Imperial College of Science, Technology and Medicine	University of London
Computer Science (CS) / Software Engineering (SE)	BEng and MEng Examinations Part I
Department of Computing	Integrated Laboratory Course

Laboratory work is a continuously assessed part of the examinations
and is a required part of the degree assessment.
Laboratory work must be handed in for marking by the due date.
Late submissions may not be marked.

Exercise: 17	Working: Individual
Title: Java Database	
Issue date: 8th March 2004	Due date: 15th March 2004
System: Linux	Language: Java

Aims

- To gain further experience with **abstract data types** (ADTs) in Java.
- To implement a tree structure with **references** ("pointers").

The Problem

- Your task is to implement a database to store data about cats living in a cattery. Each cat has a *name* - a string - and a *number of tins* of cat food the cat eats each week - a non-negative integer.
- You should write an implementation class to implement the provided interface class **DatabaseInterface.java**. This class should be called **Database.java** which is an implementation of a binary search tree. This implementation uses a class **DatabaseNode.java** which implements the underlying data structure. These two classes, along with the provided interface class **DatabaseInterface.java** should all be in the package **database**.
- The class **DatabaseNode.java** should contain the attributes for a cat staying in a cattery and its dietary requirements. The class also provides access methods for a database of cats.

The interface class and its implentation

- The database should be implemented as an **ADT** which consists of a data structure together with access methods for the data structure. The implementaion should use two

classes, **Database.java** which implements **DatabaseInterface.java** and **DatabaseNode.java** which provides the underlying data structure and its own access methods. You are provided with the interface class file **DatabaseInterface.java** and a test harness main program file **CatAdmin.java** which uses the access methods offered by the interface.

- The access methods are as follows. You should note that these access methods contain no information about the data structure used to implement the database as the structure is a *private* (or encapsulated) field of the **Database** class. You should however implement the database as a reference based binary search tree.

```
public void add (String name, int tins);
/*
 * If the named cat is not present in the database, adds a fresh entry,
 * in the correct place, with the given name and number of tins.
 * Otherwise, that existing cat has its tins changed to the given value.
 */

public int lookup (String name);
/*
 * If the cat is in the database, returns that cat's weekly quota of tins.
 * Otherwise returns -1.
 */

public int countCats();
/*
 * Returns the total number of cats in the database.
 */

public int countTins();
/*
 * Returns the total number of tins eaten each week by all the cats.
 */

public void delete (String name);
/*
 * If the named cat is present in the database, deletes it.
 * Otherwise leaves the database unchanged.
 */

public void printDatabase();
/*
 * Displays contents of the database as <Name> [tab] <Tins>
 * (e.g., Tibbles 14), in alphabetical order of cats, one to a line.
 */
```

