Name: Samruddhi Vidyadhar Pataskar
University: Indian Institute of Technology, Kharagpur
E - mail: samruddhipataskar.pune@gmail.com

Adding new extensions to index-out-of-bound type checker

Motivation:

The current type checker for checking array indices to ensure that no index results in illegal memory access limits certain programming patterns from use. I have learnt this from a case study, which I have included in this document.

I would like to add some extensions to make these common programming patterns allowed in programs which are compiled using the type checker.

I would like to add extensions for the following:

1. Using multiple variables as aliases to array length.
2. Allow use of mathematical expressions as array indices.
3. Reduce redundant checking of indices when array is passed to functions.

1. Using multiple variables as aliases to array length

It is a common practice to use a variable to store the length of an array which will be taken dynamically from the user. This variable often remains unchanged throughout the course of the program. It is known to the user that this variable stores the upper bound of the array and hence, programmers commonly use this as an alias to upper bound of the array.

The type checker insists that the length of array be used for whenever checking indices against upper bounds instead of these variables.

I would like to add the following annotations to the checker:

@UpperBound to indicate that the variable can be used as upper bound of an array when required.

example: @UpperBound int n;

It will fire an error when any one of the following rules is violated:

1. If value of n cannot be guaranteed to be at least zero.
2. If any statement attempts to modify the value of n.
3. If n remains uninitialised.

@Expr for statements which use mathematical expressions as array indices.

It will fire an error when any one of the following rules is violated:

1. The value of any variable in expression cannot be determined.
2. The mathematical expression itself can possibly go beyond bounds of the array.

@Argument to indicate that the array will be passed as function arguments to some function called.

It will prevent checking of some conditions in the called function.

It will fire an error when any one of the following rules is violated:

1. The array annotated with this is not passed as argument to any function.
2. The called function attempts same checks as the calling function.

I am the perfect fit for this project as I have been learning Java since I was 16 and I have experienced most of the common programming problems some of which I believe I can eradicate through my work on this project.

URL to a calculator I made using Java:

https://github.com/samruddhi3/calci-app/blob/master/calculator.java

URL to the code I used for my case study:

https://github.com/samruddhi3/gsoc_case_study/blob/master/Fibon.java

Additional trainings:
I have been trained in Core Java at Prompt Info Services, Pune.
I had a course on Java Programming in School when I was in the 10th grade.

Name: Samruddhi Vidyadhar Pataskar

Case Study for index-out-of-bound errors type checker

I have used the index-out-of-bound error type checker of CheckerFramework 2.1.9 on a basic Java program.

The program used for this case study includes simple operations on arrays. But there are many different ways of accessing array elements. Hence, it is a good example for the case study as it puts the type checker through many different scenarios.

The program includes:
1. Creating an array of n integers. Where n is taken as input from user i.e. creating array dynamically.
2. Accessing array elements using constant index.
3. Accessing array elements using variable as index.
4. Accessing array elements inside a loop.
5. Accessing array elements outside a loop.
6. Accessing array elements using mathematical expressions.
7. Passing the array element as argument to a function.

Program used for case study:

Program to print Fibonacci numbers. Accept number of numbers of the Fibonacci series from the user.

```
import java.io.*;
import java.util.*;
class Fibon {
  public static void print(int a[]) {
       int i;
       for(i = 0; i < a.length; i++)
            System.out.println(a[i]);
  }
  public static void calculate(int n) {
       int a[] = new int[n];
       int i;
       a[0] = a[1] = 1;
       for(i = 2; i < n; i++)
            a[i] = a[i - 1] + a[i - 2];
       print(a);
  }
  public static void main(String args[]) {
       Scanner in = new Scanner(System.in);
       int n = in.nextInt();
       calculate(n);
  }
}
```

Program 1.

Command used to compile Java program:

```
javac  -processor index Fibon.java
```

1. Declaring new array dynamically

Line number 10:

```
int a[] = new int[n];
```

Creates a new array at run time. Value of variable n is provided by user at run time through main method.

