Gruppuppgift DVI VT11

Grupp 1

Medlemmar:
Jens Persson
Hannes Landstedt
Daniel Aladics
Magnus Grönvall

Innehållsförteckning

Innehåll

Arbetsbeskrivning

Alla arbetade gemensamt.

Instruktioner för programstart

Du behöver Java installerat.
Roten ska vara mappen som innehåller två mappar: src och data.
Src ska vara uppdelad i två paket: aVLTree samt LibraryOOAD.
Data ska ha två filer: Lantagare.txt samt Media.txt
Programmet startar ifrån Catalog.java i LibraryOOAD paketet.
Man loggar normalt in med sitt namn och lösen är sitt personnummer.
En defualt användare är inlagd; admin med lösen admin.

Systembeskrivning

Det är ett bibliotek system som man kan använda för att låna och återlämna media samt leta efter media som finns inlagda i systemet. För att använda systemet måste man vara en registrerad användare. Efter att man har loggat in kan man se de olika valen man kan använda sig av systemet.
Man kan:

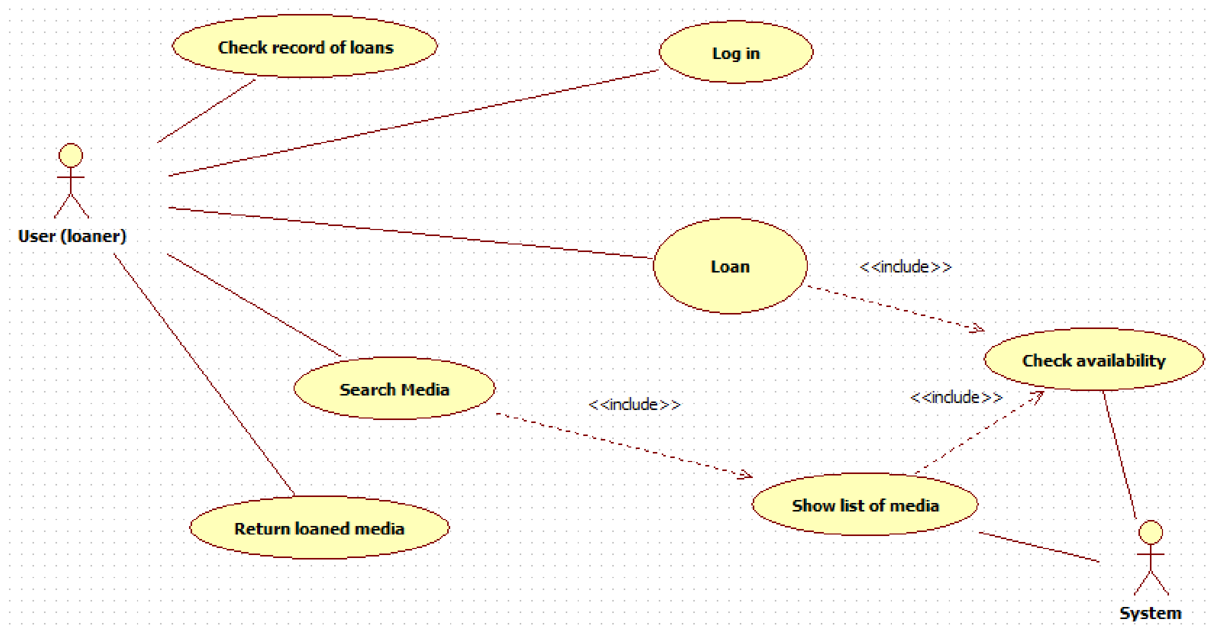
- Söka
    - Skriv något i fältet och systemet letar upp relevant media.
- Se nuvarande lån
    - Listar all nuvarande lån du har.
- Registrera ett nytt lån på media
    - Skriv in ett medias ID och tryck låna.
- Återlämna ett lån på media
    - Skriv in ett medias ID och tryck återlämna.
- Logga ut.
    - Loggar ut användaren och inloggningsfönster visas.

Användningsfall

Aktörer

- User
- System

Användningsfallsdiagram



Användningsfallsbeskrivningar

*** Check record of loans ***
Description
===========
The user retrieves a list of their current loans.

Preconditions
=============
The user is logged in.

Basic Flow
==========
1 - User clicks 'View Loans' button.
2 - Sees a list of current loans.

Alternative Flows
=================
User has no current loans.
2.1 - Display message "you have no current loans".

Postconditions
==============

*** Log in ***
Description
===========
A user logs in to the system.

Preconditions
=============
User is NOT logged in.

Basic Flow
==========
1. User enters username and password.
2. User presses 'log in' button.
3. System logs in the user with matching username and password.

Alternative Flows
=================
Incorrect username or password.
3.1 - Show error message.

Postconditions
==============
User is logged in.

*** Loan ***
Description
===========
User makes a loan.

Preconditions
=============
User is logged in.
User has chosen a book to check out.

Basic Flow
==========
1. User enters media ID.
2. System makes sure media is available.
3. System adds media to user's loans.
4. System sets return date of media.
5. System sets current loaner of media to user.

Alternative Flows
=================
Media not avaiable.
2.1 - Show 'not available' message.

Postconditions
==============
Media's loaning status (returnDate and user) has been changed.
Media has been added to user's loans.

*** Search media ***
Description
===========
The user enters something to search for and clicks the search button. A result matching user's search is displayed.

Preconditions
=============

Basic Flow
==========
1. User enters a string to be searched for.
2. User clicks 'search'.
3. Search result is displayed.

Alternative Flows
=================
No result.
3.1 - Error message 'nothing found' is displayed.

Postconditions
==============
(User knows ID of media to borrow).

*** Return loaned media ***
Description
===========
Return something borrowed.

Preconditions
=============
User has ID of media to be returned.

Basic Flow
==========
1. User enters ID of media to be returned.
2. System locates media using ID.
3. System removes media from user's loans.
4. System sets return date of media to null.
5. System sets current loaner of media to null.

Alternative Flows
=================
Media ID not found.
2.1 - Display message "Invalid ID, make sure you did not make a typo!".

Media has no user.
3.1 - Display message "media is already available".

Postconditions
==============
Media's loaning status (returnDate and user) has been changed.
Media has been removed from user's loans.

*** Show list of media ***
Description
===========
Shows a list of media.

Preconditions
=============
There is a list of media to be shown.

Basic Flow
==========
1. List media.

