DICTIONARY SYSTEM

Students' ID, Name & Surname: 1503913 İpek YASİN

1729304 Oğuzhan KALKUZ

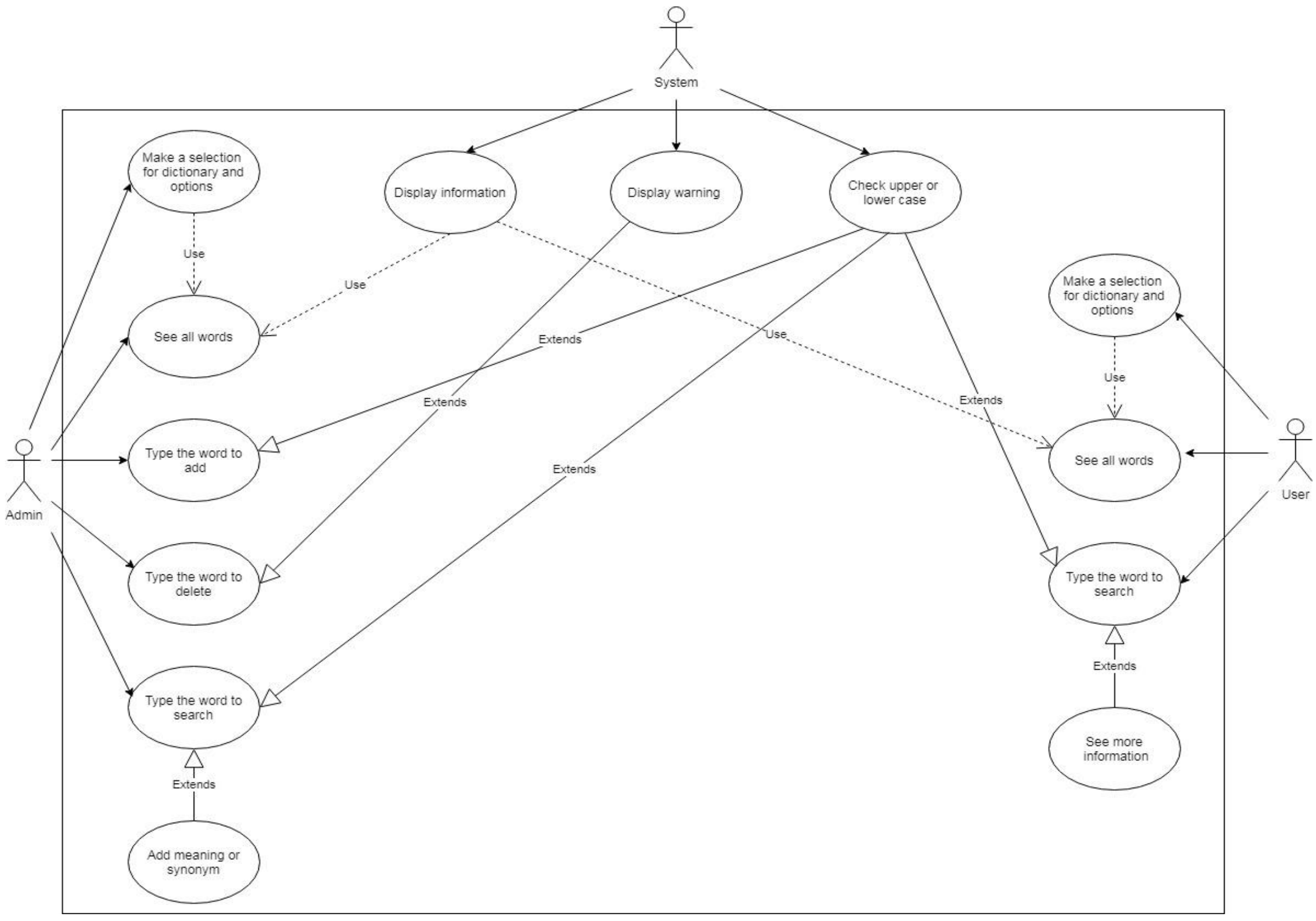
1722326 Yousef ELBAYOUMI

1. Aim of the Project

The principal goal of this project is to create a dictionary system which is used by manager and users through implementing Linked List and AVL Tree data structures. This system is going to be used for adding words as in English or Turkish, deleting them if it is needed, and searching for their synonyms and meaning.

There are three primary actors: Manager, Users, and Dictionary System. All users can log in to the system without any restrictions. The manager might add new words according to their language. Also, the manager can check out existing word information, and delete present words if they have a mistake. In addition, users may search and see any words as in English or Turkish whatever they want. Lastly, the dictionary system has a controlling part that prevents users from entering invalid input. Also, it displays words according to the request of the user or manager and it gives some warning if there is no word in which users are searching for it. Besides, the system checks lower and upper case to prevent any typos.

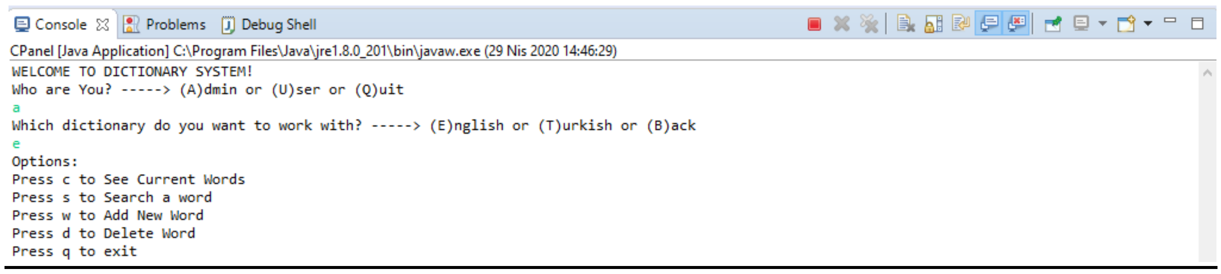
2. Use-Case Diagram of the Project



2.1.Outputs of the Program

2.1.1 English Dictionary for the Admin Part

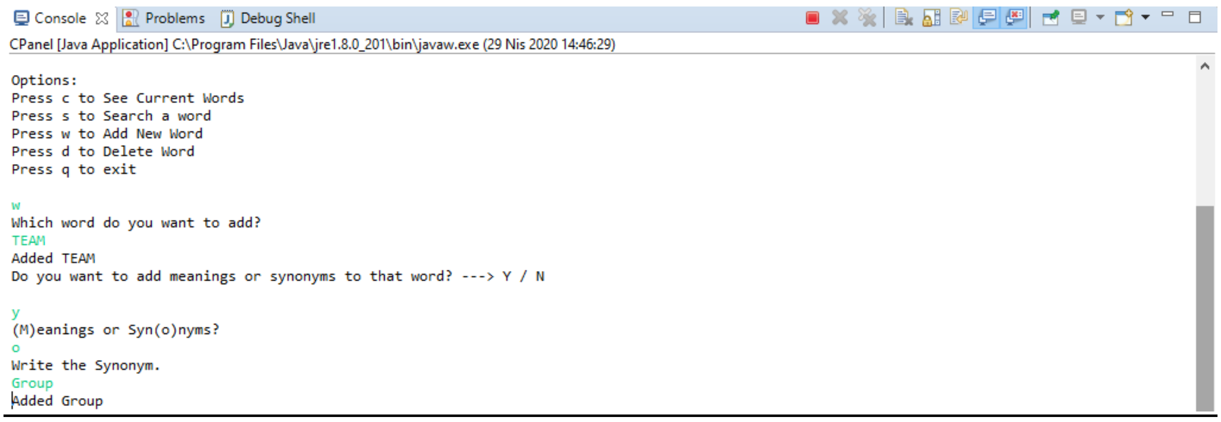
MENU



```
CPanel [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (29 Nis 2020 14:46:29)
WELCOME TO DICTIONARY SYSTEM!
Who are You? -----> (A)dmin or (U)ser or (Q)uit
a
Which dictionary do you want to work with? -----> (E)nglish or (T)urkish or (B)ack
e
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit
```

When the project is run, there are two options to select, one is for the admin and the other is for the user. After users choose their position, there is one question for dictionary language selection and following this, the menu is visible.

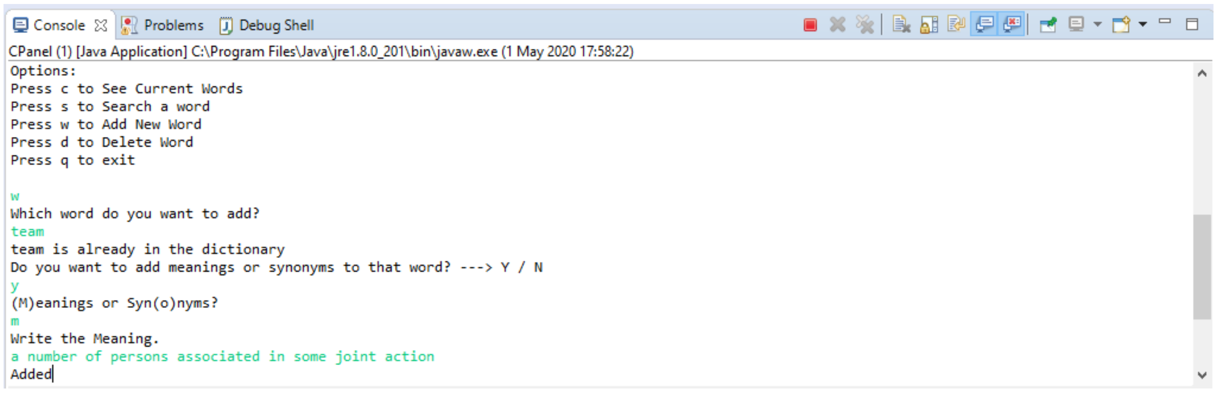
ADD NEW WORDS



```
CPanel [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (29 Nis 2020 14:46:29)

Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

w
Which word do you want to add?
TEAM
Added TEAM
Do you want to add meanings or synonyms to that word? ---> Y / N

y
(M)eanings or Syn(o)nyms?
o
Write the Synonym.
Group
Added Group
```

To add new words, admin has to press 'w'. After adding word, admin can add its synonym or meaning optionally.

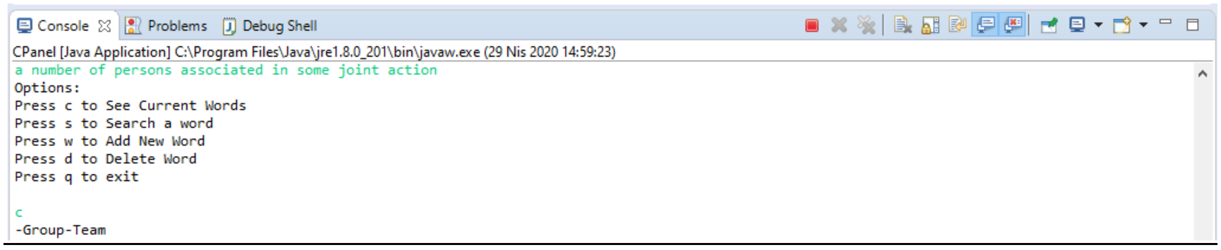
-->The system checks its case when the words are written as upper case.



```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 17:58:22)
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

w
Which word do you want to add?
team
team is already in the dictionary
Do you want to add meanings or synonyms to that word? ---> Y / N
y
(M)eanings or Syn(o)nyms?
m
Write the Meaning.
a number of persons associated in some joint action
Added
```

When the same word is tried to add, the system warns the admin as 'it is already in the dictionary.' and admin can add more information for it.

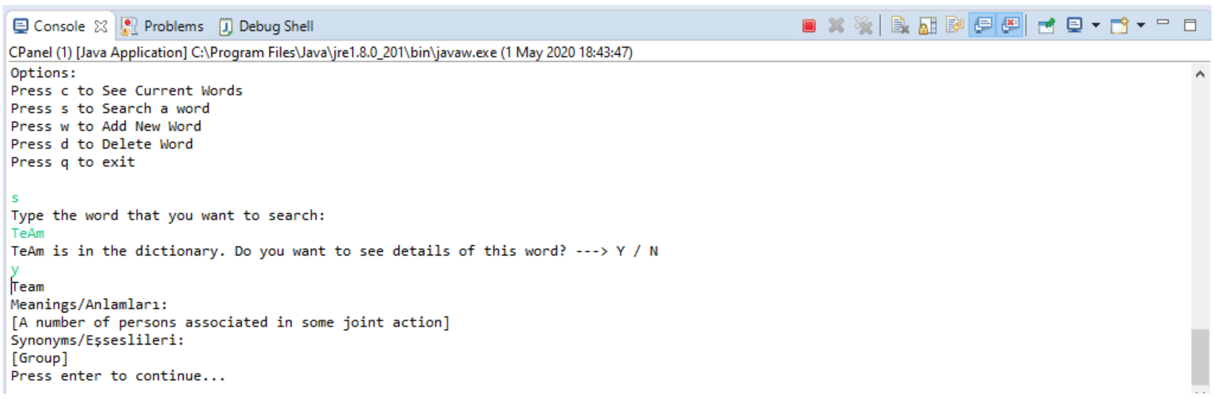
DISPLAY WORDS



```
CPanel [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (29 Nis 2020 14:59:23)
a number of persons associated in some joint action
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

c
-Group-Team
```

To view current words, admin has to press 'c'.

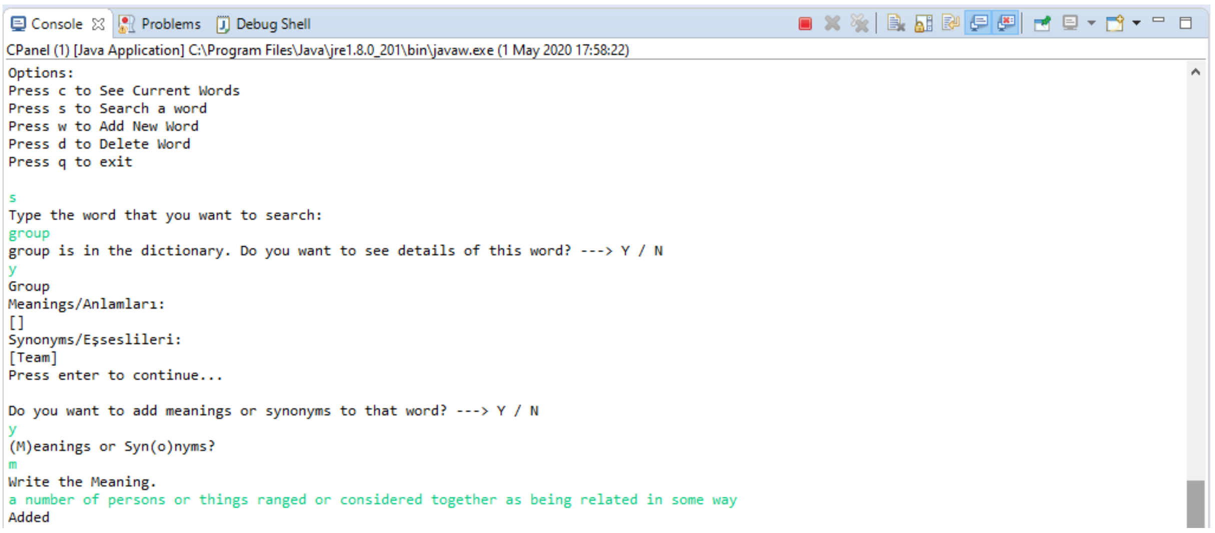
SEARCH WORDS



```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 18:43:47)
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

s
Type the word that you want to search:
TeAm
TeAm is in the dictionary. Do you want to see details of this word? ---> Y / N
y
Team
Meanings/Anlamları:
[A number of persons associated in some joint action]
Synonyms/Eşseslileri:
[Group]
Press enter to continue...
```

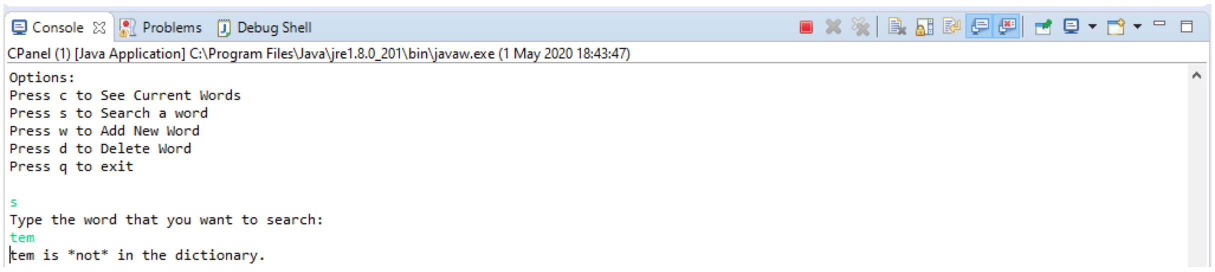
To search current words, admin has to press‘s’. After searching word, the admin can view its details optionally.

