั่นปฏิญาณต่อคำพูดดังต่อไปนี้ หากไม่สามารถทำได้ ท่านจะไม่มีสิทธิ์ส่งงาน

- *ข้าพเจ้าและเพื่อนในกลุ่มเข้าใจและตระหนักดีว่า ในการทำการบ้านนี้ ข้าพเจ้าและเพื่อนในกลุ่มจะช่วยกันทำงานนี้ให้เสร็จสิ้นเอง โดยไม่ปรึกษาหรือแบ่งปันข้อมูลกับกลุ่มอื่น ๆ หรือบุคคลภายนอก*
- *หากข้าพเจ้าเป็นรุ่นพี่ที่กลับมาเรียนวิชานี้อีกครั้ง ข้าพเจ้าตระหนักดีว่า ข้าพเจ้าจะทำงานให้เสร็จสิ้นเองอีกครั้ง โดยไม่ดูคำตอบของปีก่อน ๆ*
- *ข้าพเจ้าจะไม่เผยแพร่เนื้อหาโจทย์การบ้านนี้ออกสู่สาธารณะโดยเด็ดขาด*
- *หากข้าพเจ้าไม่สามารถปฏิบัติตามคำมั่นนี้ได้ ข้าพเจ้ายินดีที่จะยอมรับคะแนน ศูนย์คะแนน ในทุก ๆ การบ้านโดยไม่โต้แย้ง*

การบ้านนี้ให้นักศึกษา implement “Linked List ADT” โดยใช้ Java โดยให้มีคุณสมบัติดังต่อไปนี้

1. ให้สร้าง class ชื่อว่า DoublyLinkedList โดย class นี้ มีคุณสมบัติของ “Doubly-linked list with tail” ตามที่ได้เรียนในห้องเรียน
2. ให้ object ของ DoublyLinkedList ทำหน้าที่บรรจุ objects ของ class Node โดย class Node นี้ มีสมาชิกที่เป็นตัวแปรอย่างน้อยสี่ตัว คือ herold (ชนิด int), name (ชนิด String), power (ชนิด double) และ Era (ชนิด Era)
    a. ในการ implement Doubly-linked list with tail คุณสามารถใช้ references head และ tail ได้อย่างเต็มที่ แต่ถ้าจะทำการ implement Singly-linked list without tail (ตามข้อ 6) คุณสามารถใช้ reference head ได้แต่เพียงอย่างเดียว
    b. Era เป็นประเภท Enums ศึกษาได้จากที่นี่ https://www.w3schools.com/java/java_enums.asp
3. ให้ class Node มี 2 Constructors (Overloading functions) ดังนี้
    a. Node(int id, String name, double power, Era era)
        - ทำหน้าที่ กำหนดค่าเริ่มต้นของทั้งสามตัวแปรตามที่ผู้ใช้กำหนด
    b. Node(String error_msg)
        - Constructor ชนิดนี้จะถูกใช้เพื่อสร้าง dummy Node เพื่อ return ให้กับ caller ในกรณีเกิด ERROR ขึ้น
        - โดย Error Message ที่เกิดขึ้นให้เซตไปที่ค่าของตัวแปร name ส่วนค่าเริ่มต้นสำหรับตัวแปรที่เหลือ ให้ใช้ค่า default ของ Java
    c. ใน starter code ผมได้เติม constructor Node() ไว้อีกอันนึง เพื่อไม่ให้ dummy code ของผมมันฟ้อง error คุณไม่ต้องสนใจฟังก์ชันอันนี้นะครับ ให้สนใจเฉพาะสองฟังก์ชันข้างบนก็พอ
4. ให้ class Node มีฟังก์ชัน public void printHero()
        - ฟังก์ชันนี้ทำหน้าที่ พิมพ์ค่าของ herold, name, power และ era ของออกทาง console
        - ใน starter code มีให้ละครับ
5. ให้ class DoublyLinkedList มี public functions ดังต่อไปนี้
    a. public LinkedList1(String name)
        - ฟังก์ชันนี้ทำหน้าที่เป็น Constructor ของ List โดยกำหนดชื่อของ List คือค่าของตัวแปร name
    b. public void pushHead(Node node)
        - ฟังก์ชันนี้ทำหน้าที่นำ node มาเติมข้างหน้าสุดของ list (referenced by head)
    c. public Node topHead()
        - ฟังก์ชันนี้ทำหน้าที่ return node ที่อยู่ข้างหน้าสุดของ list (referenced by head)
        - ถ้า List ว่าง ให้ print ออกทาง console ก่อนเลยว่า ERROR แล้ว ให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Empty List!”
    d. public void popHead()
        - ฟังก์ชันนี้ทำหน้าที่ ลบ node ที่อยู่ข้างหน้าสุดของ list (referenced by head) ออกไป

