CS1102: Data Structure and Algorithms

Tutorial 2 - Abstract Data Types

(4, 5 February 2010)

1. a) What is the difference between abstract classes and interfaces?

   b) What fundamental principle(s) does class StudentA implement? Comparing class StudentA with class StudentB, what are the benefits of said principle(s)?

```
public class StudentA{
      private String name;
      private String matric;
      private double CAP;

      public StudentA(String name, String matric){
            this.name = name;
            this.matric = matric;
      }

      public void setCAP(double val) throws Exception{
            if (val <= 0 || val >5) throw new Exception("Illegal value!!");
            CAP = val;
      }

      public double getCAP(){return CAP;}
      ...
}

public class StudentB{
      public  String name;
      public String matric;
      public double CAP;

      public StudentB(String name, String matric){
            this.name = name;
            this.matric = matric;
      }
}

public class Test{
      public static void main(String[] args){
            StudentA s1= new StudentA("John Lee", "HT0851227A");
            try{
                  s1.setCAP(-10);
            }
            catch (Exception exp){
                  exp.printStackTrace();
            }

            StudentB s2 = new StudentB("Jane Smith", "HT051235B");
            s2.CAP = 345;
      }
}
```

c) Below is the pseudo-code for bubble sort, one of the most straightforward algorithms for sorting an integer array in an ascending order. The algorithm goes through the vector and swaps any adjacent values (`swap(items[i], items[i+1])`) if they are in the incorrect order. Write the pre and post conditions for the pseudo-code. Try to find a loop-invariant. What arrays would you use to test the method once you have implemented it?

```
method void bubbleSort(items[])
//pre-condition:
do
    swapped := false
    for each i in 0 to length(items) - 1 inclusive do: //line 1
      if items[i] > items[i+1] then
        swap(items[i], items[i+1])              //line 2
        swapped := true
      end if
    end for
  while swapped
//post-condition:
end method
```

d) Use the above pseudo-code to quickly modify the code below by implementing the method sort for the vector of `StudentA`, which sorts the students in the array in an ascending order based on their CAP.

```
public class StudentA{
      private String name;
      private String matric;
      private double CAP;

      public StudentA(String name, String matric){
            this.name = name;
            this.matric = matric;
      }

      public void setCAP(double val) throws Exception{
            if (val <= 0 || val >5) throw new Exception("Illegal value!!");
            CAP = val;
      }

      public double getCAP(){return CAP;}

      public String toString(){
            return name +"\t(" + matric+"t:\t" + CAP;
      }
      …
}

public class Test{
      public void sort(StudentA[] students){
      //implement here!! How fast can you write this?
     }//end sort


      public static void main(String[] args){
            StudentA[] students = new StudentA[3];
            StudentA s1= new StudentA("John Lee", "HT0851227A");
```

```
		StudentA s2 = new StudentA("Jane Doe", "HT0556223Y");
		StudentA s3 = new StudentA("Yixiang Chen", "HT07542324L");
		try{
			s1.setCAP(3.56);
			s2.setCAP(4.30);
			s3.setCAP(4.75);
		}
		catch (Exception exp){
			exp.printStackTrace();
		}
		students[0] = s1; students[1] = s2; students[2] = s3;

		Test t = new Test();
		t.sort(students);
		for (StudentA student : students){
			System.out.println(student);
		}
	}
}
```

2. Suppose that you are required to code a ContactList ADT using a Java Array as the data structure to store the contacts. The ADT must provide public methods for the programmer to add, edit, and delete a contact. A contact contains a person name and a phone number. Duplicate contacts (same person name and same phone number) are not allowed. Explain the following in English (with the aid of diagrams if necessary).

    a) What assumptions would you make? (Hint: a person has more than one phone numbers.)

    b) How would you implement the add method based on your assumptions? Assume that the new contact information (person name and phone number) is passed to the method.

    c) What would you do when the array is full?

    d) How would you implement the edit method based on your assumptions? Assume that both old and new contact information (person name and phone number) is passed to the method.

    e) How would you implement the delete method based on your assumptions? Assume that the contact information (person name and phone number) is passed to the method.

3. Consider the Java class `ArrayList<E>`. Suppose we define a data type called

    ```
    ArrayList<ArrayList<Integer>>
    ```

    which is an ArrayList of ArrayLists of Integers.
    Implement a method

    ```
    int getMin( ArrayList<ArrayList<Integer>> aList )
    ```

    that returns the minimum of all the integers in the `aList`.

4. Consider the following ADT of a simple linked list.

```
class ListNode<E> {
    private E element;
    private ListNode<E> next;

    public ListNode (E item, ListNode <E> n) {
        element = item;
        next = n;
    }

    public E getElement() {
        return this.element;
    }

    public ListNode<E> getNext() {
        return this.next;
    }

    public void setNext(ListNode<E> n) {
        this.next = n;
    }
}

class SimpleLinkedList<E> {
    private ListNode<E> head = null;

    public ListNode<E> getHead() {
        return this.head;
    }

    public void addHead(E item) {
        head = new ListNode<E>(item, head);
    }

    public void insertAfter(E item, E newItem) {
        for(ListNode<E> curr = head;
            curr != null;
            curr = curr.getNext())
        {
            if (curr.getElement().equals(item)) {
                ListNode<E> newNode =
                    new ListNode<E>(newItem, curr.getNext());
                curr.setNext(newNode);
                System.out.println(newItem + " is inserted after " + item);
                return;
            }
        }
        System.out.println("Cannot find " + item);
    }

    public void delete(E item) {
        for(ListNode<E> prev = null, curr = head;
            curr != null;
            prev = curr, curr = curr.getNext())
        {
            if (curr.getElement().equals(item)) {
                if (prev == null) {
                    head = curr.getNext();
                }
                else {
```

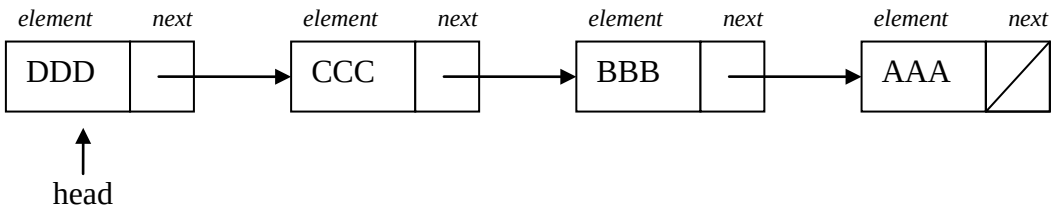
```
                    prev.setNext(curr.getNext());
                }
                System.out.println(item + " is deleted");
                return;
            }
        }
        System.out.println("Cannot find " + item);
    }

    public void print() {
        for(ListNode<E> curr = head;
            curr != null;
            curr = curr.getNext())
        {
            System.out.print(curr.getElement() + " -- ");
        }
        System.out.println();
    }
}
```

Using the above ADT, we create a simple linked list of Strings:

```
SimpleLinkedList<String> list = new SimpleLinkedList<String>();
list.addHead("AAA");
list.addHead("BBB");
list.addHead("CCC");
list.addHead("DDD");
list.print();
```

The output of list.print() is DDD -- CCC -- BBB -- AAA --
The following diagram shows the structure of our created list.



Draw diagrams for the following actions step by step and write the output.

```
a) list.insertAfter("AAA", "ZZZ"); list.print();
b) list.insertAfter("DDD", "YYY"); list.print();
c) list.delete("DDD"); list.print();
d) list.delete("BBB"); list.print();
```