Alternative Flows
=================

Postconditions
==============

*** Check availability ***
Description
===========
Displays availability information about a media.

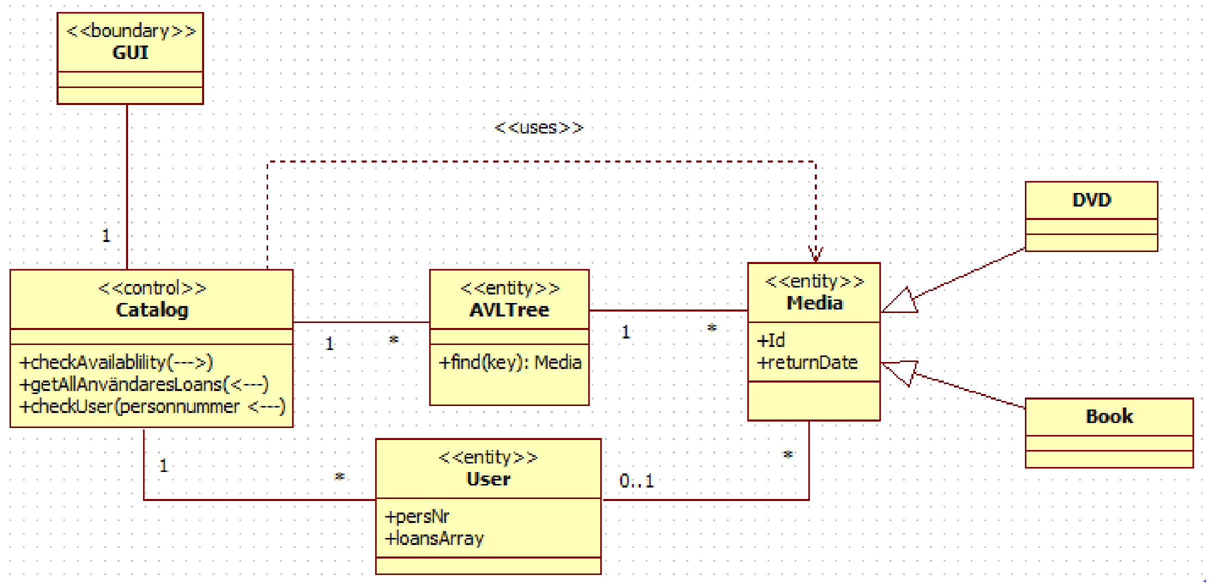
Preconditions
=============
There is a media.

Basic Flow
==========
1. Show that it is available.

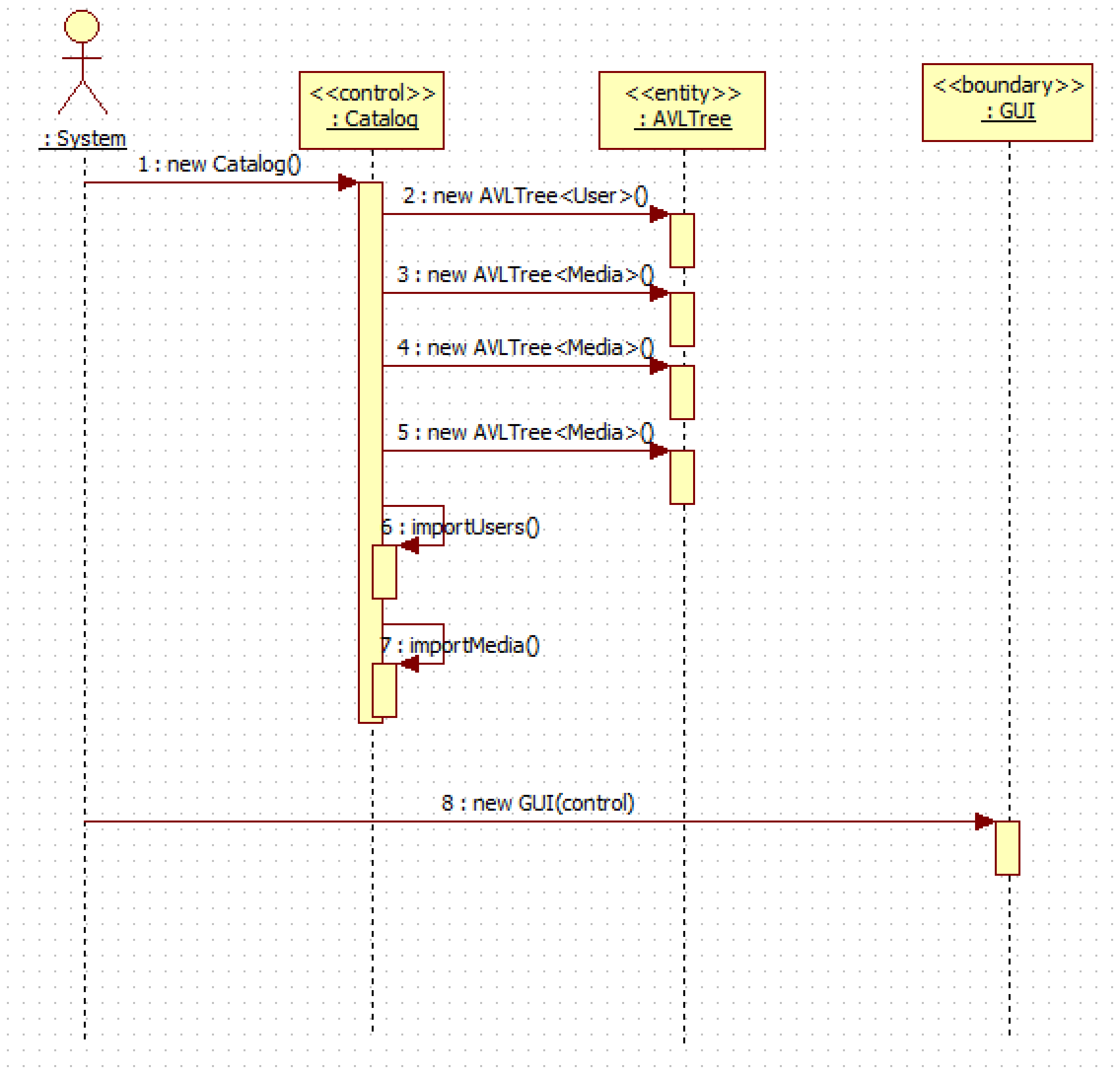
Alternative Flows
=================
Media is not available.
1.2 - Show return date of media.