- ถ้า List ว่าง ให้ print ออกทาง console ว่า ERROR

e. public void pushTail(Node node)

- ฟังก์ชันนี้ทำหน้าที่นำ node ไปต่อท้าย node ที่อยู่หลังสุด

f. public Node topTail()

- ฟังก์ชันนี้ทำหน้าที่ return node ที่อยู่ข้างหลังสุดของ list
- ถ้า List ว่าง ให้ print ออกทาง console ก่อนเลยว่า ERROR แล้ว ให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Empty List!”

g. public void popTail()

- ฟังก์ชันนี้ทำหน้าที่ ลบ node ที่อยู่ข้างหลังสุดของ list ออกไป
- ถ้า List ว่าง ให้ print ออกทาง console ว่า ERROR

h. public Node findNode(int id)

- ฟังก์ชันนี้ทำหน้าที่ return Node ที่มีค่า heroId เท่ากับ id
- ถ้า List ว่าง ให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Empty List!” (ไม่ต้องแจ้งว่า ERROR)
- ถ้า List ไม่ว่างแต่หา node นั้นไม่เจอให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Hero Not Found!”

i. public Node eraseNode(int id)

- ฟังก์ชันนี้ทำหน้าที่หา Node ที่มีค่า heroId เท่ากับ id ใน List เมื่อเจอแล้วให้ลบ Node นั้นออกจาก List นอกจากนี้ ให้ return Node ที่ลบนั้นออกไปให้ caller อีกด้วย
- ถ้า List ว่าง ให้ print ออกทาง console ก่อนเลยว่า ERROR แล้ว ให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Empty List!”
- ถ้าหา node นั้นไม่เจอให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Hero Not Found!”

j. public boolean isEmpty()

- ฟังก์ชันนี้ทำหน้าที่ return ว่า Data structure นี้ว่างหรือไม่

k. public void addNodeAfter(Node node1, Node node2)

- ฟังก์ชันนี้ทำหน้าที่นำ node2 (ซึ่งเป็น Node เดี่ยวๆ) ไปแทรกใน list โดยนำไปแทรกด้านหลังของ node1 (ซึ่งมีอยู่ใน list อยู่แล้ว)

l. public void addNodeBefore(Node node1, Node node2)

- ฟังก์ชันนี้ทำหน้าที่นำ node2 (ซึ่งเป็น Node เดี่ยวๆ) ไปแทรกใน list โดยนำไปแทรกด้านหน้าของ node1 (ซึ่งมีอยู่ใน list อยู่แล้ว)

m. public Node getHighestPowerHero()

- ฟังก์ชันนี้ทำหน้าที่หาว่า Node ไหนมี Hero ที่มี POWER สูงที่สุด
- ถ้า List ว่าง ให้ return ด้วย Node ที่ถูกสร้างด้วย Constructor แบบที่สอง โดยให้มี Error Message ว่า “Empty List!” (ไม่ต้องแจ้งว่า ERROR)
- ถ้าคนที่ได้ POWER สูงที่สุดมีมากกว่าหนึ่งคน ให้ return คนที่อยู่ไกล head (ใกล้ tail) มากที่สุด

n. public void merge(List list)

