UIC CS FACULTY SCHEDULER

BRYAN SPAHR

Introduction

Why did I choose undergraduate research? Well, I always relish the opportunity to set my own schedule (sometimes to my own detriment). I also liked the idea of a semester-long side project with Pat Troy. We've talked about various aspects of CS and coding in the past, especially feedback on the new CS curriculum and the curriculum as a whole.

When I came to Troy and told him we were doing research, we once again talked about the CS curriculum; this time, about possible gaps in subjects covered that could be filled in the future. One of these gaps is the topic of Web Development. Most CS students end up teaching themselves some web development at some point in their UIC career, but there isn't any academia related to the topic. With the possibility in mind of a future tech elective, I was tasked with researching what materials such a class would need and, most importantly, whether or not the known UIC-hosted web service, people.uic.edu, would suffice for an entire class of students.

It didn't take me long to realize that the best resources for such a course were all over the web, in fact, there was no need to create new materials for the course given the wide range of stuff available. I also quickly concluded after a little research and questioning that people.uic.edu would easily support a full class of students.

What is this?

Well, I had to do something to fulfil my research requirements for credit.

As it turns out, Professor Troy is in charge of choosing which CS professors teach each class every semester for the following semester. As such, he suggested this project as a potentially viable solution to this manual task by choosing professors using a GUI application.

This application does just that. Also included is a GUI editor for updating which classes each CS instructor teaches. Despite a relatively simple GUI, this project features an attention to detail and professional object oriented design throughout code. My main goal for this project was to make an application that could add value to what Professor Troy does, as opposed to just making something that will never see the light of day.

How it works

If you haven't already, download the zip file release here: http://bit.ly/1C18fQN

Inside you should have an executable JAR file (note: requires Java Runtime to run) and a folder called data that contains several plaintext files. The JAR file and data folder need to be in the same directory in order to use the pre-loaded data provided in the zip file. Alternatively, you can put the JAR file in any directory you want and the application will create new data files for you to input your own data from scratch.

After launching the JAR file, you should see a small window with two buttons, Faculty Editor and Generate Instructors.

Faculty Editor

Clicking on the left button in the main window opens the editor. You'll see a dropdown to select an instructor's name. Every time you switch to a different instructor, the application will pre-load the data for that instructor into his/her course slots. Each instructor has up to 8 slots for commonly taught courses, which often aren't all used, so the blank slots can just be left as "n/a". When finished editing/adding courses for a particular instructor, make sure to click the save button or your changes will be lost. You should see console output as the application saves the data to both the application and the external data files.

Generating Instructors

Clicking on the right button in the main window calculates all instructors (from the available ones listed for each course) for the following semester. The results are displayed in the Results Window, and include all CS courses in the data, even those without any listed instructors. Also in this window is a button for saving out the results as a plaintext file for long-term purposes.

Downsides/Future Improvements

- I was originally going to have the main window include check boxes for each course so you could check only the courses being offered the following semester, making for a much cleaner result window. This functionality has been shelved for the time being.
- I was playing with the concept of putting a "reject" button next to each instructor pick in the results window. This would mean the user could reject multiple picks out of the original calculation and recalculate leaving the other picks untouched. This is less important, however.
- The application doesn't take into account instructor teaching loads. This means 1 instructor could be picked for multiple classes, while another instructor could end up with none. This implies a greater need for the above point.
- Probably other things. Let me know if you think of other flaws.

What I learned/gained experience

- A few things about Java Swing and threading
- More experience building complex object-oriented applications in Java
- Javadoc commenting, the thorough kind
- How to use github.io
- How to use a compareTo to sort on multiple fields

Thanks for reading!

A complete API for this application follows this page.

Package classes

Class Summary

Class	Description
Course	Data object to represent a CS course.
GUIapp	Contains all GUI elements and calls the functions in the Worker class as needed.
Instructor	Data object to represent a CS instructor.
Main	Simply used for starting the application.
Worker	A static class that does all the work through static methods.

classes

Class Main

java.lang.Object
    classes.Main

```
public class Main
extends java.lang.Object
```

Simply used for starting the application.

