```
/*Henry Gold / Havin Lim
Lab : 2 - Getting started with Java
August 29 2022
Sources: None
Help obtained: None
We confirm that the above list of sources is complete AND that we have not talked to anyone
else (e.g., CSC 207 students) about the solution to this problem.
*/

class ArrayProblems {
    public static void main(String[] args) {
        int[] arr1 = {3, 7, -10, 2, 9, 1};
        int min =  min(arr1);
        int max = max(arr1);
        int range = range(arr1);
        System.out.format("%d%n%d%n%d",min ,max, range);
    }
    public static int min(int[] arr) {
        int len = arr.length - 1;
        int cur = arr[0];
        for (int i = 1; i <= len; i++) {
            if (arr[i] < cur) {
                cur = arr[i];
            }
        }
        return cur;
    }
    public static int max(int[] arr) {
        int len = arr.length - 1;
        int cur = arr[0];
        for (int i = 1; i <= len; i++) {
```

```
            if (cur < arr[i]) {
                cur = arr[i];
            }
        }
        return cur;
    }
    public static int range(int[] arr) {
        return max(arr) - min(arr);
    }
}
```

Henry Gold / Havin Lim

Lab : 2 - Getting started with Java

August 29 2022

Sources: None

Help obtained: None

We confirm that the above list of sources is complete AND that we have not talked to anyone else (e.g., CSC 207 students) about the solution to this problem.

1. If we declare **“int arr[5];”** produces an error message expecting a closing bracket instead of an integer.
   If we use the right syntax we can initialize the null array.
   When you try to use a null array you get a NullPointerException.
2. When the new expression is used it fills the array with zeros(0).
3. We get an out of bounds error at runtime. If you want to create an error that’s not useful we might walk off the end of an array.

Henry Gold / Havin Lim

Lab : 3 - Strings and StringBuilder

August 31 2022

Sources: None

Help obtained: None

We confirm that the above list of sources is complete AND that we have not talked to anyone else (e.g., CSC 207 students) about the solution to this problem.

```
class lab3 {
    public static void main(String[] args) {

        StringBuilder str = new StringBuilder("");

        for (int i = 1; i < args.length; i++) {
            str.append(args[i]+" ");
        }
        if (args[0].equals("reverseString")){
            System.out.println(reverseString(str.toString()));
        }
        if (args[0].equals("rot13")){
            System.out.println(rot13(str.toString()));
        }
        if (args[0].equals("multTable")){
            multTable();
        }
        if (args[0].equals("countSubstring")) {
            String str2 = new String("this");
            String str1 = new String("this is this and that is this");
            System.out.format("countSubstringTest: %d%n", countSubstring(str1, str2));
        }
```

```
}

// #1

public static String reverseString(String str) {
    StringBuilder newStr = new StringBuilder("");
    for (int i = (str.length() - 1); i >= 0; i--) {
        newStr.append(str.charAt(i));
    }
    return newStr.toString();
    }

// #2

public static String rot13(String str) {
    StringBuilder newStr = new StringBuilder(" ");
    for (int i=0; i<str.length(); i++) {
        newStr.append(rot13Helper(str.charAt(i)));
    }
    return newStr.toString();
}
public static char rot13Helper(char ch) {
    int chNumeric = (int)ch;
    if (((int)'a' <= chNumeric) && (chNumeric < (int)'n')) {
        return (char)(chNumeric + 13);
    } else if (((int)'n' <= chNumeric) && (chNumeric <= (int)'z')) {
        return (char)(chNumeric - 13);
    } else if (((int)'A' <= chNumeric) && (chNumeric  < (int)'N')) {
        return (char)(chNumeric + 13);
    } else if (((int)'N' <= chNumeric) && (chNumeric <= (int)'Z')) {
```

```
            return (char)(chNumeric - 13);
        }
        return ch;
    }

    /* #3
Because the two halves of the alphabet swaps symmetrically, encryption and decryption has the same procedure.
*/


    // #4
    public static void multTable() {
        for (int i=1; i<=10; i++) {
            for (int j=1; j<=10; j++) {
                System.out.format("%5s", (i*j));
            }
            System.out.format("%n");
        }
    }

    // #5
    public static int countSubstring(String str, String str2) {
        int counter = 0;
        for (int i=0; i<=(str.length()-str2.length()); i++) {
            if(str2.equals(str.substring(i,((str2.length()) + i)))) {
                counter ++;
            }
        }
        return counter;
    }
```