- ฟังก์ชันนี้ทำหน้าที่รวม List สองตัวเข้าด้วยกัน โดยนำ List (list) ที่นำเข้า เข้าต่อด้านหลังสุดของ List (this) ปัจจุบัน

o. public void printStructure()

- ให้ print สถานะข้อมูลของ Data structure ออกทาง console ด้วย pattern ดังต่อไปนี้
- หาก SinglyLinkedList ชื่อ list1 มีข้อมูลคือ {{59061, “V3”, 3.0, Era.SHOWA}, {59062, “X”, 2.0, Era.SHOWA }, {59063, “Skyrider”, 2.5 Era.SHOWA }, {59064, “Stronger”, 3.25 Era.SHOWA }} ให้แสดงว่า
  - list1: head –> {59061} –> {59062} –> {59063} –> {59064} –> null
- หาก DoubleLinkedList ชื่อ list2 มีข้อมูลคือ {{59064, “Kuuga”, 3.25, Era.HEISEI}, {59062, “Agito”, 2.0, Era.HEISEI}} ให้แสดงว่า
  - list2: head <–> {59064} <–> {59062} <–> tail

6. ให้สร้าง class ชื่อว่า SinglyLinkedList โดย class นี้ มีคุณสมบัติของ “Singly–linked list without tail” (ไม่มีหาง) ตามที่ได้เรียนในห้องเรียน โดยกำหนดให้คลาสนี้สามารถดำเนินการได้ตามข้อ 5a. – 5o. เหมือนกับที่คุณทำมาแล้วได้ทั้งหมด
7. ตัวอย่างการทำงาน

Java code

```
public static void main(String[] args) {
        Node node;
        DoublyLinkedList list1 = new DoublyLinkedList("list1");

        node = new Node(320, "Build", 63.4, Era.HEISEI);
        list1.pushHead(node);

        node = new Node(321, "Cross–Z", 59.3, Era.HEISEI);
        list1.pushHead(node);

        node = new Node(322, "Rogue", 72.8, Era.HEISEI);
```

```
        list1.pushHead(node);


        node = new Node(323, "Killbus", 98.4, Era.HEISEI);
        list1.pushHead(node);
        list1.printStructure();

        DoublyLinkedList list2 = new DoublyLinkedList("list2");

        list2.pushTail(new Node(324, "Grease", 85.1, Era.HEISEI));
        list2.pushTail(new Node(325, "Blood", 67.6, Era.HEISEI));
        list2.pushTail(new Node(326, "MadRogue", 51.2, Era.HEISEI));
        list2.pushTail(new Node(327, "Evol", 79.9, Era.HEISEI));

        list2.printStructure();

        list1.merge(list2);

        list1.printStructure();
}
```

Output

```
list1: head <-> [323] <-> [322] <-> [321] <-> [320] <-> tail
list2: head <-> [324] <-> [325] <-> [326] <-> [327] <-> tail
list1: head <-> [323] <-> [322] <-> [321] <-> [320] <-> [324] <-> [325] <-> [326] <-> [327] <-> tail
```

ผลลัพธ์เมื่อเปลี่ยนชื่อ class จาก DoublyLinkedList เป็น SinglyLinkedList

```
list1: head -> [323] -> [322] -> [321] -> [320] -> null
list2: head -> [324] -> [325] -> [326] -> [327] -> null
list1: head -> [323] -> [322] -> [321] -> [320] -> [324] -> [325] -> [326] -> [327] -> null
```

Java code

```
public static void main(String[] args) {
        Node node;
        DoublyLinkedList list = new DoublyLinkedList("list3");

        node = list.getHead();
        node.printHero();
```

```
        node = list.getTail();
        node.printHero();

        list.pushTail(new Node(301, "Den0", 582.0, Era.HEISEI));
        list.pushTail(new Node(302, "Kiva", 626.0, Era.HEISEI));
        list.pushTail(new Node(303, "Decade", 560.0, Era.HEISEI));

        node = list.getHead();
        node.printHero();

        node = list.getTail();
        node.printHero();
}
```