-->The system checks its case when the words are written as upper case.



```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 17:58:22)
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

s
Type the word that you want to search:
group
group is in the dictionary. Do you want to see details of this word? ---> Y / N
y
Group
Meanings/Anlamları:
[]
Synonyms/Eşseslileri:
[Team]
Press enter to continue...

Do you want to add meanings or synonyms to that word? ---> Y / N
y
(M)eanings or Syn(o)nyms?
m
Write the Meaning.
a number of persons or things ranged or considered together as being related in some way
Added
```

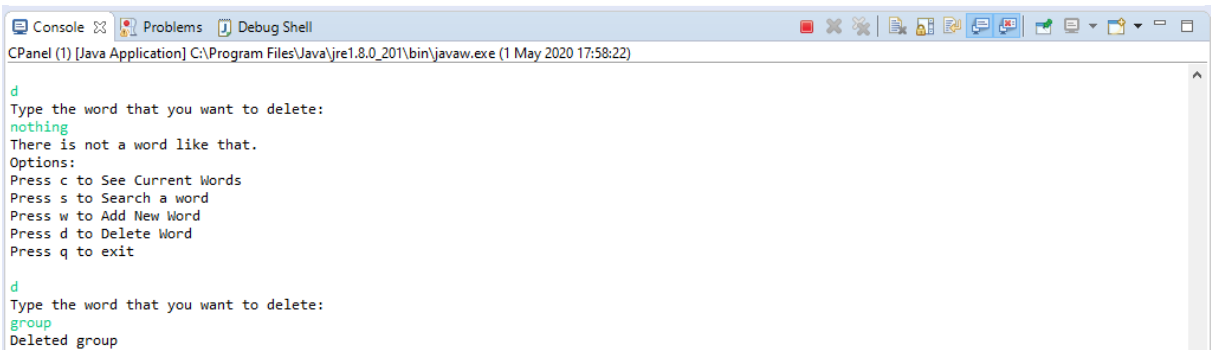
When searched word synonymously, all information is displayed. Following this, the admin can add its synonym or meaning optionally.



```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 18:43:47)
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

s
Type the word that you want to search:
tem
tem is *not* in the dictionary.
```

When a word not found is searched, the system warns the admin as ‘it is not in the dictionary.’

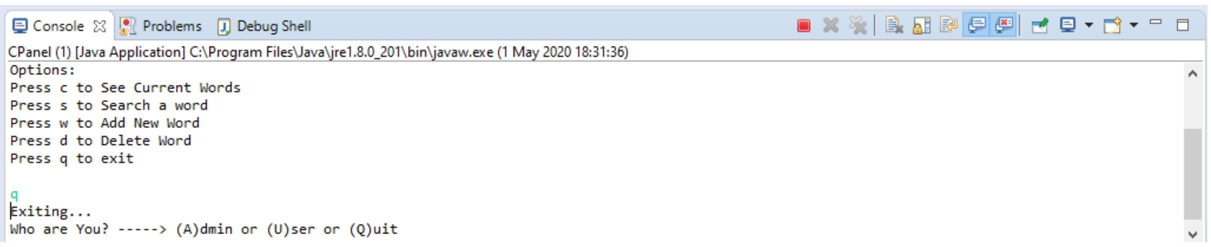
DELETE WORDS



```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 17:58:22)

d
Type the word that you want to delete:
nothing
There is not a word like that.
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

d
Type the word that you want to delete:
group
Deleted group
```

To delete words, admin has to press‘d’. When a word not found is wanted to delete the system warns the admin as ‘There is not a word like that.’

EXITING



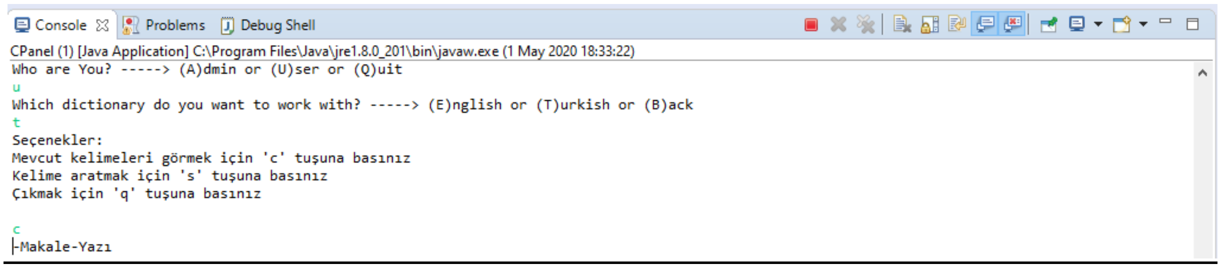
```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 18:31:36)
Options:
Press c to See Current Words
Press s to Search a word
Press w to Add New Word
Press d to Delete Word
Press q to exit

q
Exiting...
Who are You? -----> (A)dmin or (U)ser or (Q)uit
```

To exit, ‘q’ should be pressed.

✓ *There is another algorithm that is the same as the Turkish Dictionary for the admin part.*

2.1.2 Turkish Dictionary for the User Part

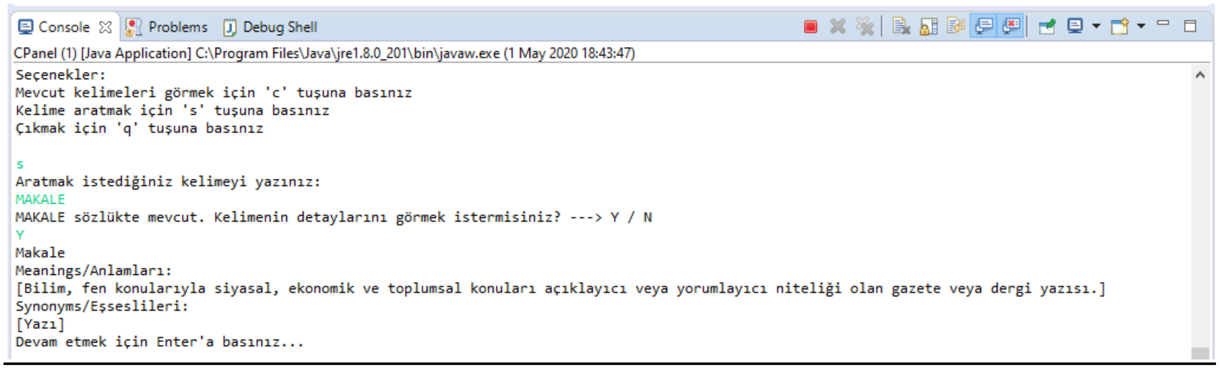
MENU & DISPLAY WORDS



```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 18:33:22)
Who are You? -----> (A)dmin or (U)ser or (Q)uit
u
Which dictionary do you want to work with? -----> (E)nglish or (T)urkish or (B)ack
t
Seçenekler:
Mevcut kelimeleri görmek için 'c' tuşuna basınız
Kelime aratmak için 's' tuşuna basınız
Çıkmak için 'q' tuşuna basınız

c
|-Makale-Yazı
```

It is the same algorithm with the admin part but now its language is in Turkish. When the project is run, there are two options to select, one is for the admin and the other is for the user. After users choose their position, there is one question for dictionary language selection and following this, the menu is visible and to view current words, user has to press ‘c’.

SEARCH WORDS

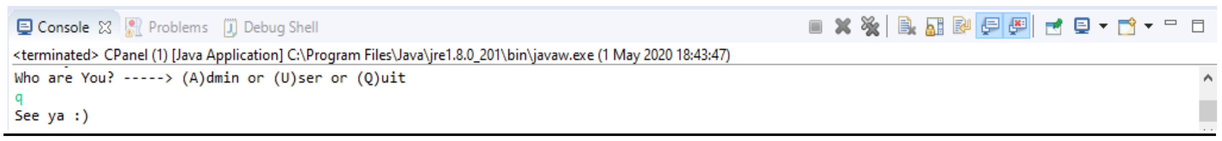


```
CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 18:43:47)
Seçenekler:
Mevcut kelimeleri görmek için 'c' tuşuna basınız
Kelime aratmak için 's' tuşuna basınız
Çıkmak için 'q' tuşuna basınız

s
Aratmak istediğiniz kelimeyi yazınız:
MAKALE
MAKALE sözlükte mevcut. Kelimenin detaylarını görmek istermisiniz? ---> Y / N
Y
Makale
Meanings/Anlamları:
[Bilim, fen konularıyla siyasal, ekonomik ve toplumsal konuları açıklayıcı veya yorumlayıcı niteliği olan gazete veya dergi yazısı.]
Synonyms/Eşseslileri:
[Yazı]
Devam etmek için Enter'a basınız...
```

To search current words, user has to press‘s’. After searching word, the user can view its details optionally. When a word not found is searched, the system warns the user.

-->The system checks its case when the words are written as upper case.

QUIT THE APPLICATION



```
<terminated> CPanel (1) [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (1 May 2020 18:43:47)
Who are You? -----> (A)dmin or (U)ser or (Q)uit
q
See ya :)
```

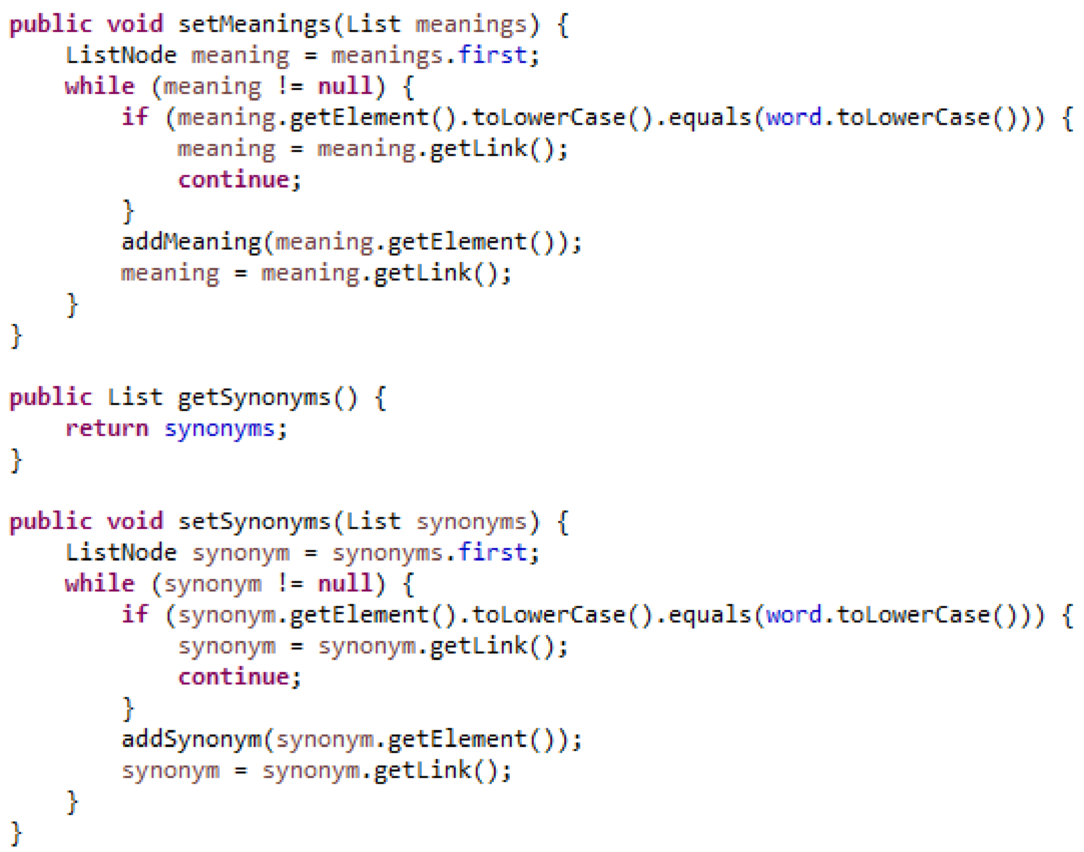
To quit the application, 'q' should be pressed when position selection is visible.

