CSE 1325

Week of 09/12/2022

Instructor : Donna French

arrays

In many programs, you will need to collect large numbers of values.

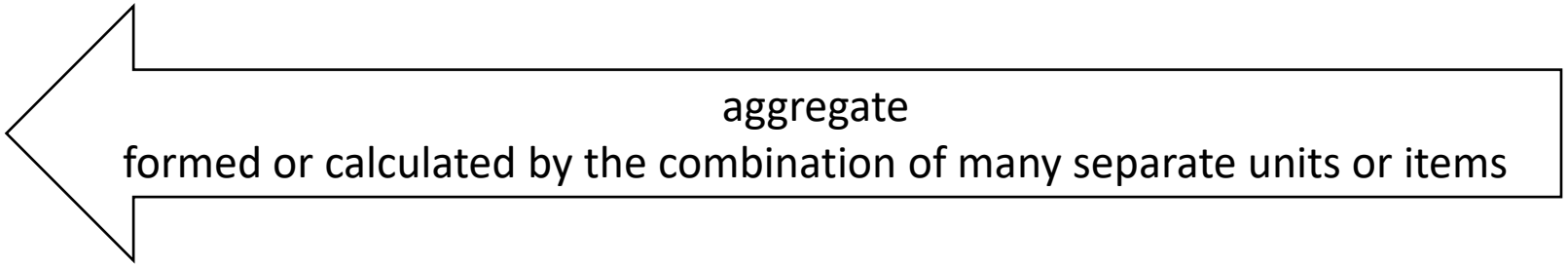
In Java, you use the array and array list constructs for this purpose.

Arrays have a more concise syntax; whereas, array lists can automatically grow to any desired size.

We are going to start with arrays.

arrays

- aggregate type



- used to store collections of related data

- multiple values of the same data type can be stored with one variable name

arrays

Create an integer array to hold the even numbers between 1 and 10.

```
int even[] = {2,4,6,8,10};
```

Create a character array to hold all of the vowels.

```
char vowels [] = {'a','e','i','o','u'};
```

arrays

Create an array to hold decimal values.

```
double myDecimals[] = {2.5,3.4,6.5,1.8,1.01,4.6};
```

Create an array to hold punctuation symbols.

```
char puncSyms[] = {'!','@','#','$','%','^','&','*'};
```

arrays

```
int even[] = {2,4,6,8,10};


System.out.printf("%d", even[0]);
System.out.printf("%d", even[1]);
System.out.printf("%d", even[2]);
System.out.printf("%d", even[3]);
System.out.printf("%d", even[4]);
```

246810

arrays

```
char vowels[] = {'a','e','i','o','u'};

System.out.printf("%c", vowels[0]);
System.out.printf("%c", vowels[1]);
System.out.printf("%c", vowels[2]);
System.out.printf("%c", vowels[3]);
System.out.printf("%c", vowels[4]);
```

```
aeiou
```

arrays

```
double myDecimals[] = {2.5,3.4,6.5,1.8,1.01,4.6};

System.out.printf("%.2f", myDecimals[0]);
System.out.printf("%.2f", myDecimals[1]);
System.out.printf("%.2f", myDecimals[2]);
System.out.printf("%.2f", myDecimals[3]);
System.out.printf("%.2f", myDecimals[4]);
System.out.printf("%.2f", myDecimals[5]);
```

2.503.406.501.801.014.60

arrays

```
char puncSyms[] = {'!','@','#','$','%','^','&','*'};

System.out.printf("%c", puncSyms[0]);
System.out.printf("%c", puncSyms[1]);
System.out.printf("%c", puncSyms[2]);
System.out.printf("%c", puncSyms[3]);
System.out.printf("%c", puncSyms[4]);
System.out.printf("%c", puncSyms[5]);
System.out.printf("%c", puncSyms[6]);
System.out.printf("%c", puncSyms[7]);
```

!@#$%^&*

String Compare

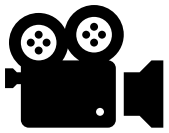
How do we compare two strings to see if they are equal?

Can we ignore the case when doing this comparison?

How can we tell if one string would come before the other in a dictionary?

String Compare

```
public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    String myWords[] = {"","",""};
    String searchWord;

    for (int i = 0; i < 3; i++)
    {
        System.out.print("Enter a word ");
        myWords[i] = in.next();
    }
```



```
Source  History

6       public static void main(String[] args)
7       {
8           Scanner in = new Scanner(System.in);
9           String myWords[] = {"","",""};
10          String searchWord;
11
12          for (int i = 0; i < 3; i++)
13          {
14              System.out.print("Enter a word ");
15              myWords[i] = in.next();
16          }
17
            System.out.print("Enter search word ");
```

studentcode.StudentCode › main

Output

Debugger Console | StudentCode (debug)

```
boolean FoundIt = false;
```

String Compare

```
System.out.print("Enter search word ");
searchWord = in.next();

for (int i = 0; i < 3; i++)
{
    if (myWords[i].equals(searchWord))
        FoundIt = true;
}

if (FoundIt)
    System.out.println("Found it!");
else
    System.out.println("Did not find it");
```

String Compare

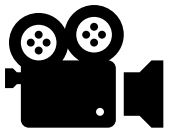
To test whether two strings are equal to each other, you must use the method called `equals`

```
if (string1.equals(string2))….
```

Do not use the == operator to compare strings. The comparison

```
if (string1 == string2)
```

has an unrelated meaning. It tests whether the two strings are stored in the same memory location. You can have strings with identical contents stored in different locations.



```
18              }
19
20              System.out.print("Enter search word ");
21              searchWord = in.next();
22
23              for (int i = 0; i < 3; i++)
24              {
25                  if (myWords[i].equals(searchWord))
26                      FoundIt = true;
27              }
28
29              if (FoundIt)
30                  System.out.println("Found it!");
31              else
```

studentcode.StudentCode > main > for (int i = 0; i < 3; i++)

Output

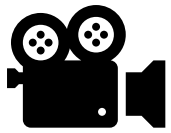
Debugger Console | StudentCode (run)

String Compare

Does the case of our strings matter when searching?

If we put "`dog`" in our array and search for "`DOG`", will it be found?

If we put "`dog`" in our array and search for "`doG`", will it be found?