Postconditions
==============

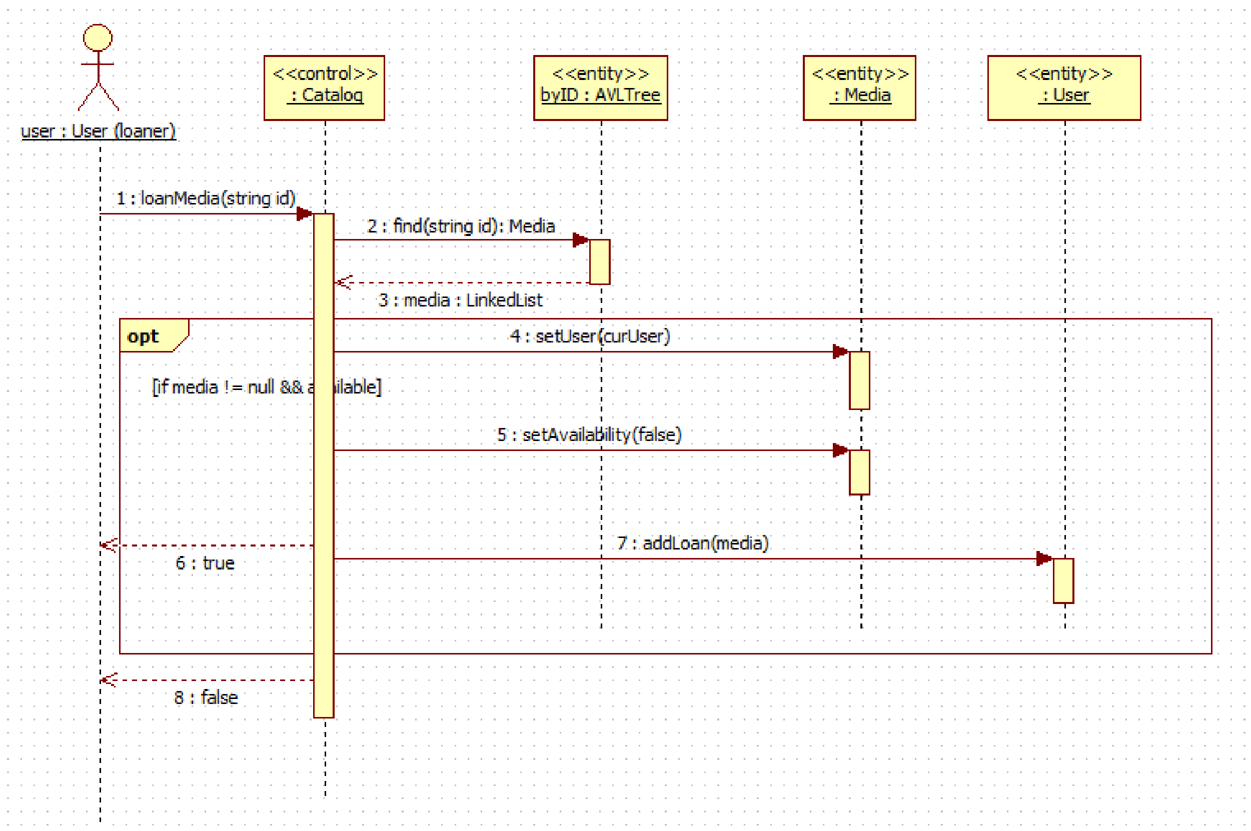
Klassdiagram



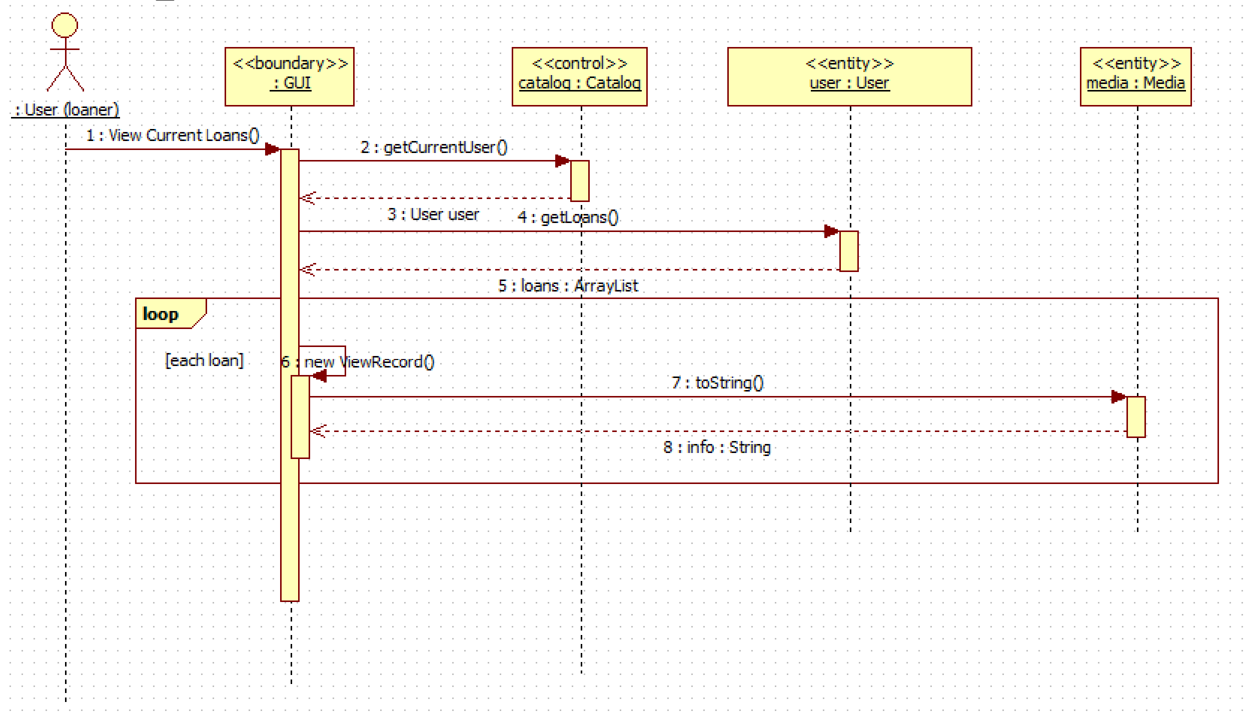
Sekvensdiagram
Uppstart av system



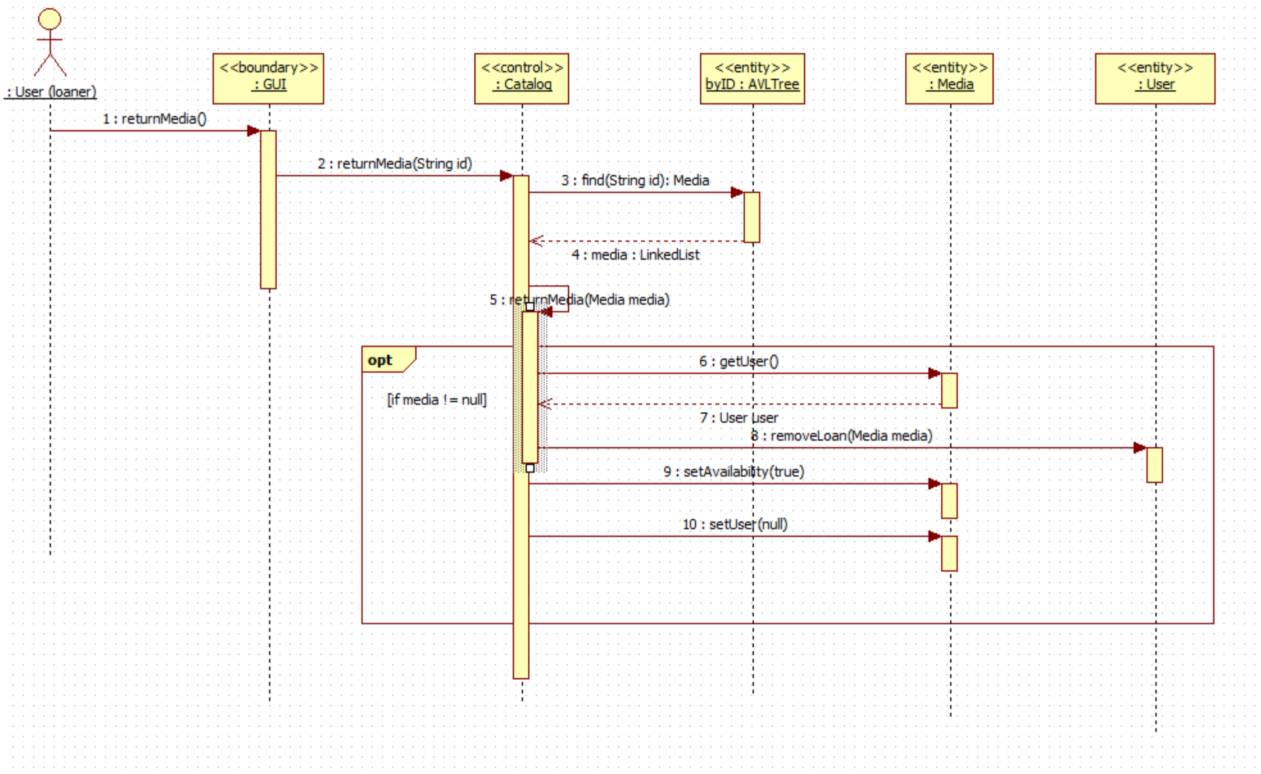
Utlåning av media



Listning av all lån



Återlämning



Källkod

Book

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package LibraryOOAD;

/**
 * A Book type media.
 *
 * @author Spellabbet
 */
public class Book extends Media {
    private String type;
    private String author;
    private String published;
    private String result;

