User Manual

Description
Runtime exceptions can cause a lot of problems in code. They can crash entire systems and lose important data. One of these exceptions is an index out of bounds exception. Our goal is to expand the type system to be able to guarantee that code will not throw that exception and crash, but instead be found at compile time. This would be extremely useful to developers because catching fixing bugs early is significantly cheaper than finding them later after building and during testing. Being able to prove that you code will not crash in this way before releasing it can avoid a lot of the problems associated with crashing programs.

The IndexChecker helps with this task. By simply running our checker over files that use array indexing, warnings will be generated at possible illegal array accesses.

Installation
We currently do not have a user release for the index checker. In order to use it in its current state, you must download its source and build it as a developer would have to. See the Developer Manual on how to:

- Obtain Source Code
- First time build

Running the Checker
Now the checker framework and index checker should be installed and set up. The index checker can be run with the following command:

javac -processor org.checkerframework.checker.index.IndexChecker <*JavaFileName*>.java

The normal javac compiler should be run with the addition of warnings generated on potentially incorrect array indexes. In the case that something does not work, see the checker framework manual's trouble shooting section:
http://types.cs.washington.edu/checker-framework/current/checker-framework-manual.html#troubleshooting

Writing Annotations:
When declaring an int variable, that is a valid index for an array a you create it as such @IndexFor(a) int index = x;

You can only access array a without warning using @IndexFor(a) int.

If you increment the index by 1, you should ensure the resulting value is strictly less than the length of the array before using it to access the array.

If you decrement the index by 1, you should check the resulting value is greater or equal to 0.

If you have an unknown value you should check it against both of those constraints.

The resulting value of applying any other arithmetic operation should be treated as unknown i.e. requiring to check both bounds.

Loops:

```
// this loop works properly no warnings
for(@IndexFor("arr") int i = 0; i < arr.length; i++){
    o = arr[i];
}

// due to an improper bounds check this will issue a warning for unsafe array access
for(@IndexFor("arr") int i = 0; i<= arr.length; i++){
    o= arr[i];
}

// this works fine
for(@IndexFor("arr") int i = arr.length -1 ; i > -1 /* or >= 0*/; i--){
    o= arr[i];
}
```

Unknown index value:

```
int index = input;

o = arr[index]; // issues a warning: index could be out of bounds

// this works fine no warnings because bound were properly checked
if(index > -1 /* or >= 0*/ && index < arr.length){
    o = arr[index];
}
```

Use Index Checker to find Non-Obvious out of bounds

Developer tries to do something with two arrays but doesn't check input properly. Because of this sometimes he runs into errors he doesn't know the cause of.

```
4⊖    public static void main(String[] args){
5         int[] values = getvalues();
6         int valIndex = getinput();
7         if(valIndex < values.length){
8             System.out.println(values[valIndex]);
9         }
10    }
```

Problems @ Javadoc Console Terminal Git Staging Error Log wa

```
<terminated> UseCase2 [Java Application] /homes/iws/jsantino/Downloads/jdk1.8.0_65/jr
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -1
        at proposal.UseCase2.main(UseCase2.java:8)
```

So he runs the checker to get warnings and find the error.

```
    public static void main(String[] args){
        int[] values = getvalues();
        int valIndex = getinput();
        if(valIndex < values.length){
            System.out.println(values[valIndex]);
        }
    }
```

roblems @ Javadoc Console Terminal Git Staging

```
Warning unsafe array access at line 8
    found       values[@LTLength("values") i]
    expected    values[@IndexFor("values") i]
```

The found type indicates that the value was not checked against zero, and therefore shows why the error happens and how to fix it.

Using the Index Checker to find possible silent errors

A developer has code that uses a lot of arrays and they find that their output isn't what they expect. In order to make sure the errors are not with the array access they use the Index Checker to catch some types of silent errors.

He expected to see 59, the highest index in arr2 but for some reason he got 0.

```
8    public static void main(String[] args){
9        int[] arr = new int[30];
10       int[] arr2 = new int[60];
11       for(int i = 0; i < arr.length; i++){
12           arr[i] = i;
13       }
14       for(int i = 0; i < arr.length; i++){
15           arr2[i] = i;
16       }
17       System.out.println(arr2[arr2.length-1]);
18   }
19 }
20
```

Problems @ Javadoc Console Terminal Git Stagi

```
<terminated> UseCaseUnAnnotated [Java Application] /homes/iws/
0
```

```
8    public static void main(String[] args){
9        int[] arr = new int[30];
0        int[] arr2 = new int[60];
1        for(@IndexFor("arr")int i = 0; i < arr.length; i++){
2            arr[i] = i;
3        }
4        for(@IndexFor("arr2")int i = 0; i < arr.length; i++){
5            arr2[i] = i;
6        }
7        System.out.println(arr2[arr2.length-1]);
8    }
9 }
0
```

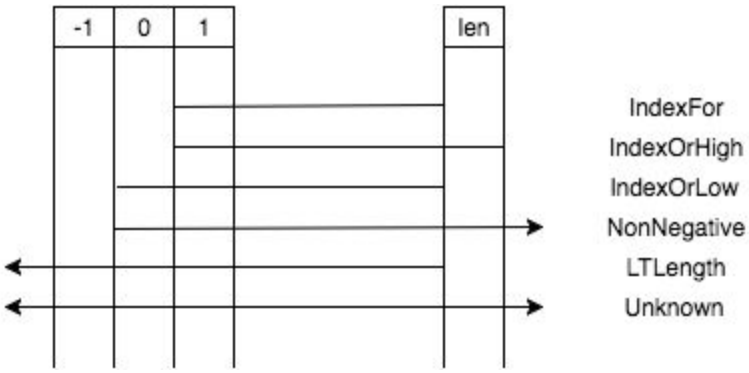
Problems @ Javadoc Console Terminal Git Staging Error Log

```
1 Warning unsafe array access at line 15
2     found       arr2[@IndexFor("arr") i]
3     expected    arr2[@IndexFor("arr2") i]
```

The warning shows that the index is for the wrong array(he checked the wrong condition). He can now fix the mistake and get the correct behaviour.