StudentCode.java | Code5_1000074079.java | ICQ7.java

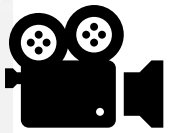
Source | History

```
20          System.out.print("Enter search word ");
21          searchWord = in.next();
22
23          for (int i = 0; i < 3; i++)
24          {
25              if (myWords[i].equals(searchWord))
26                  FoundIt = true;
27          }
28
29          if (FoundIt)
30              System.out.println("Found it!");
31          else
32              System.out.println("Did not find it");
```

studentcode.StudentCode > main > for (int i = 0; i < 3; i++)

Output

Debugger Console | StudentCode (run)



```
20          System.out.print("Enter search word ");
21          searchWord = in.next();
22
23          for (int i = 0; i < 3; i++)
24          {
                if (myWords[i].equals(searchWord))
26                  FoundIt = true;
27          }
28
29          if (FoundIt)
30              System.out.println("Found it!");
31          else
32              System.out.println("Did not find it");
```

studentcode.StudentCode > main > for (int i = 0; i < 3; i++) > if (myWords[i].equals(searchWord))

Output

Debugger Console | StudentCode (run)

String Compare

Does the case of our strings matter when searching?

YES

If case does not matter to your process (it can be case insensitive), then use

```
equalsIgnoreCase()
```

instead of

```
equals()
```

String Compare

How can we tell if one string would come before the other in a dictionary?

We compare two strings and determine their "lexicographic" order.

In general terms, lexicographic order translates to alphabetic order.

apple would come before banana in the dictionary.

Programming languages required a few more rules to be applied to determining alphabetic order.

Which comes first – apple or Banana?

Dictionaries are arranged in alphabetic order and are generally case insensitive – you don't look up the uppercase version of a word

String Compare

We order strings based on their values in the ASCII character set.

American **S**tandard **C**ode for **I**nformation **I**nterchange

ASCII is a 7-bit character set containing 128 characters. It contains the numbers from 0-9, the upper and lower case English letters from A to Z, and some special characters. The character sets used in modern computers, in HTML, and on the Internet, are all based on **ASCII**.

String Compare

- ASCII character set
  - 128 characters
  - each character has an integer value between 0 and 127
  - The ASCII value are used when determining the order of strings.

Ascii	Char	Ascii	Char	Ascii	Char	Ascii	Char
0	Null	32	Space	64	@	96	`
1	Start of heading	33	!	65	A	97	a
2	Start of text	34	"	66	B	98	b
3	End of text	35	#	67	C	99	c
4	End of transmit	36	$	68	D	100	d
5	Enquiry	37	%	69	E	101	e
6	Acknowledge	38	&	70	F	102	f
7	Audible bell	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Horizontal tab	41	)	73	I	105	i
10	Line feed	42	*	74	J	106	j
11	Vertical tab	43	+	75	K	107	k
12	Form feed	44	,	76	L	108	l
13	Carriage return	45	-	77	M	109	m
14	Shift in	46	.	78	N	110	n
15	Shift out	47	/	79	O	111	o
16	Data link escape	48	0	80	P	112	p
17	Device control 1	49	1	81	Q	113	q
18	Device control 2	50	2	82	R	114	r
19	Device control 3	51	3	83	S	115	s
20	Device control 4	52	4	84	T	116	t
21	Neg. acknowledge	53	5	85	U	117	u
22	Synchronous idle	54	6	86	V	118	v
23	End trans. block	55	7	87	W	119	w
24	Cancel	56	8	88	X	120	x
25	End of medium	57	9	89	Y	121	y
26	Substitution	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	File separator	60	<	92	\	124	|
29	Group separator	61	=	93	]	125	}
30	Record separator	62	>	94	^	126	~
31	Unit separator	63	?	95	_	127	Forward del.

String Compare

When comparing strings and a mismatch is found, the string containing the "larger" character is considered "larger"; therefore, would be ordered after the "smaller" string.

greet

great

g compared to g – result is they are the same

r compared to r – result is they are the same

e compared to e – result is they are the same

e compared to a – e is "greater than" a

great and then greet

String Compare

When comparing two strings, you compare the first letters of each word, then the second letters and, so on, until one of the strings ends or you find the first letter pair that doesn't match.

If one of the strings ends, the longer string is considered the "larger" one.

Compare "car" and "cart". The "car" parts are equivalent but then we have reached the end of "car" but not of "cart" so "car" is BEFORE "cart".

String Compare

Java uses a method called `compareTo()` to compare two strings.

```
string1.compareTo(string2)
```

If the return of `compareTo()` is < 0, then the first string is less than the second string (order is string1 and then string2)

If the return of `compareTo()` is > 0, then the first string is greater than the second string (order is string2 and then string1)

If the return of `compareTo()` is 0, then the two strings are the same.

```
System.out.print("Enter your first word ");
word1 = in.next();
System.out.print("Enter your second word ");
word2 = in.next();

if (word1.compareTo(word2) < 0)
    System.out.printf("%s < %s\n", word1, word2);
else if (word1.compareTo(word2) > 0)
    System.out.printf("%s > %s\n", word1, word2);
else
    System.out.printf("%s is equivalent to %s", word1, word2);
```

```
Enter your first word apple
Enter your second word Apple
apple > Apple


Enter your first word Pear
Enter your second word PEar
Pear > PEar


Enter your first word Banana
Enter your second word Banana
Banana is equivalent to Banana


Enter your first word Zebra
Enter your second word apple
Zebra < apple
```

```
System.out.print("Enter your first word ");
word1 = in.next();
System.out.print("Enter your second word ");
word2 = in.next();

if (word1.compareToIgnoreCase(word2) < 0)
    System.out.printf("%s < %s\n", word1, word2);
else if (word1.compareToIgnoreCase(word2) > 0)
    System.out.printf("%s > %s\n", word1, word2);
else
    System.out.printf("%s is equivalent to %s", word1, word2);
```

```
Enter your first word apple
Enter your second word Apple
apple is equivalent to Apple


Enter your first word Pear
Enter your second word PEar
Pear is equivalent to PEar


Enter your first word Banana
Enter your second word Banana
Banana is equivalent to Banana


Enter your first word Zebra
Enter your second word apple
Zebra < apple
```

String Handling

We have been using `in.next()` to read our strings.