    /**
     * Creates a book object.
     *
     * @param id Unique for this instance.
     * @param author Author of this book.
     * @param name Name of this book.
     * @param published Publication date of this book.
     */
    public Book(String id, String author, String name, String published){
        super(id);
        this.type = getClass().getName();
        this.author = author;
        this.name = name;
        this.published = published;
    }

    @Override
    public String toString(){
        result = "Type: " + this.type + "\nID: " + this.getId() + "\nName: " + this.name +
"\nAuthor: " + this.author + "\nPublished: " + this.published + "\nAvailable: " +
super.available;
        return result;
    }
}
```

Catalog

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package LibraryOOAD;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

import java.util.Arrays;
import java.util.Iterator;
import java.util.LinkedList;

import aVLTree.AVLTree;

/**
 * The catalog also known as the huge fucking awesome controller of the whole world. FUCK
EYAH!
 *
 * @author Spellabbet
 */
public class Catalog {

    protected AVLTree<User> users;
    protected User curUser;
    // TODO : index array, eek so much duplicate code!
    protected AVLTree<Media> byID;
    protected AVLTree<Media> byName;
    protected AVLTree<Media> byWild;
    // Used to split title into parts for wild index
    protected String nameRegex = " ";

    /**
     * Creates a new catalog instance.
     */
    public Catalog(){
        this.curUser = null;
        users = new AVLTree<User>();
        byID = new AVLTree<Media>();
        byName = new AVLTree<Media>();
        byWild = new AVLTree<Media>();
        importUsers();
        importMedia();

        this.users.add("admin", new User("admin", "admin", "none"));
    }
```

```
/**
 * Does an exact search.
 *
 * TODO : search(AVLTree index, String key) to avoid duplicate code.
 *
 * @param key What to search for.
 * @return A string representation of the result.
 */
public String search(String key){
   String result = "You've searched for '" + key + "'.";

   // by ID
   LinkedList<Media> mediaByID = byID.find(key);
   if (mediaByID != null)
   {
      Iterator<Media> iterator = mediaByID.iterator();
      while (iterator.hasNext())
      {
         Media media = iterator.next();
         result += "\n\n* Found by ID *\n" + media.toString();
      }
   }

   // by Name
   LinkedList<Media> mediaByName = byName.find(key);
   if (mediaByName != null)
   {
      Iterator<Media> iterator = mediaByName.iterator();
      while (iterator.hasNext())
      {
         Media media = iterator.next();
         result += "\n\n* Found by Name *\n" + media.toString();
      }
   }

   // by Wild
   LinkedList<Media> mediaByWild = byWild.find(key.toLowerCase());
   if (mediaByWild != null)
   {
      Iterator<Media> iterator = mediaByWild.iterator();
      while (iterator.hasNext())
      {
         Media media = iterator.next();
         result += "\n\n* Found by Wild *\n" + media.toString();
      }
   }

   return result;
}
```

```
    /**
     * Imports all users from a filé.
     *
     */
    public void importUsers(){
        String person;
        String[] parts;
        System.out.println("CWD: " + System.getProperty("user.dir"));
        String path = "data/Lantagare.txt";
        try {
            BufferedReader br = new BufferedReader(new FileReader(path));
//          Example:
//          891216-1111;Harald Svensson;040-471024
            person = br.readLine();
            while(person != null){
                parts = person.split(";");
                this.users.add(parts[1], new User(parts[0], parts[1], parts[2]));
                person = br.readLine();
            }
        } catch (IOException e) {
            System.out.println("File not found.");
        }
    }

    /**
     * Imports all media from a filé.
     *
     */
    public void importMedia(){
        String media;
        String[] parts;
        System.out.println("CWD: " + System.getProperty("user.dir"));
        String path = "data/Media.txt";
        try {
            BufferedReader br = new BufferedReader(new FileReader(path));
//          Examples:
//          Bok;427769;Deitel;Java how to program;2005
//                      Dvd;635492;Nile City 105,6;1994;Robert Gustavsson;Johan Rheborg;Henrik
Schyffert
            media = br.readLine();
            while(media != null){
                parts = media.split(";");
                if(parts[0].equals("Bok")){
                    Book book = new Book(parts[1], parts[2], parts[3], parts[4]);
                    addMedia(book);
                }else{
                    String[] actors = Arrays.copyOfRange(parts, 4, parts.length);
                    DVD dvd = new DVD(parts[1], parts[2], parts[3], actors);
                    addMedia(dvd);
                }
            media = br.readLine();
        }
    } catch (IOException e) {
        System.out.println("File not found.");
    }
}

/**
 * Adds the given media to 3 different AVLtrees.
 *
 * @param media The media to be added.
 */
public void addMedia(Media media){
    System.out.println("addMedia(" + media.toString() + ")");
    // by ID
    byID.add(media.getId(), media);

    // by Name
    byName.add(media.getName(), media);

    // by Wild (name)
    String[] names = media.getName().split(nameRegex);
    for ( int i = 0; i < names.length; ++i ) {
        byWild.add(names[i].toLowerCase(), media);
    }
}

/**
 * Login of given user to the system.
 *
 * @param name Name to check.
 * @param persNr Personal security number to check.
 * @return If login was a success or not.
 */
public boolean logIn(String name, String persNr) {
    LinkedList<User> temp = users.find(name);
    if(temp.size() > 0){
        if( temp.getFirst().getPersNr().equals(persNr)){
            curUser = temp.getFirst();
            return true;
        }
    }
    return false;
}

/**
 * Returns media.
 *
 * @param media The media to be returned.
 * @return If the return was a success or not.
 */
public boolean returnMedia(Media media){
    if(media.getUser().equals(curUser)){
        media.getUser().removeLoan(media);
        media.setUser(null);
        media.setAvailability(true);
        return true;
    }
    return false;
}

/**
 * Returns media.
 *
 * @param id The id to find the media.
 * @return If the return was a success or not.
*/
public boolean returnMedia(String id){
    boolean result = false;
    LinkedList<Media> medias = byID.find(id);
    if(medias.size() == 0){
        return false;
    } else {
        Media media = medias.getFirst();
        if(media != null && !media.available){
            result = returnMedia(media);
        }
    }
    return result;
}

/**
 * Loans a media.
 *
 * @param media The media to be loaned.
 */
public void loanMedia(Media media){
    media.setUser(curUser);
    media.setAvailability(false);
    curUser.addLoan(media);
}


/**
 * Loans a media.
 *
 * @param id The id of a media to be loaned.
 * @return If the loan was a success or not.
 */
public boolean loanMedia(String id){
        LinkedList<Media> medias = byID.find(id);
        if(medias.size() == 0){
            return false;
        } else {
            Media media = medias.getFirst();
            if(media != null && media.available){
                loanMedia(media);
                return true;
            }
        }
        return false;
    }
    /**
     * Gets the current user logged in.
     *
     * @return The current user logged in.
     */
    public User getCurrentUser(){
        return this.curUser;
    }

    /**
     * Starts the system.
     *
     * @param args The argument.
     */
    public static void main(String[] args) {
        Catalog catalog = new Catalog();
        GUI gui = new GUI(catalog);
        }
}
```

DVD

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package LibraryOOAD;


/**
 * A DvD type media.
 *
 * @author Spellabbet
 */
public class DVD extends Media {
   private String type;
   private String published;
   private String[] actors;
   private String result;
   private String actorsString;

