DESIGN PATTERN ASSIGNMENTS

2ns year Class

Sub: OBJECT ORIENTED SYSTEMS

1. Consider the following code.

**public class DataStructure {**

**private final static int SIZE = 15;**

**private int[] arrayOfInts = new int[SIZE];**

**public DataStructure() {**

**for (int i = 0; i < SIZE; i++) {**

**arrayOfInts[i] = i;**

**}**

**}**

**public void printEven() {**

**DataStructureIterator iterator = this.new EvenIterator();**

**while (iterator.hasNext()) {**

**System.out.print(iterator.next() + " ");**

**}**

**}**

**private class EvenIterator extends DataStructureIterator {**

**public boolean hasNext() { // Write Code Here }**

**public Integer next() { // Write Code Here }**

**}**

**public static void main(String s[]) {**

**DataStructure ds = new DataStructure();**

**ds.printEven();**

**}**

**}**

1. Complete the definition of the DataStructure class.
2. Define a method named **print(DataStructureIterator iterator)**. Invoke this method with an instance of the class **EvenIterator** so that it performs the same function as the method **printEven**.
3. Invoke the method **print(DataStructureIterator iterator)** so that it prints elements that have an odd index value. Use an anonymous class as the method's argument instead of an instance of the interface **DataStructureIterator**.
4. A Parse Tree is a binary tree with nodes as (unary or binary) operators or numeric operands. An example of a parse tree is shown in the following.

\*

/ \

+ 7

/ \

5 2

We need to implement an algorithm to compute the value of the expression represented as a parse tree. Please note that the parse tree is already formed. The elements of the parse tree are subclasses of a class *TreeNode*. These subclasses include *AddNode*, *SubtractionNode*, *DivisionNode*, *MultiplicationNode* and *NumericNode*. We plan to define the algorithm by means of a visitor.

a) Draw the class diagram of the solution.

b) Draw the sequence diagram of the solution.

c) Define the classes in Java

1. Suppose that a client wants to create and manipulate a text box. The client can edit the text by many commands (such as bold, italics, cut, paste etc.) which he can undo. Consider the following “sample” main().

main()

{

CommandManager cmdManager = new CommandManager();

Texbox tb1 = new TextBox();

BoldCommand bC = new BoldCommad(tb1);

cmdManager.ExecuteCommand(bC);

ItalicsCommand iC = new ItalicsCommand(tb1);

cmdManager.ExecuteCommand(iC);

cmdManager.undo();

cmdManager.undo();

}

Use a *Command* pattern to solve this problem.

1. Draw a class Diagram for the solution of the said problem using *Command* Pattern
2. Draw a sequence diagram for the solution of the said problem using *Command* Pattern.
3. Implement the solution using Java.