As we have seen, this works for strings/words that contain no whitespace.

What happens when we add whitespace

```
6       public static void main(String[] args)
7       {
8           Scanner in = new Scanner(System.in);
9
10          String phrase1, phrase2;
11
12          System.out.print("Enter a phrase ");
13          phrase1 = in.next();
            System.out.printf("You entered %s\n", phrase1);
15      }
    }
```

studentcode.StudentCode › main

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2

String Handling

`in.next()` only picks up the characters before the whitespace.

Anything after the whitespace is left behind the standard input buffer.

```
System.out.print("Enter a phrase ");
phrase1 = in.next();
System.out.printf("You entered %s\n", phrase1);
```

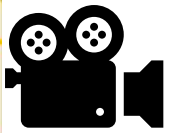
```
8           Scanner in = new Scanner(System.in);
9
10          String phrase1, phrase2;
11
12          System.out.print("Enter a phrase ");
13          phrase1 = in.next();
14          System.out.printf("You entered %s\n", phrase1);
15          System.out.print("Enter another phrase ");
16          phrase1 = in.next();
17          System.out.printf("You entered %s\n", phrase1);
18      }
```

studentcode.StudentCode > main >

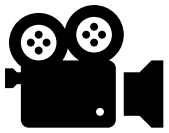
Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2

```
[ant -f C:\\Users\\Donna\\Desktop\\UTA\\CSE1310\\Programs\\StudentCode -Dnb.int
ernal.action.name=run run
```



stdin



```
Source  History
 8          Scanner in = new Scanner(System.in);
 9
10          String phrase1, phrase2;
11
12          System.out.print("Enter a phrase ");
13          phrase1 = in.next();
14          System.out.printf("You entered %s\n", phrase1);
15          System.out.print("Enter another phrase ");
16          phrase1 = in.next();
17          System.out.printf("You entered %s\n", phrase1);
18      }
```

studentcode.StudentCode › main

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2

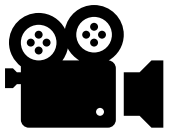
String Handling

`in.next()` only picks up the characters before the whitespace.

`in.nextLine()` reads through whitespace and stops at the newline.

```
System.out.print("Enter a phrase ");
phrase1 = in.nextLine();
System.out.printf("You entered %s\n", phrase1);
```

```
String[] phrases = {"","","","",""};

for (int i = 0; i < 5; i++)
{
    System.out.print("Enter a phrase ");
    phrases[i] = in.nextLine();
}

System.out.print("\nEnter a search phrase ");
String searchPhrase = in.nextLine();

for (int i = 0; i < 5; i++)
{
    if (searchPhrase.equals(phrases[i]))
    {
        System.out.printf("Found it in element %d\n", i);
    }
}
```



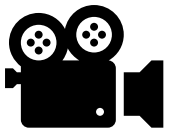
Source History

```
8           Scanner in = new Scanner(System.in);
9
10          String phrases[] = {"","","","",""};
11
12          for (int i = 0; i < 5; i++)
13          {
14              System.out.print("Enter a phrase ");
15              phrases[i] = in.nextLine();
16          }
17
18          System.out.print("\nEnter a search phrase ");
```

studentcode.StudentCode > main

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2 | StudentCode (run) #3



```
17
18          System.out.print("\nEnter a search phrase ");
19          String searchPhrase = in.nextLine();
20
21          for (int i = 0; i < 5; i++)
22          {
                if (searchPhrase.equals(phrases[i]))
24              {
25                  System.out.printf("Found it in element %d\n", i);
26              }
27          }
```

studentcode.StudentCode > main > for (int i = 0; i < 5; i++) > if (searchPhrase.equals(phrases[i]))

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2 | StudentCode (run) #3

Declaring Arrays

Up to now, we have been declaring our arrays and initializing them at the same time to set their size.

```
String phrases[] = {"","","","","",};
int SumArray[] = {0,0,0,0,0};
int dice[] = {0,0,0,0,0};
```

In most circumstances, it will be more convenient to create an array of a certain size and NOT need to initialize it at the same time.

For example, what if I want an array of 100 `int`s?

Declaring Arrays

Instead of initializing the array to set its size, we can use `new` to allocate the space.

```
String phrases[] = {"","","","",""};
```

can be stated as

```
String phrases[] = new String[5];
```

```
int SumArray[] = {0,0,0,0,0};
```

can be stated as

```
int SumArray[] = new int[5];
```

```
int dice[] = {0,0,0,0,0};
```

can be stated as

```
int dice[] = new int[5];
```

Declaring Arrays

So if we needed an array to hold 100 numbers.

```
int SumArray[] = new int[100];
```

rather than

```
int SumArray[] = {0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0};
```

Declaring Arrays

Declare a `String` array that can hold 1034 elements.

```
String Yarn[] = new String[1034];
```

Declare a `double` array that can hold 332 elements.

```
double Trouble[] = new double[332];
```

Declare a `int` array that can hold 63 elements.

```
int MyInts[] = new int[63];
```

Declaring Arrays

Now we can prompt for how big of an array to create, create it and then fill it in.

```
System.out.print("How many numbers do you need to enter? ");
int howMany = in.nextInt();

int intArray[] = new int[howMany];
```

Arrays

Arrays in Java are objects and objects know things about themselves.

Arrays in non object oriented languages do not know things about themselves.

Information about objects can be obtained by calling their methods.

Arrays

Arrays in Java know how many elements they contain. We get that information by asking the array for that information.

```
int intArray[] = new int[howMany];
```

```
intArray.length
```

`length` is one of the pieces of information stored in class `Array`

```
19          }
20
21          System.out.print("You entered numbers ");
22          for (int i = 0; i < howMany; i++)
23          {
24              System.out.printf("%3d", intArray[i]);
25          }
26          System.out.println();
27          System.out.printf("intArray has %d elements", intArray.length);
28          System.out.println();
29      }
```

studentcode.StudentCode > main > intArray

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2 | StudentCode (run) #3

Arrays

```
System.out.print("How many numbers do you need to enter max? ");
int howMany = in.nextInt();


int intArray[] = new int[howMany];


for (int i = 0; i < intArray.length; i++)
{
   System.out.print("Enter a number ");
   intArray[i] = in.nextInt();
}

System.out.print("You entered numbers ");
for (int i = 0; i < intArray.length; i++)
{
   System.out.printf("%3d", intArray[i]);
}
```

Arrays and Sorting

Because arrays are objects in Java, they have some useful abilities.