✓ *There is another algorithm that is the same as the English Dictionary for the user part.*

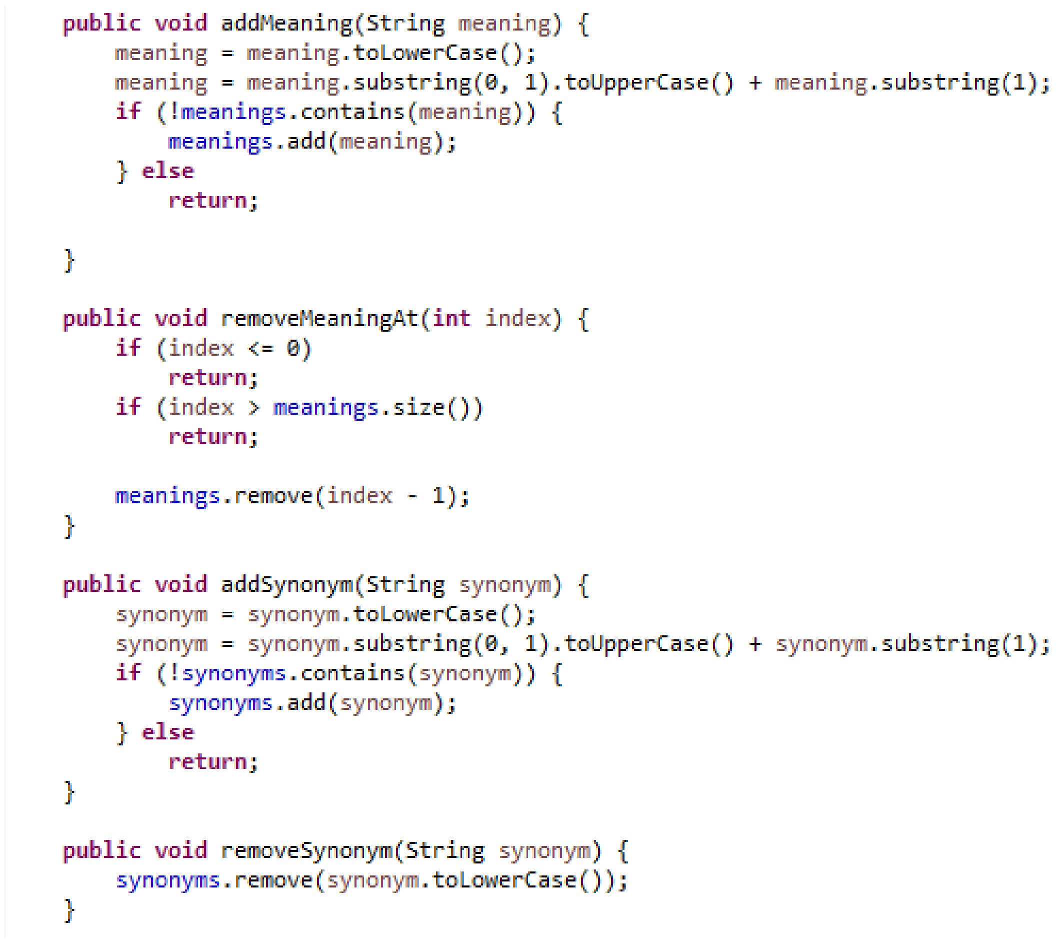
2.1.3 Codes

AVLNode.java

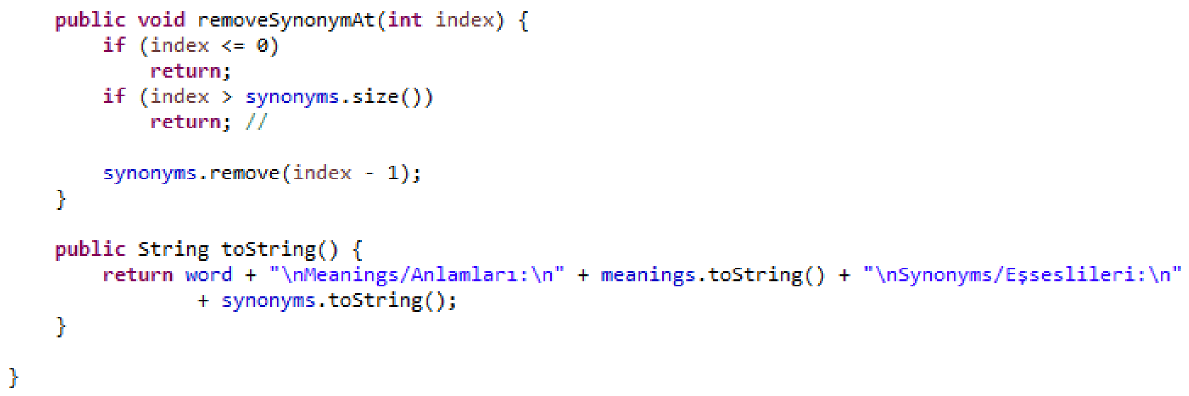
```
package Project;

public class AVLNode {

    private String word;
    private List meanings;
    private List synonyms;
    int height;

    AVLNode leftChild, rightChild;

    public AVLNode(String w) {
        setWord(w);
        height = 1;
        meanings = new List();
        synonyms = new List();
    }

    public String getWord() {
        return word;
    }

    public void setWord(String word) {
        word = word.toLowerCase();
        this.word = word.substring(0, 1).toUpperCase() + word.substring(1);
    }

    public List getMeanings() {
        return meanings;
    }
```



```java
public void setMeanings(List meanings) {
    ListNode meaning = meanings.first;
    while (meaning != null) {
        if (meaning.getElement().toLowerCase().equals(word.toLowerCase())) {
            meaning = meaning.getLink();
            continue;
        }
        addMeaning(meaning.getElement());
        meaning = meaning.getLink();
    }
}

public List getSynonyms() {
    return synonyms;
}

public void setSynonyms(List synonyms) {
    ListNode synonym = synonyms.first;
    while (synonym != null) {
        if (synonym.getElement().toLowerCase().equals(word.toLowerCase())) {
            synonym = synonym.getLink();
            continue;
        }
        addSynonym(synonym.getElement());
        synonym = synonym.getLink();
    }
}
```

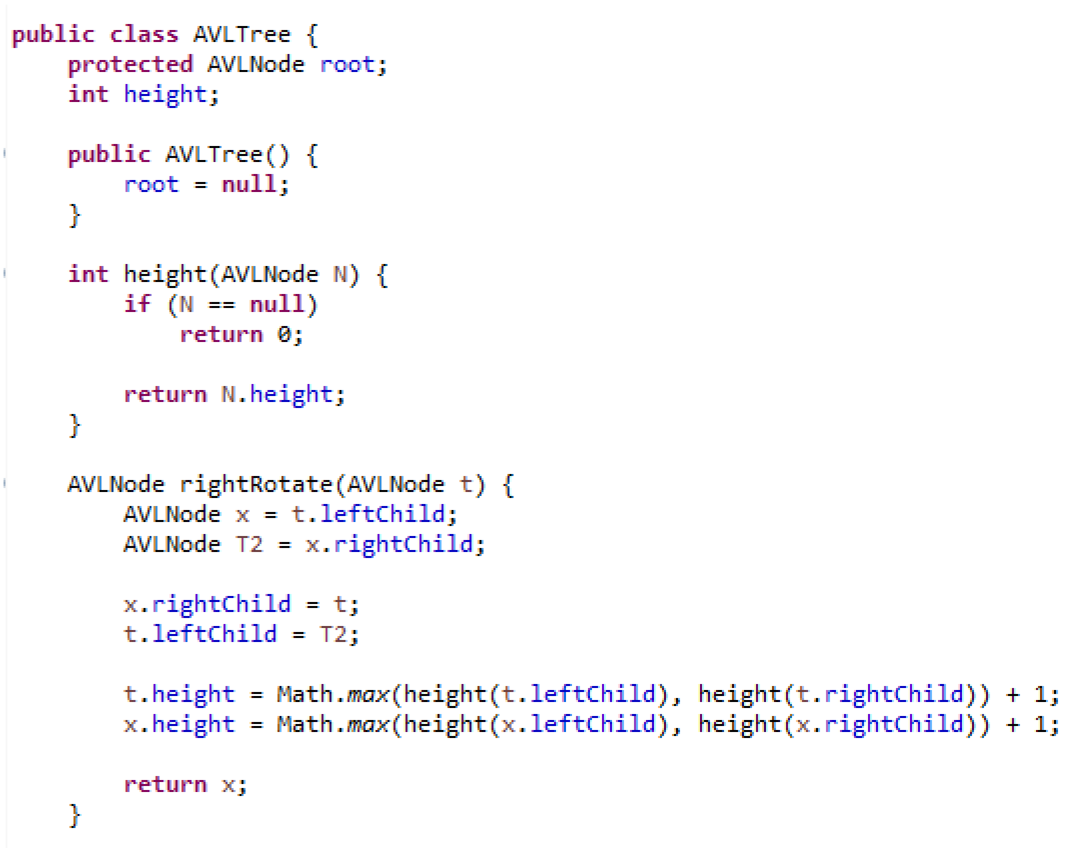


```java
public void addMeaning(String meaning) {
    meaning = meaning.toLowerCase();
    meaning = meaning.substring(0, 1).toUpperCase() + meaning.substring(1);
    if (!meanings.contains(meaning)) {
        meanings.add(meaning);
    } else
        return;

}

public void removeMeaningAt(int index) {
    if (index <= 0)
        return;
    if (index > meanings.size())
        return;

    meanings.remove(index - 1);
}

public void addSynonym(String synonym) {
    synonym = synonym.toLowerCase();
    synonym = synonym.substring(0, 1).toUpperCase() + synonym.substring(1);
    if (!synonyms.contains(synonym)) {
        synonyms.add(synonym);
    } else
        return;
}

public void removeSynonym(String synonym) {
    synonyms.remove(synonym.toLowerCase());
}
```

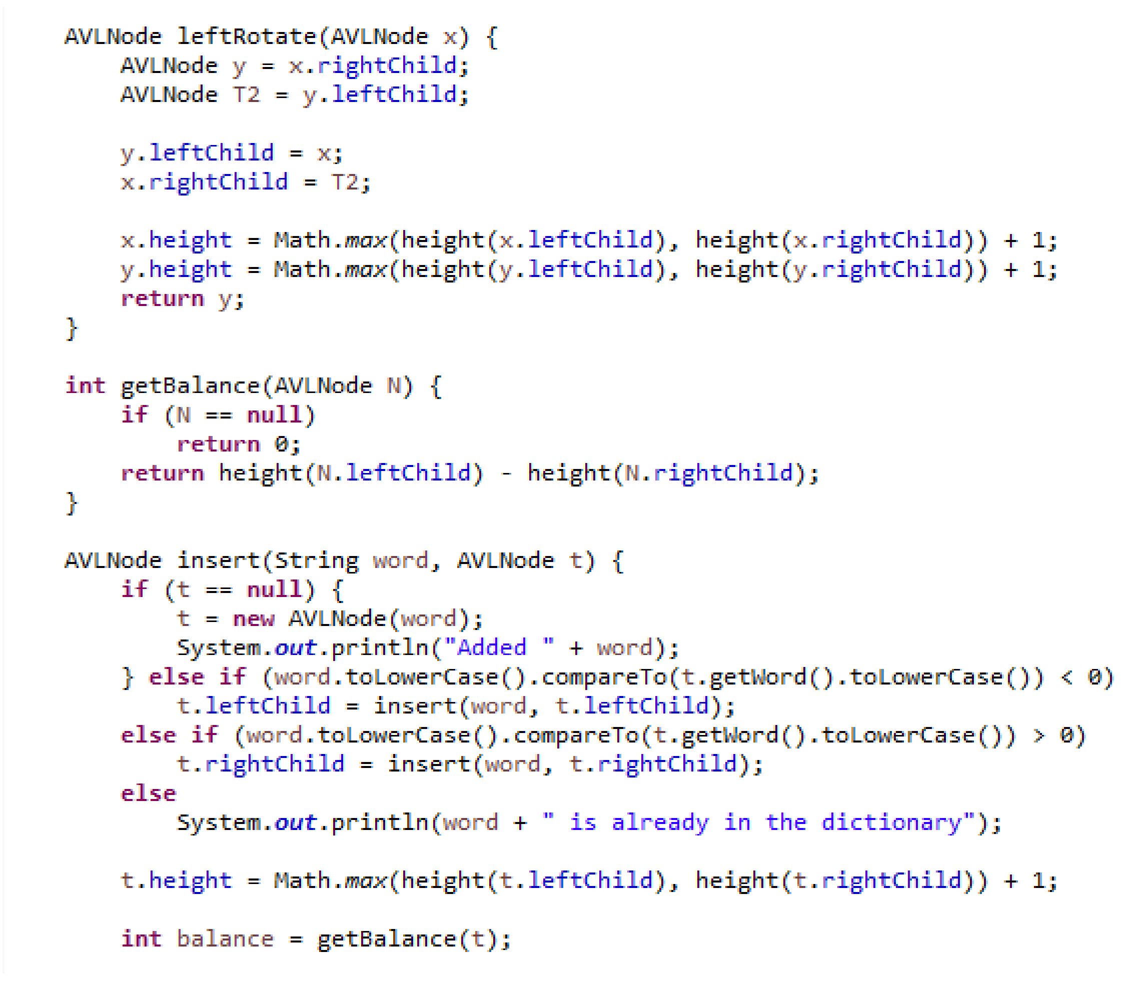


```
    public void removeSynonymAt(int index) {
        if (index <= 0)
            return;
        if (index > synonyms.size())
            return; //

        synonyms.remove(index - 1);
    }

    public String toString() {
        return word + "\nMeanings/Anlamları:\n" + meanings.toString() + "\nSynonyms/Eşseslileri:\n"
                + synonyms.toString();
    }

}
```

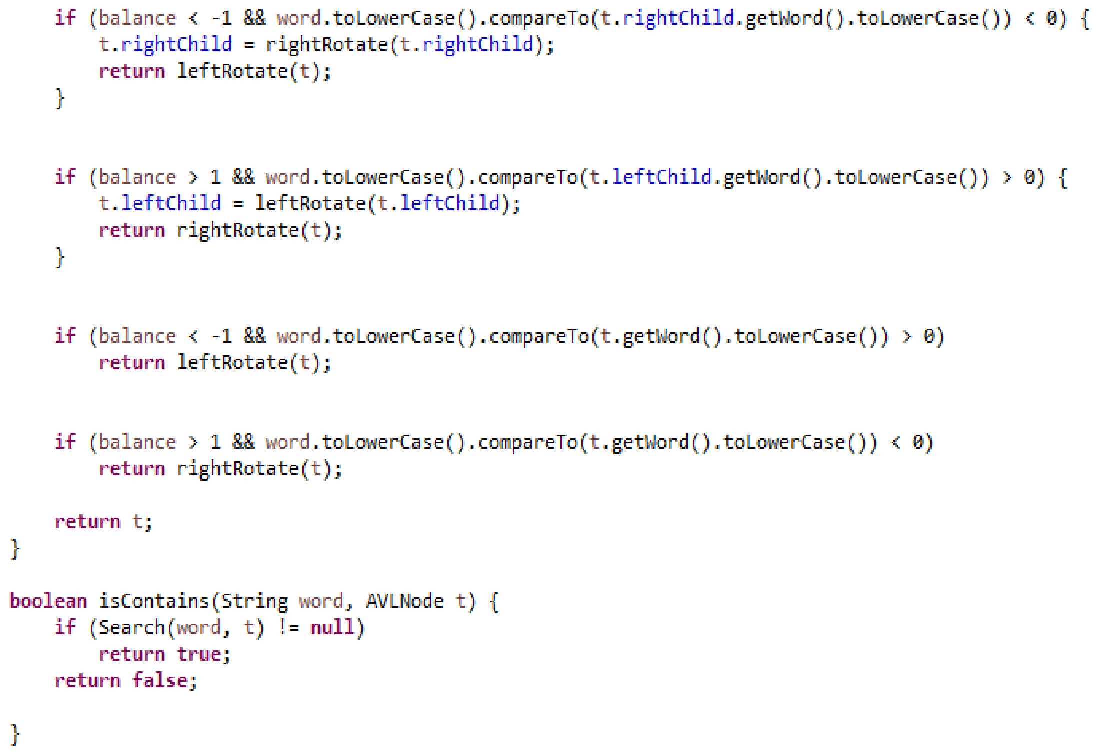
AVLTree.java Class



```
public class AVLTree {
    protected AVLNode root;
    int height;

    public AVLTree() {
        root = null;
    }

    int height(AVLNode N) {
        if (N == null)
            return 0;

        return N.height;
    }

    AVLNode rightRotate(AVLNode t) {
        AVLNode x = t.leftChild;
        AVLNode T2 = x.rightChild;

        x.rightChild = t;
        t.leftChild = T2;

        t.height = Math.max(height(t.leftChild), height(t.rightChild)) + 1;
        x.height = Math.max(height(x.leftChild), height(x.rightChild)) + 1;

        return x;
    }
```

