Peer Assisted Study Session

FIT2099 - Week 10

Monash University

Objectives

- Explore inner/nested/anonymous classes and their uses
- Explore Refactoring Basics

Estimated Time

Question 1 (20 Minutes)
Question 2 (10 Minutes)
Question 3 (10 Minutes)
Question 4 (15 Minutes)

Questions

1. Read these pages on nested classes, inner classes, and anonymous classes.

   Using an appropriate type of class (inner, anonymous, etc) complete the class by defining an Iterator and implementing the Iterable interface method (iterator) within the LinkedList class.

```
import java.util.*;


// class Main {
//   public static void main(String[] args) {
//     LinkedList<Integer> ns = new LinkedList<>(1,2,3,4,5);
//     println(ns);


//     for (int n : ns) {
//       println(n);
//     }
//   }
 //   static <T> void println(T t) {
//     System.out.println(t);
//   }
// }
```

```
class LinkedList<T> {
  private ListNode head;
  class ListNode {
   T value;
   ListNode next;

   ListNode(T value, ListNode next) {
     this.value = value;
     this.next = next;
   }
  }
   LinkedList(T... vals) {
   // loop over values from last to first
   // create a node for each value and link
   // to the next node in the chain
   ListNode next = null;
   for (int i = vals.length-1; i >= 0; i--) {
     ListNode current = new ListNode(vals[i], next);
     next = current;
   }
   head = next;
  }
   public String toString() {
   String s = "[";

   ListNode node = head;
   while (node != null) {
     s += node.value + (node.next == null ? "" : " ");
     node = node.next;
   }

   return s + "]";
  }
 }

import java.util.*;
```

```
class Main {
 public static void main(String[] args) {
   LinkedList<Integer> ns = new LinkedList<>(1,2,3,4,5);
   println(ns);

   for (int n : ns) {
     println(n);
   }
 }
  static <T> void println(T t) {
   System.out.println(t);
 }
}

class LinkedList<T>
 implements Iterable<T> {
  private ListNode head;
  class ListNode {
   T value;
   ListNode next;

   ListNode(T value, ListNode next) {
     this.value = value;
     this.next = next;
   }
 }
  LinkedList(T... vals) {
   // loop over values from last to first
   // create a node for each value and link
   // to the next node in the chain
   ListNode next = null;
   for (int i = vals.length-1; i >= 0; i--) {
     ListNode current = new ListNode(vals[i], next);
     next = current;
   }
   head = next;
```

```
 }
  public String toString() {
   String s = "[";

   ListNode node = head;
   while (node != null) {
     s += node.value + (node.next == null ? "" : " ");
     node = node.next;
   }

   return s + "]";
 }
  public Iterator<T> iterator() {
   return new Iterator<T>() {
     ListNode current = head;

     public boolean hasNext() {
       return current != null;
     }

     public T next() {
       T val = current.value;
       current = current.next;
       return val;
     }
   };
 }
}
```

2. What is refactoring? Explain.

Refactoring modifies software to improve its readability, maintainability, and extensibility without changing what it actually does.
The goal of refactoring is NOT to add new functionality
The goal is refactoring is to make code easier to maintain in the future

3. Refactor the below code in-order to remove the check for null. [Hint: use inheritance]

```
public String getName(customer){
 if (customer == null) {
      name = “occupant”;
     return name;
 } else {
      name = customer.getName();
     return name;
 }
}
```

```
Customer c = findCustomer();
name = c.getName(c)
```

```
public class NullCustomer extends Customer {

  public String getName() {
    return “occupant”
}
```

4. Refactor the below code in-order to remove the switch case statement. Draw the UML Diagram first. Then implement it.

```
class Bird {
 double getSpeed() {
     switch (_type) {
       case EUROPEAN:
         return getBaseSpeed();
       case AFRICAN:
         return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
       case NORWEGIAN_BLUE:
```

```
            return (_isNailed) ? 0 : getBaseSpeed(_voltage);
}
      throw new RuntimeException ("Should be unreachable");
 }
}
```