Name : Siddhant Kotak

Student ID : 202201410

Course : IT 314 Software Engineering

Professor : Saurabh Tiwari

Semester : Autumn 2024

Lab : Program Inspection, Debugging and Static Analysis

Stack Implementation

Given code :

```
//Stack implementation in java
import java.util.Arrays;
public class StackMethods {
   private int top;
   int size;
   int[] stack ;
   public StackMethods(int arraySize){
      size=arraySize;
      stack= new int[size];
      top=-1;
   }
   public void push(int value){
      if(top==size-1){
         System.out.println("Stack is full, can't push a value");
      }
      else{
         top--;
         stack[top]=value;
      }
   }
   public void pop(){
      if(!isEmpty())
         top++;
      else{
         System.out.println("Can't pop...stack is empty");
      }
   }
   public boolean isEmpty(){
        return top==-1;
    }
    public void display(){
        for(int i=0;i>top;i++){
            System.out.print(stack[i]+ " ");
        }
        System.out.println();
    }
}
public class StackReviseDemo {
    public static void main(String[] args) {
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);
        newStack.display();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display();
    }
}
```

```
output: 10
        1
        50
        20
        90
        10
```

Program Inspection of Stack Implementation:

1. **How many errors are there in the program? Mention the errors you have identified.**
   Errors identified: 3

   - **Error 1: Incorrect operation in push method (top-- instead of top++)**
     - **Category**: Logic Error
     - In the push method, the top-- should be top++. Since the stack starts with top initialized to -1, each time a new element is pushed, the top should increment to indicate the next position in the stack.
   - **Error 2: Incorrect condition in the display method (wrong loop condition)**
     - **Category**: Logic Error
     - The loop condition for(int i=0;i>top;i++) is incorrect. The loop should run from i = 0 to i <= top, meaning i < top + 1 for a valid display of stack elements.
   - **Error 3: Incorrect operation in pop method (top++ instead of top--)**
     - **Category**: Logic Error
     - When pop is called, the value of top should be decremented (top--) to reflect the removal of an element from the stack. The current top++ will increase the position instead of decreasing it.

2. **Which category of program inspection would you find more effective?**

Most effective category: Logic Errors

- **Reason**: The key issues in this program are related to incorrect logic in the push, pop, and display methods. Program inspection focusing on logical flow, especially with regard to array indices and boundary conditions, is crucial for detecting these errors.

3. **Which type of error you are not able to identify using the program inspection?**

Missed error type: Memory management or array overflow issues in dynamic stacks

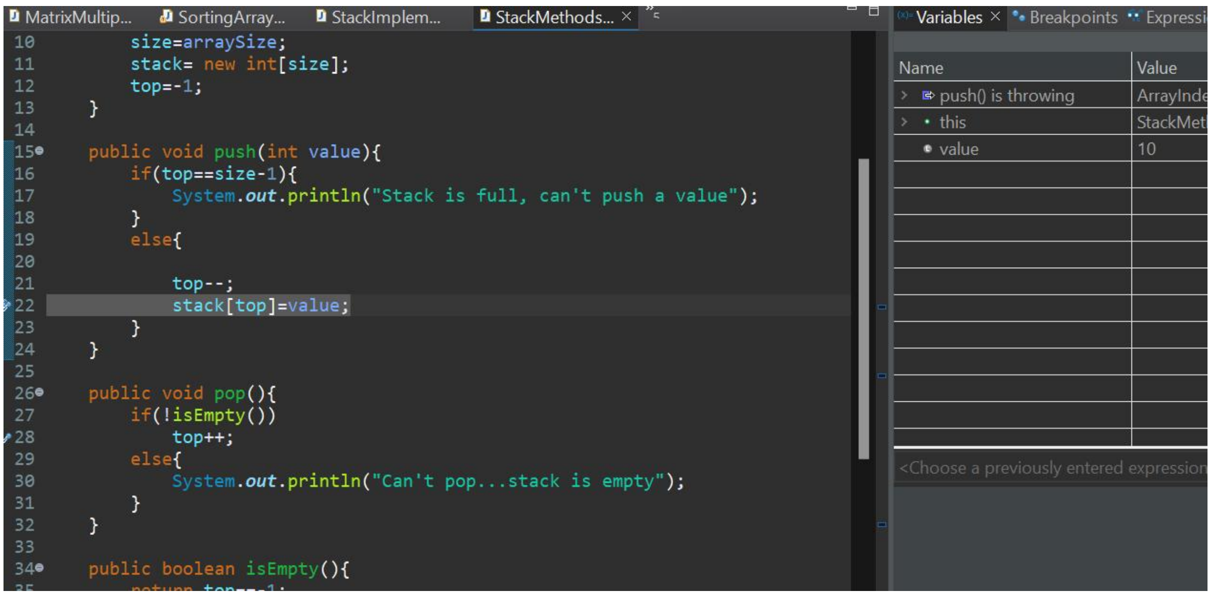
- **Reason**: Since this stack has a fixed size, memory issues may not arise in this context. However, if a dynamically sized stack were implemented, errors related to memory management (such as resizing arrays or handling large inputs) would not be detectable through static inspection alone.