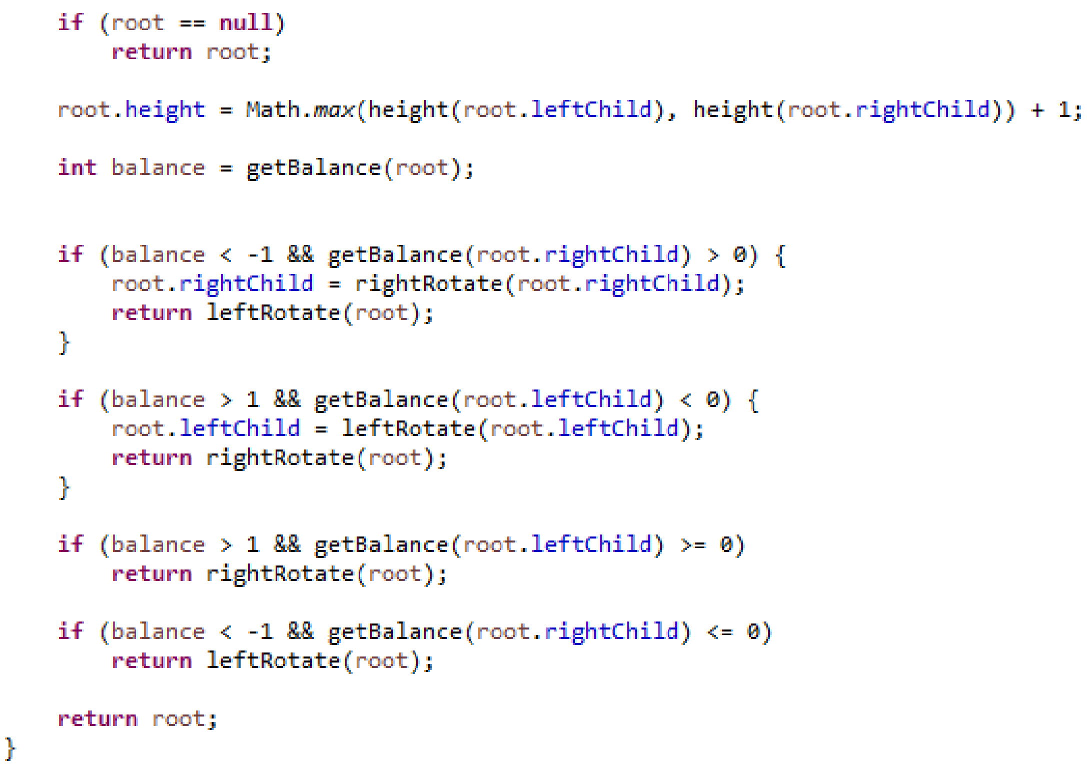


```java
AVLNode leftRotate(AVLNode x) {
    AVLNode y = x.rightChild;
    AVLNode T2 = y.leftChild;

    y.leftChild = x;
    x.rightChild = T2;

    x.height = Math.max(height(x.leftChild), height(x.rightChild)) + 1;
    y.height = Math.max(height(y.leftChild), height(y.rightChild)) + 1;
    return y;
}

int getBalance(AVLNode N) {
    if (N == null)
        return 0;
    return height(N.leftChild) - height(N.rightChild);
}

AVLNode insert(String word, AVLNode t) {
    if (t == null) {
        t = new AVLNode(word);
        System.out.println("Added " + word);
    } else if (word.toLowerCase().compareTo(t.getWord().toLowerCase()) < 0)
        t.leftChild = insert(word, t.leftChild);
    else if (word.toLowerCase().compareTo(t.getWord().toLowerCase()) > 0)
        t.rightChild = insert(word, t.rightChild);
    else
        System.out.println(word + " is already in the dictionary");

    t.height = Math.max(height(t.leftChild), height(t.rightChild)) + 1;

    int balance = getBalance(t);
```

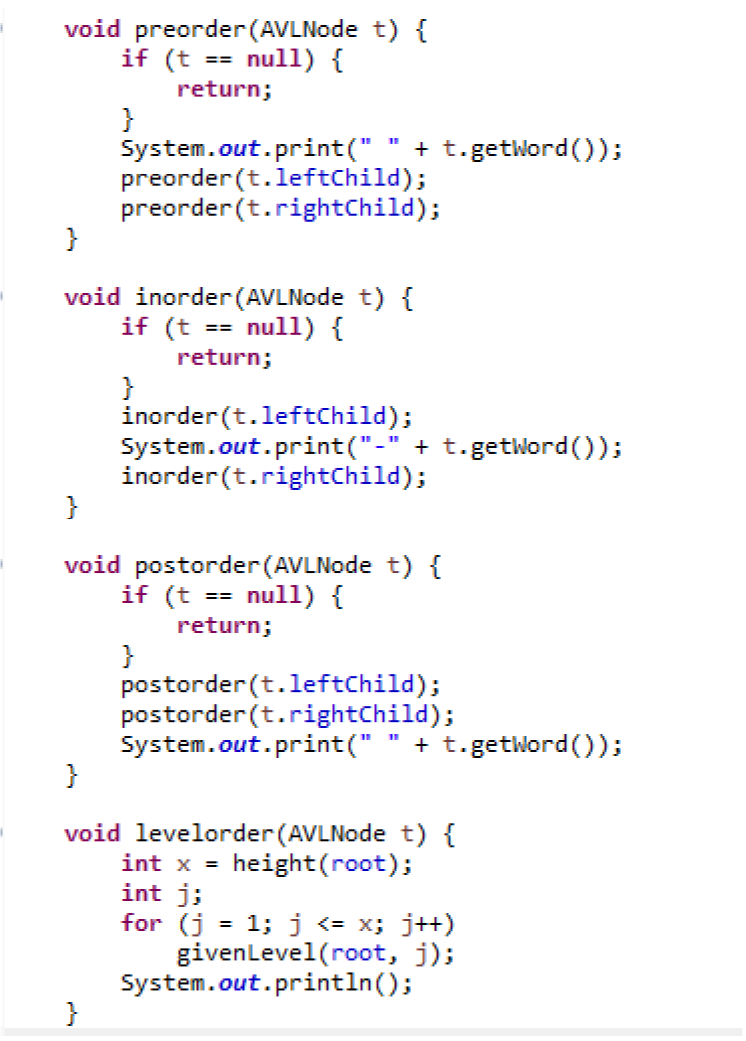


```java
    if (balance < -1 && word.toLowerCase().compareTo(t.rightChild.getWord().toLowerCase()) < 0) {
        t.rightChild = rightRotate(t.rightChild);
        return leftRotate(t);
    }


    if (balance > 1 && word.toLowerCase().compareTo(t.leftChild.getWord().toLowerCase()) > 0) {
        t.leftChild = leftRotate(t.leftChild);
        return rightRotate(t);
    }


    if (balance < -1 && word.toLowerCase().compareTo(t.getWord().toLowerCase()) > 0)
        return leftRotate(t);


    if (balance > 1 && word.toLowerCase().compareTo(t.getWord().toLowerCase()) < 0)
        return rightRotate(t);

    return t;
}

boolean isContains(String word, AVLNode t) {
    if (Search(word, t) != null)
        return true;
    return false;

}
```

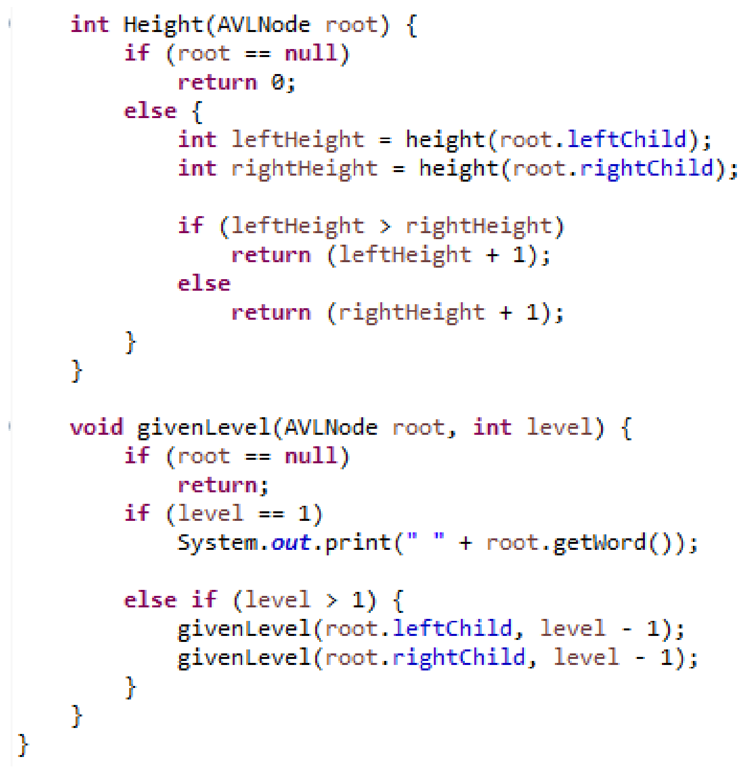
```java
AVLNode Search(String word, AVLNode t) {
    if (t == null) {
        return null;
    }
    if (word.toLowerCase().equals(t.getWord().toLowerCase())) {
        return t;
    } else if (word.toLowerCase().compareTo(t.getWord().toLowerCase()) < 0) {
        return Search(word, t.leftChild);
    } else if (word.toLowerCase().compareTo(t.getWord().toLowerCase()) > 0) {
        return Search(word, t.rightChild);
    } else
        return null;

}

AVLNode minValueNode(AVLNode node) {
    AVLNode current = node;


    while (current.leftChild != null)
        current = current.leftChild;

    return current;
}


AVLNode deleteNode(String word, AVLNode root) {

    if (root == null)
        return root;
    if (word.toLowerCase().compareTo(root.getWord().toLowerCase()) < 0)
        root.leftChild = deleteNode(word, root.leftChild);
    else if (word.toLowerCase().compareTo(root.getWord().toLowerCase()) > 0)
        root.rightChild = deleteNode(word, root.rightChild);

    else {
        if ((root.leftChild == null) || (root.rightChild == null)) {
            AVLNode temp = null;
            if (temp == root.leftChild)
                temp = root.rightChild;
            else
                temp = root.leftChild;


            if (temp == null) {
                temp = root;
                root = null;
            } else
                root = temp;

        } else {
            AVLNode temp = minValueNode(root.rightChild);
            root.setWord(temp.getWord());

            root.rightChild = deleteNode(temp.getWord(), root.rightChild);
        }
    }
```



```java
        if (root == null)
            return root;

        root.height = Math.max(height(root.leftChild), height(root.rightChild)) + 1;

        int balance = getBalance(root);


        if (balance < -1 && getBalance(root.rightChild) > 0) {
            root.rightChild = rightRotate(root.rightChild);
            return leftRotate(root);
        }

        if (balance > 1 && getBalance(root.leftChild) < 0) {
            root.leftChild = leftRotate(root.leftChild);
            return rightRotate(root);
        }

        if (balance > 1 && getBalance(root.leftChild) >= 0)
            return rightRotate(root);

        if (balance < -1 && getBalance(root.rightChild) <= 0)
            return leftRotate(root);

        return root;
    }
```

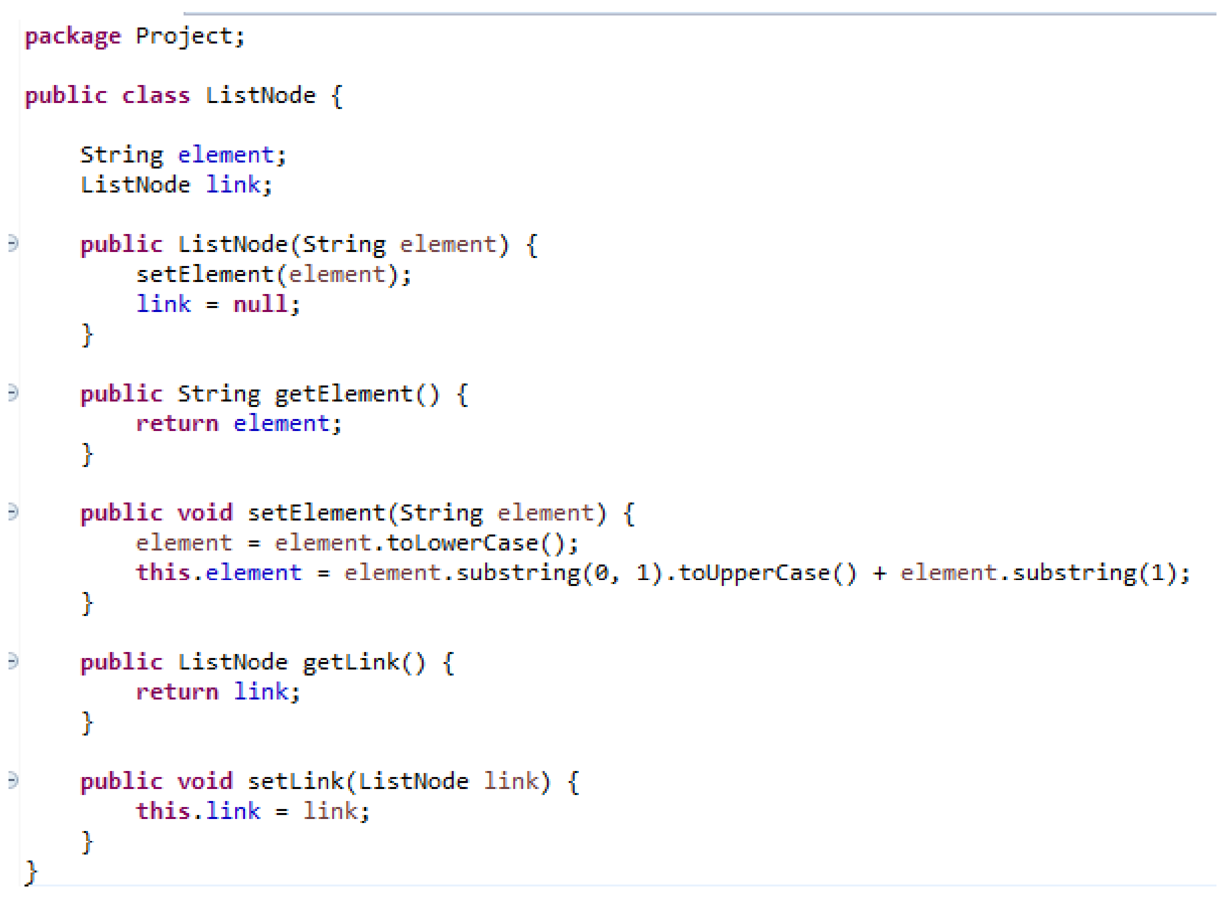


```java
    void preorder(AVLNode t) {
        if (t == null) {
            return;
        }
        System.out.print(" " + t.getWord());
        preorder(t.leftChild);
        preorder(t.rightChild);
    }

    void inorder(AVLNode t) {
        if (t == null) {
            return;
        }
        inorder(t.leftChild);
        System.out.print("-" + t.getWord());
        inorder(t.rightChild);
    }

    void postorder(AVLNode t) {
        if (t == null) {
            return;
        }
        postorder(t.leftChild);
        postorder(t.rightChild);
        System.out.print(" " + t.getWord());
    }

    void levelorder(AVLNode t) {
        int x = height(root);
        int j;
        for (j = 1; j <= x; j++)
            givenLevel(root, j);
        System.out.println();
    }
```