}

```
// Adapted by Henry Gold / Havin Lim

/** Switch -- a two-position switch

    @author  John David Stone<br />
    Department of Computer Science<br />
    Grinnell College<br />
    <tt>reseda@grinnell.edu</tt>

    @version June 26, 2018

    An object of the <code>Switch</code> class
    models a conventional two-position toggle switch,
    like the power switch on a vacuum cleaner.
*/

public class Switch {

    /** The <code>on</code> field
        stores the current position of the switch
        (true for the "on" position, false for "off").
    */

    private boolean on;

    public static boolean ON = true;
    public static boolean OFF = false;

    /** The <code>Switch</code> constructor
        allocates storage for a new <code>Switch</code>,
        initializes its current state to "off,"
```

```
    and returns it.
*/

public Switch() {
    on = false;
}

public Switch(boolean b) {
    on = b;
}

/**
   The <code>report</code> method
   returns the current state of the switch.

   @return  the current state of the switch
            (true for "on", false for "off")
*/

public boolean report() {
    return on;
}

/** The <code>turn</code> method
    imposes a specified state on the switch.

    @param  newState  the state to be imposed
                      (true for "on", false for "off")
*/

public void turn(boolean newState) {
```

```
    on = newState;
}

/** The <code>toggle</code> method
    models the operation of toggling the switch,
    changing its state.
*/

public void toggle() {
    on = !on;
}

public boolean sameState(Switch s) {
    return s.report() == this.report();
}

public void conformToConsensus (Switch s1, Switch s2, Switch s3) {
    int c = 0;
    boolean b = false;
    if (s1.report()) {
        c++;
    } else {
        c--;
    }
    if (s2.report()) {
        c++;
    } else {
        c--;
    }
    if (s3.report()) {
        c++;
```

```
        } else {
            c--;
        }

        if (c > 0) {
            b = true;
        }

        if (c < 0) {
            b = false;
        }

        this.turn(b);

    }
}
```

/* Copyright © 2008, 2018 John David Stone */

/* This program is free software.
   You may redistribute it and/or modify it
   under the terms of the GNU General Public License
   as published by the Free Software Foundation --
   either version 3 of the License
   or (at your option) any later version.
   A copy of the GNU General Public License
   is available on the World Wide Web
   at https://www.gnu.org/licenses/gpl.html.

   This program is distributed
   in the hope that it will be useful,

```
   but WITHOUT ANY WARRANTY --
   without even the implied warranty
   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the GNU General Public License for more details.
*/
```

```
/*
Henry Gold / Havin Lim
Lab : 4 - References and Objects
September 2 2022
Sources: None
Help obtained: None
We confirm that the above list of sources is complete AND that we have not talked to anyone
else (e.g., CSC 207 students) about the solution to this problem.
*/


public class SwitchTester {
    public static void SwitchTester(Switch s) {
        System.out.format("%b%n", s.report());
    }

    public static void MethodCallTest(Switch s) {
        System.out.format("In method test before: %b%n", s.report());
        s.toggle();
        System.out.format("In method test after:%b%n", s.report());
    }
    public static void main(String[] args) {
        Switch s1 = new Switch();
        Switch s2 = s1;
        SwitchTester(s1);
        SwitchTester(s2);
        s1.toggle();
        SwitchTester(s1);
        SwitchTester(s2);
        // We  expect to see that changing the first switch object also
        // changes the second object. This is correct.
```

```
MethodCallTest(s1);
System.out.format("Method call test outside of method:%b%n", s1.report());
// The effects from inside the method persisted outside as well.

Switch s3 = new Switch();
Switch s4 = new Switch();
SwitchTester(s3);
SwitchTester(s4);
s3.toggle();
SwitchTester(s3);
SwitchTester(s4);

// Yes, two separately created switches can be toggled independentely.

Switch s5 = new Switch();
Switch s6 = new Switch();
System.out.format("Equals Test : %b%n", (s5 == s6));

// Separately created switches are not "equal" even if the have the same internal states.

System.out.println(OnOffTest(s3));
System.out.println(OnOffTest(s4));

s5.sameState(s6);

System.out.format("%b%n", (s5.sameState(s6)));

// Unit tests for conformtToConsensus

Switch s7 = new Switch();
```

```
        Switch s8 = new Switch();
        Switch s9 = new Switch();
        Switch s10 = new Switch();
        s7.toggle();
        s8.toggle();
        SwitchTester(s10);
        s10.conformToConsensus(s7, s8, s9);
        SwitchTester(s10);

        // New Constructor Test

        Switch s11 = new Switch(true);
        SwitchTester(s11);

        // ON OFF Test
        s11.turn(Switch.OFF);
        SwitchTester(s11);

    }
    public static String OnOffTest(Switch s) {
        if (s.report() == true) {
            return "on";
        } else {
            return "off";
        }
    }
}
```