4. **Is the program inspection technique worth applying?**

Applicability: Yes, the program inspection technique is worth applying.

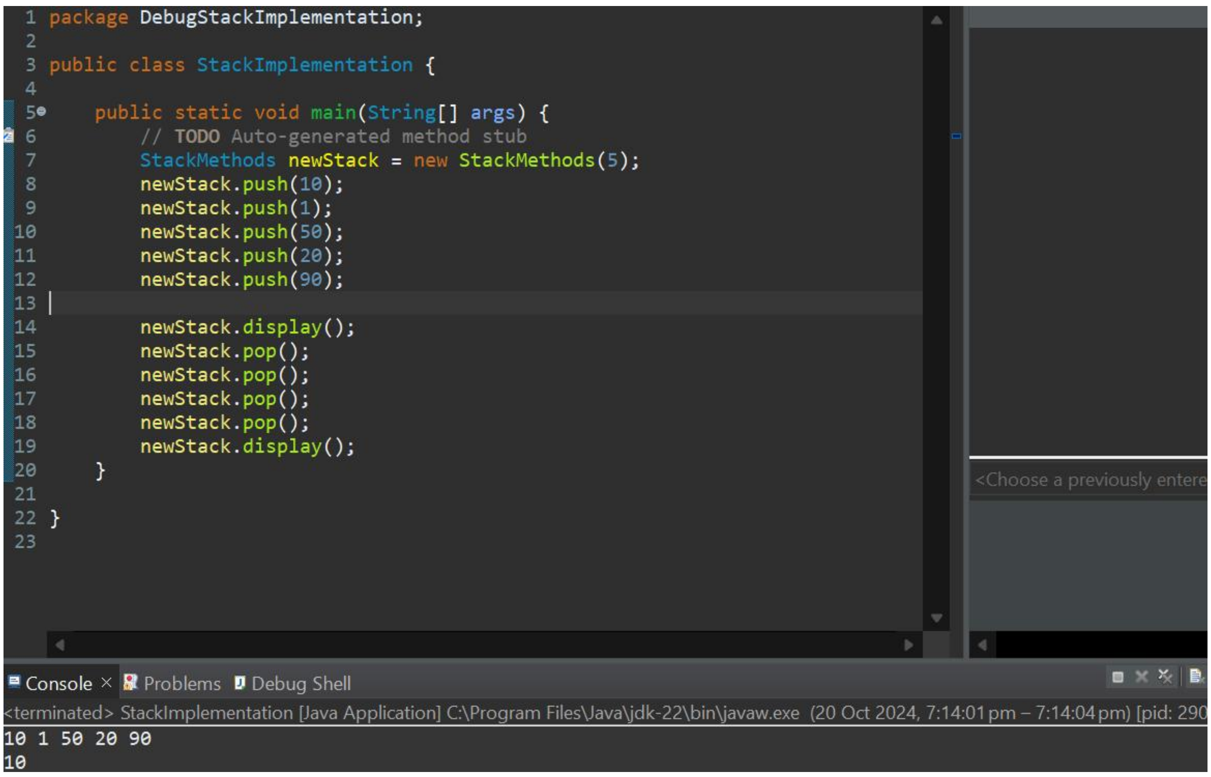
- **Reason**: The inspection process identified critical logical errors in the push, pop, and display methods, which would prevent the stack from functioning properly. By addressing these errors early, debugging and runtime issues are minimized, ensuring the stack performs as expected.