   /**
    * Creates a DVD object.
    *
    * @param id Unique for this instance
    * @param name Title of this media.
    * @param published Date of publication.
    * @param actors Actors associated with media.
    */
   public DVD(String id, String name, String published, String[] actors){
      super(id);
      this.type = getClass().getName();
      this.name = name;
      this.published = published;
      this.actors = actors;
   }

   protected String actorsToString(){
      actorsString = "";
      for(int i = 0; i<actors.length ; i++){
         actorsString = actorsString + actors[i] + ", ";
      }
      return actorsString;
   }

   @Override
   public String toString(){
        result = "Type: " + this.type + "\nID: " + this.getId() + "\nName: " + this.name +
"\nActors: " + actorsToString() + "\nPublished: " + this.published + "\nAvailable: " +
super.available;
        return result;
    }
}
```

GUI

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package LibraryOOAD;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.TextField;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.Timer;

/**
 * A GUI.
 *
 * @author Spellabbet
 */
public class GUI extends JFrame implements ActionListener, KeyListener {
   //Log in
   private JPanel pLogIn = new JPanel(new GridLayout(4, 1));
   private JLabel lLogIn = new JLabel("Please log in", (int) CENTER_ALIGNMENT);
   private JFrame fLogIn = new JFrame();
   private JButton bLogIn = new JButton("Log in");
   private JTextField name = new JTextField("admin");
   //private JTextField persNr = new JTextField("admin");
```

```
private JPasswordField persNr = new JPasswordField("admin");
//private JTextField name = new JTextField("Your name");
//private JTextField persNr = new JTextField("Your Social Security Number");
private Catalog catalog;

//library window, when logged in
private JButton search = new JButton("Search");
private JButton viewLoans = new JButton("View current loans");
private JButton newLoan = new JButton("New loan");
private JButton returnMedia = new JButton("Return media");
private JButton logOut = new JButton("Log out");
private JPanel knappar = new JPanel(new GridLayout(5,1));
private JPanel view = new JPanel(new GridLayout(1,1));

// all different view panels
private JPanel panelSearch = new JPanel(new BorderLayout());
private JPanel searchBar = new JPanel();
private JTextField searchInput = new JTextField("input");
private JButton bSearch = new JButton("Search");
private JTextArea searchResultPanel = new JTextArea();
private JScrollPane scroll = new JScrollPane(searchResultPanel);

private ViewLoansPanel panelViewLoans;
private NewLoan panelNewLoan = new NewLoan();
private ReturnLoan panelReturnMedia = new ReturnLoan();

protected WelcomePanel panelWelcome;

/**
 * Creates a new instance of GUI.
 *
 * @param catalog The catalog to control.
 */
public GUI(Catalog catalog){
   this.catalog = catalog;

   panelWelcome = new WelcomePanel();

   // settings for login panel data
   fLogIn.setTitle("HackerLibary - Log in - v3.0");
   fLogIn.setBounds(400, 400, 400, 200);
   fLogIn.setResizable(false);
   fLogIn.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   pLogIn.setBounds(400, 400, 400, 200);
   pLogIn.setBackground(Color.BLACK);
   name.setSize(100, 50);
   name.setBackground(Color.BLACK);
   persNr.setBackground(Color.BLACK);
   name.setForeground(Color.GREEN);
   persNr.setForeground(Color.GREEN);
```

```
lLogIn.setForeground(Color.GREEN);
bLogIn.setForeground(Color.GREEN);
bLogIn.setBackground(Color.BLACK);
pLogIn.add(lLogIn);
pLogIn.add(name);
pLogIn.add(persNr);
pLogIn.add(bLogIn);
fLogIn.add(pLogIn);
name.setCaretColor(Color.red);
persNr.setCaretColor(Color.red);
bLogIn.addActionListener(this);
fLogIn.setVisible(true);

// settings for library view
this.setBounds(13+37, 13+37, 500, 500);
this.setTitle("HackerLibary - v3.0");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLayout(new BorderLayout());
knappar.add(search);
knappar.add(viewLoans);
knappar.add(newLoan);
knappar.add(returnMedia);
knappar.add(logOut);
view.setForeground(Color.GREEN);
search.setForeground(Color.GREEN);
viewLoans.setForeground(Color.GREEN);
newLoan.setForeground(Color.GREEN);
returnMedia.setForeground(Color.GREEN);
logOut.setForeground(Color.GREEN);
view.setBackground(Color.BLACK);
search.setBackground(Color.BLACK);
viewLoans.setBackground(Color.BLACK);
newLoan.setBackground(Color.BLACK);
returnMedia.setBackground(Color.BLACK);
logOut.setBackground(Color.BLACK);

this.add(BorderLayout.WEST, knappar);
this.add(BorderLayout.CENTER, view);

search.addActionListener(this);
viewLoans.addActionListener(this);
newLoan.addActionListener(this);
returnMedia.addActionListener(this);
logOut.addActionListener(this);

// Search panel
searchInput.setPreferredSize(new Dimension(100, 20));
bSearch.setPreferredSize(new Dimension(100, 20));
searchInput.setForeground(Color.GREEN);
searchInput.setBackground(Color.BLACK);
```

```
        bSearch.setForeground(Color.GREEN);
        bSearch.setBackground(Color.BLACK);
        searchBar.setBackground(Color.BLACK);
        searchResultPanel.setForeground(Color.GREEN);
        searchResultPanel.setBackground(Color.BLACK);
        searchInput.addKeyListener(this);
        searchBar.add(searchInput);
        searchBar.add(bSearch);
        panelSearch.add(BorderLayout.CENTER, scroll);
        panelSearch.add(BorderLayout.NORTH, searchBar);
        bSearch.addActionListener(this);

        // default view
        setView(panelWelcome);
    }

    /**
     * Converts an array or chars to string.
     *
     * @param chars The array of characters to be converted.
     * @return The results of the conversion.
     */
    protected String charArrayToString(char[] chars)
    {
        String returnValue = "";
        for ( int i = 0; i < chars.length; ++i ) {
            returnValue += chars[i];
        }
        return returnValue;
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == bLogIn) {
            //if (catalog.logIn(name.getText(), persNr.getText())) {
            System.out.println('g' + "persNr" + persNr.getPassword().length);
            if (catalog.logIn(name.getText(), charArrayToString(persNr.getPassword()))) {
                this.setVisible(true);
                fLogIn.setVisible(false);
                setView(panelWelcome);
            } else if (name.getText().equals("din mamma")) {
                lLogIn.setText("ERROR! NUKE HAS BEEN SENT TO YOUR MOMS
LOCATION!");
            } else {
                lLogIn.setText("ERROR! NUKE SENT TO YOUR LOCATION!");
            }
        } else if (e.getSource() == search) {
            setSelected(search);
            setView(panelSearch);
        } else if (e.getSource() == viewLoans) {
            setSelected(viewLoans);
        setView(new ViewLoansPanel());
    } else if (e.getSource() == newLoan) {
        setSelected(newLoan);
        setView(panelNewLoan);
    } else if (e.getSource() == returnMedia) {
        setSelected(returnMedia);
        setView(panelReturnMedia);
    } else if (e.getSource() == logOut) {
        catalog.curUser = null;
        this.setVisible(false);
        fLogIn.setVisible(true);
        setSelected(null);
    } else if (e.getSource() == bSearch) {
        search();
    }
}

/**
 * Sets the panel to be viewed.
 *
 * @param panel The panel to be viewed.
 */
public void setView(JPanel panel){
    view.removeAll();

    if (panel == null)
        view.add(panelWelcome);
    else
        view.add(panel);

    view.revalidate();
    view.repaint();
}

protected void search(){