Result of compilation using the upper stated command:

```
Fibon.java:12: error: [array.length.negative] Variable used in array
creation could be negative.
        int a[] = new int[n];
                          ^
  found   : @LowerBoundUnknown int
  required: an integer >= 0 (@NonNegative or @Positive)
```

Compiling without using the checker framework does not fire any error on this line. The value of n being negative will only be detected as a run time error. After which the program will crash. The following exception resulted when I gave -3 as input to the program for array size :

```
Exception in thread "main" java.lang.NegativeArraySizeException
    at Fibon.calculate(Fibon.java:12)
    at Fibon.main(Fibon.java:22)
```

Since the program contains no explicit check on the value of n the type checker fired the corresponding error.

The type checker must not fire this error if it is known that n is at least zero. To test this I added some more lines to the program and compiled it again.

```
import java.io.*;
import java.util.*;
class Fibon {
  public static void print(int a[]) {
       int i;
       for(i = 0; i < a.length; i++)
           System.out.println(a[i]);
  }
  public static void calculate(int n) {
       if(n < 0)
           System.exit(0);
       int a[] = new int[n];
       int i;
       a[0] = a[1] = 1;
       for(i = 2; i < n; i++)
           a[i] = a[i - 1] + a[i - 2];
       print(a);
  }
  public static void main(String args[]) {
       Scanner in = new Scanner(System.in);
       int n = in.nextInt();
       calculate(n);
  }
}
```

Program 2.

The new lines ensure that n is greater than zero before it is used to create the new array.

After compilation the type checker fires no error for the line which creates the new array dynamically.

2. Accessing array elements using constant index

Line 12 of Program 1. attempts to access the first two elements of array a.

```
    a[0] = a[1] = 1;
```

The error given by the type checker for this line is as follows.

```
Fibon.java:12: error: [array.access.unsafe.high] Potentially unsafe
array access: the index could be larger than the array's bound
        a[0] = a[1] = 1;
          ^
  found   : @UpperBoundUnknown int
  required: an integer less than a's length (@LTLengthOf(“a"))


Fibon.java:12: error: [array.access.unsafe.high] Potentially unsafe
array access: the index could be larger than the array's bound
        a[0] = a[1] = 1;
                 ^
  found   : @UpperBoundUnknown int
  required: an integer less than a's length (@LTLengthOf(“a"))
```

The upper bound is unknown for array a. Hence, this error was encountered.

As similar to the previous case, I included some more lines to try and eradicate this error.

```
1. import java.io.*;
2. import java.util.*;
3. class Fibon {
4.   public static void print(int a[]) {
5.        int i;
6.        for(i = 0; i < a.length; i++)
7.             System.out.println(a[i]);
8.   }
9.   public static void calculate(int n) {
10.       if(n < 0)
11.            System.exit(0);
12.       int a[] = new int[n];
13.       int i;
14.       if(n > 1)
15.            a[0] = a[1] = 1;
16.       else
17.            System.exit(0);
18.       for(i = 2; i < n; i++)
19.            a[i] = a[i - 1] + a[i - 2];
20.       print(a);
21.  }
22.  public static void main(String args[]) {
23.       Scanner in = new Scanner(System.in);
24.       int n = in.nextInt();
25.       calculate(n);
26.  }
27.}
```

Program 3.

After compiling again I see the same error that was as before. Although the size of array is now explicitly guaranteed to be at least 2 the type checker still thinks that accessing the first two members is invalid as it could still be out-of-index of the array.

In case of dynamically created arrays the type checker fails to check against upper bound of the array.

But if we use a.length instead of n in line 14(Program 3.) The program fires no errors.

Since n was used to take input from the user and not modified in the process, there is no problem in using it as upper bound of the array. The type checker does not allow this.

This limits the programmers ability to use other representatives of a.length.

3. Accessing array elements using variables as index

In the method print(), line 7 of program 1. accesses the array elements for printing one by one.

The type checker fires no error.

The for loop used for the purpose of printing already puts constraints on the way elements are accessed. The loop variable i is well within the range which allows for safe access to all array elements.

I tweaked the program to some extent to see errors produced by the type checker:

I got the following error for some of the tweaks:

```
Fibon.java:7: error: [array.access.unsafe.high] Potentially unsafe array
access: the index could be larger than the array's bound
            System.out.println(a[i]);
                                 ^
```

The tweaks were aimed to increase the range of i beyond length of array a.