Author:

bryan

Constructor Summary

Constructors

Constructor and Description
`Main()`

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`static void`	`main(java.lang.String[] args)` Creates a new instance of GUIapp, which creates the GUI and gets things rolling.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Main

```
public Main()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Creates a new instance of GUIapp, which creates the GUI and gets things rolling.

Parameters:

`args - Standard args for main.`

classes

Class Course

java.lang.Object
    classes.Course

All Implemented Interfaces:

`java.lang.Comparable<Course>`

```
public class Course
extends java.lang.Object
implements java.lang.Comparable<Course>
```

Data object to represent a CS course.

Author:

bryan

Field Summary

Fields

Modifier and Type	Field and Description
`private int`	**`gradHours`** The number of grad hours this course is worth.
`private java.util.List<Instructor>`	**`instructors`** An ArrayList of instructors that teach this course.
`private java.lang.String`	**`name`** Official title of course.
`private int`	**`number`** Course number.
`private int`	**`underGradHours`** The number of undergrad hours this course is worth.

Constructor Summary

Constructors

Constructor and Description
`Course``()` Just a regular old constructor.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	**addInstructor**(**Instructor** dude) Adds an instructor to this course's list of instructors.
Instructor	**chooseInstructor**() Determines a random instructor to teach this course, chosen from the ArrayList instructors field.
int	**compareTo**(**Course** c) For sorting an ArrayList of courses by their course number.
java.lang.String	**fileBlock**() Creates a nice, readable plaintext representation of this object's attributes.
int	**getGradHours**() Standard getter.
java.lang.String	**getName**() Standard getter.
int	**getNumber**() Standard getter.
int	**getUnderGradHours**() Standard getter.
boolean	**hasInstructor**(**Instructor** dude) Determines whether or not the given instructor is already in the list.
void	**setGradHours**(int gradHours) Standard setter.
void	**setName**(java.lang.String name) Standard setter.
void	**setNumber**(int number) Standard setter.
void	**setUnderGradHours**(int underGradHours) Standard setter.
java.lang.String	**toString**() Standard toString method.
java.lang.String	**tsvLine**() Creates a single-line TSV style representation of this object's data.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

name

```
private java.lang.String name
```

Official title of course.

number

```
private int number
```

Course number.

underGradHours

```
private int underGradHours
```

The number of undergrad hours this course is worth.

gradHours

```
private int gradHours
```

The number of grad hours this course is worth.

instructors

```
private java.util.List<Instructor> instructors
```

An ArrayList of instructors that teach this course.

Constructor Detail

Course

```
public Course()
```

Just a regular old constructor.

Method Detail

chooseInstructor

```
public Instructor chooseInstructor()
```

Determines a random instructor to teach this course, chosen from the ArrayList instructors field.

Returns:

An instructor object.

addInstructor

```
public void addInstructor(Instructor dude)
```

Adds an instructor to this course's list of instructors.

Parameters:

dude - The instructor to add to this course.

hasInstructor

```
public boolean hasInstructor(Instructor dude)
```

Determines whether or not the given instructor is already in the list. This way, he/she can't be added twice.

Parameters:

dude - The instructor to check.

Returns:

A boolean indicating whether or not the list contains the given instructor.

fileBlock

```
public java.lang.String fileBlock()
```

Creates a nice, readable plaintext representation of this object's attributes.

Returns:

A string of plaintext containing this object's attributes.

tsvLine

```
public java.lang.String tsvLine()
```

Creates a single-line TSV style representation of this object's data.

Returns:

A single TSV line.

toString

```
public java.lang.String toString()
```

Standard toString method.

Overrides:

toString in class java.lang.Object

getName

```
public java.lang.String getName()
```

Standard getter.

Returns:

The name of this course.

setName

```
public void setName(java.lang.String name)
```

Standard setter.

Parameters:

name - A string to set as this course's name.

getNumber

```
public int getNumber()
```

Standard getter.

Returns:

The number of this course.

setNumber

```
public void setNumber(int number)
```

Standard setter.

Parameters:

number - An int to set as this course's number.

getUnderGradHours

```
public int getUnderGradHours()
```

Standard getter.

Returns:

The number of undergrad hours this course has.

setUnderGradHours

```
public void setUnderGradHours(int underGradHours)
```

Standard setter.

Parameters:

underGradHours - An int to set as this course's number of undergrad hours.

getGradHours