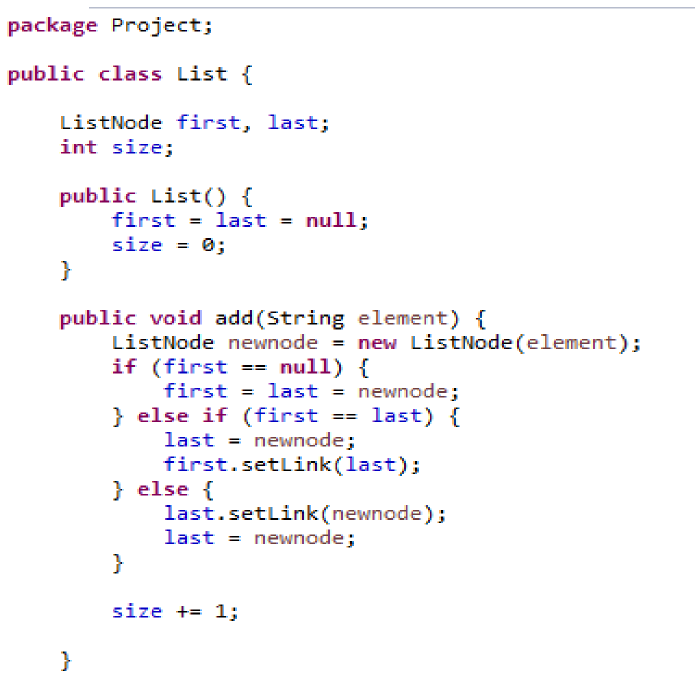
```java
    int Height(AVLNode root) {
        if (root == null)
            return 0;
        else {
            int leftHeight = height(root.leftChild);
            int rightHeight = height(root.rightChild);

            if (leftHeight > rightHeight)
                return (leftHeight + 1);
            else
                return (rightHeight + 1);
        }
    }

    void givenLevel(AVLNode root, int level) {
        if (root == null)
            return;
        if (level == 1)
            System.out.print(" " + root.getWord());

        else if (level > 1) {
            givenLevel(root.leftChild, level - 1);
            givenLevel(root.rightChild, level - 1);
        }
    }
}
```

ListNode.java Class

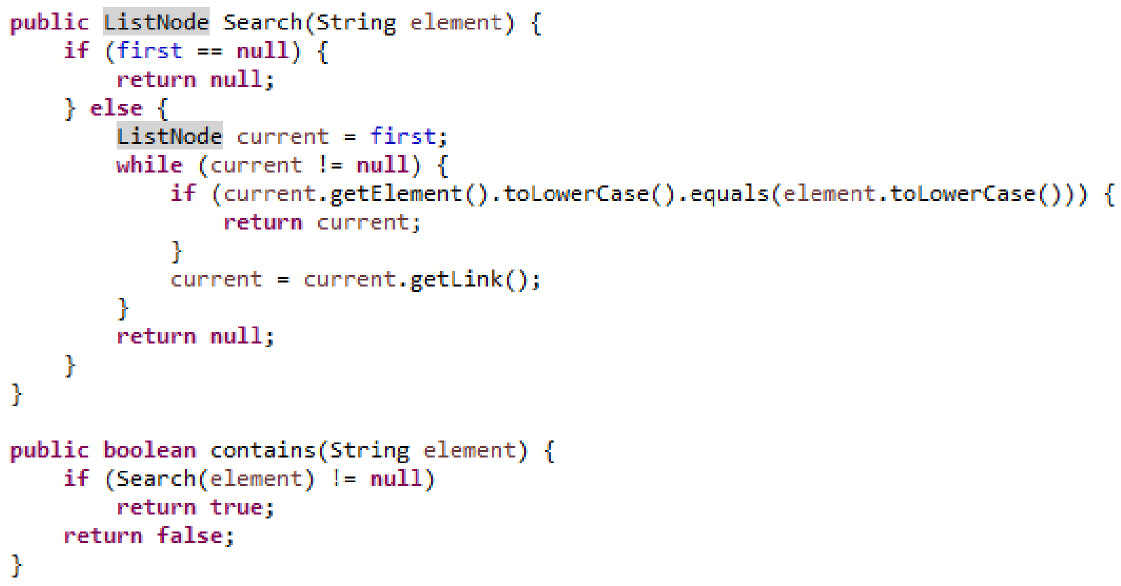


```java
package Project;

public class ListNode {

    String element;
    ListNode link;

    public ListNode(String element) {
        setElement(element);
        link = null;
    }

    public String getElement() {
        return element;
    }

    public void setElement(String element) {
        element = element.toLowerCase();
        this.element = element.substring(0, 1).toUpperCase() + element.substring(1);
    }

    public ListNode getLink() {
        return link;
    }

    public void setLink(ListNode link) {
        this.link = link;
    }
}
```

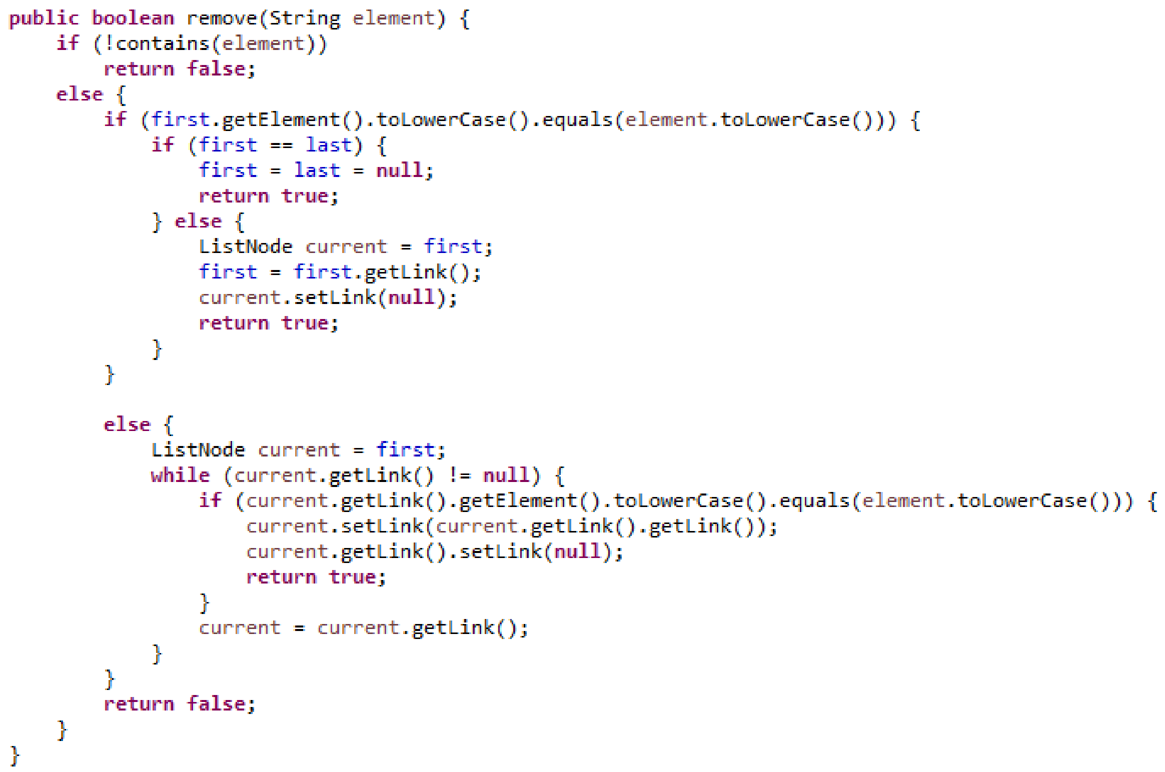
List.java Class



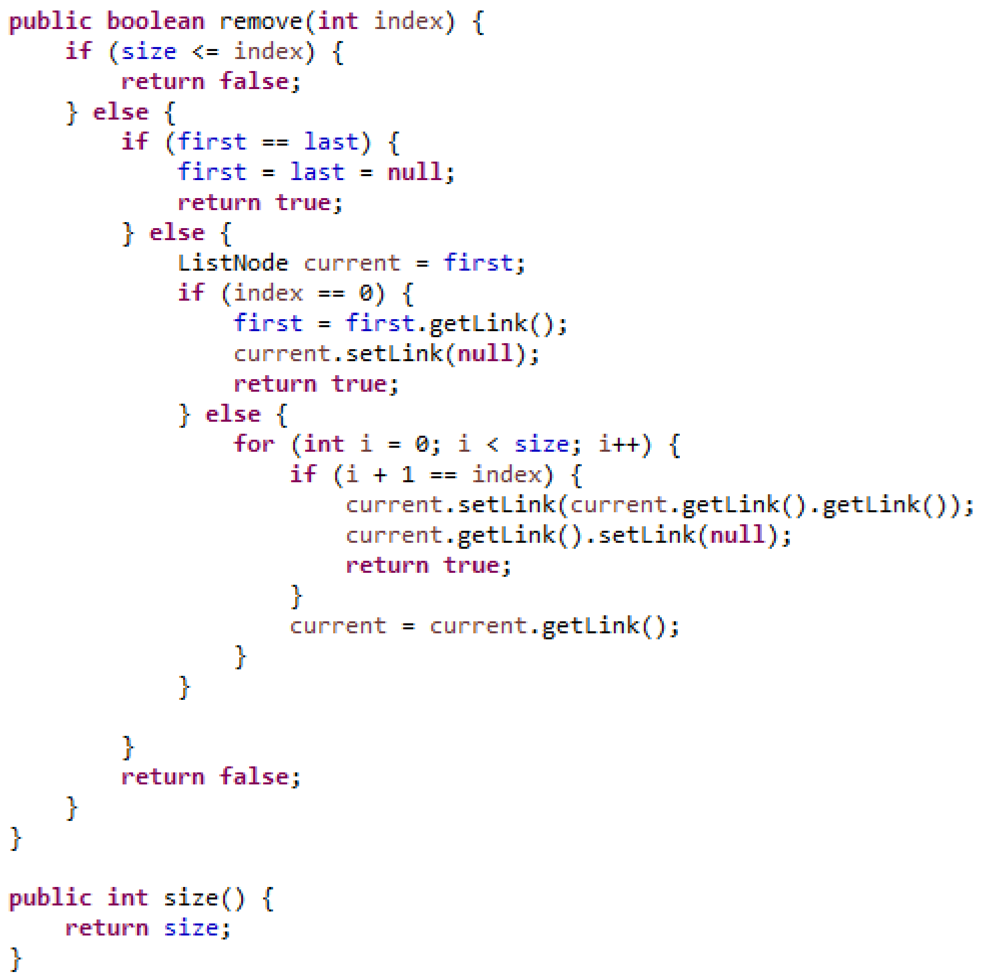
```java
package Project;

public class List {

    ListNode first, last;
    int size;

    public List() {
        first = last = null;
        size = 0;
    }

    public void add(String element) {
        ListNode newnode = new ListNode(element);
        if (first == null) {
            first = last = newnode;
        } else if (first == last) {
            last = newnode;
            first.setLink(last);
        } else {
            last.setLink(newnode);
            last = newnode;
        }

        size += 1;

    }
```



```java
    public ListNode Search(String element) {
        if (first == null) {
            return null;
        } else {
            ListNode current = first;
            while (current != null) {
                if (current.getElement().toLowerCase().equals(element.toLowerCase())) {
                    return current;
                }
                current = current.getLink();
            }
            return null;
        }
    }

    public boolean contains(String element) {
        if (Search(element) != null)
            return true;
        return false;
    }
```



```java
public boolean remove(String element) {
    if (!contains(element))
        return false;
    else {
        if (first.getElement().toLowerCase().equals(element.toLowerCase())) {
            if (first == last) {
                first = last = null;
                return true;
            } else {
                ListNode current = first;
                first = first.getLink();
                current.setLink(null);
                return true;
            }
        }

        else {
            ListNode current = first;
            while (current.getLink() != null) {
                if (current.getLink().getElement().toLowerCase().equals(element.toLowerCase())) {
                    current.setLink(current.getLink().getLink());
                    current.getLink().setLink(null);
                    return true;
                }
                current = current.getLink();
            }
        }
        return false;
    }
}
```

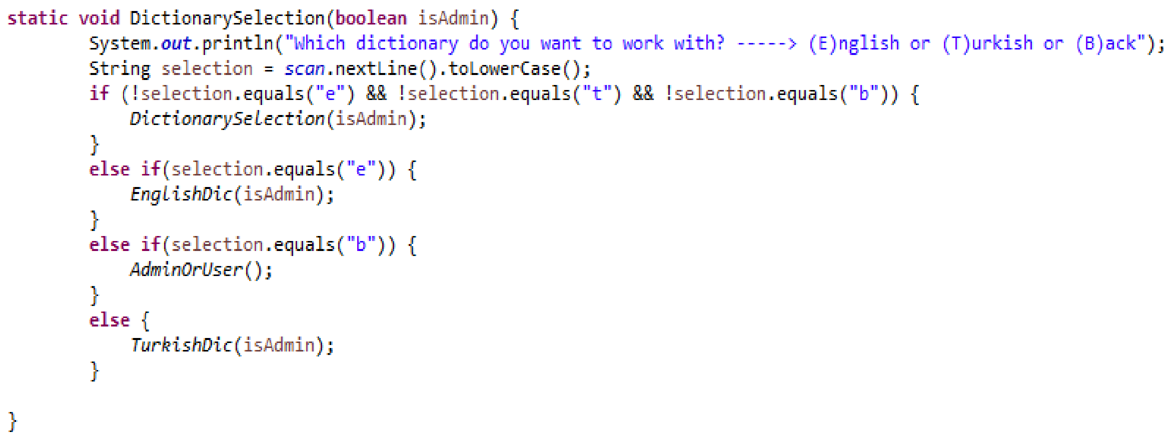


```java
public boolean remove(int index) {
    if (size <= index) {
        return false;
    } else {
        if (first == last) {
            first = last = null;
            return true;
        } else {
            ListNode current = first;
            if (index == 0) {
                first = first.getLink();
                current.setLink(null);
                return true;
            } else {
                for (int i = 0; i < size; i++) {
                    if (i + 1 == index) {
                        current.setLink(current.getLink().getLink());
                        current.getLink().setLink(null);
                        return true;
                    }
                    current = current.getLink();
                }
            }

        }
        return false;
    }
}

public int size() {
    return size;
}
```

