



Expand All

Collapse All

Huntington's disease detector

▼ Is this really how doctors diagnose Huntington's disease?

Yes, pretty much. The genomics era has reduced many problems in biology and medicine to DNA sequencing and string processing.

▼ Why doesn't `removeWhitespace(s)` remove the whitespace in the string referenced by `s`?

Strings in Java are *immutable*, so there is no way to change a string's length or individual characters. The method `removeWhitespace(s)` returns a reference to a *new* string, which you can then assign to the variable `s` via the assignment statement `s = removeWhitespace(s)`. Now, `s` refers to the new string, which has the same characters as the original string, but with the whitespace removed.

▼ Are there other Java whitespace character besides spaces, tabs, and newlines?

Yes. There are form feeds (`\f`), carriage returns (`\r`), and vertical tabs (`\x0B`), but you do not need to worry about them on this assignment.


▼ What's the difference between `replace()` and `replaceAll()`?

Both methods will replace all occurrences of the first argument with the second argument. However, when passed a `String` as the first argument, `replace()` treats it as a string, whereas `replaceAll()` treats it as a *regular expression*.

▼ Can I use regular expressions?


No. We won't introduce regular expressions until *Computer Science: Algorithms, Theory, and Machines*.

▼ Can I implement `removeWhitespace()` by creating a new string and concatenating together all of the non-whitespace characters?


Do not do so using string concatenation: repeatedly concatenating n characters (one at a time) to a `String` object takes time proportional to n^2 . You may do so using the [StringBuilder](#)  data type (a mutable version of the `String` data type): repeatedly appending n characters (one at a time) to a `StringBuilder` object takes time proportional to n .

Kernel filter


▼ How do I get the width and height of a `Picture`? Create a new `Picture`? Get/set the color of a pixel in a given row and column?

Here is a subset of the [Picture API](#) .

▼ How do I get the red, green, and blue components of a pixel? Create a new `Color` object?

Here is a subset of the [Color API](#) .

▼ My program has many cases to handle the periodic boundary conditions. Any tips for simplifying my code?

Use either the remainder operator (`%`) or the [Math.floorMod\(\)](#)  function.

▼ When implementing a 3-by-3 kernel filter, how do I iterate over the 9 pixels centered at a given pixel?

Use a double nested loop, as in the Minesweeper problem (from the Arrays assignment). Same idea also works for a 9-by-9 kernel filter.

▼ Can I define a helper method that applies an arbitrary linear filter to an image?

Yes. In your final design, we recommend that you implement the following helper function:

```
// Returns a new picture that applies an arbitrary kernel filter to the given picture.
private static Picture kernel(Picture picture, double[][] weights)
```

All of the other kernel filters can be implemented by calling this function with a particular matrix. Note that the method is *private*, which means that clients cannot call it directly.

However, to get started, we recommend that you implement `sharpen()` directly, to gain experience with manipulating `Picture` and `Color` objects, and then attempt `kernel()`.

▼ Is applying a kernel filter the same as matrix multiplication?

No. Mathematically, it corresponds to the *convolution* operation. For this reason, the kernel is sometimes referred to as the *convolution matrix*.

▼ Why do the matrix elements sum to 1 in most of the kernels?

It ensures that the resulting image has the same brightness as the original image.

▼ Do kernels always have an equal number of rows and columns? Is the number of rows and columns always odd?

Both properties are very typical, but neither is required. If the number of rows or columns is even, you would need to specify how to align the kernel to the image when processing pixel p .

▼ Why use periodic boundary conditions?

Periodic boundary conditions (also known as *circular* or *toroidal* boundary conditions) are simple to implement, but not the only reasonable choice. Other common boundary conventions include

- *Dirichlet*: treat out-of-bounds pixels as black or zero.
- *Neumann*: compute out-of-bounds pixels by mirror-reflecting the array across the array border.
- *Replicate*: treat out-of-bounds pixel as equal to the nearest array border pixel.

Different boundary conditions work best in different situations. Image-processing libraries often provide the user a choice of boundary conditions.