```
public int getGradHours()
```

Standard getter.

Returns:

The number of grad hours this course has.

setGradHours

```
public void setGradHours(int gradHours)
```

Standard setter.

Parameters:

gradHours - An int to set as this course's number of grad hours.

compareTo

```
public int compareTo(Course c)
```

For sorting an ArrayList of courses by their course number.

Specified by:

compareTo in interface java.lang.Comparable<Course>

classes

Class Instructor

java.lang.Object
    classes.Instructor

All Implemented Interfaces:

`java.lang.Comparable<Instructor>`

```
public class Instructor
extends java.lang.Object
implements java.lang.Comparable<Instructor>
```

Data object to represent a CS instructor.

Author:

`bryan`

Field Summary

Fields

Modifier and Type	Field and Description
`private java.lang.String`	**`background`** This instructor's professional background.
`private int`	**`course1`** A course slot to store a course this instructor teaches.
`private int`	**`course2`** A course slot to store a course this instructor teaches.
`private int`	**`course3`** A course slot to store a course this instructor teaches.
`private int`	**`course4`** A course slot to store a course this instructor teaches.
`private int`	**`course5`** A course slot to store a course this instructor teaches.
`private int`	**`course6`** A course slot to store a course this instructor teaches.
`private int`	**`course7`** A course slot to store a course this instructor teaches.

private int	**course8** A course slot to store a course this instructor teaches.
private java.lang.String	**degName** The name of this instructor's degree.
private int	**degYear** The year this instructor's degree was completed.
private java.lang.String	**email** This instructor's email address.
private java.lang.String	**firstName** The first name of this instructor.
private java.lang.String	**lastName** The last name of this instructor.
private java.lang.String	**name** The full name of this instructor.
private java.lang.String	**title** The title of this instructor.

Constructor Summary

Constructors

Constructor and Description
Instructor() Just a regular constructor.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
int	**compareTo**(**Instructor** dude) For sorting an ArrayList of instructors by their last name, then their first name, then their degree year.
boolean	**doesTeachCourse**(**Course** c) Determines whether or not this instructor teaches the given course.
java.lang.String	**fileBlock**() Creates a nice, readable plaintext representation of this object's attributes.
java.lang.String	**getBackground**() Standard getter.

int	**getCourse1**() Standard getter.
int	**getCourse2**() Standard getter.
int	**getCourse3**() Standard getter.
int	**getCourse4**() Standard getter.
int	**getCourse5**() Standard getter.
int	**getCourse6**() Standard getter.
int	**getCourse7**() Standard getter.
int	**getCourse8**() Standard getter.
java.lang.String	**getDegName**() Standard getter.
int	**getDegYear**() Standard getter.
java.lang.String	**getEmail**() Standard getter.
java.lang.String	**getFirstName**() Standard getter.
java.lang.String	**getLastName**() Standard getter.
java.lang.String	**getName**() Standard getter.
java.lang.String	**getTitle**() Standard getter.
void	**setBackground**(java.lang.String background) Standard setter.
void	**setCourse1**(int course1) Standard setter.
void	**setCourse2**(int course2) Standard setter.
void	**setCourse3**(int course3) Standard setter.

void	**setCourse4**(int course4) Standard setter.
void	**setCourse5**(int course5) Standard setter.
void	**setCourse6**(int course6) Standard setter.
void	**setCourse7**(int course7) Standard setter.
void	**setCourse8**(int course8) Standard setter.
void	**setDegName**(java.lang.String degName) Standard setter.
void	**setDegYear**(int degYear) Standard setter.
void	**setEmail**(java.lang.String email) Standard setter.
void	**setFirstName**(java.lang.String firstName) Standard setter.
void	**setLastName**(java.lang.String lastName) Standard setter.
void	**setName**(java.lang.String name) Standard setter.
void	**setTitle**(java.lang.String title) Standard setter.
java.lang.String	**toString**() Standard toString method.
java.lang.String	**tsvLine**() Creates a single-line TSV style representation of this object's data.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

name

private java.lang.String name

The full name of this instructor.

firstName

```
private java.lang.String firstName
```

The first name of this instructor.

lastName

```
private java.lang.String lastName
```

The last name of this instructor.

title

