

Lab 9 – User Interfaces

Aim

The main aim of this lab is for you to create an interface for the Zoo that lets you control it. A user should be able to:

- Add an animal
- Remove an animal
- Save to a text file

Code Understanding

Based on your lecture notes, some initial code is provided, and is on ICE.

- Download this week's Zoo project folder from ICE. Open the main class, Zoo 1, and look at the main method. It finishes with a new line:

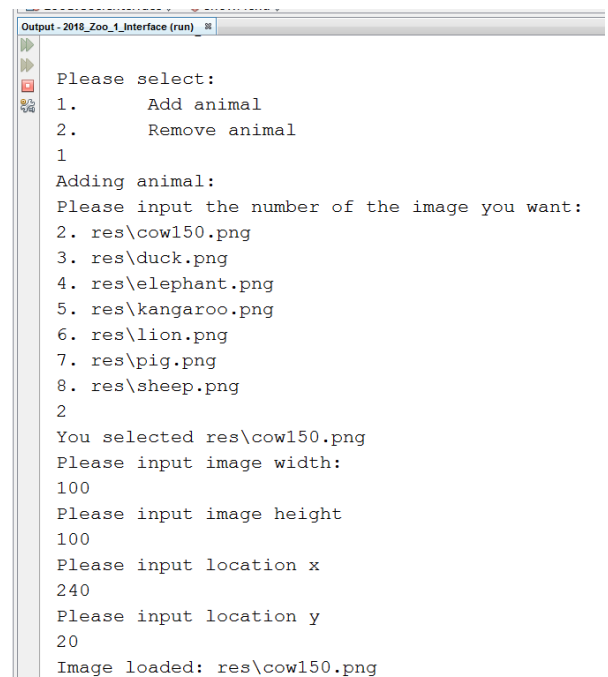
```
UserInterface.showMenu( );
```

- Open the UserInterface class. Look at the showMenu method. Follow what it does.
- Notice how it gets the user input with the getIntInput method. Check that method. How does it work?
- Run the application. What happens? Why?

Adding an Animal to the Zoo

The menu has an option to add an animal to the zoo. We will implement it. The user should be able to choose which image they want to use, and the location, width and the height of the image to be added.

- Review the lecture ppt from around slide 30
- You are going to complete the getImageInput method in the UserInterface class
- Our FileUtils class has a new method, getFileNames(String dirPath)
- It returns an ArrayList<String> of the path+filename of all files in that directory (folder)
- You are going to use this to create a menu for the user to choose which animal image they want
- Slide 32 from the lecture shows you how



```
Output - 2018_Zoo_1_Interface (run)
Please select:
1.      Add animal
2.      Remove animal
1
Adding animal:
Please input the number of the image you want:
2. res\cow150.png
3. res\duck.png
4. res\elephant.png
5. res\kangaroo.png
6. res\lion.png
7. res\pig.png
8. res\sheep.png
2
You selected res\cow150.png
Please input image width:
100
Please input image height:
100
Please input location x:
240
Please input location y:
20
Image loaded: res\cow150.png
```

- When you have got this working, you also need to ask the user to input x, y, width, height. Then finish the method with:

```
Zoo1.addImage(path, imageX, imageY, width, height);
```

- This will add the image to the Zoo
- At the end of the menu, we need to call the showMenu() method to return the user to the menu

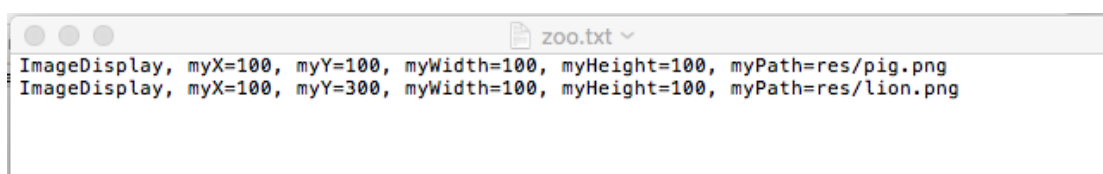
Saving to File

Adding a new animal each time you need to run the zoo can be very boring. You may also want to save your zoo so you can look at it again in the future. To do this, we can save objects to file!

- Review the lecture ppt from slide 37 to 47. This discusses how to save and load data
- Override toString in your ImageDisplay class to save information about each animal in your file. Be very careful with spaces, commas etc. Use the lecture notes, and the text should be in the format in the lecture notes, and also shown below
- Test it with a System.out to display it on screen
- There is a static method in FileUtils that will save the zoo to a file:

```
public static boolean saveZooToFile(ArrayList<ImageDisplay> toWrite)
```

- Add a menu option in the UserInterface class for the user to save to file
- Test it by adding a number of animals to your zoo, and saving it
- When it is working, you should see a text file in your project folder called “zoo.txt”. Open it. Does it look like this?



Loading from File

In the previous task, you created a text file that stored all your zoo animals. Now, you want to change your program so that it will load the file, and automatically add animals to the zoo.

- You need to create new ImageDisplay objects by reading your text file
- FileUtils has a new method readZooFromFile(). It reads the file line by line
- Each line is passed to the imageFromString method:

```
imageFromString(line);
```

- Currently, this method does not do anything. You need to complete it, so that it ends with the line

```
Zoo1.addImage(path, x, y, w, h);
```

- You will need to take that line of text, divide it into individual strings, and then extract only the useful data (i.e. the numbers and the paths)

Tips:

- `line.split(",")` returns an array of strings by splitting *line* at every comma, or whatever you pass as an argument, like this:

```
String[] data = line.split(",");
```

data should look like this:

data[0]	"ImageDisplay"
data[1]	" myX=100"
data[2]	" myY=100"
data[3]	" myWidth=100"
data[4]	" myHeight=100"
data[5]	" myPath=res/lion.png"

Of course, you may have to check your `toString` method for the exact formatting.

Then you can use the other methods of `String`, such as `substring()`, or `split()`, to get the data you want. You will also need to use `Integer.parseInt(String)`.

Working with ArrayLists

In this week's task, you worked with an `ArrayList`. The lecture gave you examples of how to use a basic `ArrayList` for strings. We will extend on this.

- Create a new Java project and import your `Person` and `Student` files from week 6. Create a new controller class, with a main method that creates students.
- Using your lecture notes as a guide, create an `ArrayList` of 4 students by creating 4 students and adding them to an `ArrayList`. Check your week 8 and 9 lecture notes
- Create a static method in your controller class that will loop through the `ArrayList` and call each student's `toString()` method.

Searching through an ArrayList

You can search through an array list, much like you can search through an array.

- Create a new static method named `renameStudent`. This method will allow you to change the name of a student that matches a certain name (such as “Ting”)
- Using your knowledge from previous labs and lectures, in this method, you want to get an input from the user of the name to find, and also the new name, and search through your arraylist to see if there is a matching name. If a matching name is found and no modifications have been made, then change the name.
- If the name is not on the list, then inform the user that nothing has been changed

Homework - Back to the Zoo – Removing an Animal!

We can now add animals to our zoo, and we can save and write to files. We can also load from our zoo. But we also want to be able to remove animals from our zoo. How might we do this? Experiment in your own time, and try to find the best solution