UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS212 - Data structures and algorithms

Practical 1 Specifications: Recursive Array

Release date: 26-02-2024 at 06:00
Due date: 01-03-2024 at 23:59
Total marks: 130

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you start coding.***
- This assignment should be completed individually; no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- You may not import any of the built-in Java data structures. Doing so will result in a zero mark. You may only use native 1-dimensional arrays where applicable. If you require additional data structures, you will have to implement them yourself.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a zero mark. Runtime errors are considered unsafe programming.
- Read the entire specification before you start coding.
- **Ensure your code compiles with Java 8**
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.

2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent), and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers to avoid any misunderstanding.** Also note that the OOP principle of code reuse does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

On completion of this practical, you will have gained experience with the following:

- Implementing a dynamically sized array using recursion.
- Writing basic array operations using recursion.
- Writing a sorting algorithm using recursion.

4 Introduction

Recursion is an important tool for programming, as some problems have solutions that can be easily solved using recursion. Recursion occurs when a function calls itself directly or indirectly through other functions. Recursion is used in conjunction with iteration, as both techniques have their own advantages and disadvantages. Languages like Scheme or LISP does not allow iteration, and the only solution is recursion. For this practical, iteration is not allowed, to ensure that students understand how to use recursion for tasks which they would not usually solve recursively.

Dynamically sized arrays are arrays whose size grows and shrinks as needed. This is done by recreating larger or smaller arrays as needed, and then deleting the old array. A lot of improvements can be made to make this more efficient, but to keep this practical simple, we will follow the less efficient approach, of resizing after every insertion or removal.

5 Warning

For this practical, you are **only allowed to use recursion**. The following rules must be adhered to for the entirety of this practical. Failure to comply with any of these rules **will result in a mark of 0**.

- The words "for","do" and "while" may not appear anywhere in any of the files you upload.
- You are not allowed to create any extra classes, and not allowed to upload any java files not in the following list:
  - RecursiveArray.java
  - Main.java
- Note that you are also not allowed to add extra classes in the allowed files. If the word "class" appears twice in any file you will also receive 0.
- You are not allowed to add any global variables to the classes, or change the global variables given. The only global variables allowed are as follows:
  - RecursiveArray
    - public Integer[] *array*

6 How to convert iteration to recursion

If you are struggling to implement some of the functions recursively, try to first create an iterative solution, as this might be more natural for you to implement. After you have a working iterative solution, then use the following examples, to convert the iterative solution to a recursive solution.

6.1 For loop

Assuming we have a basic for loop, like the one given below.

```
for(int i = 10 ; i < 20 ; i++){
    System.out.println(i);
}
```

The steps to convert this to a recursive function are listed below:

- The looping variable initialization is done in the normal function call.
- The looping variable should be passed as a parameter.
- The looping condition is the base case of the recursive function.
- The looping update statement is passed down in the recursive call.

Thus the recursive version is given below

```
// This is the normal function. It is used to initialize the variables
public static void recursivePrint(){
    recursivePrint(10); // The recursive function is called with the
       initialization condition
}
// This is the recursive function. Note that the return type and parameters can
   be different from the original
private static void recursivePrint(int i){ // All loop variables are passed in
    if(!(i < 20)){ // The loop condition is used as the base case.
        return;
    }
    System.out.println(i); // The body of the loop is called here
    recursivePrint(i+1); // The function is then recursively called with the
       updated loop variable
}
```

6.2 While loop

Let's assume we have a while loop, with a variable outside of the loop which we want to update, like the one below.

```
int i = 0;
String print = "";
while(i <= max){
    if(i != 0){
        print += ",";
    }
    print += String.valueOf(i);
    i += 1;
}
System.out.println(print);
```

The same concept as the **for loop** can be used. The loop condition is the base case. The loop variable is the parameter passed into the recursive function. Techinically you can pass the value which you want to update (String print) as a parameter, but for EO's this is not always possible, as you are not allowed to add helper functions or change the signature. Thus, it is recommended to rather return the value which you want to update. The solution to this is to return the value in the recursive function.

```
// This is the normal function.
public static void recursiveString(int max){
    // Since we want to update a String variable, the recursive helper function
       returns a String
    // Thus we print out the result of the recursive call.
    System.out.println(recursiveString(max,0));
}

// The max variable is passed in, which won't change and the i variable is the
   loop variable
private static String recursiveString(int max,int i){
    // This is not quite the loop condition. This is because with the normal
       loop condition, it is
    difficult to remove the extra , at the end. Thus we end one earlier and
       leave the comma out.
    if(i == max){
        return String.valueOf(i);
    }
    // Note that we can append the recursive call to the back of the string.
    return String.valueOf(i) + "," + recursiveString(max,i + 1);
}
```

7 Tasks

For this practical you are only allowed to work in one class, the `RecursiveArray` class. You are also not allowed to use any iteration. The UML for this class is given below. Please note that you are not allowed to add extra members, but you are allowed to add helper functions.