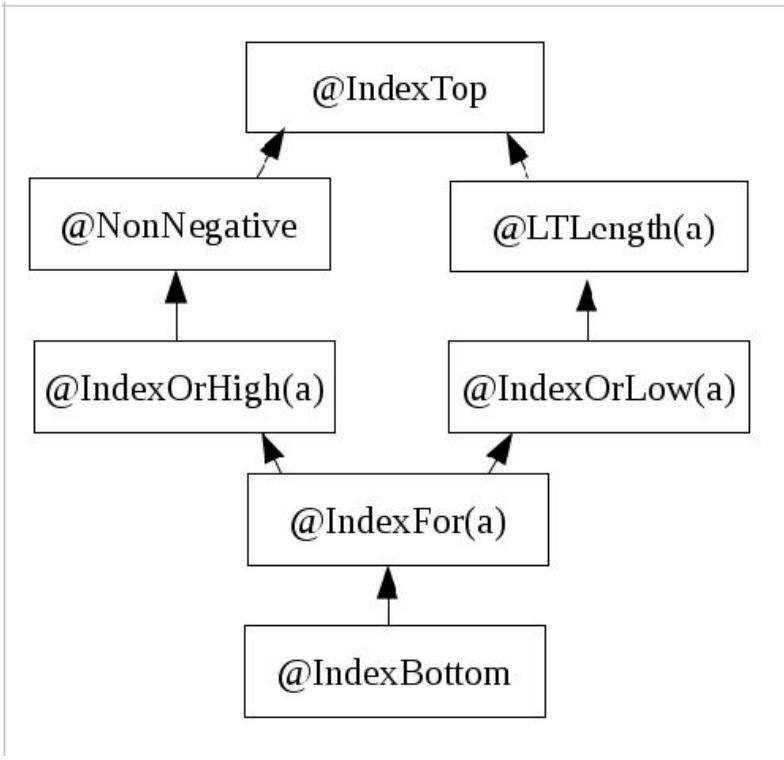
What are the Types:

The following is a description of every annotated type included in this checker. It is not expected that any but IndexFor() will need to be explicitly written by developers but they may be helpful to understand warnings and for debugging purposes.



Type	Value	Description
@Unknown int i	i can be any value	
@IndexOrHigh(a) int i	i >= 0 && i < a.length + 1	Possibly one too high
@IndexOrLow(a) int i	i >= -1 && i < a.length	Possibly one too low
@NonNegative int i	i >= 0	Possibly too high
@LTLength(a) int i	i < a.length	Possibly too low
@IndexFor(a) int i	i >= 0 && i < a.length	Safe
@IndexBottom int i	i can’t be anything	

Lattice:



Intro Rules:

Literals 0 and above have type @NonNegative
a.length has type @IndexOrHigh(a)
Literal -1 has type @IndexOrLow()
Any other int has type @Unknown

Type Rules:

Only @IndexFor("a") can be used for accessing array a.

Data Flow Rules:

Note that the mention of a type represents all proper heirs and the type itself. Rules can be overwritten by heirs.

@IndexFor(a)	+ 1	= @IndexOrHigh(a)
@IndexOrLow(a)	+ 1	= @IndexOrHigh(a)
@IndexFor(a)	+ @NonNegative	= @NonNegative
@IndexFor(a)	- 1	= @IndexOrLow(a)
@IndexOrHigh(a)	- 1	= @IndexOrLow(a)
@IndexFor(a)	- @NonNegative	= @LTLength(a)

Any other arithmetic operation performed on any index results in @Unknown.

This table describes how types are refined through composition operations.
If any type in the Type column passes any comparison in the same row, it will have the new type shown in the scope the comparison is valid.

Type	Comparison	New type	Reason
@IndexOrHigh(a) @NonNegative(a)	< @IndexOrHigh(a) <= @LTLength(a)	@IndexFor(a)	Proves < length
@IndexOrLow(a) @LTLength(a)	> @IndexOrLow(a) >= @NonNegative	@IndexFor(a)	Proves > -1
@Unknown	> @IndexOrLow >= @NonNegative	@NonNegative	Proves > -1
@Unknown	< @IndexOrHigh(a) <= @LTLength(a)	@LTLength(a)	Proves < length

For example this first line describes this situation:

```
@IndexOrHigh(a) int i;
if(i < a.length){
        //Inside here i is @IndexFor(a);
}
```

Bug Reporting
Our issue tracker can be found at https://github.com/gyhughes/checker-framework/issues.
From here, click New Issue in the top right, and submit a bug report there.
More info can be found at https://guides.github.com/features/issues/.