CS 2210a — Data Structures and Algorithms
Assignment 4: Simple Database Manager

Due Date: November 18, 11:59 pm
Total marks: 20

1 Overview

For this assignment you are to write a simple database manager that will store a set of records in an ordered dictionary implemented using a binary search tree (you do not need to implement an AVL tree). Each record is of the form `(key,data)`, where `data` is a string and `key` is a pair `(word,type)`, where `word` is a string of one or more letters and `type` is an integer value as follows:

- `type = 1`, if the `data` in a record is just a text string
- `type = 2`, if the `data` in a record is the name of an audio file. The only audio files that we will consider are of type ".wav" or ".mid".
- `type = 3`, if the `data` in a record is the name of an image file. The only image files that we will consider are of type ".gif" or ".jpg".

When comparing two keys k_1 =`(string1,type1)` and k_2 =`(string2,type2)`, we use the following rules:

- $k_1 = k_2$ if `string1 = string2` and `type1 = type2`.
- $k_1 < k_2$ if `string1` lexicographically precedes `string2` or if `string1 = string2` and `type 1 < type2`.

Consider, for example, a database storing the following records:

- r_1 =((“computer”,1), “An electronic machine frequently used by Computer Science students.”)
- r_2 =((“computer”,3), “computer.gif”)
- r_3 =((“flower”,3), “flower.jpg”)
- r_4 =((“ping”,2), “ping.wav”)

Note that the key (“computer”,1) of r_1 is smaller than the key (“computer”,3) of r_2, and the key of r_2 is smaller than the key (“flower”,3) of r_3.

Your program will provide a simple text-based interface that will allow users to interact with the database through the use of the following commands.

- `search word`

  For each record `((word,type),data)` with the specified `word` as part of the key, your program will do the following:

  - print `data` on the screen if `type = 1`
  - play the audio file named in `data`, if `type = 2`
  - display the image stored in the file named in `data`, if `type = 3`

  If no record in the database has the specified `word` in its key attribute an appropriate error message must be displayed. So, for example, if in the above database the user enters the command `search ping` your program must play the file “ping.wav”. If the user enters the command `search computer`, first your program must print the text “An electronic machine

frequently used by Computer Science students." and then it must display the image in file "computer.gif". If the user enters the command `search homework` your program should print a message indicating that there is no record in the database with "homework" in its key attribute.

- `remove word type`

  Removes from the database the record with key `(word,type)` or it prints an appropriate message is no such record exists.

- `insert word type data`

  Inserts the record `((word,type),data)` into the database if there is no record with key `(word,type)` already there; otherwise an appropriate error message is printed.

- `next word type`

  Prints the key attribute `(w,t)` of the record in the database that follows the record with key `(word,type)`; if such a record does not exist because there is no record in the database with key larger than `(word,type)` then an appropriate message is printed.

  For example, for the above database if the user types the command `next computer 3` your program must print `(flower,3)`. For the command `next ping 2` the program should print a message indicating that `(ping,2)` is the largest key in the database.

- `prev word type`

  Prints the key attribute `(w,t)` of the record in the database that precedes the record with key `(word,type)`; if such a record does not exist because there is no record in the database with key smaller than `(word,type)` then an appropriate message is printed.

  For example, for the above database if the user types the command `prev computer 3` your program must print `(computer,1)`. For the command `prev computer 1` the program should print a message indicating that `(computer,1)` is the smallest key in the database.

- `first`

  Prints the key `(w,t)` of the record with smallest key in the database. For the above database the command `first` must print `(computer,1)`.

- `last`

  Prints the key `(w,t)` of the record with largest key in the database. For the above database the command `last` must print `(ping,2)`.

- `end`

  This command terminates the program.

2 Classes to Implement

You are to implement these Java classes: `Key`, `Record`, `OrderedDictionary`, `DictionaryException`, and `UI`. You can implement more classes if you need to. **You must write all the code yourself**. You **cannot** use code from the textbook, the Internet, or any other sources: however, you may implement the algorithms discussed in class.