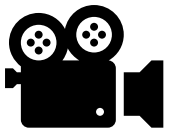
We can ask an array to sort itself.

First, we need to import another util

```
import java.util.Arrays;
```

This contains the methods for handling arrays.

```
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;
```



StudentCode.java | Code5_1000074079.java | ICQ7.java | CompareTo.java

Source | History

```
16          for (int i = 0; i < intArray.length; i++)
17          {
18              System.out.print("Enter a number ");
19              intArray[i] = in.nextInt();
20          }
21
22          System.out.print("You entered numbers ");
23          for (int i = 0; i < intArray.length; i++)
24          {
25              System.out.printf("%3d", intArray[i]);
26          }
```

studentcode.StudentCode > main > for (int i = 0; i < intArray.length; i++)

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2 | StudentCode (run) #3 | StudentCode (run) #4

```
for (int i = 0; i < intArray.length; i++)
{
    System.out.print("Enter a number ");
    intArray[i] = in.nextInt();
}

Arrays.sort(intArray);

System.out.print("You entered numbers ");
for (int i = 0; i < intArray.length; i++)
{
    System.out.printf("%3d", intArray[i]);
}
```

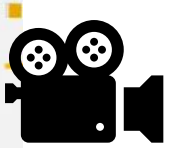
```
System.out.print("How many words do you want to alphabetize? ");
int howMany = in.nextInt();

String[] stringArray = new String[howMany];

for (int i = 0; i < stringArray.length; i++)
{
    System.out.print("Enter a word ");
    stringArray[i] = in.next();
}

Arrays.sort(stringArray);

System.out.print("Your alphabetized list is ");
for (int i = 0; i < stringArray.length; i++)
{
    System.out.printf("%s ", stringArray[i]);
}
```

```
17          {
18              System.out.print("Enter a word ");
19              stringArray[i] = in.next();
20          }
21
22          Arrays.sort(stringArray);
23
24          System.out.print("Your alphabetized list is ");
            for (int i = 0; i < stringArray.length; i++)
26          {
27              System.out.printf("%s ", stringArray[i]);
28          }
```

studentcode.StudentCode > main

Output

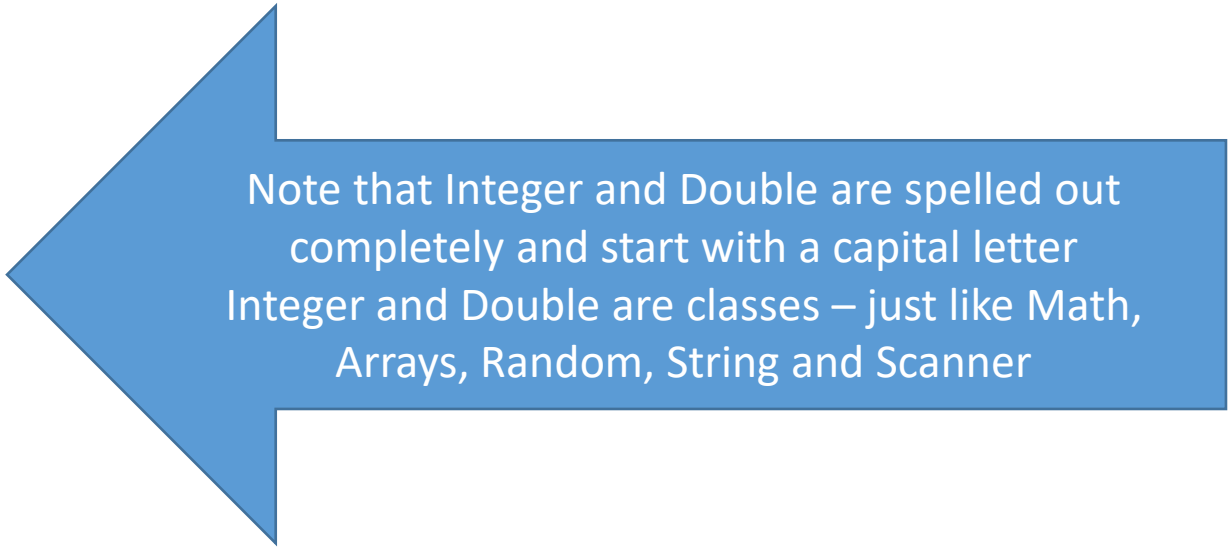
compareTo (run) | StudentCode (run) #4



Minimum and Maximum Values in Java

Java has preset values to represent the minimum and maximum values that integers and doubles can hold.

```
Integer.MAX_VALUE

Integer.MIN_VALUE

Double.MAX_VALUE

Double.MIN_VALUE
```



```java
public class StudentCode
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);

        System.out.println(Integer.MAX_VALUE);
        System.out.println(Integer.MIN_VALUE);
        System.out.println(Double.MAX_VALUE);
        System.out.println(Double.MIN_VALUE);

```

studentcode.StudentCode > main

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2

```
Updating property file: C:\Users\Donna\Desktop\UTA\CSE1310\Programs\StudentCode\build\built-jar.properties
compile:
run:
2147483647
-2147483648
1.7976931348623157E308
4.9E-324
BUILD SUCCESSFUL (total time: 0 seconds)
```

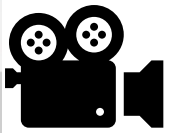
```
4   public class StudentCode
5   {
6       public static void main(String[] args)
7       {
8           Scanner in = new Scanner(System.in);
9
10          System.out.println(Integer.MAX_VALUE);
11          System.out.println(Integer.MIN_VALUE);
12          System.out.println(Double.MAX_VALUE);
13          System.out.println(Double.MIN_VALUE);
14
```

studentcode.StudentCode > main

Output

Debugger Console | compareTo (run) | StudentCode (run) | StudentCode (run) #2

Re-run



Arrays

```
String[] stringArray = new String[howMany];

for (int i = 0; i < stringArray.length; i++)
{
    System.out.print("Enter a word ");
    stringArray[i] = in.next();
}

Arrays.sort(stringArray);

System.out.println(stringArray[6]);
```

What happens if `howMany` is 6?