I had made the following changes to the loop condition, one by one:
change a.length to a.length + 20,
change a.length to 64,
change a.length to a.length - 5

After this I changed the program to see some more errors produced by the code.

I got the following error for another change that I made:

```
Fibon.java:7: error: [array.access.unsafe.low] Potentially unsafe array
access: the index could be negative.
            System.out.println(a[i]);
                                 ^
  found   : @LowerBoundUnknown int
  required: an integer >= 0 (@NonNegative or @Positive)
```

I had changed the initialization of i from 0 to -5.

The type checker uses conditions of the loop to check illegal access of array elements.

Even here it is not allowed to use other representatives of array length to check for upper bound of array.

4. Accessing array elements inside a loop

Lines 6 and 7 are used to access elements in a loop, as I have discussed before the type checker uses the conditions of the loop variable to see if there is any illegal memory access.

The type checker successfully checks the upper bound of the array by referring to the loop condition.

It checks the lower bound by seeing if `i` has been initialised to a nonnegative value. That is the only check performed.

If `i` has been initialised to any positive value the type checker does not fire an error.

There is no need to check if the initialisation makes `i` greater than `a.length` because if this is the case then the loop statement will not be executed at all and hence, there will be no illegal memory access.

5. Accessing array elements outside a loop

Any access outside a loop requires that the element being used as index for the array is subjected to check explicitly before the statement used to access the element.

Here a.length must be used. No other variable which is an alternative to length of array can be used.

6. Accessing array elements using mathematical expressions

Line 14 of program 1. attempts to calculate values of next numbers in the Fibonacci series by using the recurring relation of Fibonacci numbers.

But the type checker fires the error:

```
Fibon.java:19: error: [array.access.unsafe.low] Potentially unsafe array
access: the index could be negative.
            a[i] = a[i - 1] + a[i - 2];
                                  ^
  found   : @GTENegativeOne int
  required: an integer >= 0 (@NonNegative or @Positive)
```

The type checker does not check the expression. Although the expression and loop conditions guarantee that the expression does not result in illegal memory access.

The programmer cannot use logic in the program which results naturally from the situation at hand. Instead he must use new variables for the logical expressions and also place extra checking conditions on them.

I changed the program to see if explicit conditions can make a difference:

```java
import java.io.*;
import java.util.*;
class Fibon {
  public static void print(int a[]) {
      int i;
      for(i = 0; i < a.length; i++)
          System.out.println(a[i]);
  }
  public static void calculate(int n) {
      if(n < 0)
          System.exit(0);
      int a[] = new int[n];
      int i;
      if(a.length > 1)
          a[0] = a[1] = 1;
      else
          System.exit(0);
      for(i = 2; i < n; i++)
          if(i - 2 >= 0)
              a[i] = a[i - 1] + a[i - 2];
      print(a);
  }
  public static void main(String args[]) {
      Scanner in = new Scanner(System.in);
      int n = in.nextInt();
      calculate(n);
  }
}
```

This still produces the same error even though the index expression explicitly is within bounds of legal memory access.

7. Passing array as argument to a function

`print` method needs an array as input. If the array is being passed to another function then all the checks performed by the checker on the array in the calling function must be performed on the array in the called function.

This is helpful if the function takes many different arrays at different times as input.

But is inefficient if the array before being passed already has been checked for certain restrictions such as,

Line 12 in program 1 checks if a[0] and a[1] cause illegal memory access. It causes the user to put explicit checking conditions. If this line is copied into `print` function at line 6 the type checker fires the error:

```
Fibon.java:6: error: [array.access.unsafe.high] Potentially unsafe array
access: the index could be larger than the array's bound
        a[0] = a[1] = 1;
          ^
  found   : @UpperBoundUnknown int
  required: an integer less than a's length (@LTLengthOf(“a"))


Fibon.java:6: error: [array.access.unsafe.high] Potentially unsafe array
access: the index could be larger than the array's bound
        a[0] = a[1] = 1;
                 ^
  found   : @UpperBoundUnknown int
  required: an integer less than a's length (@LTLengthOf("a"))
```

But it is known that this whenever this function is called from `calculate` method, this check is redundant.