2.1 Key

This class represents the key attribute of a record in the database. Each key has two parts: a word and a a type. A word is a string of one or more letters. Letters in a word must be converted to lower case. The type of a key can be 1, 2, or 3 as specified above. For this class you must implement all and only the following `public` methods:

- `public Key(String word, int type)`: A constructor which returns a new `Key` with the specified word and type.
- `public String getWord()`: Returns the word in the `Key`.
- `public int getType()`: Returns the type in the `Key`.
- `public int compareTo(Key k)`: Returns 0 if `this` key is equal to `k`, returns -1 if `this` key is smaller than `k`, and it returns 1 otherwise.

You can implement any other methods that you want to in this class, but they must be declared as `private` methods (i.e. not accessible to other classes).

2.2 Record

This class represents a record in the database. Each record consists of two parts: a key and the data associated with the key. For this class, you must implement all and only the following `public` methods:

- `public Record(Key k, String data)`: A constructor which returns a new `Record` with the specified key and data. If the type in the key is 2 or 3, then `data` stores the name of the corresponding audio or image file.
- `public Key getKey()`: Returns the key in the `Record`.
- `public String getData()`: Returns the data in the `Record`.

You can implement any other methods that you want to in this class, but they must be declared as `private` methods.

2.3 OrderedDictionary

This class implements the database through an ordered dictionary. You must implement the ordered dictionary using a binary search tree. You must use a `Record` object to store the data contained in each node of the tree. In your binary search tree **only the internal nodes will store information**. The leaves will be java objects storing null `Record`objects. The **key** for an internal node will be the Key object from the `Record` stored in that node.

Hint. You might want to implement a class `Node` to represent the nodes of the binary search tree. Each node will store a Record, and references to its left child, right child, and parent. When comparing two node keys you need to use method `compareTo` from class Key.

In the OrderedDictionary class you must implement all the public methods specified in the `OrderedDictionaryADT` interface, shown below, and the constructor

```
public OrderedDictionary()
```

You can download `OrderedDictionaryADT.java` from the course's website.

```
public interface OrderedDictionaryADT {
   /* Returns the Record object with key k, or it returns null if such a record is
      not in the dictionary.  */
   public Record find (Key k);

   /* Inserts r into the ordered dictionary.  It throws a DictionaryException if a
      record with the same key as r is already in the dictionary.  */
   public void insert (Record r) throws DictionaryException;

   /* Removes the record with Key k from the dictionary.  It throws a
      DictionaryException if the record is not in the dictionary.  */
   public void remove (Key k) throws DictionaryException;

   /* Returns the successor of k (the record from the ordered dictionary with
      smallest key larger than k); it returns null if the given key has no
      successor.  The given key DOES NOT need to be in the dictionary.  */
   public Record successor (Key k);

   /* Returns the predecessor of k (the record from the ordered dictionary with
      largest key smaller than k; it returns null if the given key has no
      predecessor.  The given key DOES NOT need to be in the dictionary.  */
   public Record predecessor (Key k);

   /* Returns the record with smallest key in the ordered dictionary.  Returns null
      if the dictionary is empty.  */
   public Record smallest ();

   /* Returns the record with largest key in the ordered dictionary.  Returns null
      if the dictionary is empty.  */
   public Record largest ();
}
```

To implement this interface, you need to declare your `OrderedDictinary` class as follows:

```
public class OrderedDictionary implements OrderedDictionaryADT {
          ⋮
}
```

You can implement any other methods that you want to in this class, but they must be declared as `private` methods (i.e. not accessible to other classes).

2.4 DictionaryException

This is the class implementing the class of exceptions thrown by the `insert` and `remove` methods of `OrderedDictionary`. See the class notes on exceptions.

