Mini Project # 2

Date Assigned: 19/11/2008

Due Date: 04/12/2008

Before reading the statements please check your current level in the course from 07-CP-Level-191108.pdf (uploaded on the course pages). You have to do the project according to your level. Level - II student can do the Level - I project.

Level - I

Question 1:

In this problem you will be the using the wheels.users package to show polymorphic behavior of shapes. Program specification is as follow.

Develop a class ShapesCollection which extends wheels.users.Frame.

It has following member variables

- **Shape ShapesArray []**

  It is used to store different shapes passed by user. Can also use java.util.LinkedList <Shape>list for better dynamic management.

- **int count**

  It is used to store current count of Shapes. Not to be used if LinkedList is used.

It has following member functions:

A default constructor to initialize ShapesArray to any number of Shapes.

- **public void draw( )**

  To draw shapes using show( ) function.

- **public void hide( )**

  To hide shapes using hide( ) function.

- **public void addShape(Shape s)**

  This adds different subclasses of Shape in array.

- **public void removeShape(int index)**

  To remove shape at index.

Develop a new class PolymorphicShapes that creates a ShapesCollection object and fill it with different shapes and call its hide and draw function.

Question 2:

Quick Sorting

Quicksort is an algorithm of the divide and conquer type. The main purpose of this task is to implement the algorithm of quicksort using java. The basic information for the algorithm can be found in 'Seymour Lipschutz' book named 'Data Structures' or from any other place you want.

REQUIREMENTS:

1 *User Input Array*:
Numbers which are to be sorted must be inserted by the user at run time.

2 *Dynamic Array Length:*
The length of the array which is to be sorted must be defined by the user at run time

3 *Each Pass Display*
Each pass carried out in the process of sorting must be displayed showing step by step sorting of an array and elaborating the algorithm.

Level -II

Question 1:

Implement a superclass Person, then create two classes, Student and Instructor, that inherit from Person. A Person has a name and a year of birth. A Student has a major, and an Instructor has a salary. All instance variables must be declared private! Use the following specification to write the class definitions for all the classes. Make sure you include the methods toString for all classes.

The Person class is described as follows:

Two attributes:

- **Name**
  Stores String representing name of Person.
- **BirthYear**
  Stores Integer representing of birth year of person.

One constructor:

- **public Person(String name, int YearOfBirth)**
  Creates a person with the name and birth year passed.

Four methods:

- **getName( )**
  Returns the name of the Person
- **getYearOfBirth( )**
  Returns the year of birth of Person.
- **toString()**
  Returns a String containing the name followed by the constant String "born in the year" followed by the year the person was born in.
- **equals(Object obj)**
  Returns true if this object is equal both in Name and YearOfBirth

The Student class is described as follows

One attribute:
- **Major**

Stores the string representation of Student's major

One constructor:
- **public Student(String name, int YearOfBirth , String major)**

  Creates a Student with the name, major and birth year passed.

Two methods:
- **getMajor()**

  Returns the major of Student
- **toString()**

  Returns a String containing the name followed by the constant String "born in the year" followed by the year the person was born in, followed by the major.

The Instructor class is described as follows:

One attribute:
- **Salary**

  Stores the integer representation of Instructor's salary

One constructor:
- **public Insturctor(String name, int YearOfBirth , int salary)**

  Creates an Instructor with the name, salary and birth year passed.

Two methods:
- **getSalary( )**

  Returns the salary of Insturctor
- **toString( )**

  Returns a String containing the name followed by the constant String "born in the year" followed by the year the person was born in, followed by the salary.

Use this code to test your program

```
public class TestPeople
{
      public static void main(String args[])
      {
            Person p = new Person("Aslam Khan", 1847);
            System.out.println(p);
            System.out.println(p.getClass());
            System.out.println(p.getName() + " " + p.getYearOfBirth() +
                         "\n");

            Student s = new Student("Ali Raza" , 1931, "Cricket");
            System.out.println(s);
            System.out.println(s.getClass());
            System.out.println(s.getName() + " " + s.getYearOfBirth() +
                         " " + s.getMajor() + "\n");
            Instructor i = new Instructor("Junaid Tahir", 1955, 85000);
            System.out.println(i);
            System.out.println(i.getClass());
            System.out.println(i.getName() + " " + i.getYearOfBirth() +
                        " " +i.getSalary() + "\n");

            System.out.println("\n");


            p = s;
            System.out.println(p);
            System.out.println(p.getClass());
            ystem.out.println(p.getName() + " " +
                                  p.getYearOfBirth() + " " +
                                  ((Student)p).getMajor() + "\n");
            p = i;
            System.out.println(p);
            System.out.println(p.getClass());
            System.out.println(p.getName() + " " + p.getYearOfBirth() +
                                  " " +
                         ((Instructor)p).getSalary() + "\n");

            Instructor  is2  =  new  Instructor("Usman  Shahid",  1982,
      65000);
            if (p.equals(s))
               System.out.println(p.getName() + " and " + s.getName() +
                          " are the same.");
            else
                  System.out.println(p.getName()   +   "   and   "   +
                  s.getName() + " are different.");
            if (p.equals(is2))
                  System.out.println(p.getName()   +   "   and   "   +
                  is2.getName() + " are the same.");
            else
                  System.out.println(p.getName()   +   "   and   "   +
                  is2.getName() + " are different.");

      }
}
```

Question 2:

Infix → Prefix and Postfix

The basic purpose of this assignment is to develop a program that will get an arithmetic expression from the user in the **Infix** form and shows it in both **Prefix** and **Postfix** form, and also show each step while converting from infix to both postfix and prefix form. e.g.

Infix: **A + { B * C - (D / E ^ F) * G } * H**
Postfix: **A B C * D E F ^ / G * - H * +**
Prefix: + A * - * B C * / D ^ E F G H

REQUIREMENTS:

1 *Operators Supported*:
The operators that the program must support are " ^ / * + - "

2 *Bracket Support:*

The program must support three different types of brackets [{ ( ) }]

3 *User Input Expression*

The expression to be converted is to be entered by the user at the run time.

4 *Fault Tolerant*

The expression entered by the user must be checked if is actually an arithmetic expression or not. If not then indicated that it is not a valid expression otherwise show Postfix and Prefix conversion with complete steps.

Design Information and Requirements

- The code should be well indented and easy to read.
- Follow naming convention discussed in the session for naming classes, variables, function and constants.
- No project submissions are accepted after due date.
- Maximum two persons are allowed in one project group. But both must be from the same level(Level I/Level II) and same group(G1/G2)
- Avoid plagiarism, negative marking will be conducted in case of plagiarism.
- Also Email me the zip file of project before 12.00 Noon due date.

Project Grading Scheme

Item	**Marks**
Code Quality	30 %
Code Testing	30 %
Complete Execution	20 %
Question & Answers	20 %