Conclusion:

This particular type checker does not need many annotations as other preliminary checks such as if conditions and loop conditions suffice to keep check on index values.

The type checker for index-out-of-bound errors makes use of the fact that lower bound of an array has index 0 and upper bound can be calculated using `<array_name>.length.`

The user cannot use any other variable to represent the length of the array. In some cases other variables may be used to store the array length naturally, especially when dynamically created.

It does not support the common coding practice of using mathematical expressions as array indices.

It is inefficient when tasked with passing array to functions as it performs preliminary checks on those arrays again.

I would like to work on this project to fix these shortcomings.

SAMRUDDHI PATASKAR

samruddhipataskar.pune@gmail.com

+91 7477787222

Kharagpur,W.Bengal

EDUCATION

Master of Science (M.sc.), Chemistry

(2016-2021)

Indian Institute of Technology (IIT), Kharagpur, W.Bengal.

XII (Intermediate),Science

Year of Completion : 2016

Telangana State Board of Intermediate Education

Percentage : 97.20 %

X (Secondary)

Year of Completion : 2014

ICSE Board (St Mary's School,Pune)

Percentage : 96.80 %

TRAININGS

National Science (Vijyoshi) camp at IISc,Bengaluru 2015:

Participated in the National Science (VIJYOSHI) Camp 2015 organized by the Kishore Vaigyanik Protsahan Yojana (KVPY) in association with INSPIRE Program, DST and IISER- Kolkata during 18-20 Dec 2015 , at IISC, Bangaluru.

Training in Core Java

Training in programming in Core Java at Prompt Info Services, Pune in 2014.

PROJECTS

Dr. Homi Bhabha Young Scientist's Talent Search Competiton(2012 - 2013)

Worked on a project towards eradication of dengue as part of this competition at the state - level(senior group).

SKILLS

Writing:

Won All India Essay Writing Event 2016 conducted by Shri Ram Chandra Mission-**ZONAL WINNER 1st** (ENGLISH –Category II)

Communication:

Trinity Guildhall award, LONDON :

Awarded TRINITY GUILDHALL in foundation Level Graded examination-" Grade 2 Speech & Drama" with merit, conducted by Trinity College, London (Nov 2008).

LAMDA Level 1 award in communication:

The London Academy of Music and Dramatic Art awarded LAMDA Level 1
Award in communication –"Speaking Verse & Prose –Grade 3" with distinction (March 2010).

Best Speaker award:

Awarded Inter House Extempore Winner Best Speaker on 3rd September, 2013, at st. Mary's School, Pune.

ADDITIONAL DETAILS

NTSE Scholarship Award 2012

Appeared in National Talent Search Examination 2012 conducted by NCERT New Delhi & qualified for the award of scholarship & certificate of merit.

KVPY Fellowship Award 2014:

Based on the performance in the Aptitude test as well as Interview held during November 2014 and January/February 2015, rewarded KVPY fellowship award under the stream SA-2014.

Dr. Homi Bhabha Young Scientist's Talent Search competition:

Honoured by conferring upon her **Silver Medal** for displaying proficiency at Dr. Homi Bhabha Young Scientist's Science Talent Search competition

2012-13 (State Level-Senior Group). This is organized by the Mumbai Science Teacher's Association.

Bhaskar Genius Scholarship award 2012:

Awarded a certificate for securing consolation position in Zone 4 (Maharashtra) in Bhaskar Genius Scholarship award- 2012 conducted by Divya Marathi & Manipal University.

ICSE-2014:

Class X: Scored 100% marks in Mathematics & won the Dr. Neelkant A.

Kalyani Prize (28-6-2014).

Scored 100% in Computer Applications also.