2.5 UI

This class implements the user interface and it contains the `main` method, declared with the usual method header:

```
public static void main(String[] args)
```

You can implement any other methods that you want to in this class, but they must be declared as `private` methods (i.e. not accessible to other classes). The input to the program will be a file

containing the records that are to be stored in the ordered dictionary. Therefore, to run the program you will type this command:

`java UI` *inputFile*

where *inputFile* is the name of the file containing the input for the program. The format for this file is as follows. The first line contains a word and the data associated with it appears in the second line. The third line contains another word with its associated data in the fourth line, and so on. For example an input file might be the following one:

```
course
A series of talks or lessons, for example, CS2210.
computer
An electronic machine frequently used by Computer Science students.
homework
Very enjoyable work that students need to complete outside the classroom.
roar
roar.wav
flower
flower.jpg
```

In this example, the first 3 words have type 1, while `roar` has type 2 and `flower` has type 3. The type needs to be inferred from the data. If the data only contains one string of the form "x.y", where y = ".wav" or y = ".mid" then the type is 2, if y = ".jpg" or y = ".gif" then the type is 3; otherwise the type is 1. The `Record` object for the word `"course"` will store `(("course",1),"A series of talks or lessons, for example, CS2210.")`. The `Record` object for the word `"roar"` will store `(("roar",2),"roar.wav")`; and the `Record` for `"flower"` will store `("flower",3),"flower.jpg")`.

Important Notes

- If the main method is as above, then the name of the input file will be stored in `args[0]`.
- All words in the key attributes (not in the data associated with the keys) will be converted to lower case before being stored in the dictionary, so that capitalization does not matter when looking for a key in the dictionary.
- To make it easier for the TA's to test your program, you are required to use method

  `read (String label)`

  from the provided `StringReader` class to read user commands entered from the keyboard. Class `StringReader.java` can be downloaded from the course's website. The above method prints on the screen the label supplied as parameter and then it reads one line of input from the keyboard; the line read is returned as a String to the invoking method. So, for example, to query the user for input you might use this in your program:

  ```
  StringReader keyboard = new StringReader();
  String line = keyboard.read("Enter next command:  ");
  ```

- To play a sound file you must use the provided Java class `SoundPlayer.java`, which can be downloaded from the course's website; you will use method `play(String fileName)` to play the named audio file. To display an image, you must use the `PictureViewer.java` class, which

can be downloaded from the course's website; you will use method `show(String fileName)` to display a `.jpg` or `.gif` file.

A `MultimediaException` will be thrown by methods `play` and `show` if the named files cannot be found or if they cannot be processed. Your program must catch these exceptions and print appropriate messages.

- Your program must print an appropriate message if an invalid command is entered.

Hint. You might find the StringTokenizer Java class useful.

3 Testing your Program

We will run a test program called `TestDict` to make sure that your implementation of `OrderedDictionary` has the properties specified above. We will supply you with a copy of `TestDict` to test your implementation. We will also run other tests on your software to check whether it works properly.

4 Coding Style

Your mark will be based partly on your coding style. Among the things that we will check, are

- Variable and method names should be chosen to reflect their purpose in the program.
- Comments, indenting, and white spaces should be used to improve readability.
- No variable declarations should appear outside methods ("instance variables") unless they contain data which is to be maintained in the object from call to call. In other words, variables which are needed only inside methods, whose values do not have to be remembered until the next method call, should be declared inside those methods.
- All variables declared outside methods ("instance variables") should be declared `private` (not `protected`), to maximize information hiding. Any access to the variables should be done with accessor methods (like `getWord` and `getType` for class `Key`).

5 Marking

Your mark will be computed as follows.

- Program compiles, produces meaningful output: 2 marks.
- `TestDict` tests pass: 5 marks.
- `UI` tests pass: 3 marks
- Coding style: 2 marks.
- Ordered Dictionary implementation: 5 marks.
- `UI` program implementation: 3 marks.

6 Submitting Your Program

You are required to submit an electronic copy of your program through OWL. Please do not put your code in sub-directories as this makes it harder for the TAs to mark the assignments. Also, please do not compress your files.

If you submit your program more than once, please send me an email message to let me know. We will take the latest program submitted as the final version, and we will deduct marks accordingly if it is late.