```
private java.lang.String title
```

The title of this instructor.

background

```
private java.lang.String background
```

This instructor's professional background.

email

```
private java.lang.String email
```

This instructor's email address.

degName

```
private java.lang.String degName
```

The name of this instructor's degree.

degYear

```
private int degYear
```

The year this instructor's degree was completed.

course1

```
private int course1
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course2

```
private int course2
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course3

```
private int course3
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course4

```
private int course4
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course5

```
private int course5
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course6

```
private int course6
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course7

```
private int course7
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

course8

```
private int course8
```

A course slot to store a course this instructor teaches. If the "slot" is empty, 99 is used as a placeholder.

Constructor Detail

Instructor

```
public Instructor()
```

Just a regular constructor.

Method Detail

getName

```
public java.lang.String getName()
```

Standard getter.

Returns:

This instructor's name.

doesTeachCourse

```
public boolean doesTeachCourse(Course c)
```

Determines whether or not this instructor teaches the given course.

Parameters:

c - The course to check.

Returns:

A boolean indicating whether or not this instructor teaches the given course.

setName

```
public void setName(java.lang.String name)
```

Standard setter. Also sets the first and last name fields.

Parameters:

name - The name to be assigned to this instructor.

fileBlock

```
public java.lang.String fileBlock()
```

Creates a nice, readable plaintext representation of this object's attributes.

Returns:

A string of plaintext containing this object's attributes.

tsvLine

```
public java.lang.String tsvLine()
```

Creates a single-line TSV style representation of this object's data.

Returns:

A single TSV line.

toString

```
public java.lang.String toString()
```

Standard toString method.

Overrides:

`toString in class java.lang.Object`

Returns:

`The name of this instructor.`

getEmail

```
public java.lang.String getEmail()
```

Standard getter.

Returns:

`The referenced field.`

setEmail

```
public void setEmail(java.lang.String email)
```

Standard setter.

Parameters:

`email - The value to assign to this field.`

getDegName

```
public java.lang.String getDegName()
```

Standard getter.

Returns:

`The referenced field.`

setDegName

```
public void setDegName(java.lang.String degName)
```

Standard setter.

Parameters:

`degName - The value to assign to this field.`

getDegYear

```
public int getDegYear()
```

Standard getter.

Returns:

`The referenced field.`

setDegYear

```
public void setDegYear(int degYear)
```

Standard setter.

Parameters:

`degYear - The value to assign to this field.`

getTitle

```
public java.lang.String getTitle()
```

Standard getter.

Returns:

`The referenced field.`

setTitle

```
public void setTitle(java.lang.String title)
```

Standard setter.

Parameters:

`title - The value to assign to this field.`

getBackground

```
public java.lang.String getBackground()
```

Standard getter.

Returns:

`The referenced field.`

setBackground

```
public void setBackground(java.lang.String background)
```

Standard setter.

Parameters:

`background - The value to assign to this field.`

getFirstName

```
public java.lang.String getFirstName()
```

Standard getter.

Returns:

The referenced field.

setFirstName

```
public void setFirstName(java.lang.String firstName)
```

Standard setter.

Parameters:

`firstName - The value to assign to this field.`

getLastName

```
public java.lang.String getLastName()
```

Standard getter.

Returns:

`The referenced field.`

setLastName

```
public void setLastName(java.lang.String lastName)
```

Standard setter.

Parameters:

`lastName - The value to assign to this field.`

getCourse1

```
public int getCourse1()
```

Standard getter.

Returns:

`The referenced field.`

setCourse1

```
public void setCourse1(int course1)
```

Standard setter.

Parameters:

`course1 - The value to assign to this field.`

getCourse2

```
public int getCourse2()
```

Standard getter.

Returns:

The referenced field.

setCourse2

```
public void setCourse2(int course2)
```

Standard setter.

Parameters:

course2 - The value to assign to this field.

getCourse3

```
public int getCourse3()
```

Standard getter.

Returns:

The referenced field.

setCourse3

```
public void setCourse3(int course3)
```

Standard setter.

Parameters:

course3 - The value to assign to this field.

getCourse4

```
public int getCourse4()
```

Standard getter.