Class VII: Rewarded General Proficiency first prize (30-9-2011)

Class VIII: Rewarded General Proficiency Third Prize (28- 9-2012)

Inter school maths Quiz competition-2011:

Gold medal winner- Junior group
(Conducted by Rotary club & Pune District Mathematics Teacher's Association)

Inter school Maths Quiz competition 2013:

Gold medal winner- Senior
(Conducted by Rotary club & Pune District Mathematics Teacher's Association)

NSTSE 2014:

Participated in the National Level Science Talent Search Examination 2014 conducted nation wide on 2nd February, 2014 by Unified Council with excellent performance securing 80 percentage of marks. Secured All India Rank-1019 & with State Rank -71

MTSE 2012:

Awarded a certificate in appreciation of meritorious performance in Maharashtra Talent Search Examination for STD VIII conducted at state level by Centre for Talent Search & excellence, N.Wadia College, Pune-1 on 21st April 2012 & Secured State Level Rank - 19.

MTSE 2013:

Awarded the certificate in appreciation of meritorious performance in Maharashtra Talent Search Examination for STD IX conducted at state level by Centre for Talent Search & excellence, N.Wadia College, Pune-1 on 27th April, 2013& Secured State Level Rank - 13.

MTSE 2014:

Awarded a certificate in appreciation of meritorious performance in Maharashtra Talent Search Examination for STD X conducted at state level by Centre for Talent Search & excellence, N.Wadia College. Pune-1 on 28th April 2014 & Secured State Level Rank - 10.

IMO 2014:

Secured International Rank - 1078 and State Rank - 110 in Maharashtra & Goa Zone in the finals of 7th Science Olympiad Foundation (SOF)- International Mathematics Olympiad Competition held in February 2014.

NSO 2014:

Secured International Rank - 1090 & State Rank - 99 in Maharashtra & Goa Zone in the finals of 16th Science Olympiad Foundation (SOF)- National Science Olympiad Competition held in February 2014.

NSO 2013:

Secured International Rank - 445 & State Rank - 35 in Maharashtra & Goa zone in the finals of 15th SOF - National Science Olympiad Competition held in February 2013.

IMO 2013:

Secured International Rank - 89 and State Rank - 12 in Maharashtra & Goa zone in the finals of 6th Science Olympiad foundation (SOF) -International Mathematics Olympiad Competition held in February 2013.

IMO 2011:

Secured Rank - 130 in the final round of 4th International Mathematics Olympiad conducted by Science Olympiad Foundation (SOF) in March 2011.

All India Open Mathematics Scholarship Examination:

Shown excellent performance in the finals of All India Open Mathematics Scholarship Examination during the year 2010 (std 6), year 2011 (std 7) and also shown meritorious performance after entering into the Mega finals of the same competition during the year 2012 (std 8) and year 2013 (std 9). This is the competition conducted by Institute of promotions of Mathematic, Pune.

IMO 2011:

Participated & secured State Rank - 6 and achieved Olympiad rank in the 2nd level of International Olympiad of Mathematics 2011, conducted by Mathematics Olympiad Foundation, New Delhi, India.

Ganit Olympiad 2011-12:

Rewarded certificate of Merit for achieving grand success in the "Ganit Olympiad 2011-12" held on 19th February, 2012 organized by Pune District Mathematics Teacher's Association & Lions club of Poona, Ganeshkhind.

Mathematics NISHNAT examination 2013-14:

Stood first in NISHNAT exam conducted by Pune District Mathematics Teacher's Association, Pune.

Rotary Ganit Olympiad 2013:

Won 1st prize in Rotary Ganit Olympiad held on 6th October, 2013 Organized by Pune District Mathematics Teacher's Association & Rotary club of Pune Pride.

Ganit Pravinya Pariksha 2008:

Achieved first class in the Ganit Pravinya Pariksha 2008 (std. 5th) conducted by Pune Zilla Ganit Adhyapak Mandal, Pune

Ganit Pravinya Pariksha 2012

Achieved first class with distinction in Ganit Pravinya Pariksha 2012 (std. 8th) conducted by Pune Zilla Ganit Adhyapak Mandal, Pune.

भरतनाट्यम – Dance exams

Conducted by आखिल भारतीय गंधर्व महाविद्यालय मंडळ, मुंबई.

प्रारंभिक -	Nov 2008	-	passed in Grade I.
प्रवेशिका PRATHAM -	Nov 2009	-	passed in Grade III.
प्रवेशिका PURNA -	Nov 2010	-	passed in Grade I.