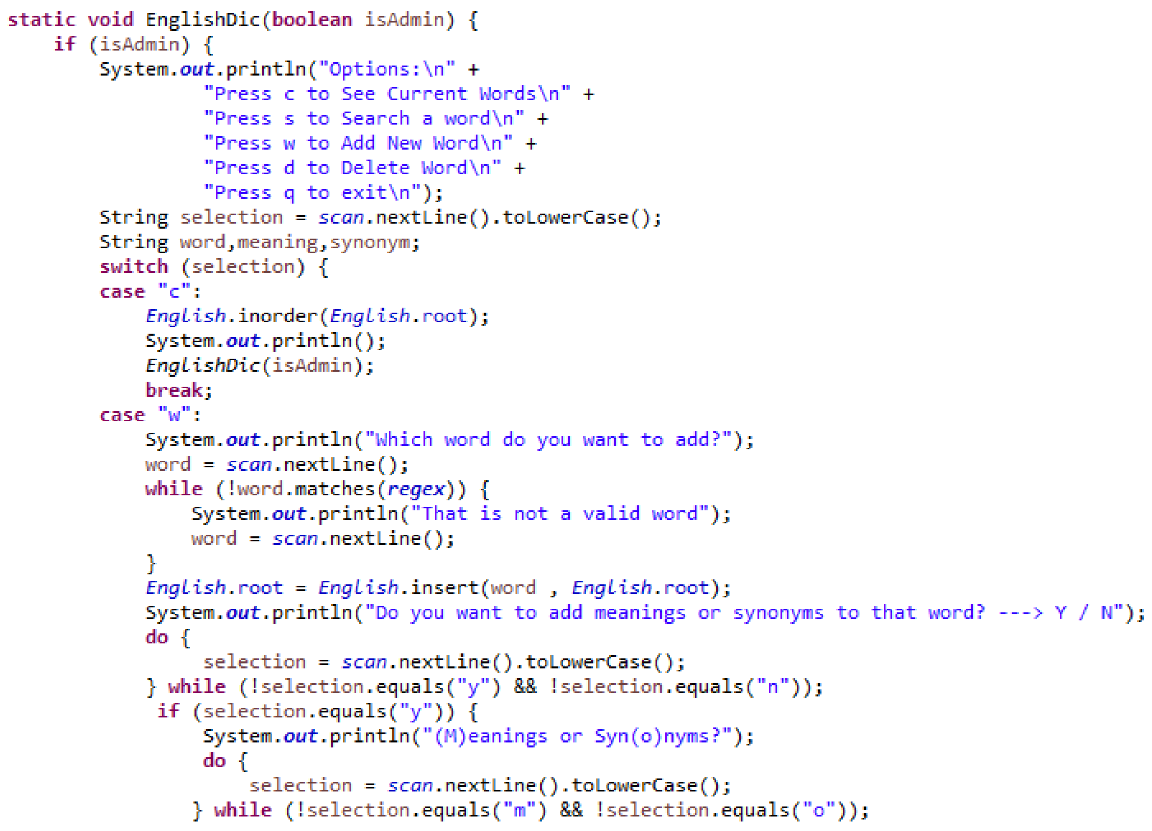
```java
    public String toString() {
        String result = "";
        ListNode current = first;
        while (current != null) {
            result += current.getElement();
            current = current.getLink();
            if (current != null) {
                result += ", ";
            }
        }
        return "[" + result + "]";
    }
}
```

<u>CPanel.java Class</u>

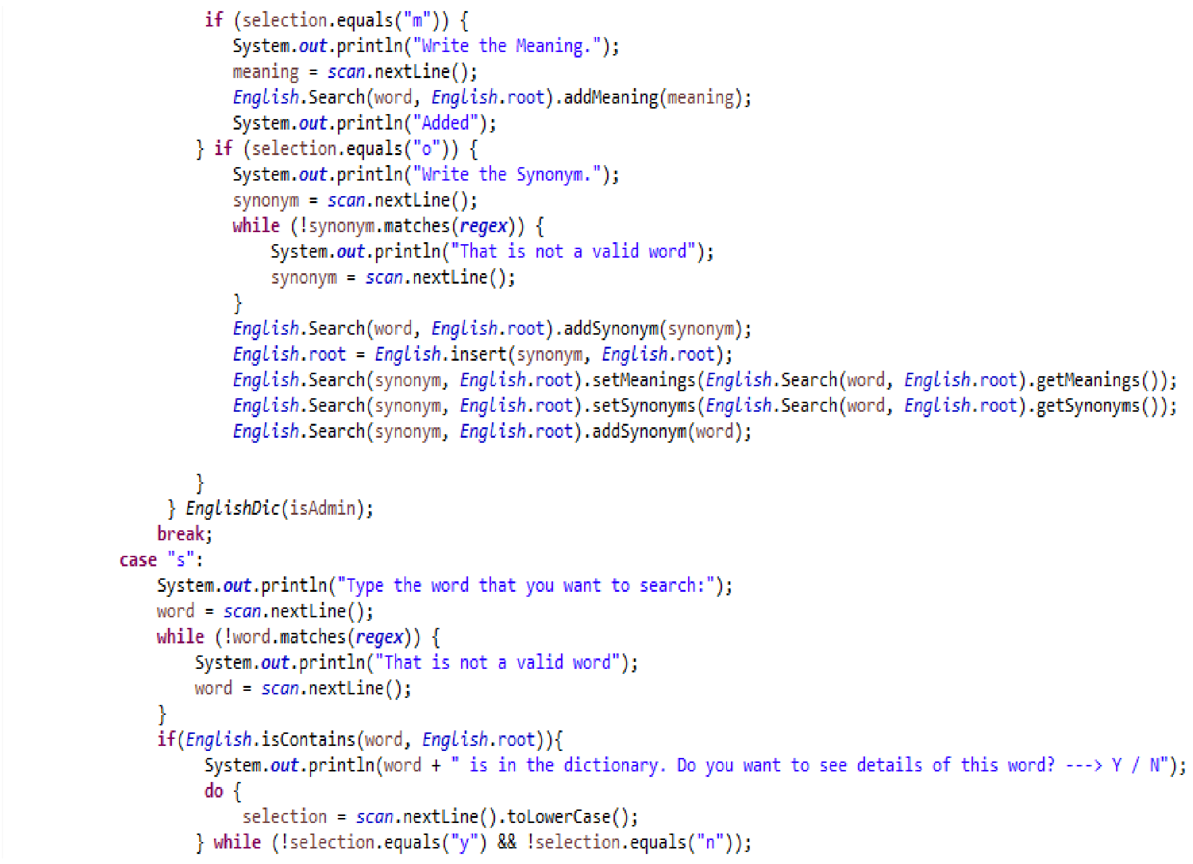
```java
package Project;

import java.util.Scanner;

public class CPanel {
    static AVLTree English = new AVLTree();
    static AVLTree Turkish = new AVLTree();
    static Scanner scan = new Scanner(System.in);
    static final String regex = "^\\p{L}+$" ;

    public static void main(String[] args) {
        System.out.println("WELCOME TO DICTIONARY SYSTEM!");
        AdminOrUser();

    }

    static void AdminOrUser() {
        System.out.println("Who are You? -----> (A)dmin or (U)ser or (Q)uit");
        String selection =  scan.nextLine().toLowerCase();
        if (!selection.equals("a") && !selection.equals("u") && !selection.equals("q")) {
            AdminOrUser();
        }
        else if(selection.equals("a")) {
            DictionarySelection(true);
        }else if (selection.equals("q")) {
            System.out.println("See ya :)");
        }
        else DictionarySelection(false);

    }
```



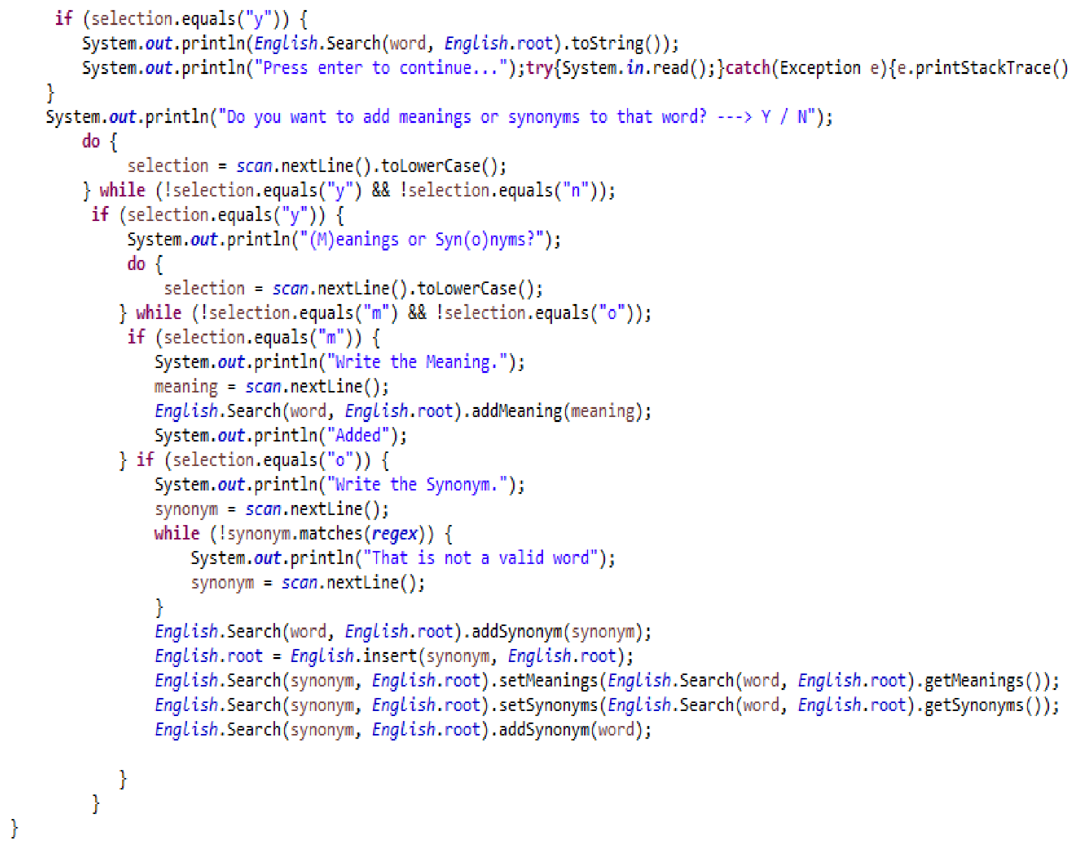
```java
static void DictionarySelection(boolean isAdmin) {
        System.out.println("Which dictionary do you want to work with? -----> (E)nglish or (T)urkish or (B)ack");
        String selection = scan.nextLine().toLowerCase();
        if (!selection.equals("e") && !selection.equals("t") && !selection.equals("b")) {
            DictionarySelection(isAdmin);
        }
        else if(selection.equals("e")) {
            EnglishDic(isAdmin);
        }
        else if(selection.equals("b")) {
            AdminOrUser();
        }
        else {
            TurkishDic(isAdmin);
        }

}
```



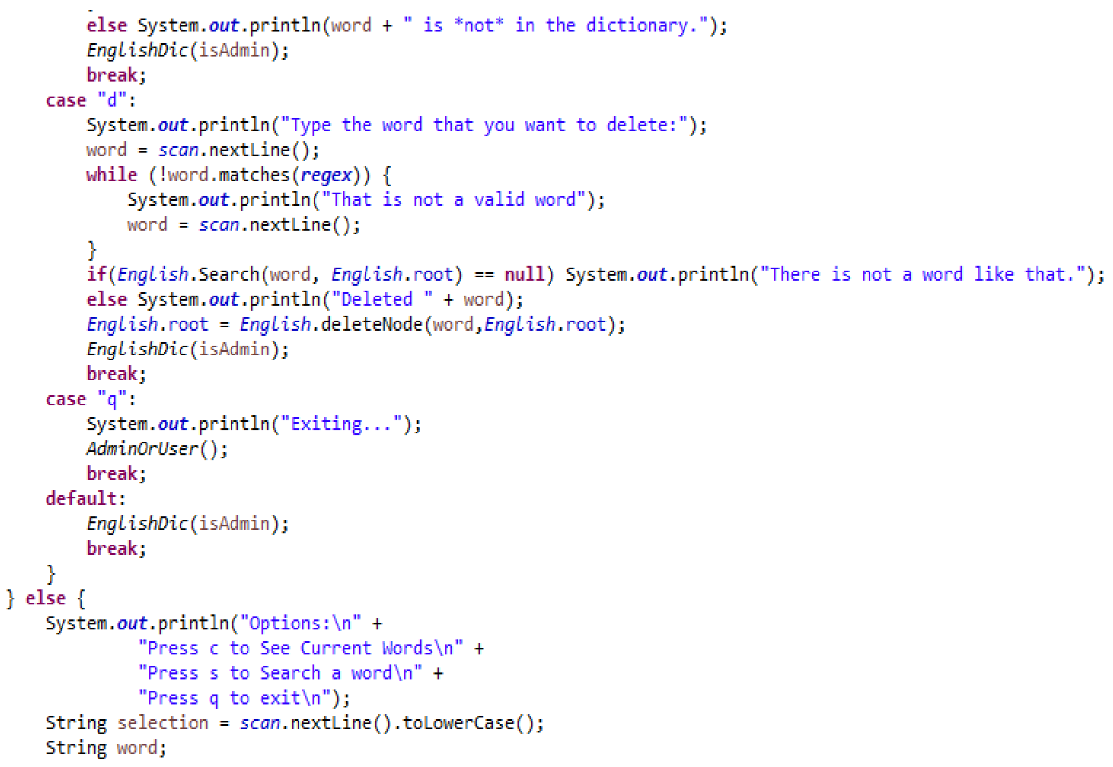
```java
static void EnglishDic(boolean isAdmin) {
    if (isAdmin) {
        System.out.println("Options:\n" +
                 "Press c to See Current Words\n" +
                 "Press s to Search a word\n" +
                 "Press w to Add New Word\n" +
                 "Press d to Delete Word\n" +
                 "Press q to exit\n");
        String selection = scan.nextLine().toLowerCase();
        String word,meaning,synonym;
        switch (selection) {
        case "c":
            English.inorder(English.root);
            System.out.println();
            EnglishDic(isAdmin);
            break;
        case "w":
            System.out.println("Which word do you want to add?");
            word = scan.nextLine();
            while (!word.matches(regex)) {
                System.out.println("That is not a valid word");
                word = scan.nextLine();
            }
            English.root = English.insert(word , English.root);
            System.out.println("Do you want to add meanings or synonyms to that word? ---> Y / N");
            do {
                 selection = scan.nextLine().toLowerCase();
            } while (!selection.equals("y") && !selection.equals("n"));
             if (selection.equals("y")) {
                 System.out.println("(M)eanings or Syn(o)nyms?");
                 do {
                      selection = scan.nextLine().toLowerCase();
                } while (!selection.equals("m") && !selection.equals("o"));
```