Source History

```
11          System.out.print("How many words do you want to alphabetize? ");
12          int howMany = in.nextInt();
13
14          String[] stringArray = new String[howMany];
15
16          for (int i = 0; i < stringArray.length; i++)
17          {
18              System.out.print("Enter a word ");
19              stringArray[i] = in.next();
20          }
21
22          Arrays.sort(stringArray);
23
24          System.out.println(stringArray[6]);
```

studentcode.StudentCode > main > in

Output - StudentCode (run)

```
ant -f C:\\Users\\Donna\\Desktop\\UTA\\CSE1310\\Programs\\StudentCode -Dnb.internal.a
ction.name=run run
```

Source History

```
17              {
18                  System.out.print("Enter a word ");
19                  stringArray[i] = in.next();
20              }
21
22              Arrays.sort(stringArray);
23
24              System.out.println(stringArray[6]);
25
26              System.out.print("Your alphabetized list is ");
27              for (int i = 0; i < stringArray.length+1; i++)
28              {
29                  System.out.printf("%s ", stringArray[i]);
```

studentcode.StudentCode › main › in

Output - StudentCode (run)

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsExcepti : Index 6 out of b
ounds for length 6
        at studentcode.StudentCode.main(StudentCode.java:24)
C:\Users\Donna\Desktop\UTA\CSE1310\Programs\StudentCode\nbproj
: The following error occurred while executing this line:
C:\Users\Donna\Desktop\UTA\CSE1310\Programs\StudentCode\nbproject\build-impl.xml:948:
 Java returned: 1
BUILD FAILED (total time: 10 seconds)
```

Error on line 24

Arrays

What if we declare an array variable but we don't allocate the memory for it?

- not initialized to value

or

- did not use `new` to reserve memory

```
double elephant[];
elephant[0] = 123.456;
```

Slide - Apache NetBeans IDE 12.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config> 437.7/572.0MB

FinalDemo1325.java | HelloWorld1325.java | NewLineLeftBehind.java | Slide.java

Source History

```
9   {
10      public static void main(String[] args)
11      {
            double elephant[];
            elephant[0] = 123.456;
14      }
15  }
16
```

Output - Slide (run)

