Programming 2:
Tutorial 5

Set by: Luke Dickens

29th Oct – 2nd Nov, 2018

Reminder about the tutorial sheets

Remember that the best way to learn a programming language and understand the concepts is to do lots of programming. This involves a good deal of problem solving, and that requires you to think, experiment and test things. Please try all the standard questions (those not [hard] or [harder]), and spend some time thinking carefully about them, before asking for help. If you are still stuck:

- ask me, or the lab helpers, for help at the lab sessions
- or post a question on the moodle course page

Questions marked as *harder* are there to make you think. You only need to *sketch* a solution to these, and model solutions may not be provided. Do not worry if you cannot complete the harder questions without help.

1 In-place cyclic shift of array

As usual, the classes for this question are in the appropriate folder of the provided code. Look at the `Sorter` class and the `InputShifterProg` programme.

a) Consider an in-place operation that acts on an int array, and shifts the positions of all elements forward by one step, moving the very last element to the start of the array.

For instance, when this operation is applied to the `int` array {`1,2,3,4,5`} the array would become {`5,1,2,3,4`}. Likewise, if the operation were applied to that output, then the array would become {`4,5,1,2,3`}.

Write a class method for the `Sorter` class called `cyclicShift`, that takes an int array, and performs this operation in-place. It should return nothing (e.g. void return type), instead it will update the input array. Remember to include preconditions and postconditions in your comments.

Now write some code in `InputShifterProg` to test your method.

Hint: *If you are struggling to see how this might work, then first try doing it on paper. Remember you can only perform one operation at a time, and may need additional space to store variables temporarily.*

b) The `cyclicShift` method above is a special case of a more general operation. Consider the operation that shifts elements in an int array by `n` steps, where `n` can be any appropriate integer. For instance, when `n == 3` this would change {`1,2,3,4,5`} to {`3,4,5,1,2`}.

Add an overloaded method, `cyclicShift`, to the `Sorter` class, that takes two arguments: an int array to be modified, and an int that specifies how many steps to shift positions by. You should use the single step `cyclicShift` method within the body of your new `cyclicShift` method. Again, include preconditions and postconditions, and update the test code in `InputShifterProg` to test your new method.

Hint: *Think about applying a one step cyclic shift, over and over again.*

c) [**optional**] The method you have just written is a little inefficient, meaning it does more computation than is necessary. Can you rewrite the two `cyclicShift` methods, so that the general n-step `cyclicShift` method is more efficient, and so that the single-step `cyclicShift` operation uses the more general method within its body.

Hint: *One way to implement the n-step shift is with a temporary array of int variables. The single-step shift is now just one simple case of this more general approach.*

d) [**optional**] Now implement a new method `reorderArray`. This is an in-place array operation method that takes two arguments: an int array to be reordered, called `array`; and a second int array containing the new positions for the elements, called `newPositions`. `newPositions` is an array of desired new positions for the elements of `array`. Therefore, if the value of `newPositions[i]` is `j`, then the `i`th element of `array`, should be moved to position `j`.

For instance, if `array` is initially {`7,8,9,10`} and `newPositions` is {`3,1,2,0`}, then `array` should become {`10,8,9,7`}. What are the preconditions and postconditions?

2 Cards in your Hand [harder]

This will be returning to the `Card`, `Deck` and `Dealer` class that you worked on in Tutorial 2. Make sure you have looked at the solutions to that question before you start this one. Also, you may want to watch the two part video on that question on lecturecast.

Now copy your solution code (or my solution code) to this week's tutorial folder into a subfolder called `cards_in_your_hand`, before starting this question. Make sure you update the package name to reflect the new folder structure.

Many card games involve players holding a small number of cards in their hand. These cards are together often called a **hand**. We are going to try to write a class that represents a hand, and which has some useful functionality associated with it.

a) Create a new class called `Hand` (you will need a new file). This should have:

- an array of `Card`s which is just large enough to hold the cards as an attribute. What is its type? What should you call it? What is its visibility?
- It should have a constructor. When constructed a `Hand` object should have zero `Card`s in it. How many arguments should your constructor take.
- It should have a method `dealTo` that takes a card and inserts it into the `Hand`.

- It should have a method `playFrom` which takes an int indicating the position of the `Card` which should be played, and should return that `Card` and update the `Hand` so that the played `Card` is no longer in the `Hand`.
- a `toString` method that returns a `String` representation of all the cards in the `Hand`.

b) Write some code in the `Dealer` class that tests your class?

c) Can you rewrite the `selectionSort` algorithm from lectures so that it sorts a `Hand` in order?

Hint: *You may want to add a method to the* `Card` *class which compares two cards to determine which should appear earlier in the hand.*