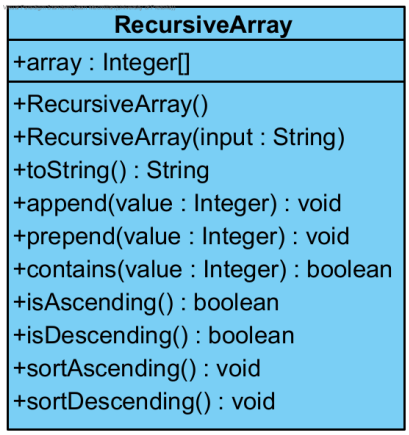

Figure 1: UML

- Members
  - array: Integer[]
    - This is an array of Integers, which will be the data for this object.
    - This array will always be perfectly sized, thus there will never be a null in the array.
    - If the array is empty, then it should be a size of 0.

- Functions
  - RecursiveArray()
    - This is the default constructor, which should create an empty array.
  - RecursiveArray(input: String)
    - This constructor takes in a String. This should then be converted to an array.
    - If the String is empty, then create an empty array.
    - If the String is not empty you may assume that the input is valid. The input will be an unknown number of integers, which will be seperated by commas.
    - An example of this is below:

```
1,2,3,4,5
```

  - toString(): String
    - This should return a String representation of the array.
    - If the array is empty, then return

```
Empty Array
```

    - If the array is not empty, then print all of the values, each value seperated by a comma. The String should start with "[" and end with "]".
    - Example

```
[1,2,3,4,5]
```

  - append(value: Integer): void
    - This should append the passed-in parameter to the end of the array.
    - You may assume that the passed-in parameter is not null.
    - The array should be perfectly sized, so the size should increase by 1.
  - prepend(value: Integer): void
    - This should prepend the passed-in parameter to the start of the array.
    - You may assume that the passed-in parameter is not null.
    - The array should be perfectly sized, so the size should increase by 1.
  - contains(value: Integer): boolean
    - This should return true if there is a value in the array which matches the passed-in array, otherwise return false.
  - isAscending(): boolean
    - This should return true if the array is in ascending order.
    - This means that every element is less than or equal to all elements coming after it.
    - The following are also considered in ascending order:

```
[]
[1,1,1,1]
```

- isDescending(): boolean
  - This should return true if the array is in descending order.
  - This means that every element is greater than or equal to all elements coming after it.
  - The following are also considered in descending order

```
[]
[1,1,1,1]
```

- sortAscending(): void
  - This function should sort the array in ascending order.
- sortDescending(): void
  - This function should sort the array in descending order.

8 Testing

As testing is a vital skill that all software developers need to know and be able to perform daily, approximately 10% of the assignment marks will be assigned to your testing skills. To do this, you will need to submit a testing main (inside the Main.java file) that will be used to test an Instructor-provided solution. You may add any helper functions to the Main.java file to aid your testing. In order to determine the coverage of your testing the jacoco tool. The following set of commands will be used to run jacoco:

```
javac *.java
rm -Rf cov
mkdir ./cov
java -javaagent:jacocoagent.jar=excludes=org.jacoco.*,destfile=./cov/output.exec
    -cp ./ Main
mv *.class ./cov
java -jar ./jacococli.jar report ./cov/output.exec --classfiles ./cov --html
    ./cov/report
```

This will generate output which we will use to determine your testing coverage. The following coverage ratio will be used:

$$\frac{\text{number of lines executed}}{\text{number of source code lines}}$$

and we will scale this ratio according to the size of the class.
The mark you will receive for the testing coverage task is determined using table 1:

Coverage ratio range	% of testing mark
0%-5%	0%
5%-20%	20%
20%-40%	40%
40%-60%	60%
60%-80%	80%
80%-100%	100%

Table 1: Mark assignment for testing

Note the top boundary for the Coverage ratio range is not inclusive except for 100%. Also, note that only the function stipulated in this specification will be considered to determine your mark. Remember that your main will be testing the instructor-provided code and as such it can only be assumed that the functions outlined in this specification are defined and implemented.

9 Upload checklist

The following files should be in the root of your archive

- Main.java
- RecursiveArray.java

10 Allowed libraries

- None

11 Submission

You need to submit your source files on the FitchFork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place the above-mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code must be able to be compiled with the Java 8 standard.

For this practical, you will have 5 upload opportunities and your best mark will be your final mark. Upload your archive to the appropriate slot on the FitchFork website well before the deadline. **No late submissions will be accepted!**