```
ant -f C:\\Users\\frenc\\Documents\\NetBeansProjects\\Slide -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\frenc\Documents\NetBeansProjects\Slide\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\frenc\Documents\NetBeansProjects\Slide\build\built-jar.properties
Compiling 1 source file to C:\Users\frenc\Documents\NetBeansProjects\Slide\build\classes
C:\Users\frenc\Documents\NetBeansProjects\Slide\src\slide\Slide.java:13: error: variable elephant might not have
        elephant[0] = 123.456;
        ^
1 error
BUILD FAILED (total time: 0 seconds)
```

Partially Filled Arrays

An array cannot change size once it is declared. It is allocated when you create it; therefore, the size is set.

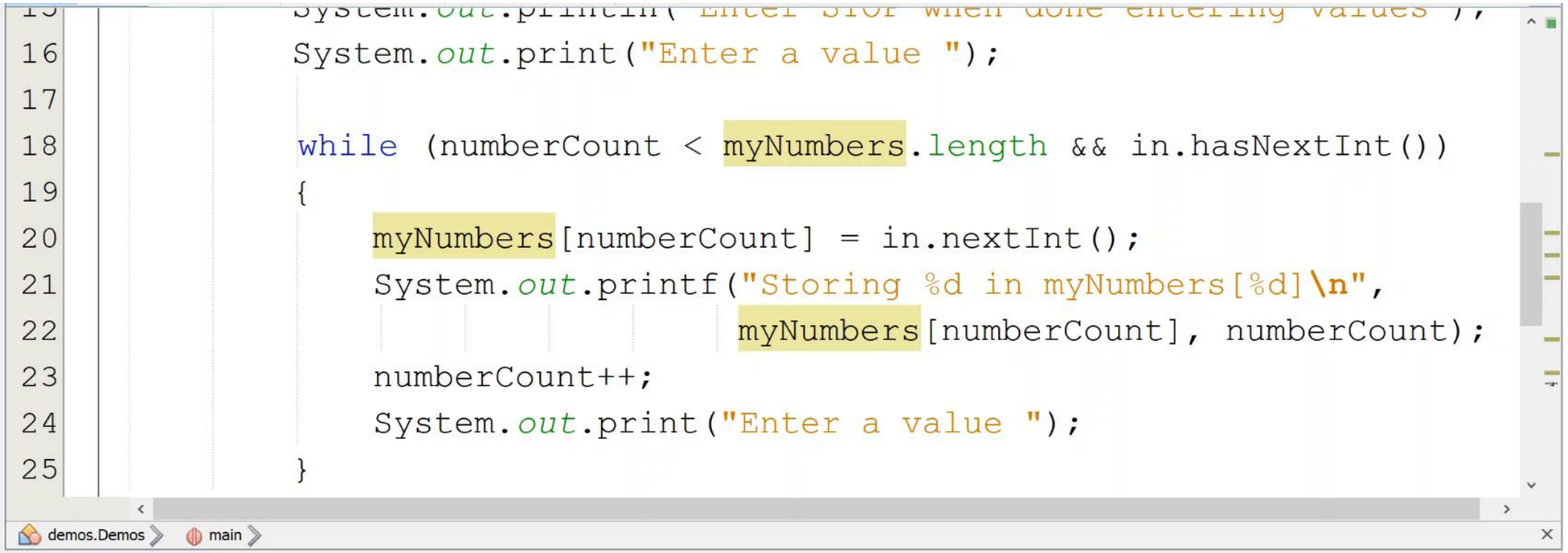
If you don't know ahead of time how many elements your array will have, then you may need to make a good guess of the maximum. This can be a difficult process.

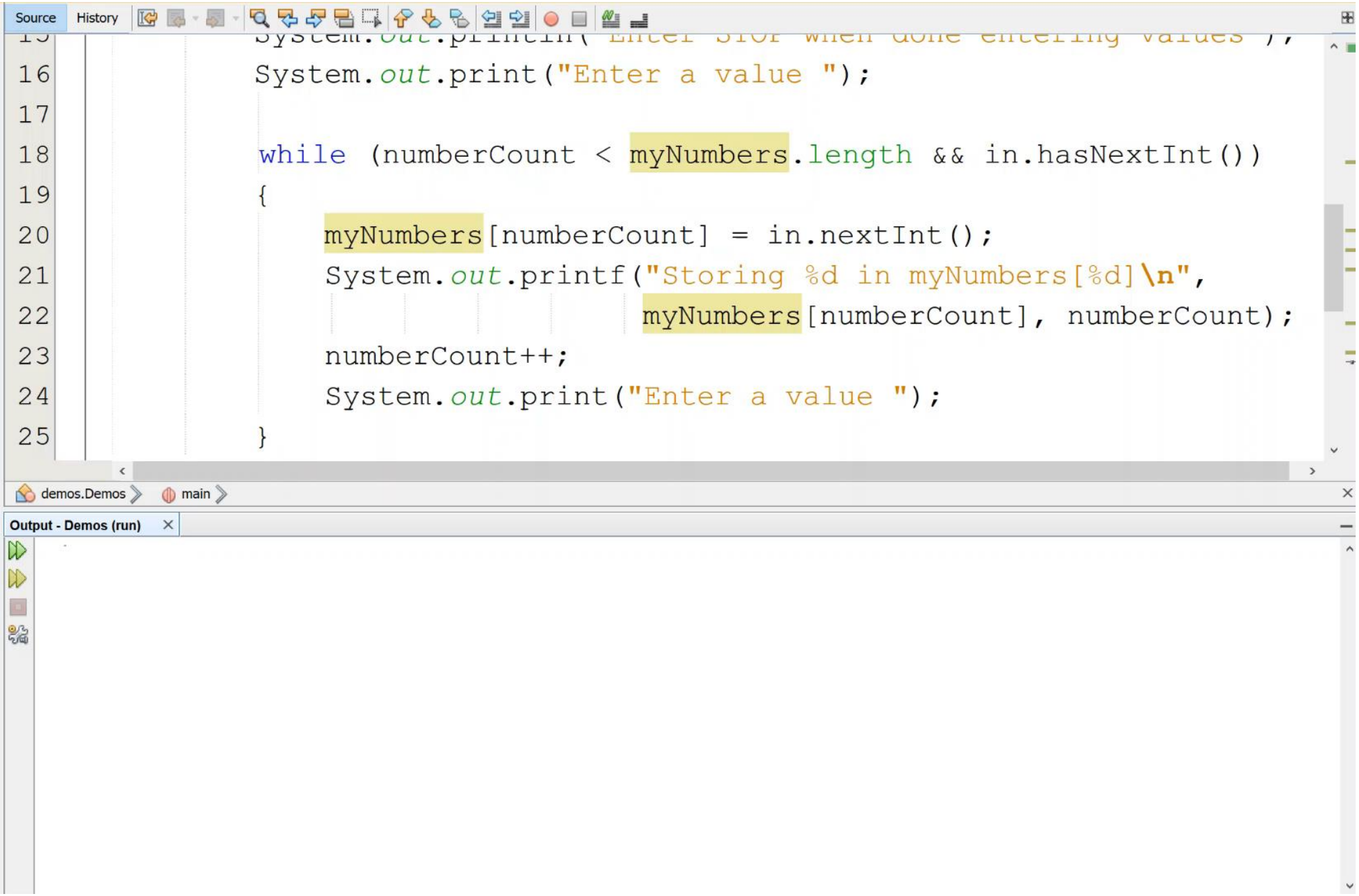
To make the process easier, we need to do two things when programming.

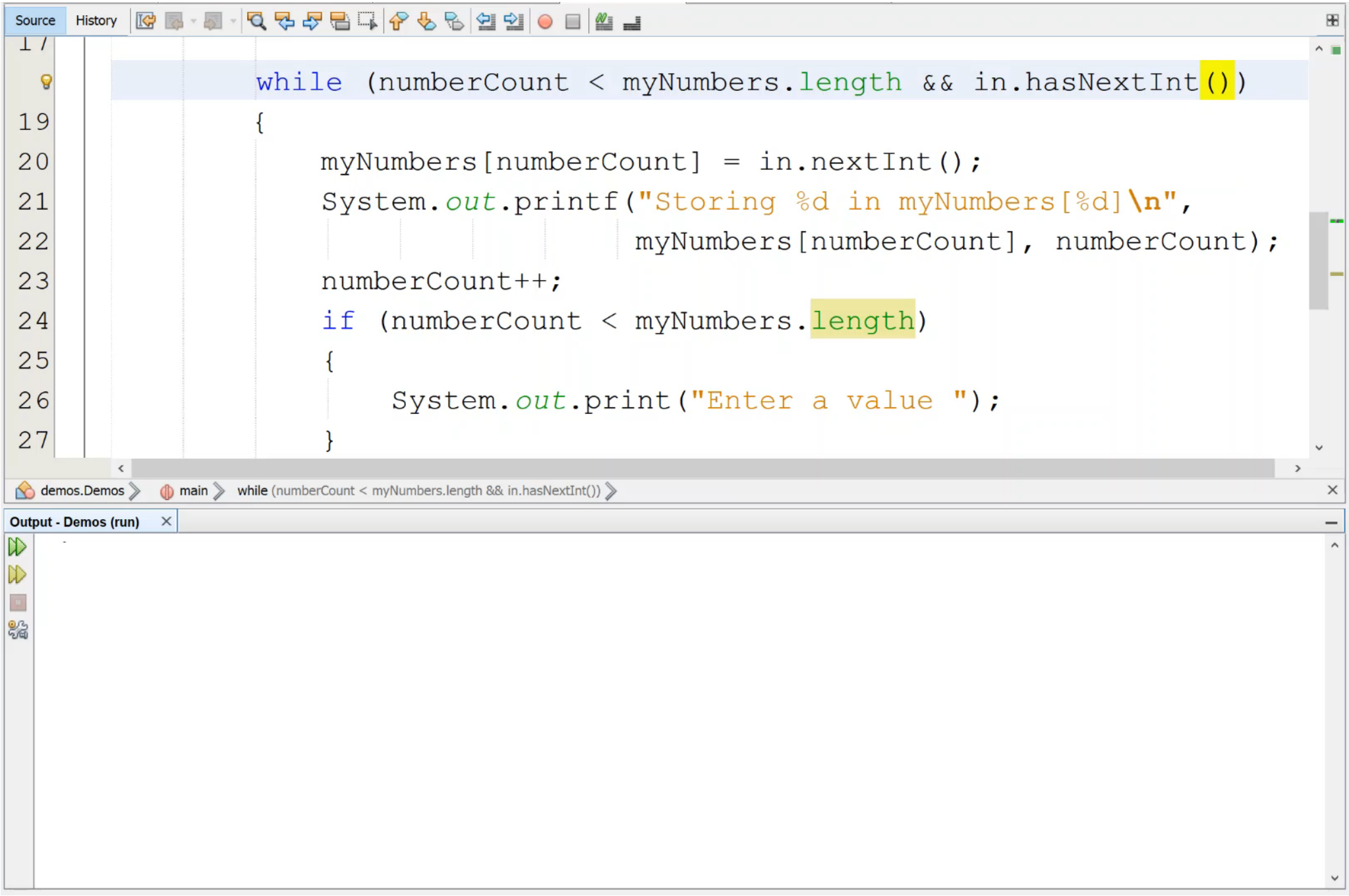
- Count the number of elements added
- Ensure that the number of elements added does not exceed the size of the array

We need to be aware that our array is partially filled.