Returns:

The referenced field.

setCourse4

```
public void setCourse4(int course4)
```

Standard setter.

Parameters:

course4 - The value to assign to this field.

getCourse5

```
public int getCourse5()
```

Standard getter.

Returns:

```
The referenced field.
```

setCourse5

```
public void setCourse5(int course5)
```

Standard setter.

Parameters:

```
course5 - The value to assign to this field.
```

getCourse6

```
public int getCourse6()
```

Standard getter.

Returns:

```
The referenced field.
```

setCourse6

```
public void setCourse6(int course6)
```

Standard setter.

Parameters:

```
course6 - The value to assign to this field.
```

getCourse7

```
public int getCourse7()
```

Standard getter.

Returns:

```
The referenced field.
```

setCourse7

```
public void setCourse7(int course7)
```

Standard setter.

Parameters:

```
course7 - The value to assign to this field.
```

getCourse8

```
public int getCourse8()
```

Standard getter.

Returns:

`The referenced field.`

setCourse8

```
public void setCourse8(int course8)
```

Standard setter.

Parameters:

`course8 - The value to assign to this field.`

compareTo

```
public int compareTo(Instructor dude)
```

For sorting an ArrayList of instructors by their last name, then their first name, then their degree year.

Specified by:

`compareTo in interface java.lang.Comparable<Instructor>`

classes

Class Worker

java.lang.Object
    classes.Worker

```
public class Worker
extends java.lang.Object
```

A static class that does all the work through static methods. Contains and manages the data.

Author:

bryan

Field Summary

Fields

Modifier and Type	Field and Description
static java.util.List<Course>	**courses** An ArrayList of courses that will hold live data while the application is running.
static java.lang.String	**currCourse1** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currCourse2** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currCourse3** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currCourse4** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currCourse5** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currCourse6** Holds the currently selected course in the corresponding GUI dropdown.

static java.lang.String	**currCourse7** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currCourse8** Holds the currently selected course in the corresponding GUI dropdown.
static java.lang.String	**currName** Holds the currently selected instructor in the GUI dropdown.
static java.util.List<Instructor>	**instructors** An ArrayList of instructors that will hold live data while the application is running.

Constructor Summary

Constructors

Constructor and Description
Worker()

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method and Description
static boolean	**checkCourseListFile**() Checks that the user-friendly course data file exists.
static boolean	**checkCourseTSV**() Checks that the course TSV data file exists.
static boolean	**checkInstructorListFile**() Checks that the user-friendly instructor data file exists.
static boolean	**checkInstructorTSV**() Checks that the instructor TSV data file exists.
static java.lang.String	**chooseInstructors**(java.util.List<Course> offered) Generates a selection of instructors for the following semester.
static java.lang.String[]	**coursesToArray**() Takes the main data list of courses and returns a string array of their course numbers.
static void	**downloadAndParseCourses**() Parses through the CS course web page and grabs useful course data.
static void	**downloadAndParseInstructors**() Parses through the UIC CS Faculty web page and grabs useful instructor data.

static boolean	**init**() Initializes global variables and checks to see if the main data files are in existence.
static void	**initPrint**() Prints initial statements about the tool and what's happening.
static java.lang.String[]	**instructorsToArray**() Takes the main data list of instructors and returns a string array of their names.
static void	**loadCourses**() Loads course data from data file.
static void	**loadInstructors**() Loads instructor data from data file.
static void	**saveInfo**() Saves the courses for the currently selected instructor in the GUI dropdown and updates the data file.
static void	**setCurrCourse1**(java.lang.String currCourse1) Setter for field currCourse1.
static void	**setCurrCourse2**(java.lang.String currCourse2) Setter for field currCourse2.
static void	**setCurrCourse3**(java.lang.String currCourse3) Setter for field currCourse3.
static void	**setCurrCourse4**(java.lang.String currCourse4) Setter for field currCourse4.
static void	**setCurrCourse5**(java.lang.String currCourse5) Setter for field currCourse5.
static void	**setCurrCourse6**(java.lang.String currCourse6) Setter for field currCourse6.
static void	**setCurrCourse7**(java.lang.String currCourse7) Setter for field currCourse7.
static void	**setCurrCourse8**(java.lang.String currCourse8) Setter for field currCourse8.
static int[]	**setCurrName**(java.lang.String currName) Setter for field currName.
static void	**updateCourses**() Redownloads course information from the web page and updates the data file.
static void	**updateInstructors**() Redownloads instructor information from the web page and updates the data file.
static void	**writeCourses**() Writes the data from the main course ArrayList field to the data files.

