Instructions

• Write your name and version number on the top of the yellow paper.
• Answer all questions on the yellow paper.
• One question per page.
• Use only one side of the yellow paper.

1. (16 Points) Multiple Choice:

A. (2 Points) Which of the following loop headers will arrange for the loop body to execute exactly 10 times?
   a. for (int i = 1; i < 10; ++i)
   b. for (int i = 0; i <= 10; ++i)
   c. for (int i = -5; i < 5; ++i)
   d. for (int i = 2; i < 20; ++i)

B. (2 Points) Which access modifier, used when defining a method, indicates that only one such method is available for all instances of the class?
   a. final
   b. private
   c. protected
   d. static

C. (2 Points) Which of the following is an example of a syntax error?
   a. a program encounters an instruction to divide by zero
   b. an array subscript in a program goes out of range
   c. the beginning of a while loop is written as “whilele” instead of “while”
   d. an algorithm that calculates the monthly payment of a loan displays incorrect results

D. (2 Points) Data structures are part of an ADT’s
   a. definition
   b. implementation
   c. specifications
   d. usage

E. (2 Points) In the following list:
   John, Kate, Fred, Mark, Jon, Adam, Drew
   which element does not have a predecessor?
   a. John
   b. Mark
   c. Drew
   d. Adam

F. (2 Points) If the array: {6, 2, 7, 13, 5, 4} is added to a stack, in the order given, which number will be the first number to be removed from the stack?
   a. 6
   b. 2
   c. 5
   d. 4

G. (2 Points) The last node of a linear linked list
   a. has the value null
   b. has a next reference whose value is null
   c. has a next reference which references the first node of the list
   d. cannot store any data

H. (2 Points) Which of the following statements deletes the node that curr references?
   a. prev.setNext(curr);
   b. curr.setNext(prev);
   c. curr.setNext(curr.getNext());
   d. prev.setNext(curr.getNext());
2. (20 Points) Given the following generic MyArray class that contains syntax and logical errors:

```java
public class MyArray<I> {

    private I[] array = new Object[100];
    private int currentLocation;

    public void addElement(I element) {
        array[currentLocation] = element;
    }

    public void replaceElement(I newElement, index) {
        if (((index <= currentLocation) ||
             ((index < 0) || (index < array.length)))
             System.out.println("Error");
    
    array[index] = newElement;
    }

    public void removeElement(int index) {
        if (((index <= currentLocation) && (index < array.length)) ||
            ((index < 0) || (index < array.length)))
            System.println("Error");

        for (int i = index ; i < currentLocation ; i++) {
            array[i-1] = array[i+1];
        }
        array[--currentLocation] = null;
    }

    public void clear() {
        for (int i = 0 ; i < array.length ; i-- ) {
            array[i] = null;
        }
        currentLocation = 0;
    }

    public int numberOfElements() {
        return currentLocation;
    }
}
```

Re-write the MyArray class and fix the 10 syntax and logical errors.
3. (40 Points) Given the following definitions for Node<I> and QueueInterface<I>:

```java
public class Node<I> {
    private I element;
    private Node<I> next;

    public Node() {
        this.element = null;
        this.next = null;
    }

    public Node(I element) {
        this.element = element;
        this.next = null;
    }

    public I getElement() {
        return element;
    }

    public void setElement(I element) {
        this.element = element;
    }

    public Node<I> getNext() {
        return next;
    }

    public void setNext(Node<I> next) {
        this.next = next;
    }
}
```

```java
import java.util.Vector;

public interface QueueInterface<I> {
    // returns true if Queue is empty
    // returns false otherwise
    public boolean isEmpty();

    // returns the size of the Queue
    public int size();

    // adds the specified element
    // to the Queue
    public void enqueue(I element);

    // removes and returns the front
    // of the Queue
    public I dequeue();

    // tests if this Queue is equal to the
    // Queue specified by oQueue
    // Two Queues are equal if they have
    // the same size and all their elements
    // are equal
    public boolean equals(Object oQueue);

    // returns a Vector containing all the
    // elements in the Queue
    public Vector<I> peekAll();
}
```

Write the complete Java class for the linked-based LinkedQueue that implements the given QueueInterface.
4. (30 Points) Given the following list:

And the following method:

```java
public void doStuff1() {
    Node[] nodes = new Node[listSize];
    Node node = head;
    int i = 0;
    while (node != null) {
        nodes[i++] = node;
        node = node.getNext();
    }
    for (i = (listSize - 1); i > 0; i--) {
        nodes[i].setNext(nodes[i-1]);
    }
    head = nodes[listSize - 1];
    tail = nodes[0];
    tail.setNext(null);
}
```

Draw the list after doStuff1() has finished executing.
Instructions
• Write your name and version number on the top of the yellow paper.
• Answer all questions on the yellow paper.
• One question per page.
• Use only one side of the yellow paper.