Code Debugging :



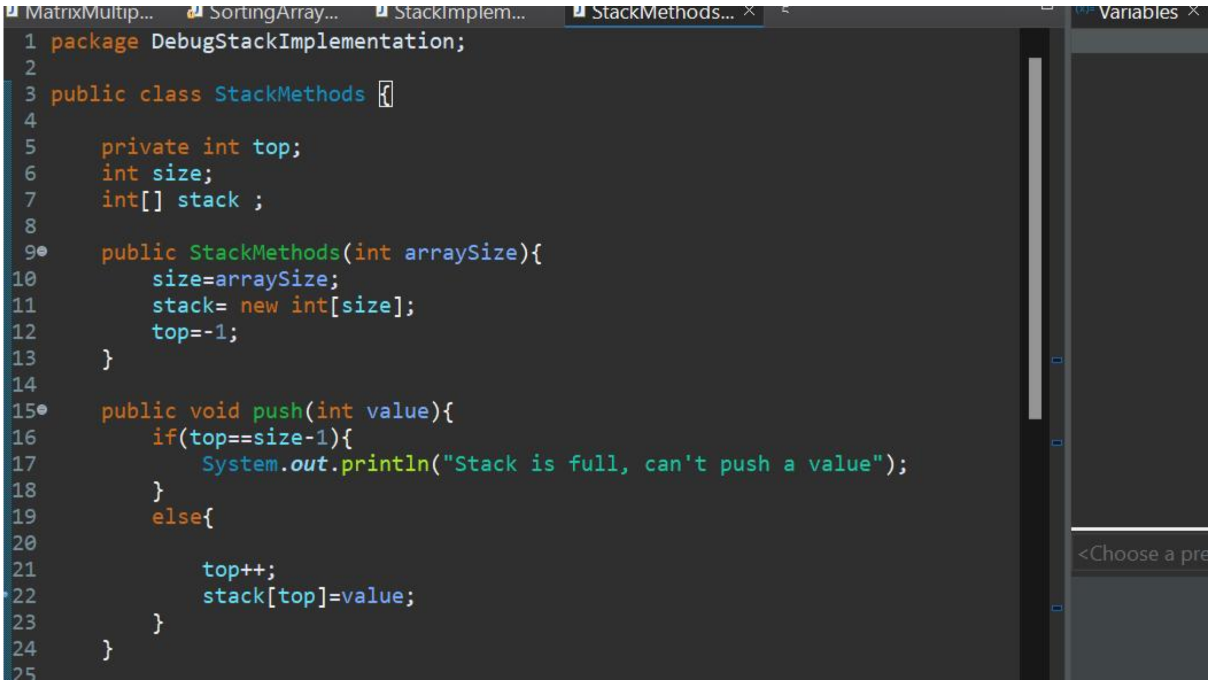
```java
        size=arraySize;
        stack= new int[size];
        top=-1;
    }

    public void push(int value){
        if(top==size-1){
            System.out.println("Stack is full, can't push a value");
        }
        else{

            top--;
            stack[top]=value;
        }
    }

    public void pop(){
        if(!isEmpty())
            top++;
        else{
            System.out.println("Can't pop...stack is empty");
        }
    }

    public boolean isEmpty(){
        return top==-1;
```