Output

```
ERROR
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
ERROR
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
HeroID: 301 , Name: Den0 , Power: 582.0, Era: Heisei rider
HeroID: 303 , Name: Decade , Power: 560.0, Era: Heisei rider
```

ผลลัพธ์เมื่อเปลี่ยนชื่อ class จาก DoublyLinkedList เป็น SinglyLinkedList → เหมือนเดิม

Java code

```
public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList("list4");

        list.printStructure();
        list.popHead();

        list.pushTail(new Node(401, "V3", 648.0, Era.SHOWA));
        list.pushTail(new Node(402, "Gaim", 516.0, Era.HEISEI));
        list.pushTail(new Node(403, "Wizard", 604.0, Era.HEISEI));
        list.pushTail(new Node(404, "ZX", 560.0, Era.SHOWA));
```

```
        list.printStructure();
        list.popHead();
        list.printStructure();
        list.popTail();
        list.printStructure();
        list.popHead();
        list.printStructure();
        list.popTail();
        list.printStructure();
        list.popTail();
}
```

Output

```
list4: head <-> tail
ERROR
list4: head <-> [401] <-> [402] <-> [403] <-> [404] <-> tail
list4: head <-> [402] <-> [403] <-> [404] <-> tail
list4: head <-> [402] <-> [403] <-> tail
list4: head <-> [403] <-> tail
list4: head <-> tail
ERROR
```

ผลลัพธ์เมื่อเปลี่ยนชื่อ class จาก DoublyLinkedList เป็น SinglyLinkedList

```
list4: head -> null
ERROR
list4: head -> [401] -> [402] -> [403] -> [404] -> null
list4: head -> [402] -> [403] -> [404] -> null
list4: head -> [402] -> [403] -> null
list4: head -> [403] -> null
list4: head -> null
ERROR
```

Java code

```
public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList("list5");
        (list.findNode(201)).printHero();
        (list.eraseNode(201)).printHero();
```

```
        list.pushTail(new Node(201, "Geats", 560.0, Era.REIWA));
        list.pushTail(new Node(202, "Zeztz", 494.0, Era.REIWA));
        list.pushTail(new Node(203, "Gotchard", 563.0, Era.REIWA));
        list.pushTail(new Node(204, "Super-1", 582.0, Era.SHOWA));
        list.pushTail(new Node(205, "X", 516.0, Era.SHOWA));
        list.pushTail(new Node(206, "Amazon", 562.25, Era.SHOWA));

        list.printStructure();

         (list.findNode(201)).printHero();
         (list.findNode(206)).printHero();
         (list.findNode(207)).printHero();

        Node node = list.findNode(203);
        list.addNodeAfter(node, new Node(207, "Gavv", 626.0, Era.REIWA));

        list.printStructure();

         (list.eraseNode(201)).printHero();
        list.printStructure();

         (list.eraseNode(206)).printHero();
        list.printStructure();

         (list.eraseNode(203)).printHero();
        list.printStructure();

         (list.eraseNode(203)).printHero(); // duplicate
        list.printStructure();
}
```

Output

```
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
ERROR
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
```

```
list5: head <-> [201] <-> [202] <-> [203] <-> [204] <-> [205] <-> [206] <-> tail
HeroID: 201 , Name: Geats , Power: 560.0, Era: Reiwa rider
HeroID: 206 , Name: Amazon , Power: 562.3, Era: Showa rider
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list5: head <-> [201] <-> [202] <-> [203] <-> [207] <-> [204] <-> [205] <-> [206] <-> tail
HeroID: 201 , Name: Geats , Power: 560.0, Era: Reiwa rider
list5: head <-> [202] <-> [203] <-> [207] <-> [204] <-> [205] <-> [206] <-> tail
HeroID: 206 , Name: Amazon , Power: 562.3, Era: Showa rider
list5: head <-> [202] <-> [203] <-> [207] <-> [204] <-> [205] <-> tail
HeroID: 203 , Name: Gotchard , Power: 563.0, Era: Reiwa rider
list5: head <-> [202] <-> [207] <-> [204] <-> [205] <-> tail
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list5: head <-> [202] <-> [207] <-> [204] <-> [205] <-> tail
```