1. (16 Points) Multiple Choice:
   A. (2 Points) The Java expression 9 / 5 + 9 % 5 equals _____.
      a. 0  
      b. 1  
      c. 3  
      d. 5  
      e. 6  

   B. (2 Points) Consider the following code that appears in a test class.
      A a = new A();
      int c = a.b;
   In order for this code to work, which statement must be true?
      a. a must be declared public inside class A
      b. b must be declared public inside class A
      c. c must be declared public inside class A
      d. Method b( ) must return int

   C. (2 Points) The syntax errors of a program are removed during the _____ phase of the program's life cycle
      a. verification
      b. coding
      c. testing
      d. refining
      e. maintenance

   D. (2 Points) An ADT's _____ govern(s) what its operations are and what they do.
      a. specifications
      b. implementation
      c. documentation
      d. data structure

   E. (2 Points) In the following list:
      John, Kate, Fred, Mark, Jon, Adam, Drew
      which element does not have a successor?
      a. John
      b. Mark
      c. Drew
      d. Adam

   F. (2 Points) If the array: {6, 21, 35, 3, 6, 2, 13} is added to a stack, in the order given, which of the following is the top of the stack?
      a. 2
      b. 6
      c. 3
      d. 13
      e. 35

   G. (2 Points) Which of the following will be true when the reference variable curr references the last node in a linear linked list?
      a. curr == null
      b. head == null
      c. curr.getNext() == null
      d. head.getNext() == null

   H. (2 Points) Which of the following statements deletes the first node of a linear linked list that has 10 nodes?
      a. head.setNext(curr.getNext());
      b. prev.setNext(curr.getNext());
      c. head = head.getNext();
      d. head = null;
2. (20 Points) Given the following generic MyArray class that contains syntax and logical errors:

```java
public class MyArray<I> {

    private I[] array = (I[]) new Object[];
    private int currentLocation;

    public void addElement(Object element) {
        array[currentLocation++] = element;
    }

    public void replaceElement(I newElement, int index) {
        if (((index >= currentLocation) || ((index > 0) && (index > array.length))) {
            System.println("Error");
        }
        array[index] = newElement;
    }

    public void removeElement(int index) {
        if (((index >= currentLocation) && (index > array.length)) || ((index < 0) || (index > array.length))) {
            System.out.println("Error");
        }
        for (int i = index - 1; i < currentLocation; i-- ) {
            array[i-1] = array[i];
        }
        array[--currentLocation] = null;
    }

    public void clear() {
        for (int i = 0 ; i >= array.length ; i++ ) {
            array[i-1] = null;
        }
        currentLocation = 0;
    }

    public int numberOfElements() {
        return currentLocation;
    }
}
```

Re-write the MyArray class and fix the 10 syntax and logical errors.
3. (40 Points) Given the following generic definitions for Node and StackInterface

```java
public class Node<I> {
    private I element;
    private Node<I> next;

    public Node() {
        this.element = null;
        this.next = null;
    }

    public Node(I element) {
        this.element = element;
        this.next = null;
    }

    public I getElement() {
        return element;
    }

    public void setElement(I element) {
        this.element = element;
    }

    public Node<i> getNext() {
        return next;
    }

    public void setNext(Node<i> next) {
        this.next = next;
    }
}
```

```java
import java.util.Vector;

public interface StackInterface<I> {
    // returns true if Stack is empty
    // returns false otherwise
    public boolean isEmpty();

    // returns the size of the Stack
    public int size();

    // pushes the specified element
    // onto the stack
    public void push(I element);

    // pops and returns the element
    // at the top of the stack
    public I pop();

    // tests if Stack is equal to the
    // Stack specified by oStack
    // Two Stacks are equal if they have
    // the same size and all their elements
    // are equal
    public boolean equals(Object oStack);

    // returns a Vector containing all the
    // elements in the Stack
    public Vector<i> peekAll();
}
```

Write the complete Java class for the linked-based LinkedStack that implements the given StackInterface.
4. (30 Points) Given the following list:

And the following method:

```java
public void doStuff2() {
    Node[] nodes = new Node[listSize];

    Node node = head;
    int i = 0;
    while (node != null) {
        nodes[i++] = node;
        node = node.getNext();
    }

    for (i = 1; i < (listSize - 3); i++) {
        nodes[i].setNext(nodes[i+2]);
    }

    for (i = (listSize - 1); i > 3; i--) {
        nodes[i].setNext(nodes[i-4]);
    }

    nodes[0].setNext(nodes[listSize - 1]);

    head = nodes[2];
    tail = nodes[3];
    tail.setNext(null);
}
```

Draw the list after `doStuff2()` has finished executing.