    searchResultPanel.setText(catalog.search(searchInput.getText()));
    scroll.revalidate();
}

/**
 * {@inheritDoc}
 * @param e
 * @see KeyListener#keyTyped(KeyEvent)
 */
public void keyTyped(KeyEvent e)
{
}

/**
 * {@inheritDoc}
 * @param e
 * @see KeyListener#keyPressed(KeyEvent)
 */
public void keyPressed(KeyEvent e)
{
}

/**
 * {@inheritDoc}
 * @param e
 * @see KeyListener#keyReleased(KeyEvent)
 */
public void keyReleased(KeyEvent e)
{
   search();
}

/**
 * Select the button given and set all others to unselected.
 *
 * @param button The button to be selected.
 */
protected void setSelected(JButton button)
{
   if (search == button)
      selected(search, true);
   else
      selected(search, false);

   if (viewLoans == button)
      selected(viewLoans, true);
   else
      selected(viewLoans, false);

   if (newLoan == button)
      selected(newLoan, true);
   else
      selected(newLoan, false);

   if (returnMedia == button)
      selected(returnMedia, true);
   else
      selected(returnMedia, false);
}

/**
 * Sets the selection status of the button.
 *
 * @param button The botton to change status.
 * @param select The status.
 */
protected void selected(JButton button, boolean select)
{
   if (select)
   {
      button.setBackground(new Color(80,240,14));
      button.setForeground(Color.BLACK);
   }
   else
   {
      button.setBackground(Color.BLACK);
      button.setForeground(Color.GREEN);
   }
}

protected class WelcomePanel extends JPanel implements ActionListener
{
   protected int width, height;
   protected Graphics g;
   protected Font font;
   protected int currentColor, initialColor;
   protected Color[] colors;
   protected Timer matrixEffect;

   protected WelcomePanel()
   {
      setBackground(Color.BLACK);
      setForeground(Color.GREEN);
      font = new Font("monospaced", Font.PLAIN, 12);

      // some nice green colors
      colors = new Color[]
      {
         new Color(50, 255, 50),
         new Color(60, 255, 60),
         new Color(70, 255, 70),
         new Color(80, 255, 80),
         new Color(90, 255, 90),
         new Color(100, 255, 100),
         new Color(110, 255, 110),
         new Color(120, 255, 120),
         new Color(130, 255, 130),
         new Color(140, 255, 140),
         new Color(150, 255, 150),
         new Color(160, 255, 160),
         new Color(170, 255, 170),
         new Color(180, 255, 180),
         new Color(190, 255, 190),
         new Color(200, 255, 200),
         new Color(210, 255, 210),
      };

      matrixEffect = new Timer(1000/18, this);

      // TODO : ? Start and stop when appropriate.
      matrixEffect.start();
   }

   /**
    * {@inheritDoc}
    * @see JComponent#paintComponent(Graphics)
    */
   protected void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      this.g = g;
      width = getWidth();
      height = getHeight();

      nextColor();
      drawWelcomeText();
   }

   protected Color nextColor()
   {
      if (++currentColor >= colors.length)
         currentColor = 0;
      return colors[currentColor];
   }

   protected Color nextInitialColor()
   {
      if (--initialColor < 0)
         initialColor=colors.length-1;
      currentColor = initialColor;
      return colors[currentColor];
   }

   protected void drawWelcomeText()
   {
      g.setFont(font);

      g.setColor(nextInitialColor());
      g.drawString("Welcome", 20, 20);
      g.setColor(nextColor());
      g.drawString(" elcome", 20, 30);
      g.setColor(nextColor());
      g.drawString("  lcome", 20, 40);
      g.setColor(nextColor());
      g.drawString("   come", 20, 50);
      g.setColor(nextColor());
      g.drawString("     ome", 20, 60);
      g.setColor(nextColor());
      g.drawString("      me", 20, 70);
      g.setColor(nextColor());
      g.drawString("       e", 20, 80);

      g.setColor(nextColor());
      g.drawString("* Awesome Terminator Matrix Library System *", 20, 190);
      g.setColor(nextColor());
      g.drawString("Created by:", 20, 205);
      g.setColor(nextColor());
      g.drawString(" - Daniel Aladics", 20, 220);
      g.setColor(nextColor());
      g.drawString(" - Hannes Landstedt", 20, 230);
      g.setColor(nextColor());
      g.drawString(" - Jens Persson", 20, 250);
      g.setColor(nextColor());
      g.drawString(" - Magnus Grönvall", 20, 240);
   }

   /**
    * {@inheritDoc}
    * @param e
    * @see ActionListener#actionPerformed(ActionEvent)
    */
   public void actionPerformed(ActionEvent e)
   {
      repaint();
   }
}

protected class ViewLoansPanel extends JPanel
{

   public ViewLoansPanel(){

      setBackground(Color.BLACK);
      setForeground(Color.GREEN);
      User user = catalog.getCurrentUser();
      ArrayList<Media> loans = user.getLoans();
      int size = loans.size();
      setLayout(new GridLayout(size, 1));
      for(int i = 0; i<size; i++){
         add(new ViewRecord(loans.get(i)));
      }

   }

   protected class ViewRecord extends JPanel implements ActionListener{
      Media media;

      public ViewRecord(Media mediaIn){
         media = mediaIn;
         setBackground(Color.BLACK);
         setForeground(Color.GREEN);
         setLayout(new GridLayout(1, 2));
         JTextArea loans = new JTextArea(media.toString());
         loans.setBackground(Color.BLACK);
         loans.setForeground(Color.GREEN);
         add(loans);
         JButton returnButton = new JButton("Return");
         returnButton.setBackground(Color.BLACK);
         returnButton.setForeground(Color.GREEN);
         returnButton.addActionListener(this);
         add(returnButton);
      }
      public void actionPerformed(ActionEvent e) {
         boolean result = catalog.returnMedia(media);
         if(result){
             setSelected(viewLoans);
             setView(new ViewLoansPanel());
         }
      }
   }
}

protected class NewLoan extends JPanel implements ActionListener{
   private JTextField idField = new JTextField("Insert ID");
   private JButton doLoan = new JButton("Loan");

   public NewLoan(){
      setForeground(Color.GREEN);
      setBackground(Color.BLACK);
      idField.setForeground(Color.GREEN);
      idField.setBackground(Color.BLACK);
      idField.setPreferredSize(new Dimension(100, 20));
      doLoan.setForeground(Color.GREEN);
      doLoan.setBackground(Color.BLACK);
      setLayout(new FlowLayout());
      doLoan.addActionListener(this);
      add(idField);
      add(doLoan);
   }

   public void actionPerformed(ActionEvent e) {
      boolean result = catalog.loanMedia(idField.getText());
      idField.setText("Failed.");
      if(result){
         setSelected(viewLoans);
         setView(new ViewLoansPanel());
         idField.setText("Insert ID");
      }
   }
}

protected class ReturnLoan extends JPanel implements ActionListener{
   private JTextField idField = new JTextField("Insert ID");
   private JButton doLoan = new JButton("Return");

   public ReturnLoan(){
      setForeground(Color.GREEN);
      setBackground(Color.BLACK);
      idField.setForeground(Color.GREEN);
      idField.setBackground(Color.BLACK);
      idField.setPreferredSize(new Dimension(100, 20));
      doLoan.setForeground(Color.GREEN);
      doLoan.setBackground(Color.BLACK);
      setLayout(new FlowLayout());
      doLoan.addActionListener(this);
      add(idField);
      add(doLoan);
   }

   public void actionPerformed(ActionEvent e) {
      boolean result = catalog.returnMedia(idField.getText());
      idField.setText("Failed.");
      if(result){
         setSelected(viewLoans);
         setView(new ViewLoansPanel());
         idField.setText("Insert ID");
      }
   }
}
}
```