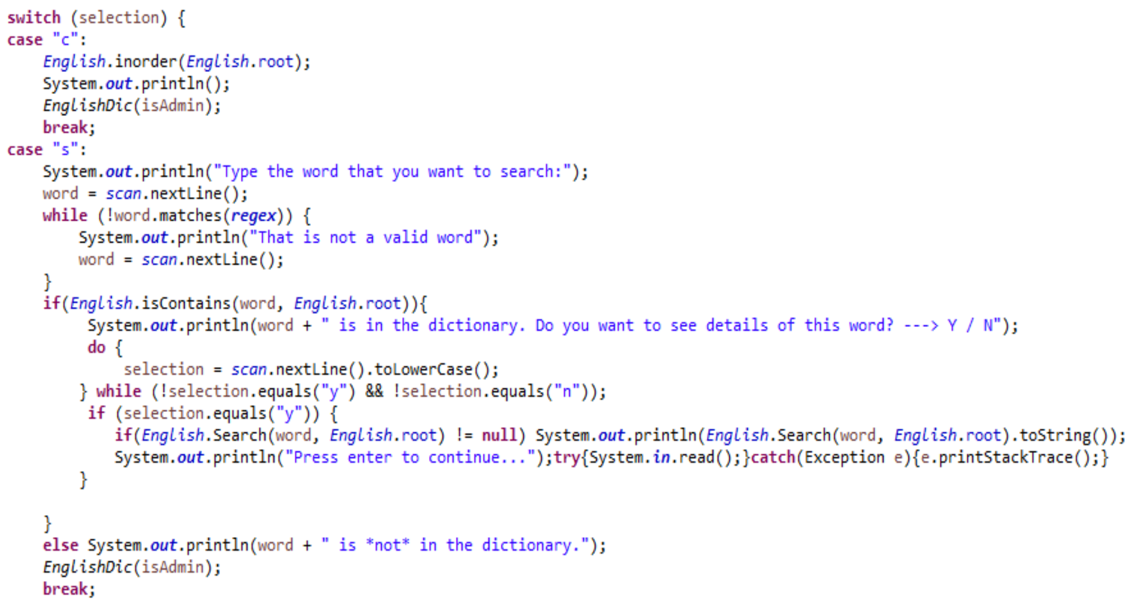
```
         if (selection.equals("m")) {
            System.out.println("Write the Meaning.");
            meaning = scan.nextLine();
            English.Search(word, English.root).addMeaning(meaning);
            System.out.println("Added");
        } if (selection.equals("o")) {
            System.out.println("Write the Synonym.");
            synonym = scan.nextLine();
            while (!synonym.matches(regex)) {
                System.out.println("That is not a valid word");
                synonym = scan.nextLine();
            }
            English.Search(word, English.root).addSynonym(synonym);
            English.root = English.insert(synonym, English.root);
            English.Search(synonym, English.root).setMeanings(English.Search(word, English.root).getMeanings());
            English.Search(synonym, English.root).setSynonyms(English.Search(word, English.root).getSynonyms());
            English.Search(synonym, English.root).addSynonym(word);

        }
     } EnglishDic(isAdmin);
    break;
case "s":
    System.out.println("Type the word that you want to search:");
    word = scan.nextLine();
    while (!word.matches(regex)) {
        System.out.println("That is not a valid word");
        word = scan.nextLine();
    }
    if(English.isContains(word, English.root)){
         System.out.println(word + " is in the dictionary. Do you want to see details of this word? ---> Y / N");
         do {
             selection = scan.nextLine().toLowerCase();
        } while (!selection.equals("y") && !selection.equals("n"));
```



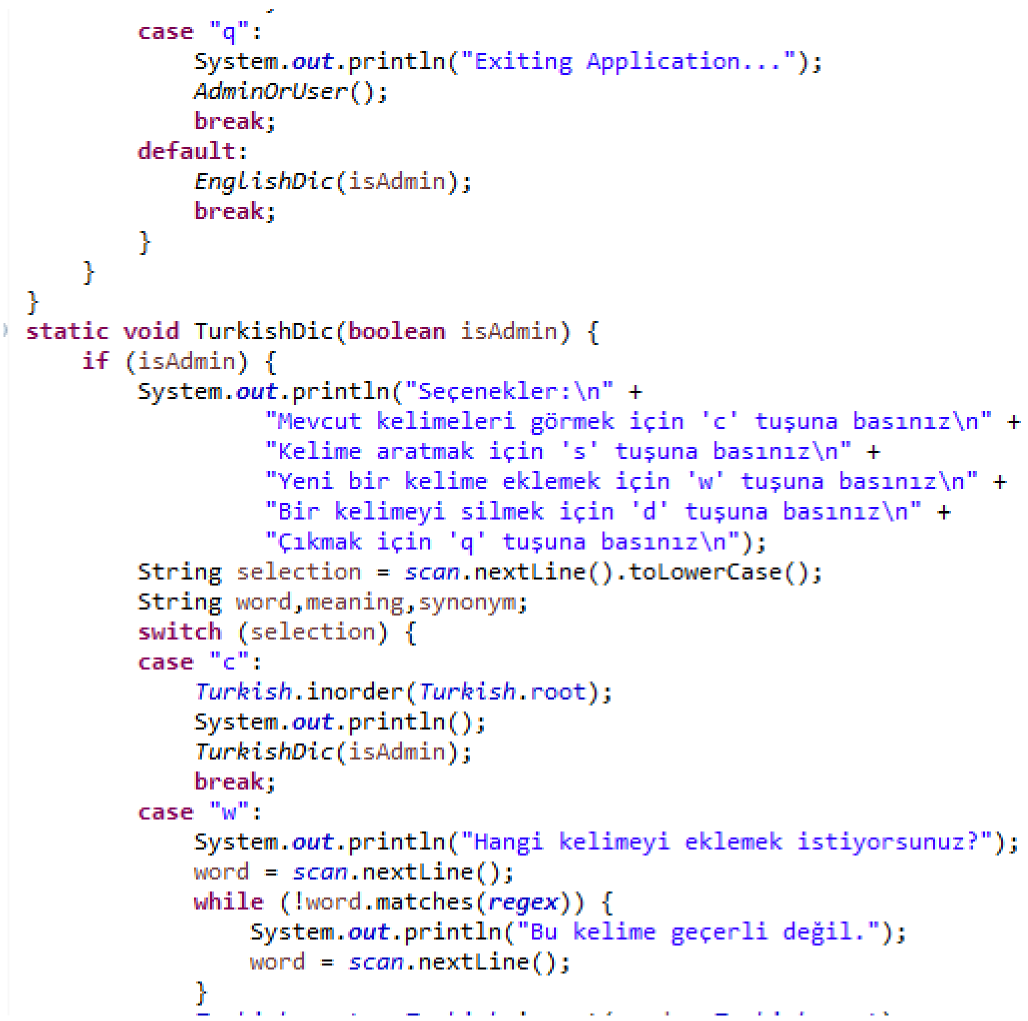
```
        if (selection.equals("y")) {
           System.out.println(English.Search(word, English.root).toString());
           System.out.println("Press enter to continue...");try{System.in.read();}catch(Exception e){e.printStackTrace()
       }
       System.out.println("Do you want to add meanings or synonyms to that word? ---> Y / N");
           do {
               selection = scan.nextLine().toLowerCase();
           } while (!selection.equals("y") && !selection.equals("n"));
            if (selection.equals("y")) {
               System.out.println("(M)eanings or Syn(o)nyms?");
               do {
                   selection = scan.nextLine().toLowerCase();
              } while (!selection.equals("m") && !selection.equals("o"));
               if (selection.equals("m")) {
                  System.out.println("Write the Meaning.");
                  meaning = scan.nextLine();
                  English.Search(word, English.root).addMeaning(meaning);
                  System.out.println("Added");
              } if (selection.equals("o")) {
                  System.out.println("Write the Synonym.");
                  synonym = scan.nextLine();
                  while (!synonym.matches(regex)) {
                      System.out.println("That is not a valid word");
                      synonym = scan.nextLine();
                  }
                  English.Search(word, English.root).addSynonym(synonym);
                  English.root = English.insert(synonym, English.root);
                  English.Search(synonym, English.root).setMeanings(English.Search(word, English.root).getMeanings());
                  English.Search(synonym, English.root).setSynonyms(English.Search(word, English.root).getSynonyms());
                  English.Search(synonym, English.root).addSynonym(word);

              }
            }
   }
```



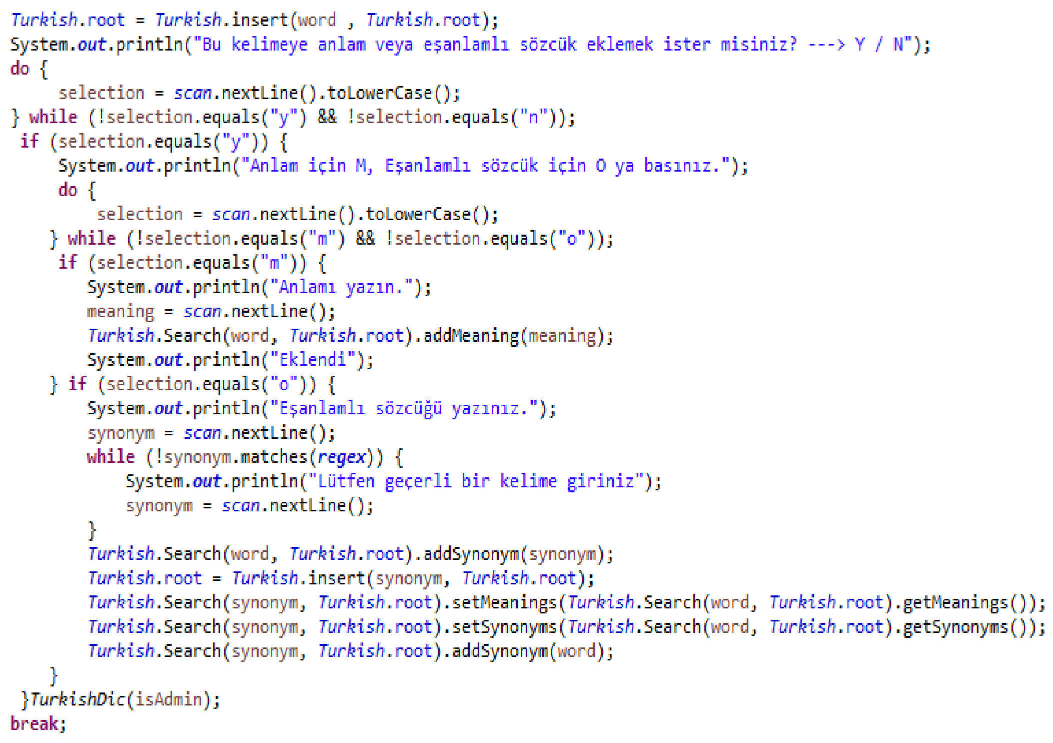
```
        else System.out.println(word + " is *not* in the dictionary.");
        EnglishDic(isAdmin);
        break;
    case "d":
        System.out.println("Type the word that you want to delete:");
        word = scan.nextLine();
        while (!word.matches(regex)) {
            System.out.println("That is not a valid word");
            word = scan.nextLine();
        }
        if(English.Search(word, English.root) == null) System.out.println("There is not a word like that.");
        else System.out.println("Deleted " + word);
        English.root = English.deleteNode(word,English.root);
        EnglishDic(isAdmin);
        break;
    case "q":
        System.out.println("Exiting...");
        AdminOrUser();
        break;
    default:
        EnglishDic(isAdmin);
        break;
    }
} else {
    System.out.println("Options:\n" +
             "Press c to See Current Words\n" +
             "Press s to Search a word\n" +
             "Press q to exit\n");
    String selection = scan.nextLine().toLowerCase();
    String word;
```



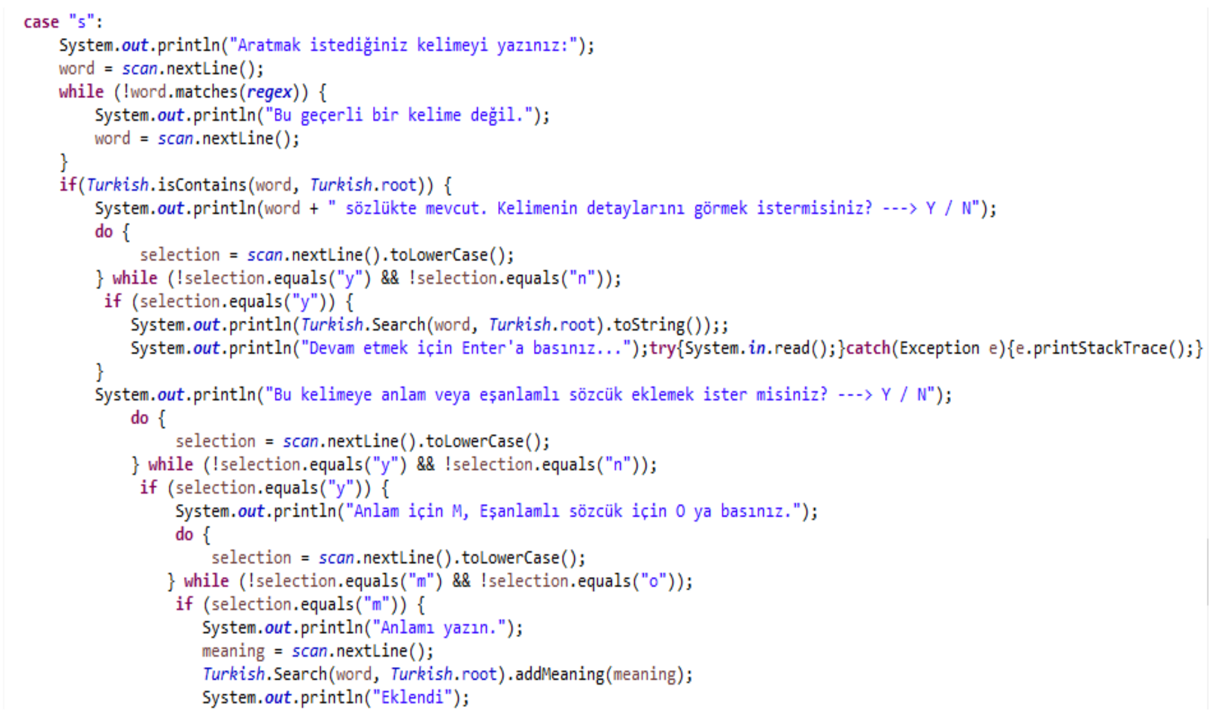
```
switch (selection) {
case "c":
    English.inorder(English.root);
    System.out.println();
    EnglishDic(isAdmin);
    break;
case "s":
    System.out.println("Type the word that you want to search:");
    word = scan.nextLine();
    while (!word.matches(regex)) {
        System.out.println("That is not a valid word");
        word = scan.nextLine();
    }
    if(English.isContains(word, English.root)){
         System.out.println(word + " is in the dictionary. Do you want to see details of this word? ---> Y / N");
         do {
             selection = scan.nextLine().toLowerCase();
        } while (!selection.equals("y") && !selection.equals("n"));
         if (selection.equals("y")) {
            if(English.Search(word, English.root) != null) System.out.println(English.Search(word, English.root).toString());
            System.out.println("Press enter to continue...");try{System.in.read();}catch(Exception e){e.printStackTrace();}
        }

    }
    else System.out.println(word + " is *not* in the dictionary.");
    EnglishDic(isAdmin);
    break;
```