static void	**writeInstructors**() Writes the data from the main instructor ArrayList field to the data files.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

instructors

```
public static java.util.List<Instructor> instructors
```

An ArrayList of instructors that will hold live data while the application is running.

courses

```
public static java.util.List<Course> courses
```

An ArrayList of courses that will hold live data while the application is running.

currName

```
public static java.lang.String currName
```

Holds the currently selected instructor in the GUI dropdown.

currCourse1

```
public static java.lang.String currCourse1
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse2

```
public static java.lang.String currCourse2
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse3

```
public static java.lang.String currCourse3
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse4

```
public static java.lang.String currCourse4
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse5

```
public static java.lang.String currCourse5
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse6

```
public static java.lang.String currCourse6
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse7

```
public static java.lang.String currCourse7
```

Holds the currently selected course in the corresponding GUI dropdown.

currCourse8

```
public static java.lang.String currCourse8
```

Holds the currently selected course in the corresponding GUI dropdown.

Constructor Detail

Worker

```
public Worker()
```

Method Detail

init

```
public static boolean init()
```

Initializes global variables and checks to see if the main data files are in existence. If not, it creates the files and does a fresh download on the data from the web page.

Returns:

`a boolean (true by default) indicating whether or not there were missing data files`

initPrint

```
public static void initPrint()
```

Prints initial statements about the tool and what's happening. Gets called as soon as the GUI is loaded.

chooseInstructors

```
public static java.lang.String chooseInstructors(java.util.List<Course> offered)
```

Generates a selection of instructors for the following semester.

Parameters:

`offered - A list of courses for which instructors should be scheduled.`

Returns:

`A string representing the printout of the scheduler.`

instructorsToArray

```
public static java.lang.String[] instructorsToArray()
```

Takes the main data list of instructors and returns a string array of their names. For populating JComboBox dropdown menus.

Returns:

`A string array of instructor names.`

coursesToArray

```
public static java.lang.String[] coursesToArray()
```

Takes the main data list of courses and returns a string array of their course numbers. For populating JComboBox dropdown menus.

Returns:

`A string array of course numbers.`

saveInfo

```
public static void saveInfo()
```

Saves the courses for the currently selected instructor in the GUI dropdown and updates the data file.

updateInstructors

```
public static void updateInstructors()
```

Redownloads instructor information from the web page and updates the data file.

updateCourses

```
public static void updateCourses()
```

Redownloads course information from the web page and updates the data file.

checkInstructorTSV

```
public static boolean checkInstructorTSV()
```

Checks that the instructor TSV data file exists. Also makes sure the data directory exists.

Returns:

```
A boolean indicating whether or not the file existed.
```

checkCourseTSV

```
public static boolean checkCourseTSV()
```

Checks that the course TSV data file exists. Also makes sure the data directory exists.

Returns:

```
A boolean indicating whether or not the file existed.
```

checkInstructorListFile

```
public static boolean checkInstructorListFile()
```

Checks that the user-friendly instructor data file exists. Also makes sure the data directory exists.

Returns:

```
A boolean indicating whether or not the file existed.
```

checkCourseListFile

```
public static boolean checkCourseListFile()
```

Checks that the user-friendly course data file exists. Also makes sure the data directory exists.

Returns:

```
A boolean indicating whether or not the file existed.
```

loadInstructors

```
public static void loadInstructors()
```

Loads instructor data from data file.

loadCourses

```
public static void loadCourses()
```

Loads course data from data file.

downloadAndParseInstructors

```
public static void downloadAndParseInstructors()
```

```
                                        throws java.io.IOException
```

Parses through the UIC CS Faculty web page and grabs useful instructor data. Populates the main data ArrayList of instructors in this class.

Throws:

`java.io.IOException - In case a connection cannot be made.`

downloadAndParseCourses

```
public static void downloadAndParseCourses()
                                    throws java.io.IOException
```

Parses through the CS course web page and grabs useful course data. Populates the main data ArrayList of courses in this class.