ผลลัพธ์เมื่อเปลี่ยนชื่อ class จาก DoublyLinkedList เป็น SinglyLinkedList

```
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
ERROR
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
list5: head -> [201] -> [202] -> [203] -> [204] -> [205] -> [206] -> null
HeroID: 201 , Name: Geats , Power: 560.0, Era: Reiwa rider
HeroID: 206 , Name: Amazon , Power: 562.3, Era: Showa rider
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list5: head -> [201] -> [202] -> [203] -> [207] -> [204] -> [205] -> [206] -> null
HeroID: 201 , Name: Geats , Power: 560.0, Era: Reiwa rider
list5: head -> [202] -> [203] -> [207] -> [204] -> [205] -> [206] -> null
HeroID: 206 , Name: Amazon , Power: 562.3, Era: Showa rider
list5: head -> [202] -> [203] -> [207] -> [204] -> [205] -> null
HeroID: 203 , Name: Gotchard , Power: 563.0, Era: Reiwa rider
list5: head -> [202] -> [207] -> [204] -> [205] -> null
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list5: head -> [202] -> [207] -> [204] -> [205] -> null
```

Java code

```
public static void main(String[] args) {
        Node node;

        DoublyLinkedList list1 = new DoublyLinkedList("list6");

        node = new Node(320, "Kuuga", 63.4, Era.HEISEI);
        list1.pushHead(node);

        node = new Node(321, "Agito", 59.3, Era.HEISEI);
```

```
list1.pushHead(node);

node = new Node(322, "Ryuki", 72.8, Era.HEISEI);
list1.pushHead(node);

node = new Node(323, "Blade", 98.4, Era.HEISEI);
list1.pushHead(node);

list1.printStructure();

DoublyLinkedList list2 = new DoublyLinkedList("list7");

list2.pushTail(new Node(324, "Faiz", 85.1, Era.HEISEI));
list2.pushTail(new Node(325, "Hibiki", 67.6, Era.HEISEI));
list2.pushTail(new Node(326, "000", 51.2, Era.HEISEI));
list2.pushTail(new Node(327, "Drive", 79.9, Era.HEISEI));

list2.printStructure();

list1.merge(list2);

list1.printStructure();

Node strongest = list1.getHighestPowerHero();
System.out.println("Highest Power Hero:");
strongest.printHero();

Node exaid = new Node(328, "Exaid", 102.6, Era.REIWA);
list1.addNodeAfter(strongest, exaid);

Node grease = list1.findNode(324);
Node zio = new Node(329, "Zio", 77.7, Era.HEISEI);
list1.addNodeBefore(grease, zio);
```

```
        list1.printStructure();
}
```

Output

```
list6: head <-> [323] <-> [322] <-> [321] <-> [320] <-> tail
list7: head <-> [324] <-> [325] <-> [326] <-> [327] <-> tail
list6: head <-> [323] <-> [322] <-> [321] <-> [320] <-> [324] <-> [325] <-> [326] <-> [327] <-> tail
Highest Power Hero:
HeroID: 323 , Name: Blade , Power: 98.4, Era: Heisei rider
list6: head <-> [323] <-> [328] <-> [322] <-> [321] <-> [320] <-> [329] <-> [324] <-> [325] <-> [326] <-> [327] <-> tail
```

ผลลัพธ์เมื่อเปลี่ยนชื่อ class จาก DoublyLinkedList เป็น SinglyLinkedList

```
list6: head -> [323] -> [322] -> [321] -> [320] -> null
list7: head -> [324] -> [325] -> [326] -> [327] -> null
list6: head -> [323] -> [322] -> [321] -> [320] -> [324] -> [325] -> [326] -> [327] -> null
Highest Power Hero:
HeroID: 323 , Name: Blade , Power: 98.4, Era: Heisei rider
list6: head -> [323] -> [328] -> [322] -> [321] -> [320] -> [329] -> [324] -> [325] -> [326] -> [327] -> null
```

