Exercises: Lambda Expressions Part 3

Some general notes:

- Arrays.asList is a simple way to make a List. E.g.:
  List<String> words = Arrays.asList("hi", "hello", ...);
- List has a useful toString method, so you can directly print a List (unlike an array). E.g.:
  System.out.println(words);
- Remember that Predicate (problems 1 and 2) and Function (problems 3 and 4) are in the java.util.function package.

1. Make a static method called allMatches. It should take a List of Strings and a Predicate<String>, and return a new List of all the values that passed the test. Test it with several examples. E.g.:

- List<String> shortWords = StringUtils.allMatches(words, s -> s.length() < 4);
- List<String> wordsWithB = StringUtils.allMatches(words, s -> s.contains("b"));
- List<String> evenLengthWords = StringUtils.allMatches(words, s -> (s.length() % 2) == 0);

2. Redo allMatches so it works on any List and associated Predicate, not just on Strings. Verify that your examples from #1 still work. But now, you should be able to also do things like this:

- List<Integer> nums = Arrays.asList(1, 10, 100, 1000, 10000);
- List<Integer> bigNums = ElementUtils.allMatches(nums, n -> n>500);

3. Make a static method called transformedList. It should take a List of Strings and a Function<String,String> and return a new List that contains the results of applying the Function to each element of the original List. E.g.:

- List<String> excitingWords = StringUtils.transformedList(words, s -> s + "!");
- List<String> eyeWords = StringUtils.transformedList(words, s -> s.replace("i", "eye"));
- List<String> upperCaseWords = StringUtils.transformedList(words, String::toUpperCase);

4. Redo transformedList so it works with generic types. Verify that your examples from #3 still work. But now, you should also be able to also do things like this:

- List<Integer> wordLengths = ElementUtils.transformedList(words, String::length);

Notice above that I am passing in a List of Strings, but getting out a List of Integer. Make sure your generic types support this idea.