IFS 353 Final Assignment 2022.

Open date: 28 October 2022, 12:00
Due date: 12 November 2022, 12:00
Total marks: 100

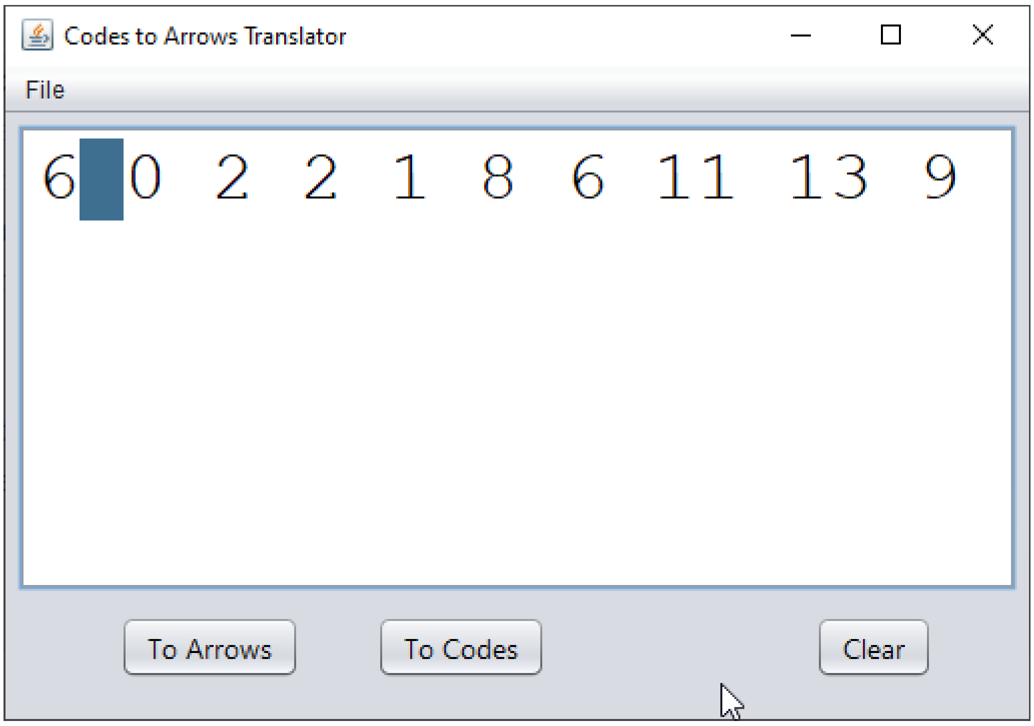
Your assignment is as follows:

- You must create a NetBeans project to produce a Java `JFrame` form GUI.
- Write a helper class named `ArrowSymbols` that contains an instance variable `public final ArrayList<Character> arrowChars`. This list must contain the following Unicode arrow chars, **in this order**:
  `'\u2190', '\u2191', '\u2192', '\u2193', '\u21A4',`
  `'\u21A5', '\u21A6', '\u21A7', '\u21E4', '\u21E5',`
  `'\u2196', '\u2197', '\u2198', '\u2199'`
- Add to the class a method
  `public String toArrowsString(int[] codes)`
  that converts an array of integers into a string of the corresponding arrow chars i.e., the integers are index values in the list of arrow chars.
- Add to the class a second method
  `public String toIntegersString(char[] arrows)`
  that converts its argument char array of arrow chars into a string of integers, separated by spaces.
- Code the GUI to contain a `JTextPane` – *not* a `JTextArea` – into which integers that correspond to the index values of arrow chars in the list, separated by spaces, can be typed. Place a button in the GUI that will make use of an `ArrowSymbols` object method to convert the integers into their arrow char equivalents. The images in the next page show the GUI with integers (note the single space between each), thereafter the integers converted into arrow chars. Make the font of the text pane monospace and large enough to show the arrows and integers clearly, as in the images.