Media

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package LibraryOOAD;


/**
 * General Media object.
 *
 * @author Spellabbet
 */
public abstract class Media {
   protected String id;
   protected boolean available;
   protected User loaner;
   protected String name;

   /**
    * Creates new instance of Media.
    *
    * @param id Unique for this Media object.
    */
   public Media( String id ) {
      this.id = id;
      this.available = true;
      this.loaner = null;
      name = "unnamed";
   }

   /**
    * Returns ID of Media object.
    *
    * @return The ID.
    */
   public String getId() {
      return id;
   }

   /**
    * Gets the name for this instance.
    *
    * @return The name.
    */
   public String getName()
   {
      return this.name;
```

```
}

/**
 * Sets the name for this instance.
 *
 * @param name The name.
 */
public void setName(String name)
{
   this.name = name;
}

/**
 * Gets the availability of this instance.
 *
 * @return Available or not.
 */
public boolean getAvailability(){
   return this.available;
}

/**
 * Gets the the current loaner of this instance.
 *
 * @return The loaner.
 */
public User getUser(){
   return this.loaner;
}

/**
 * Sets the loaner of this instance.
 *
 * @param user The loaner.
 */
public void setUser(User user){
   this.loaner = user;
}

/**
 * Sets the availbility of this instance.
 *
 * @param avail The availability.
 */
public void setAvailability(boolean avail){
   if(avail){
      this.available = true;
   }else{
      this.available = false;
   }
```

```
    }

    /**
     * {@inheritDoc}
     * @see Object#equals()
     */
    @Override public boolean equals( Object obj ) {
        Media media = (Media)obj;
        return id.equals( media.getId() );
    }

    /**
     * {@inheritDoc}
     * @see Object#toString()
     */
    public String toString()
    {
        return "ID='" + id + "', name='" + name + "'";
    }
}
```

User

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package LibraryOOAD;

import java.util.*;

/**
 * A User.
 *
 * @author Spellabbet
 */
public class User {
    private String persNr;
    private String name;
    private String tele;
    private ArrayList<Media> loansArray;

    /**
     * Creates a new instance of User.
     *
     * @param nr Personal security number of this user.
     * @param name Name of this user.
     * @param tele Telephone number to this user.
     */
    public User(String nr, String name, String tele){
        this.persNr = nr;
        this.name = name;
        this.tele = tele;
        this.loansArray = new ArrayList<Media>();
    }

    /**
     * Gets this user's personal security number.
     *
     * @return The personal security number.
     */
    public String getPersNr(){
        return this.persNr;
    }
    /**
     * Gets this user's name.
     *
     * @return The name.
     */
    public String getName(){
        return this.name;
    }
    /**
     * Gets this user's telephone number.
     *
     * @return The telephone number.
     */
    public String getTele(){
        return this.tele;
    }
     /**
      * Removes a media from the current loans of this instance.
      *
      * @param media The media to be removed from the users loans.
      */
     public void removeLoan(Media media){
        loansArray.remove(media);
    }
    /**
     * Adds a media to the current loans of this instance.
     *
     * @param media The media to be added to the users loans.
     */
    public void addLoan(Media media){
        loansArray.add(media);
    }
    /**
     * Gets all of this user's loans.
     *
     * @return All of the loans.
     */
    public ArrayList<Media> getLoans(){
        return loansArray;
    }
}
```