Java code

```
public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList("list7");
        (list.findNode(5901)).printHero();
        (list.eraseNode(5901)).printHero();
        list.pushTail(new Node(5901, "Chalice", 350, Era.REIWA));
        list.pushTail(new Node(5902, "Meteor", 275, Era.HEISEI));
        list.pushTail(new Node(5903, "Accel", 300, Era.HEISEI));
        list.pushTail(new Node(5904, "Valen", 375, Era.SHOWA));
        list.pushTail(new Node(5905, "Tycoon", 325, Era.REIWA));
        list.pushTail(new Node(5906, "Buffa", 285, Era.HEISEI));
        list.printStructure();
        (list.findNode(5901)).printHero();
        (list.findNode(5906)).printHero();
        (list.findNode(5907)).printHero();
        Node node = list.findNode(5903);
        list.addNodeAfter(node, new Node(5907, "Revice", 250, Era.SHOWA));
        list.printStructure();
        (list.eraseNode(5901)).printHero();
        list.printStructure();
```

```
        (list.eraseNode(5906)).printHero();
        list.printStructure();
        (list.eraseNode(5903)).printHero();
        list.printStructure();
        (list.eraseNode(5903)).printHero();
        list.printStructure();
}
```

Output

```
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
ERROR
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
list7: head <-> [5901] <-> [5902] <-> [5903] <-> [5904] <-> [5905] <-> [5906] <-> tail
HeroID: 5901 , Name: Chalice , Power: 350.0, Era: Reiwa rider
HeroID: 5906 , Name: Buffa , Power: 285.0, Era: Heisei rider
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list7: head <-> [5901] <-> [5902] <-> [5903] <-> [5907] <-> [5904] <-> [5905] <-> [5906] <-> tail
HeroID: 5901 , Name: Chalice , Power: 350.0, Era: Reiwa rider
list7: head <-> [5902] <-> [5903] <-> [5907] <-> [5904] <-> [5905] <-> [5906] <-> tail
HeroID: 5906 , Name: Buffa , Power: 285.0, Era: Heisei rider
list7: head <-> [5902] <-> [5903] <-> [5907] <-> [5904] <-> [5905] <-> tail
HeroID: 5903 , Name: Accel , Power: 300.0, Era: Heisei rider
list7: head <-> [5902] <-> [5907] <-> [5904] <-> [5905] <-> tail
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list7: head <-> [5902] <-> [5907] <-> [5904] <-> [5905] <-> tail
```

ผลลัพธ์เมื่อเปลี่ยนชื่อ class จาก DoublyLinkedList เป็น SinglyLinkedList

```
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
ERROR
HeroID: 0 , Name: Empty List! , Power: 0.0, Era: null
list7: head -> [5901] -> [5902] -> [5903] -> [5904] -> [5905] -> [5906] -> null
HeroID: 5901 , Name: Chalice , Power: 350.0, Era: Reiwa rider
HeroID: 5906 , Name: Buffa , Power: 285.0, Era: Heisei rider
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list7: head -> [5901] -> [5902] -> [5903] -> [5907] -> [5904] -> [5905] -> [5906] -> null
HeroID: 5901 , Name: Chalice , Power: 350.0, Era: Reiwa rider
list7: head -> [5902] -> [5903] -> [5907] -> [5904] -> [5905] -> [5906] -> null
HeroID: 5906 , Name: Buffa , Power: 285.0, Era: Heisei rider
list7: head -> [5902] -> [5903] -> [5907] -> [5904] -> [5905] -> null
HeroID: 5903 , Name: Accel , Power: 300.0, Era: Heisei rider
list7: head -> [5902] -> [5907] -> [5904] -> [5905] -> null
HeroID: 0 , Name: Hero Not Found! , Power: 0.0, Era: null
list7: head -> [5902] -> [5907] -> [5904] -> [5905] -> null
```

8. โปรดใช้ Starter code ที่อาจารย์แนบให้