Throws:

`java.io.IOException - In case a connection cannot be made.`

writeInstructors

```
public static void writeInstructors()
                             throws java.io.IOException
```

Writes the data from the main instructor ArrayList field to the data files.

Throws:

`java.io.IOException - If for some reason the files don't exist.`

writeCourses

```
public static void writeCourses()
                         throws java.io.IOException
```

Writes the data from the main course ArrayList field to the data files.

Throws:

`java.io.IOException - If for some reason the files don't exist.`

setCurrName

```
public static int[] setCurrName(java.lang.String currName)
```

Setter for field currName. In addition to setting the value, this function also looks up all associated courses for said instructor so the GUI dropdowns can be automatically populated.

Parameters:

`currName - A string to be set as the current instructor name.`

Returns:

`An int array containing the indices of course numbers that should be selected in each GUI dropdown for courses.`

setCurrCourse1

```
public static void setCurrCourse1(java.lang.String currCourse1)
```

Setter for field currCourse1.

Parameters:

`currCourse1 - A string to be set as the current course 1.`

setCurrCourse2

```
public static void setCurrCourse2(java.lang.String currCourse2)
```

Setter for field currCourse2.

Parameters:

`currCourse2 - A string to be set as the current course 2.`

setCurrCourse3

```
public static void setCurrCourse3(java.lang.String currCourse3)
```

Setter for field currCourse3.

Parameters:

`currCourse3 - A string to be set as the current course 3.`

setCurrCourse4

```
public static void setCurrCourse4(java.lang.String currCourse4)
```

Setter for field currCourse4.

Parameters:

`currCourse4 - A string to be set as the current course 4.`

setCurrCourse5

```
public static void setCurrCourse5(java.lang.String currCourse5)
```

Setter for field currCourse5.

Parameters:

`currCourse5 - A string to be set as the current course 5.`

setCurrCourse6

```
public static void setCurrCourse6(java.lang.String currCourse6)
```

Setter for field currCourse6.

Parameters:

currCourse6 - A string to be set as the current course 6.

setCurrCourse7

```
public static void setCurrCourse7(java.lang.String currCourse7)
```

Setter for field currCourse7.

Parameters:

currCourse7 - A string to be set as the current course 7.

setCurrCourse8

```
public static void setCurrCourse8(java.lang.String currCourse8)
```

Setter for field currCourse8.

Parameters:

currCourse8 - A string to be set as the current course 8.

classes

Class GUIapp

java.lang.Object
    java.awt.event.WindowAdapter
        classes.GUIapp

All Implemented Interfaces:

`java.awt.event.WindowFocusListener, java.awt.event.WindowListener, java.awt.event.WindowStateListener, java.lang.Runnable, java.util.EventListener`

```
public class GUIapp
extends java.awt.event.WindowAdapter
implements java.awt.event.WindowListener, java.lang.Runnable
```

Contains all GUI elements and calls the functions in the Worker class as needed. Taken from: http://www.comweb.nl/java/Console/Console.html and adapted for use with this application by Bryan Spahr.

Author:

bryan

Field Summary

Fields

Modifier and Type	Field and Description
`private javax.swing.JFrame`	**`editorFrame`** Popup frame that can be opened to edit faculty info.
`private javax.swing.JFileChooser`	**`fc`** Save dialog for saving the results to a plaintext file.
`private javax.swing.JFrame`	**`mainFrame`** Main GUI window with buttons.
`private java.io.PipedInputStream`	**`pin`** For piping sysout text to the editor window.
`private java.io.PipedInputStream`	**`pin2`** For piping sysout text to the editor window.
`private boolean`	**`quit`** Used for properly closing threads when the application is terminated.
`private java.lang.Thread`	**`reader`** For capturing the text stream from sysout.

private java.lang.Thread	**reader2** For capturing the text stream from sysout.
private java.io.File	**resultFile** File pointer for saving the results to a plaintext file.
private javax.swing.JFrame	**resultFrame** Display window for showing calculated results.
private java.lang.String	**resultString** For storing the result string as a field in this class.
private javax.swing.JTextArea	**resultText** For displaying the results text to the user.
private javax.swing.JTextArea	**textArea** Recipient of sysout print statements in the editor window.