aVLTree.AVLTree

```
package aVLTree;

import java.util.LinkedList;

/**
 * The AVLTree data structure.
 *
 * @author Spellabbet
 * @param <Value>
 */
public class AVLTree<Value>
{
   int size;

   AVLTreeNode<Value> root;

   // TODO : ? move to node class or create iterator class
   /**
    * A pointer used for iteration.
    *
    */
   public AVLTreeNode<Value> pointer;

   /**
    * {@inheritDoc}
    * @param key
    * @param value
    * @see Dictionary#add(String,Value)
    */
   public void add(String key, Value value)
   {
      if (root == null)
         root = new AVLTreeNode<Value>(key, value);
      else
      {
         AVLTreeNode<Value> current = root;
         while (true)
         {
            // allow for duplicate entries
            // TODO : Test how this effects the speed of adding.
            int difference = key.compareTo(current.key);
            if (difference == 0)
            {
               current.values.add(value);
               break;
            }
            else if (difference < 0)
            {
                if (current.left == null)
                {
                    current.left = new AVLTreeNode<Value>(key, value);
                    current.left.parent = current;
                    fixHeight(current);
                    balance(current);
                    break;
                }
                else
                    current = current.left;
            }
            else // difference > 0
            {
                if (current.right == null)
                {
                    current.right = new AVLTreeNode<Value>(key, value);
                    current.right.parent = current;
                    fixHeight(current);
                    balance(current);
                    break;
                }
                else
                    current = current.right;
            }
        }
    }

    size++;
}

protected void balance(AVLTreeNode<Value> node)
{
    while (node != null)
    {
        // check if right rotate is needed
        if (getHeight(node.left) - getHeight(node.right) > 1)
        {
            // is left child is right heavy, rotate it left first (double right)
            if (getHeight(node.left.right) > getHeight(node.left.left))
                rotateLeft(node.left);

            rotateRight(node);
        }
        // left rotate
        else if (getHeight(node.right) - getHeight(node.left) > 1)
        {
            // double left
            if (getHeight(node.right.left) > getHeight(node.right.right))
                rotateRight(node.right);
            rotateLeft(node);
        }

        node = node.parent;
    }
}

/**
 * {@inheritDoc}
 * @param key
 * @return
 * @see Dictionary#remove(String)
 */
public LinkedList<Value> remove(String key)
{
    AVLTreeNode<Value> node = removeNode(key);

    if (node == null)
        return null;

    return node.values;
}

/**
 * {@inheritDoc}
 * @param key
 * @return
 * @see Dictionary#remove(String)
 */
protected AVLTreeNode<Value> removeNode(String key)
{
    //System.out.println("removeNode():");

    AVLTreeNode<Value> node = (AVLTreeNode<Value>)findNode(key);
    if (node != null)
    {
        //   O
        //    \
        if (node.right == null)
        {
            // assuming balanced tree
            if (node == root)
            {
                root = node.left;
                if (node.left != null)
                    node.left.parent = root;
            }
            else
            {
                if (node.parent.right == node)
                    node.parent.right  = node.left;
                else
                    node.parent.left  = node.left;

                if (node.left  != null)
                    node.left.parent  = node.parent;

                fixHeight(node.parent);
                balance(node.parent);
            }
        }

        //   O
        //  / \
        //     o
        else if (node.left  == null)
        {
            // assuming  balanced  tree
            if (node == root)
            {
                root = node.right;
                root.parent  = null;
            }
            else
            {
                node.right.parent  = node.parent;
                if (node.parent.right  == node)
                    node.parent.right  = node.right;
                else
                    node.parent.left  = node.right;

                fixHeight(node.parent);
                balance(node.parent);
            }
        }

        //   O
        //  / \
        // o   o
        else
        {
            //   O
            //  / \
            // o   o
            //  \
            if (node.left.right  == null)
            {
                if (node == root)
                    root = node.left;
                else if (node.parent.right  == node)
```

```
                node.parent.right = node.left;
            else
                node.parent.left = node.left;
            node.left.right = node.right;
            node.left.parent = node.parent;
            node.right.parent = node.left;
            fixHeight(node.left);
            balance(node.left);
        }
        //   O
        //  / \
        // o   o
        //  \
        //   o
        else
        {
            AVLTreeNode<Value> newRoot = getRightMost(node.left);
            AVLTreeNode<Value> toBalance = newRoot.parent;
            if (newRoot.left != null)
                newRoot.left.parent = newRoot.parent;
            newRoot.parent.right = newRoot.left;

            newRoot.left = node.left;
            newRoot.left.parent = newRoot;
            newRoot.right = node.right;
            newRoot.right.parent = newRoot;
            if (node == root)
                root = newRoot;
            else if (node.parent.right == node)
                node.parent.right = newRoot;
            else
                node.parent.left = newRoot;
            newRoot.parent = node.parent;

            fixHeight(toBalance);
            balance(toBalance);
        }
    }
  }

  size--;
  return node;
}


protected AVLTreeNode<Value> getRightMost(AVLTreeNode<Value> node)
{
  AVLTreeNode<Value> rightMost = node;
  while (rightMost.right != null)
    rightMost = rightMost.right;
```

```
   return rightMost;
}


protected int getHeight(AVLTreeNode<Value> node)
{
   return node == null ? 0 : node.height;
}


protected void fixHeight(AVLTreeNode<Value> node)
{
   while (node != null)
   {
      int max = Math.max(getHeight(node.left), getHeight(node.right));
      node.height = max + 1;
      node = node.parent;
   }
}

protected AVLTreeNode<Value> findNode(String key)
{
   AVLTreeNode<Value> node = (AVLTreeNode<Value>)root;
   while (node != null)
   {
      if (key.compareTo(node.key) < 0)
         node = node.left;
      else if (key.compareTo(node.key) > 0)
         node = node.right;
      else
         return node;
   }
   return null;
}

/**
 * {@inheritDoc}
 * @param key
 * @return
 * @see Dictionary#find(String)
 */
public LinkedList<Value> find(String key)
{
   AVLTreeNode<Value> node = (AVLTreeNode<Value>)findNode(key);
   if (node == null)
      return new LinkedList<Value>();
   else
      return node.values;
   // TODO : ? Return array rather than list. It's a shame you can't
```

```
    // create generic arrays in Java.
}

/**
 * {@inheritDoc}
 * @return
 * @see Dictionary#size()
 */
public int size()
{
    return size;
}

protected AVLTreeNode<Value> getFirst()
{
    //System.out.println("getFirst():");
    AVLTreeNode<Value> first = root;

    if (first == null)
    {
        //System.out.println("getFirst(): null");
        pointer = null;
        return null;
    }

    while (first.left != null)
        first = first.left;

    pointer = first;
    return first;
}

protected AVLTreeNode<Value> getNext()
{
    pointer = getNext(pointer);
    return pointer;
}

/**
 * Get next item, in-order. Can return null if empty tree or lonely node.
 *
 * {@inheritDoc}
 * @param currentNode
 * @return
 * @see Dictionary#getNext()
 */
protected AVLTreeNode<Value> getNext(AVLTreeNode<Value> currentNode)
{
    return getNext(root, currentNode);
}
```

```
  /**
   * Get next item, in-order. Can return null if empty tree or lonely node.
   *
   * {@inheritDoc}
   * @param root
   * @param currentNode
   * @return
   * @see Dictionary#getNext()
   */
  protected AVLTreeNode<Value> getNext(AVLTreeNode<Value> root,
AVLTreeNode<Value> currentNode)
  {
    AVLTreeNode<Value> next;

    // no node
    if (currentNode == null)
    {
      //System.out.println("getNext(): Node is 'null', can't get next.");
      return null;
    }
    else
    {
      //System.out.println("getNext(): " + currentNode.toString());
    }

    // on right, get left most or self
    //   O
    //    \
    //     o
    //    /
    //   x
    if (currentNode.right != null)
    {
      //System.out.println("getNext(): >");
      next = currentNode.right;
      while (next.left != null)
      {
        //System.out.println("getNext():<");
        next = next.left;
      }
      return next;
    }

    // first right parent
    //   x
    //  /
    // o
    //  \
    //   O
    next = currentNode;
    while (next != root)
    {
        if (next == next.parent.left)
        {
            //System.out.println("getNext():parent was next yay..");
            return next.parent;
        }
        next = next.parent;
    }

    // no next
    //    x
    return null;
}

protected void rotateRight(AVLTreeNode<Value> node)
{
    // in case we do manual rotate right in GUI
    if (node.left == null)
        return;

    AVLTreeNode<Value> newRoot = node.left;
    newRoot.parent = node.parent;
    if (node == root)
        root = newRoot;
    else if (node.parent.left == node)
        node.parent.left = newRoot;
    else
        node.parent.right = node.left;

    node.left = newRoot.right;
    if (node.left != null)
        node.left.parent = node;
    newRoot.right = node;
    node.parent = newRoot;

    fixHeight(node);
}

protected void rotateLeft(AVLTreeNode<Value> node)
{
    // in case we do manual rotate left in GUI
    if (node.right == null)
        return;

    AVLTreeNode<Value> newRoot = node.right;
    newRoot.parent = node.parent;

    if (node == root)
```

```
            root = newRoot;
        else if (node.parent.right == node)
            node.parent.right = newRoot;
        else
            node.parent.left = newRoot;

        node.right = newRoot.left;
        if (node.right != null)
            node.right.parent = node;
        newRoot.left = node;
        node.parent = newRoot;

        fixHeight(node);
    }
}
```

aVLTree.AVLTreeNode

```
package aVLTree;

import java.util.LinkedList;

/**
 * A node beloning to the AVLTree.
 *
 * @author Spellabbet
 * @param <Value>
 */
public class AVLTreeNode<Value>
{
   public String key;
   public LinkedList<Value> values;
   public int height = 1;
   public AVLTreeNode<Value> parent;
   public AVLTreeNode<Value> left;
   public AVLTreeNode<Value> right;

   /**
    * Creates a new node instance.
    *
    * @param key Unique key for this instance.
    * @param value Value to be added to this instance.
    */
   public AVLTreeNode(String key, Value value)
   {
      this.key = key;
      values = new LinkedList<Value>();
      values.add(value);
   }

   @Override
   public String toString()
   {
      String parent;
      String self;
      String left;
      String right;

      if (this.parent == null)
         parent = "null";
      else
         parent = this.parent.key;

      self = key;

      if (this.left == null)
```

```
            left  = "null";
        else
            left  = this.left.key;

        if (this.right  == null)
            right  = "null";
        else
            right  = this.right.key;


        return "SELF='" + self + "', <<<'" + left + "<<< >>>" + right + ">>> , ^^^" + parent +
"^^^, height='"  + height  + "'";
        //return "parent='" + parent + "', self='" + self + "', left='" + left + "', right='" + right + "',
height='"  + height  + "'";
    }
}
```

aVLTree.Dictionary

```
package aVLTree;

/**
 * Represents a dictionary.
 *
 * @author Spellabbet
 * @param <Value> The type of value contained in the dictionary.
 */
public interface Dictionary<Value>
{
    /**
     * Add to the dictionary.
     *
     * @param key The key for finding the object.
     * @param value The value for this key.
     */
    public void add(String key, Value value);
    /**
     * Remove from the dictionary.
     *
     * @param key The key to find the object to remove.
     * @return Item removed.
     */
    public Value remove(String key);
    /**
     * Finds a item.
     *
     * @param key Key to find.
     * @return Value of this belonging having the Key.
     */
    public Value find(String key);
}
```