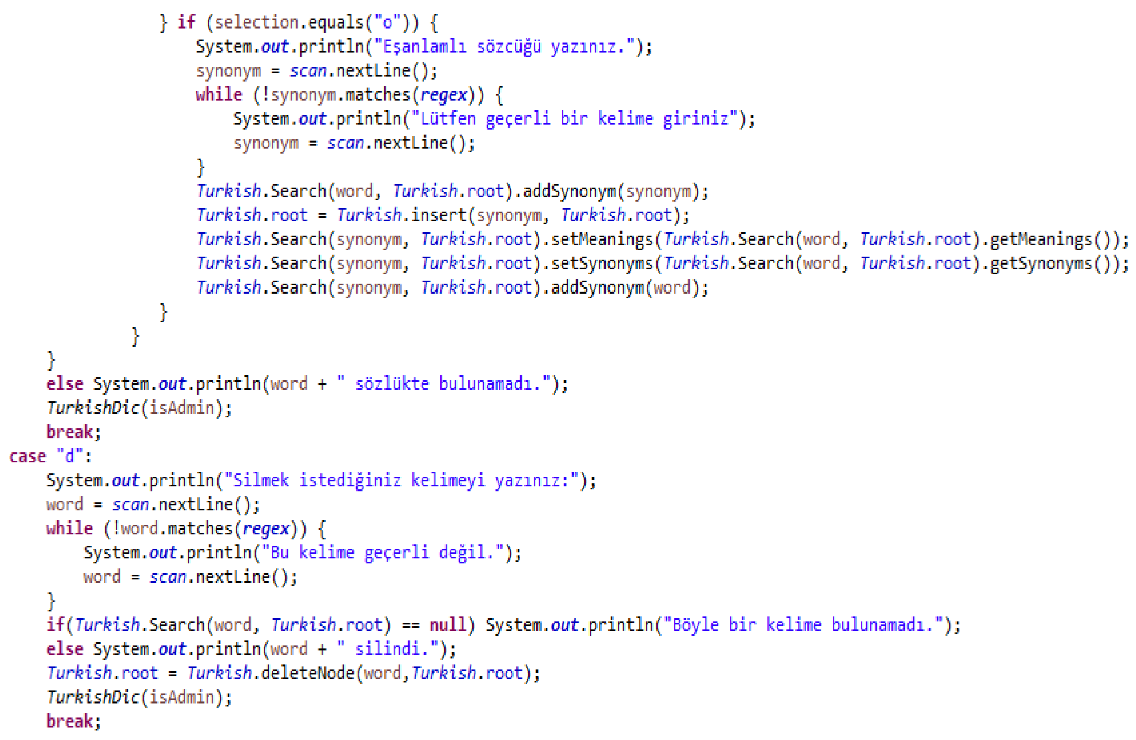
```
            case "q":
                System.out.println("Exiting Application...");
                AdminOrUser();
                break;
            default:
                EnglishDic(isAdmin);
                break;
            }
        }
    }
    static void TurkishDic(boolean isAdmin) {
        if (isAdmin) {
            System.out.println("Seçenekler:\n" +
                    "Mevcut kelimeleri görmek için 'c' tuşuna basınız\n" +
                    "Kelime aratmak için 's' tuşuna basınız\n" +
                    "Yeni bir kelime eklemek için 'w' tuşuna basınız\n" +
                    "Bir kelimeyi silmek için 'd' tuşuna basınız\n" +
                    "Çıkmak için 'q' tuşuna basınız\n");
            String selection = scan.nextLine().toLowerCase();
            String word,meaning,synonym;
            switch (selection) {
            case "c":
                Turkish.inorder(Turkish.root);
                System.out.println();
                TurkishDic(isAdmin);
                break;
            case "w":
                System.out.println("Hangi kelimeyi eklemek istiyorsunuz?");
                word = scan.nextLine();
                while (!word.matches(regex)) {
                    System.out.println("Bu kelime geçerli değil.");
                    word = scan.nextLine();
                }
```



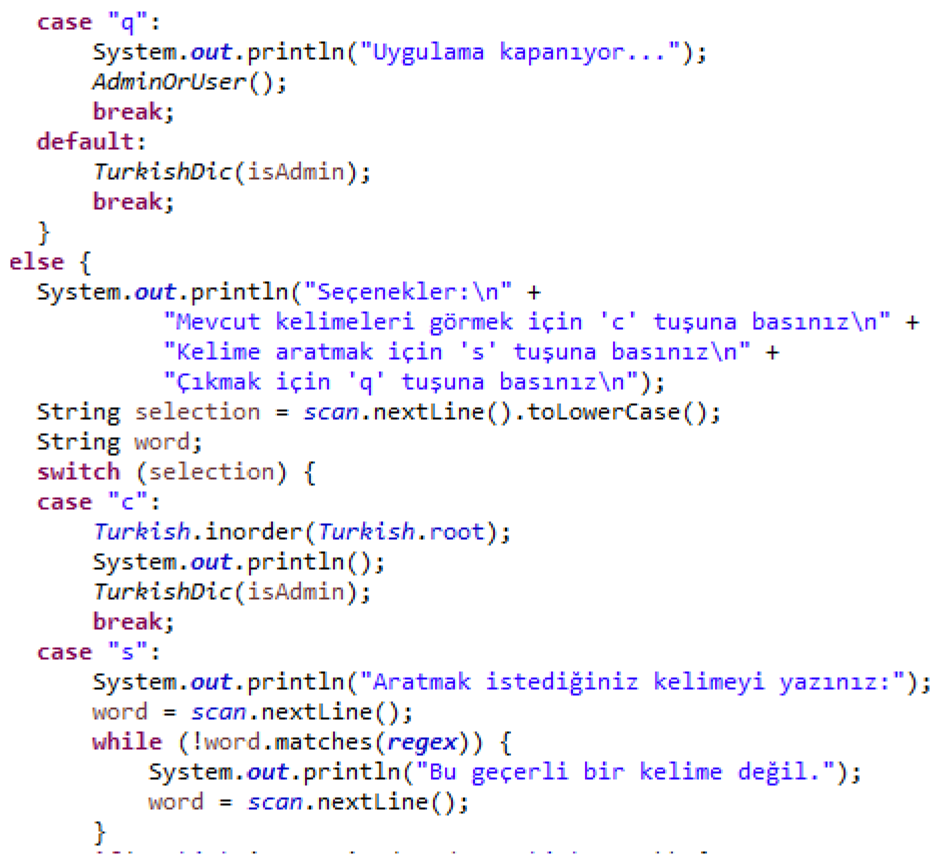
```
Turkish.root = Turkish.insert(word , Turkish.root);
System.out.println("Bu kelimeye anlam veya eşanlamlı sözcük eklemek ister misiniz? ---> Y / N");
do {
     selection = scan.nextLine().toLowerCase();
} while (!selection.equals("y") && !selection.equals("n"));
 if (selection.equals("y")) {
     System.out.println("Anlam için M, Eşanlamlı sözcük için O ya basınız.");
     do {
         selection = scan.nextLine().toLowerCase();
    } while (!selection.equals("m") && !selection.equals("o"));
     if (selection.equals("m")) {
        System.out.println("Anlamı yazın.");
        meaning = scan.nextLine();
        Turkish.Search(word, Turkish.root).addMeaning(meaning);
        System.out.println("Eklendi");
    } if (selection.equals("o")) {
        System.out.println("Eşanlamlı sözcüğü yazınız.");
        synonym = scan.nextLine();
        while (!synonym.matches(regex)) {
            System.out.println("Lütfen geçerli bir kelime giriniz");
            synonym = scan.nextLine();
        }
        Turkish.Search(word, Turkish.root).addSynonym(synonym);
        Turkish.root = Turkish.insert(synonym, Turkish.root);
        Turkish.Search(synonym, Turkish.root).setMeanings(Turkish.Search(word, Turkish.root).getMeanings());
        Turkish.Search(synonym, Turkish.root).setSynonyms(Turkish.Search(word, Turkish.root).getSynonyms());
        Turkish.Search(synonym, Turkish.root).addSynonym(word);
    }
 }TurkishDic(isAdmin);
break;
```



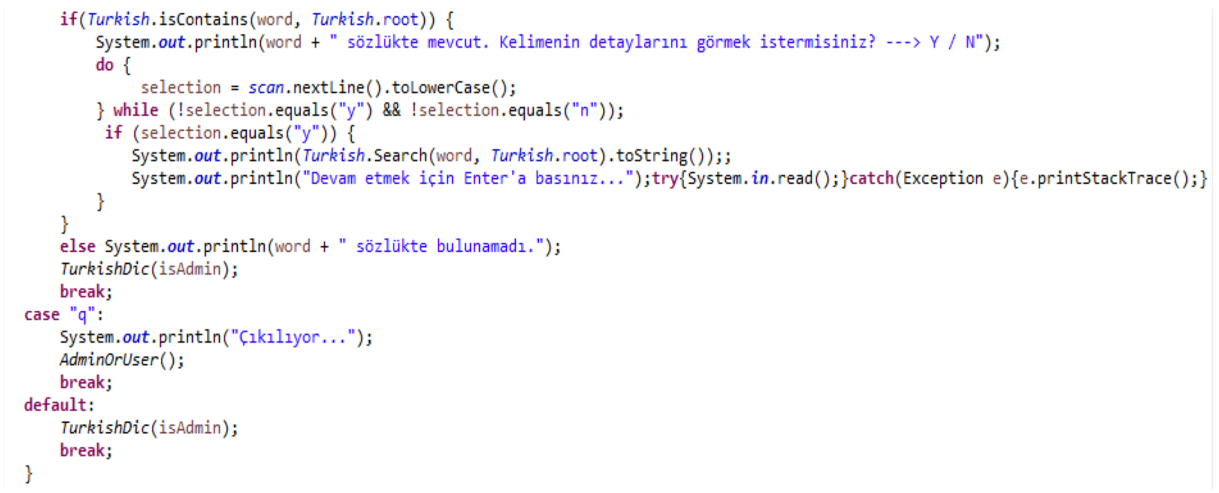
```java
case "s":
    System.out.println("Aratmak istediğiniz kelimeyi yazınız:");
    word = scan.nextLine();
    while (!word.matches(regex)) {
        System.out.println("Bu geçerli bir kelime değil.");
        word = scan.nextLine();
    }
    if(Turkish.isContains(word, Turkish.root)) {
        System.out.println(word + " sözlükte mevcut. Kelimenin detaylarını görmek istermisiniz? ---> Y / N");
        do {
             selection = scan.nextLine().toLowerCase();
        } while (!selection.equals("y") && !selection.equals("n"));
         if (selection.equals("y")) {
            System.out.println(Turkish.Search(word, Turkish.root).toString());;
            System.out.println("Devam etmek için Enter'a basınız...");try{System.in.read();}catch(Exception e){e.printStackTrace();}
        }
        System.out.println("Bu kelimeye anlam veya eşanlamlı sözcük eklemek ister misiniz? ---> Y / N");
            do {
                 selection = scan.nextLine().toLowerCase();
            } while (!selection.equals("y") && !selection.equals("n"));
             if (selection.equals("y")) {
                System.out.println("Anlam için M, Eşanlamlı sözcük için O ya basınız.");
                do {
                     selection = scan.nextLine().toLowerCase();
                } while (!selection.equals("m") && !selection.equals("o"));
                if (selection.equals("m")) {
                   System.out.println("Anlamı yazın.");
                   meaning = scan.nextLine();
                   Turkish.Search(word, Turkish.root).addMeaning(meaning);
                   System.out.println("Eklendi");
```



```java
                } if (selection.equals("o")) {
                    System.out.println("Eşanlamlı sözcüğü yazınız.");
                    synonym = scan.nextLine();
                    while (!synonym.matches(regex)) {
                        System.out.println("Lütfen geçerli bir kelime giriniz");
                        synonym = scan.nextLine();
                    }
                    Turkish.Search(word, Turkish.root).addSynonym(synonym);
                    Turkish.root = Turkish.insert(synonym, Turkish.root);
                    Turkish.Search(synonym, Turkish.root).setMeanings(Turkish.Search(word, Turkish.root).getMeanings());
                    Turkish.Search(synonym, Turkish.root).setSynonyms(Turkish.Search(word, Turkish.root).getSynonyms());
                    Turkish.Search(synonym, Turkish.root).addSynonym(word);
                }
            }
    }
    else System.out.println(word + " sözlükte bulunamadı.");
    TurkishDic(isAdmin);
    break;
case "d":
    System.out.println("Silmek istediğiniz kelimeyi yazınız:");
    word = scan.nextLine();
    while (!word.matches(regex)) {
        System.out.println("Bu kelime geçerli değil.");
        word = scan.nextLine();
    }
    if(Turkish.Search(word, Turkish.root) == null) System.out.println("Böyle bir kelime bulunamadı.");
    else System.out.println(word + " silindi.");
    Turkish.root = Turkish.deleteNode(word,Turkish.root);
    TurkishDic(isAdmin);
    break;
```



```
  case "q":
      System.out.println("Uygulama kapanıyor...");
      AdminOrUser();
      break;
  default:
      TurkishDic(isAdmin);
      break;
  }
else {
  System.out.println("Seçenekler:\n" +
           "Mevcut kelimeleri görmek için 'c' tuşuna basınız\n" +
           "Kelime aratmak için 's' tuşuna basınız\n" +
           "Çıkmak için 'q' tuşuna basınız\n");
  String selection = scan.nextLine().toLowerCase();
  String word;
  switch (selection) {
  case "c":
      Turkish.inorder(Turkish.root);
      System.out.println();
      TurkishDic(isAdmin);
      break;
  case "s":
      System.out.println("Aratmak istediğiniz kelimeyi yazınız:");
      word = scan.nextLine();
      while (!word.matches(regex)) {
          System.out.println("Bu geçerli bir kelime değil.");
          word = scan.nextLine();
      }
```



```
    if(Turkish.isContains(word, Turkish.root)) {
        System.out.println(word + " sözlükte mevcut. Kelimenin detaylarını görmek istermisiniz? ---> Y / N");
        do {
            selection = scan.nextLine().toLowerCase();
        } while (!selection.equals("y") && !selection.equals("n"));
         if (selection.equals("y")) {
            System.out.println(Turkish.Search(word, Turkish.root).toString());;
            System.out.println("Devam etmek için Enter'a basınız...");try{System.in.read();}catch(Exception e){e.printStackTrace();}
        }
    }
    else System.out.println(word + " sözlükte bulunamadı.");
    TurkishDic(isAdmin);
    break;
case "q":
    System.out.println("Çıkılıyor...");
    AdminOrUser();
    break;
default:
    TurkishDic(isAdmin);
    break;
}
```

3. References

- 2019-2020 Data Structures & Algorithm 2 Lecture Notes
- App.diagrams.net
- Stackoverflow.com