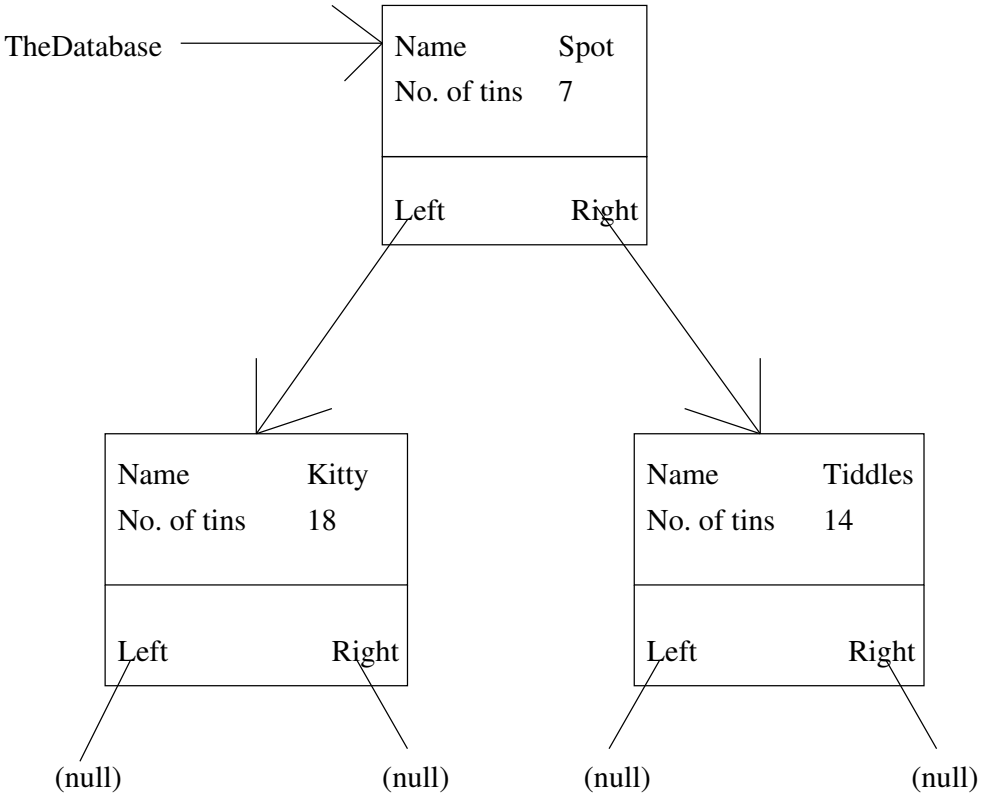
Submit by Monday 15th March 2004

What To Do

- Copy the test harness main program file **CatAdmin.java** and the interface class file **DatabaseInterface.java** with the command **exercise 17**.
- **CatAdmin.java** provides a simple menu-driven facility for performing operations on the database. **DatabaseInterface.java** advertises access methods to perform those operations. Create a sub-directory for the package **database** and move the file **DatabaseInterface.java** into the sub-directory. Before writing any code, you should **study these given files**.
- Write **Database.java** and **DatabaseNode.java** in the **database** sub-directory to implement the interface **DatabaseInterface.java** and provide the functionality required for **CatAdmin.java**.
- The database should be implemented as a tree structure using references (sometimes called "pointers") - see below. Note that the database itself never appears as a parameter to any of the access methods; rather, it should be an **encapsulated variable**: that is, a private variable, global *within* your implementation class file **Database.java** but invisible outside it.
- **Test your code thoroughly** by running the test harness and trying out a wide variety of user command sequences. We recommend that you write the access methods in the order **add**, **printDatabase**, **lookup**, **countCats**, **countTins**, **delete**. If you want to test your code before you have written all of these you will need to write "stubs" for the ones you have not yet written.

The database structure

- The database should be implemented as a **tree structure** using a recursive class, **DatabaseNode**, containing **references** to any further nodes of the tree. If null, a database node represents an empty database (or empty part of one). If non-null, it should store a cat's name and weekly number of tins, and references to two further database nodes in recursive style, representing the left and right **sub-trees** dangling off that node. (These sub-trees may in turn be null or non-null as appropriate.)



- It is *physically* possible for such a tree structure to contain two or more entries for the same named cat, or to have the cats jumbled up out of alphabetical order, but **your access methods should be written to keep the database ordered and free of duplicates**.

  A database is *ordered and free of duplicates* if either it is empty, or else all the cat names stored in its left sub-tree are *less* than the cat name stored in the node itself and all the cat names stored in its right sub-tree are *greater* than the cat name stored in the node itself, **and** each sub-tree, regarded as a smaller database in its own right, is ordered and free of duplicates in the same sense.

- Your implementation code **should all be stored in the files Database.java**. (implementing the interface using the database node class) and **DatabaseNode.java** (providing the database node class) respectively.

- You should initialize your database encapsulated variable to **null** to represent an initially empty database.

Submission

- To submit **Database.java** and **DatabaseNode.java** first *cd* into the **database** subdirectory and then type **submit 17**.

Assessment

add	1.0
lookup	0.5
countCats	0.5
countTins	0.5
delete	1.0
printDatabase	0.5
Database node class	1.0
Design, style, readability	5.0
Total	10.0