CSC103 Spring 2016
Sara Wexler

Assignment #5

.

Binary Search Tree

Purpose: Write a program, which will utilize the Binary Search Tree
This assignment is to be done by a group of two people.

Database is a collection of related pieces of information on Employees that is organized for easy retrieval.
The data base file is shown below

Record No.	Id	First Name	Last name	Salary
0	**6274**	**John**	**Johnson**	**50,000.00**
1	**2843**	**Mark**	**Wilson**	**60,000.00**
2	**4892**	**Dana**	**Albright**	**70,000.00**
:				

this information is in the file Employees.txt

6274	**John**	**Johnson**	**315.56**
2843	**Mark**	**Wilson**	**7217.23**
4892	**Dana**	**Albright**	**31462.56**
8337	**Debra**	**Douglas**	**127.26**
1892	**Bruce**	**Gold**	**719.32**
9523	**John**	**Carlson**	**1496.24**
3165	**Mary**	**Smith**	**918.26**
3924	**Simon**	**Becker**	**485.85**
6023	**John**	**Edgar**	**9.65**
8529	**Ellen**	**Fairchild**	**86.77**
1144	**Donald**	**Williams**	**4114.26**

Write a client program which will do the following:

1. Create an Employee class

```
class Employee implements Cloanable, Comparable
 {
        private int acctID;
        private String firstName;
        private String LastName;
        private double salary;

        public Employee( id, l_name,f_name,amount){….}
        public Employee( id){….}
```

```
            public int getId(){…}
            public String getLastName(){…}

            public String getFirstName{…}

            public double getSalary{…}

            public void setSalary(double amount){….}

            public string tostring();
            // a method to print information about the employee

            public int compareTo (Object obj) { …..};
           // a method to compare if object that invokes this method is greater then, equal or less
           // then obj.
          // the method returns an integer that is less than, equal to, or greater then zero if the
         //executing object is less than, equal to, or greater than the parameter, respectively.
    }
```

2. Create a program called **Company**, your program will:

- Create a menu to the user with the following options:

  1. Read from a file the list of employees and store each employee in a Binary Search Tree, based on the employee id (unique number)
  2. Add an employee to the tree
  3. Remove an employee from the tree
  4. Retrieve an employee *
  5. Update an employee *
  6. Display the firm's employees in **ascending** order based on the employee id (think of the appropriate traversal)
  7. Quit

  ***After the call to the retrieve and update operations you should display the record**

- Create an output file called **outEmployees.txt,** which have all the nodes that are existed in the tree.
  i.e. a deleted node should not be in the output file, and an added node should be in it.

3. Use the **BTNode<E>** class, and the class **TreeBag** (TreeBag is a version of the IntTreeBag, where the data it hold is an object)
   You need to implement only the **add, remove, retrieve,** and **display** methods.
   You are not required to implement the rest of the methods in the class.
   Incorporate the changes to the following methods in the **TreeBag**

- **public boolean remove (Comparable target)**

The **remove** method returns boolean therefore you need first to call retrieve to find if the object is in the tree, if not return false, if yes, then delete the node from the tree

- **public void add (Comparable element)**

- **public Object retrieve (Comparable element).**
  The **retrieve** method will search the Binary Search tree for the element, the method will return the object that was found. If the object is not found the method returns null.

- **public void display()**
  Display the firm's employees in **ascending** order based on the employee id (think of the appropriate traversal)

You may need other methods to complete the project

Provide clear documentation describing what your program is doing, the kind of objects you are storing in the tree, and what is the key for comparison between the objects.

Items to be turned in:

- Well documented source code
- Test cases of your program
- Disk with all source code files and an executable

<u>Grading Criteria</u>

design - use of existing classes, design of new classes, decomposition of client program
correctness - correct results are given for any valid input
robustness - detection and handling of errors (including user entry of information)
user interface - format of prompts, messages and output
documentation - behavior of each method that you add to the project clearly described, and description of what the program is doing,
testing - thorough and thoughtful testing of your program

<u>Due Date</u>: Last Lab (week 14- Monday May 9)

<u>Note</u>: Late lab <u>will not be accepted</u>!
Part of the grade is to demonstrate that the program is working.

Run your program with the following test - the test show output to the screen. You can change it to output to a file.
Beware that you are required to output also the records after each operation

```
class TreeTest
{
  public static void main (String[] args)
   {
      Company company = new BankEmployee();
      System.out.println ("create the tree from an input file Employee.txt");
      System.out.println ("----------------");
      company.menu("1 – Employee.txt”);
      System.out.println ();
      System.out.println ("------------------------------------------------------------------------");
      System.out.println ("add an employee to the tree");
      System.out.println ("--------------------------------");
      company.menu("2 - 5290 George  Truman          16110.68");
      System.out.println ();
      System.out.println ("------------------------------------------------------------------------");
      System.out.println ("remove an employee from the tree");
      System.out.println ("-------------------------------------");
      company.menu("3 - 4892);
      System.out.println ();
      System.out.println ("------------------------------------------------------------------------");
      System.out.println ();
      System.out.println ("retrieve an employee from the tree and print the employee record");
      System.out.println ("---------------------------------------------------------------------------");
      company.menu("4 –3924”);
      System.out.println ();
      System.out.println ("------------------------------------------------------------------------");
      System.out.println ();
      System.out.println ("update an employee from the tree and print the new ");
      System.out.println ("------------------------------------------------------------");
      company.menu("5 – 3924 20000.00);
      System.out.println ();
      System.out.println ("------------------------------------------------------------------------");
      System.out.println ();
      System.out.println ("display the tree ");
      System.out.println ("------------------");
      company.menu("6” );
      System.out.println ("------------------------------------------------------------------------");
      System.out.println ();
      System.out.println ();

   }
}
```