```
int myNumbers[] = new int[10];
Scanner in = new Scanner(System.in);
int numberCount = 0;

System.out.println("Enter STOP when done entering values");
System.out.print("Enter a value ");

while (numberCount < myNumbers.length && in.hasNextInt())
{
      myNumbers[numberCount] = in.nextInt();
      System.out.printf("Storing %d in myNumbers[%d]\n",
                        myNumbers[numberCount], numberCount);
      numberCount++;
      System.out.print("Enter a value ");
}
```


System.out.println("Enter Stop when done entering values ");
System.out.print("Enter a value ");
while (numberCount < myNumbers.length && in.hasNextInt())
{
myNumbers[numberCount] = in.nextInt();
System.out.printf("Storing %d in myNumbers[%d]\n",
myNumbers[numberCount], numberCount);
numberCount++;
System.out.print("Enter a value ");
}
demos.Demos
main
Output - Demos (run)

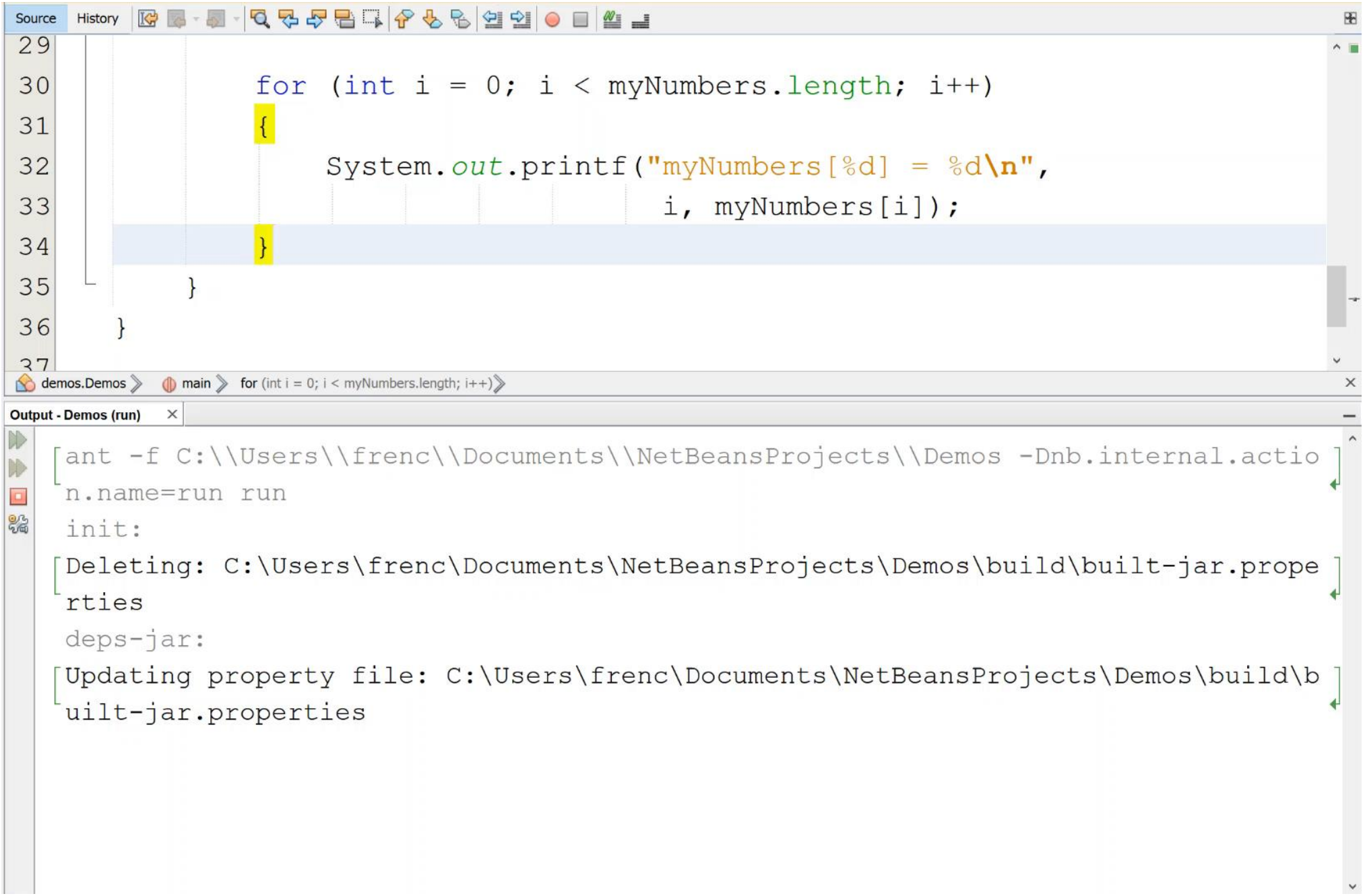

Source
History
System.out.println("Enter stop when done entering values ");
System.out.print("Enter a value ");
while (numberCount < myNumbers.length && in.hasNextInt())
{
myNumbers[numberCount] = in.nextInt();
System.out.printf("Storing %d in myNumbers[%d]\n",
myNumbers[numberCount], numberCount);
numberCount++;
System.out.print("Enter a value ");
}
demos.Demos
main
Output - Demos (run)


Source
History
while (numberCount < myNumbers.length && in.hasNextInt())
{
myNumbers[numberCount] = in.nextInt();
System.out.printf("Storing %d in myNumbers[%d]\n",
myNumbers[numberCount], numberCount);
numberCount++;
if (numberCount < myNumbers.length)
{
System.out.print("Enter a value ");
}
demos.Demos
main
while (numberCount < myNumbers.length && in.hasNextInt())
Output - Demos (run)

Partial Arrays

So what happens if we partially fill an array and then use `length` as the loop test condition to print the array?

