New Java SE Classes

Overview

In this lab, you will write an application that makes use of new classes in Java SE.

Source folders

Student project: `StudentAdditionalJavaSEClasses`
Solution project: `SolutionAdditionalJavaSEClasses`

Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Using the `StringBuilder` class
2. Using formatting techniques
3. Using regular expressions
4. Additional suggestions

Exercise 1: Using the StringBuilder class

Open `UsingStringBuilder.java`, and write an application that explores the capabilities of the `StringBuilder` class.

Suggestions and requirements:

- Write a simple `main()` method that creates a `StringBuilder` object with an initial capacity of 50 characters (say), to give it some initial space to grow into.
- Call `append()` on the `StringBuilder` object, to append some text.
- Call `delete()` to delete some of the text.
- Call `deleteCharAt()` to delete a character at a specific position.
- Call `insert()` to insert new text at various positions.
- Call `replace()` to replace text at various positions.
- Call `setCharAt()` to set a character at a specific position.
- Call `reverse()` to reverse the text.
- Finally, call `toString()` to create an immutable `String` from the text.

Exercise 2: Using formatting techniques

Open `Formatting.java`, and write some code to explore the formatting capabilities of the `Formatter` class and the `String` class.

Suggestions and requirements:

- First, take a look at the demo code that accompanies this chapter (see the `DemoNewJavaSEClasses` project, and the `Formatting.java` source file).
- In your application, create a `Formatter` object, and use it to format strings using various format specifiers (see the `demoFormatter()` method in the demo code, and experiment with the options).
- Use `String.format()` to achieve the same effect (see the `demoStringFormat()` method in the demo code).
- Experiment with number formatting (see the `demoNumberFormatting()` method in the demo code).
- Experiment with time formatting (see the `demoTimeFormatting()` method in the demo code).
- Experiment with date formatting (see the `demoDateFormatting()` method in the demo code).

Exercise 3: Using regular expressions

Open `RegularExpressions.java`, and write some code to make use of regular expressions.

Suggestions and requirements:

- First, take a look at the demo code that accompanies this chapter (see the `DemoNewJavaSEClasses` project, and the `RegularExpressions.java` source file).
- In your application, write code to do simple pattern matching, as illustrated by the `demoSimpleRegex()` method in the demo code.
- Write code to explore the matching functionality provided by the `Matcher` class, as illustrated by the `demoMatching()` method in the demo code.
- Write code to perform repeated searches on a string, as illustrated by the `demoRepeatedFinds()` method in the demo code.
- Write code to explore the regular expression support built into the `String` class, as illustrated by the `demoStringRE()` method in the demo code.

Exercise 4 (If time permits): Additional suggestions

Experiment with locales in your "formatting" code.

Experiment with group captures and replacements in your "regular expression code". For some ideas of what's possible, open the `RegularExpressions.java` demo code and take a look at the `demoGroups()` and `demoReplacement()` methods.