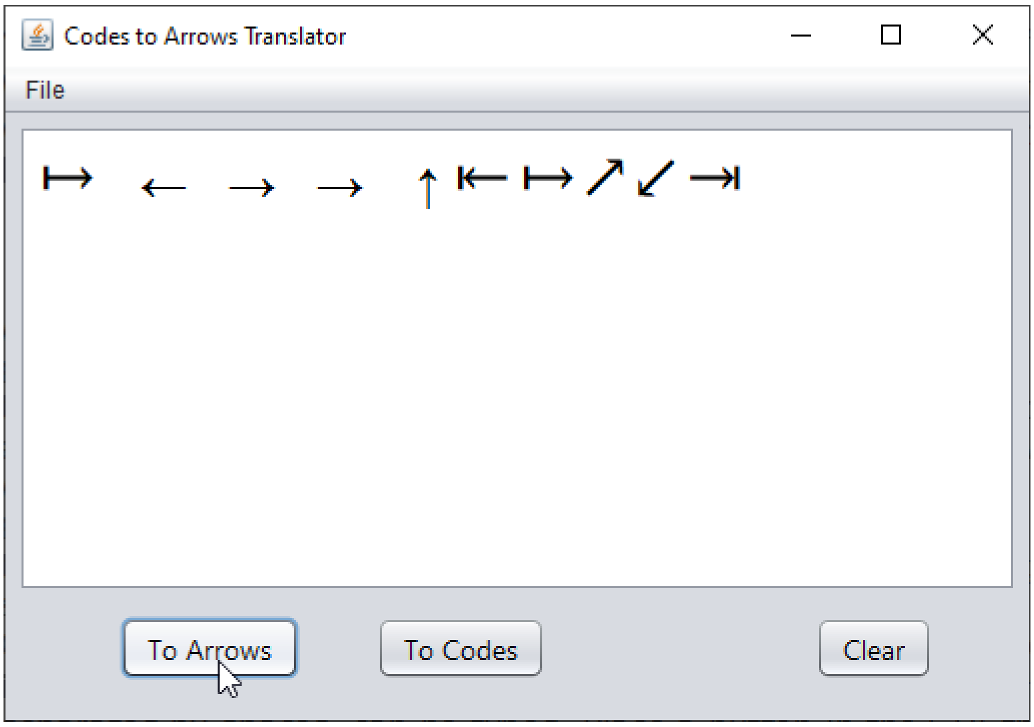
- Note that a string of arrow chars **must** begin with ↦ and end with ⇥ which means that a string of integers must begin with 6 and end with 9: see the images on the next page. Note also that if an integer in the integer string is one of 0, 1, 2, 3 then when translated to an arrow string, that arrow char must have a space inserted in front of it; all other arrow chars do not get leading spaces.
- Add a second button that will convert an arrow string back into a string of integers, separated by spaces, by making use likewise of an

`ArrowSymbols` object method.

- Add a button that when clicked will clear the text pane of text.
- Add a 'File' menu with a single 'Open file...' menu item, which will allow the user to open a text file containing a string of integers separated by spaces, for conversion to arrow chars.



```
6 0 2 2 1 8 6 11 13 9
```



```
↦ ← → → ↑⇤↦↗↙⇥
```

Run the JAR executable bundled with this PDF file by double-clicking it, to see the GUI and its functionality. Your code must handle any and all exceptions thrown by translation errors or data errors, such as an integer string that does not begin with 6 and/or end with 9, or an arrow string that does not begin with `0x21A6` and/or end with `0x21E5`. Experiment with the given GUI executable to see how an illegal string is handled – leave off the 9 or add two spaces between two integers, etc. Do the same for a string of arrow chars by leaving off the ↦ or ⇥!

The Java documentation for the `ArrowSymbols` type is also included in this archive to assist you in writing the class.

Note: the GUI is intended to process strings that *do not contain newline chars*, so do not try to process anything other than a series of integers separated by spaces (no space at the end). A `JTextPane` will wrap text automatically, so you don't need to set a property to achieve this. Note also that in the normal scheme of things, reading in a text file from a `JFileChooser` involves adding newline chars: obviously you cannot do that in this GUI or you will get a `NumberFormatException`!

The secret to coding this application is to understand that the integers are just the index numbers of the arrow chars in the order given, so by taking the entire text out of the text pane and splitting it, you can get an array of integers, each of which references its respective arrow char. The arrow chars are different – they can't be split because not all are preceded by spaces, so you have to process each char in the string manually to build an array of arrow chars only – no spaces – for conversion back to integers. So, this is just an exercise of matching indices with characters, plus the reverse operation of matching characters top indices. Make sure to use the `java.lang.String` and `java.util.ArrayList` API documentation to help you figure out how to process the arrow chars string into its separate chars, plus how to use indices with lists.

Do your best to code this application: the more code you submit, the more marks you will get, even if your code is not quite right - you can't get any marks for code you *didn't* submit.

Project submission: to submit your final assignment, zip the entire NetBeans project into an archive named with your student number and surname, in the standard way, e.g. `1234567YourSurname.zip;` upload this single file to iKamva before the due date and time. **Warning!** The due date and time is absolute and *under no circumstances will late submissions be accepted*. The iKamva server time is the only time that will be considered, not the time on your watch or computer. Make sure therefore that you take load shedding into account and submit with enough time in hand to allow you to deal with any submission problems that may occur. You leave submission to the last minute at your peril – you are playing with your future!