Name	Value
push() is throwing	ArrayInde
this	StackMet
value	10

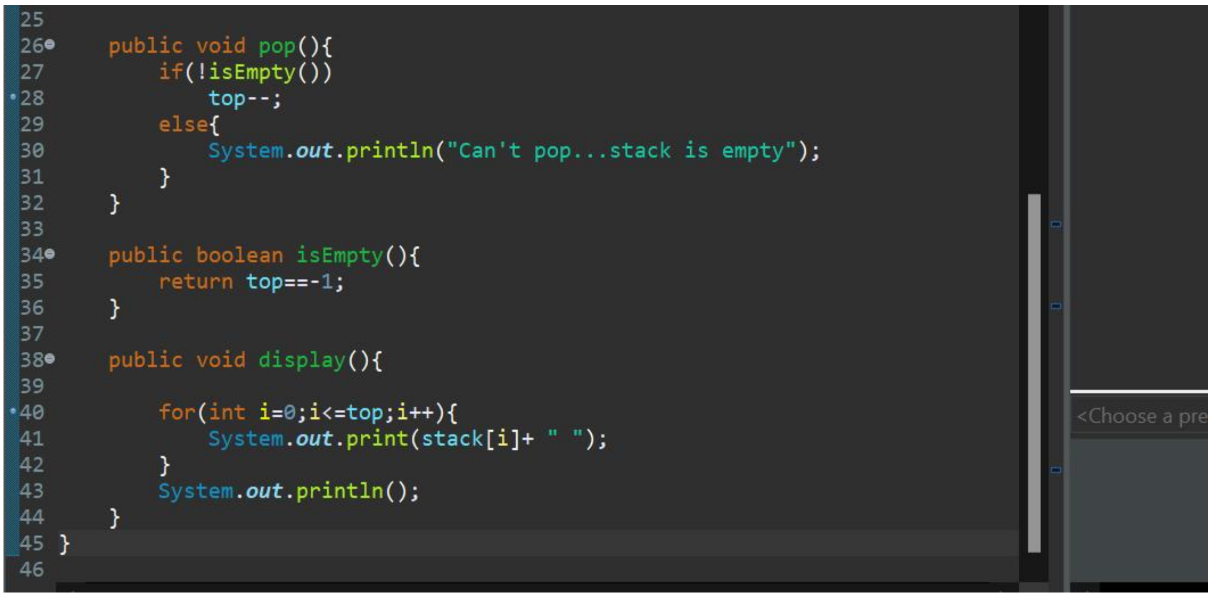


```java
package DebugStackImplementation;

public class StackImplementation {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);

        newStack.display();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display();
    }

}
```

```
<terminated> StackImplementation [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (20 Oct 2024, 7:14:01 pm – 7:14:04 pm) [pid: 290
10 1 50 20 90
10
```



```
 1 package DebugStackImplementation;
 2
 3 public class StackMethods {
 4
 5     private int top;
 6     int size;
 7     int[] stack ;
 8
 9     public StackMethods(int arraySize){
10         size=arraySize;
11         stack= new int[size];
12         top=-1;
13     }
14
15     public void push(int value){
16         if(top==size-1){
17             System.out.println("Stack is full, can't push a value");
18         }
19         else{
20
21             top++;
22             stack[top]=value;
23         }
24     }
25
```



```
25
26     public void pop(){
27         if(!isEmpty())
28             top--;
29         else{
30             System.out.println("Can't pop...stack is empty");
31         }
32     }
33
34     public boolean isEmpty(){
35         return top==-1;
36     }
37
38     public void display(){
39
40         for(int i=0;i<=top;i++){
41             System.out.print(stack[i]+ " ");
42         }
43         System.out.println();
44     }
45 }
46
```

Identified Errors

1. **Push Method (Line 18)**:
   - **Error**: top-- was incorrectly used; it should be top++ to increment the top index when adding a new element to the stack.
2. **Pop Method (Line 26)**:
   - **Error**: top++ was incorrectly used; it should be top-- to decrement the top index when removing an element from the stack.

3. **Display Method (Line 35)**:
   - **Error**: The loop condition i > top was incorrect; it should be i <= top to iterate through the stack elements properly.

Debugging Submission

1. **Errors Identified**:
   - Total of 3 errors.
   - Errors were in the push, pop, and display methods as described above.
2. **Breakpoints Needed**:
   - You may need breakpoints at the start of each method (push, pop, display) to examine how values change as the stack is manipulated.
3. **Steps Taken to Fix Errors**:
   - Modified the logic in the push and pop methods to correctly adjust the top index.
   - Corrected the loop condition in the display method to properly iterate through the stack.
4. **Complete Executable Code**:
   - You can submit the fixed code provided above as your executable code.

Fixed Code

```
import java.util.Arrays;


public class StackMethods {
  private int top;
  int size;
  int[] stack;


  public StackMethods(int arraySize) {
    size = arraySize;
    stack = new int[size];
    top = -1;
  }


  public void push(int value) {
    if (top == size - 1) {
        System.out.println("Stack is full, can't push a value");
    } else {
        top++; // Increment top before adding the value
        stack[top] = value;
    }
  }

  public void pop() {
    if (!isEmpty()) {
        top--; // Decrement top when popping
    } else {
        System.out.println("Can't pop...stack is empty");
    }
  }

  public boolean isEmpty() {
    return top == -1;
  }

  public void display() {
    if (isEmpty()) {
        System.out.println("Stack is empty");
        return;
    }
    for (int i = 0; i <= top; i++) { // Corrected loop to iterate up to top
        System.out.print(stack[i] + " ");
    }
    System.out.println();
  }
}
```

```
public class StackReviseDemo {
    public static void main(String[] args) {
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);
        newStack.display(); // Displays the stack before popping
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display(); // Displays the stack after popping
    }
}
```