

Exercise Sheet 3: Operations on Strings and ArrayLists

This is a set of exercises to support the “Java’s Built-in Classes” section

- 1) Download the files `StringTest4.java`, `UseArrayLists4.java` and `UseArrayLists8.java` from the code index for this section, and check you can compile and run this code. You can make use of the support code in these files to write front-ends (or “drivers”) to test the methods you have to write in the other parts of this exercise for strings, `ArrayList`s of integers and `ArrayList`s of strings.
- 2) Write a static method which takes two strings and returns a boolean saying whether the first string occurs as a substring within the second. For example, if the strings are “blis” and “antidisestablishment”, the method should return `true`. Do not make use of any of the built-in methods provided by Java in class `String`, apart from `length`, `substring` and `charAt`. See if you can write two versions of the method, one using iteration, the other using recursion. Remember when you are asked to write “a method” it is perfectly acceptable to write extra helper methods which that method calls.
- 3) A palindrome is a string which is exactly the same read backwards or forwards. An example is “redivider”. Write a static method which takes a string and returns a boolean saying whether it is a palindrome. Modify your method so that it works with palindromic sentences, an example being: “Madam, I’m Adam”. As you can see, this involves ignoring spaces and punctuation symbols, and also ignoring case differences in letters. In this case, you may make use of methods from Java’s class `Character` to test properties of individual characters.
- 4) Write a static method which takes an `ArrayList` of `Strings` and an integer and changes the `ArrayList` destructively to remove all `Strings` whose length is less than the integer argument. So if the integer argument is 4 and the `ArrayList` is

| | | | | | | | |
|--------|--------|-------|-------|-----|--------|-----|------|
| tomato | cheese | chips | fruit | pie | butter | tea | buns |
|--------|--------|-------|-------|-----|--------|-----|------|

the `ArrayList` should be changed to:

| | | | | | |
|--------|--------|-------|-------|--------|------|
| tomato | cheese | chips | fruit | butter | buns |
|--------|--------|-------|-------|--------|------|

- 5) Write a static method which performs the same operation as the one in question 4, but constructively rather than destructively. Show using a front-end which gives aliasing that your method for part 4 is destructive and your method for part 5 is not.
- 6) Write generic static methods which take an `ArrayList` and two objects of its element type. The methods must replace the first occurrence of an object equal to the first object by the second object. As with questions 4 and 5, give a destructive and a constructive version. For example, if a call takes an `ArrayList` of integers `[5, 12, 4, 16, 4, 2, 2]` and the integers 4 and 7, one method should change the actual `ArrayList` object to `[5, 12, 7, 16, 4, 2, 2]` and the other should return a new `ArrayList` which has that value. The file `UseArrayLists10.java` shows supporting code for creating `ArrayList`s of `Strings` and `ArrayList`s of integers, and an example of a generic method.

You may use Java’s built-in `ArrayList` methods. Note the significance of the words “occurrence of an object equal to”. It means equality testing using the method `equals`, as opposed to testing for an occurrence of the actual object using the alias test given by the `==` symbol. You could modify the front-end code in `UseArrayLists16.java` to test your method, with an element class `Pair` rather than `String` or `Integer`. You will then also need the classes `Pair` and `NoPairException`, whose `.class` files you can download.

- 7) Write a static method which takes an `ArrayList` of `DrinksMachine` objects (as used for exercise 1) and returns the result of buying a can of Coke from the cheapest drinks machine which has Coke available. The result returned will actually be a reference to a `Can` object. Your method should insert the correct amount of money for the can into the machine. If all the `DrinksMachine` objects in the `ArrayList` are empty of Cokes, the method should throw an `EmptyMachineException`.
- 8) Write a static method which takes an object of type `ArrayList<ArrayList<Integer>>` and an integer and returns an `ArrayList<ArrayList<Integer>>` object consisting of all those `ArrayList<Integer>` objects from the original `ArrayList<ArrayList<Integer>>` which do not contain the integer argument. For example, if the original `ArrayList<ArrayList<Integer>>` is `[[1,2,3],[7,5],[4,4,2],[8,12,3]]` and the integer is 2, the object returned should be `[[7,5],[8,12,3]]`. You may use appropriate methods from class `ArrayList<E>` in your code for this method.
- 9) Write a static method which performs the same task as for question 8, but does it completely constructively. So, the correct answer to question 8 should construct a new `ArrayList<ArrayList<Integer>>` object whose internal `ArrayList<Integer>` objects are shared with the original `ArrayList<ArrayList<Integer>>` object, but the correct answer to question 9 should construct a new `ArrayList<ArrayList<Integer>>` object whose internal `ArrayList<Integer>` objects are copies of those from the original `ArrayList<ArrayList<Integer>>` object.
- 10) Write a static method that takes the name of a file given as a `String`, and returns an `ArrayList<ArrayList<String>>` object, where each line from the file is converted to an `ArrayList<String>`, with each `String` being a word from that line. You may use whatever classes are necessary from the Java library to assist with this. The files `ScannerTest1.java` and `ScannerTest2.java` in the code directory for this section may help.
- To give you sample data for this, some text files are provided in the code. For example, if you download the file `twinkle.txt` into your directory, and call the method with `"twinkle.txt"` as its argument, it should return an `ArrayList` of size 4, whose first element is an `ArrayList` of `Strings` of size 4, second element an `ArrayList` of `Strings` of size 6, and so on.
- 11) Modify the methods that answer questions 8 and 9 to make them generic, and apply them to the `ArrayList<ArrayList<String>>` objects returned from the answer to question 10.
- 12) Write a generic method that takes an `ArrayList` and two objects of its element type, and returns an `ArrayList` consisting of all those elements of the original `ArrayList` starting with the first occurrence of the first element argument and ending with the next occurrence of the second element argument, putting them in reverse order. So if the original `ArrayList` is:

| | | | | | | | |
|--------|--------|-------|-------|-----|--------|-----|------|
| tomato | cheese | chips | fruit | pie | butter | tea | buns |
|--------|--------|-------|-------|-----|--------|-----|------|

and the other two arguments are `"chips"` and `"buns"`, the `ArrayList` returned should be:

| | | | | | |
|------|-----|--------|-----|-------|-------|
| buns | tea | butter | pie | fruit | chips |
|------|-----|--------|-----|-------|-------|

Again, you can modify the code from `UseArrayLists16.java` to provide a front end to test this method with a type which is not `String` or `Integer`.

Matthew Huntbach