Constructor Summary

Constructors

Constructor and Description
GUIapp() Creates a new instance of GUIapp (duh), also calls the Worker class's init method and all GUI methods and init functions in this class.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
void	**editorWindow**(boolean show) Creates the editor window with all swing elements.
void	**mainWindow**() Creates and shows the main GUI window, including all Swing elements.
java.lang.String	**readLine**(java.io.PipedInputStream in) Threading manager for sysout piping from aforementioned example code.
void	**resultWindow**() Creates the window for results but doesn't show it.
void	**run**() Standard threaded GUI method.
void	**textStreamInit**() Taken from the example code referenced above.

void	**windowClosed**(java.awt.event.WindowEvent evt) Standard JSwing event method.
void	**windowClosing**(java.awt.event.WindowEvent evt) No idea what this does.

Methods inherited from class java.awt.event.WindowAdapter

windowActivated, windowDeactivated, windowDeiconified, windowGainedFocus, windowIconified, windowLostFocus, windowOpened, windowStateChanged

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.awt.event.WindowListener

windowActivated, windowDeactivated, windowDeiconified, windowIconified, windowOpened

Field Detail

editorFrame

```
private javax.swing.JFrame editorFrame
```

Popup frame that can be opened to edit faculty info.

mainFrame

```
private javax.swing.JFrame mainFrame
```

Main GUI window with buttons.

textArea

```
private javax.swing.JTextArea textArea
```

Recipient of sysout print statements in the editor window.

reader

```
private java.lang.Thread reader
```

For capturing the text stream from sysout.

reader2

```
private java.lang.Thread reader2
```

For capturing the text stream from sysout.

quit

```
private boolean quit
```

Used for properly closing threads when the application is terminated. I think.

resultFrame

```
private javax.swing.JFrame resultFrame
```

Display window for showing calculated results.

resultText

```
private javax.swing.JTextArea resultText
```

For displaying the results text to the user.

resultString

```
private java.lang.String resultString
```

For storing the result string as a field in this class. In case it's needed later.

resultFile

```
private java.io.File resultFile
```

File pointer for saving the results to a plaintext file.

fc

```
private javax.swing.JFileChooser fc
```

Save dialog for saving the results to a plaintext file.

pin

```
private final java.io.PipedInputStream pin
```

For piping sysout text to the editor window.

pin2

```
private final java.io.PipedInputStream pin2
```

For piping sysout text to the editor window.

Constructor Detail

GUIapp

```
public GUIapp()
```

Creates a new instance of GUIapp (duh), also calls the Worker class's init method and all GUI methods and init functions in this class.

Method Detail

mainWindow

```
public void mainWindow()
```

Creates and shows the main GUI window, including all Swing elements. Closing this windows is the only way to exit the application.

resultWindow

```
public void resultWindow()
```

Creates the window for results but doesn't show it.

editorWindow

```
public void editorWindow(boolean show)
```

Creates the editor window with all swing elements.

Parameters:

`show - Indicates whether or not the editor window should be shown at the start.`

textStreamInit

```
public void textStreamInit()
```

Taken from the example code referenced above. Evidently it sets up the text piping from sysout to the text area in the editor window.

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent evt)
```

Standard JSwing event method. When the GUI window is closed, all threads are closed safely. Only the main window uses this method, so that only closing the main window closes the application.

Specified by:

`windowClosed in interface java.awt.event.WindowListener`

Overrides:

windowClosed in class java.awt.event.WindowAdapter

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent evt)
```

No idea what this does. Google it.

Specified by:

windowClosing in interface java.awt.event.WindowListener

Overrides:

windowClosing in class java.awt.event.WindowAdapter

run

```
public void run()
```

Standard threaded GUI method. Here it's used to manage the sysout text threads.

Specified by:

run in interface java.lang.Runnable

readLine

```
public java.lang.String readLine(java.io.PipedInputStream in)
                          throws java.io.IOException
```

Threading manager for sysout piping from aforementioned example code. No, I don't know what it does.

Parameters:

in - A PipedInputStream, duh

Returns:

some sort of string obviously

Throws:

java.io.IOException - if there's an IOException, why do I have to describe this