```
for (int i = 0; i < myNumbers.length; i++)
{
     System.out.printf("myNumbers[%d] = %d\n", i, myNumbers[i]);
}
```


Source
History
29
30 for (int i = 0; i < myNumbers.length; i++)
31 {
32 System.out.printf("myNumbers[%d] = %d\n",
33 i, myNumbers[i]);
34 }
35 }
36 }
37
demos.Demos
main
for (int i = 0; i < myNumbers.length; i++)
Output - Demos (run)
ant -f C:\\Users\\frenc\\Documents\\NetBeansProjects\\Demos -Dnb.internal.actio
n.name=run run
init:
Deleting: C:\Users\frenc\Documents\NetBeansProjects\Demos\build\built-jar.prope
rties
deps-jar:
Updating property file: C:\Users\frenc\Documents\NetBeansProjects\Demos\build\b
uilt-jar.properties

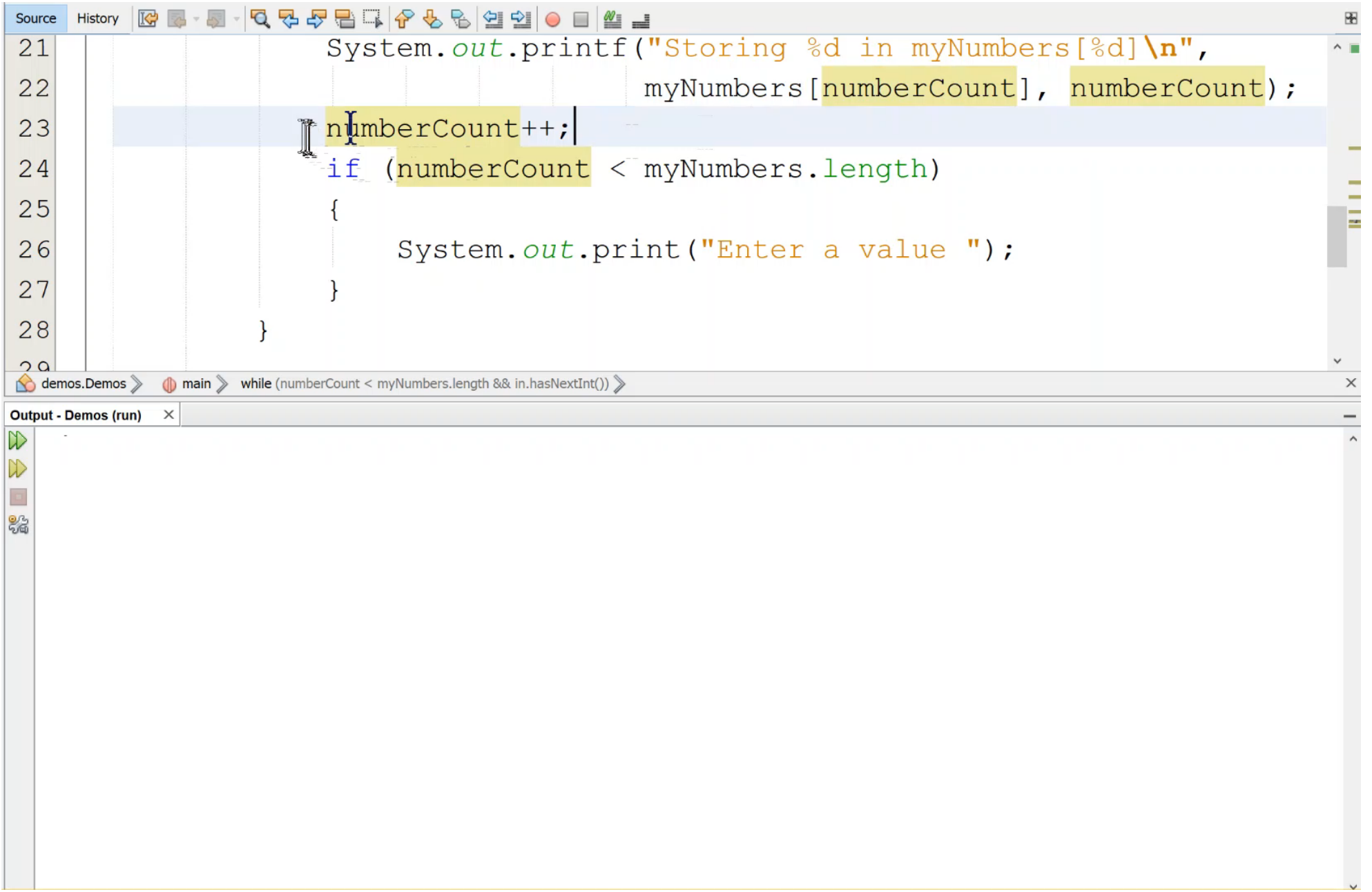
Partial Arrays

We printed out the elements containing nothing (zeroes)

```
for (int i = 0; i < myNumbers.length; i++)
{
     System.out.printf("myNumbers[%d] = %d\n", i, myNumbers[i]);
}
```

So if we know we might partially fill the array, then we should track how elements we added and use that counter for the loop test condition.

```
for (int i = 0; i < numberCount; i++)
{
     System.out.printf("myNumbers[%d] = %d\n", i, myNumbers[i]);
}
```

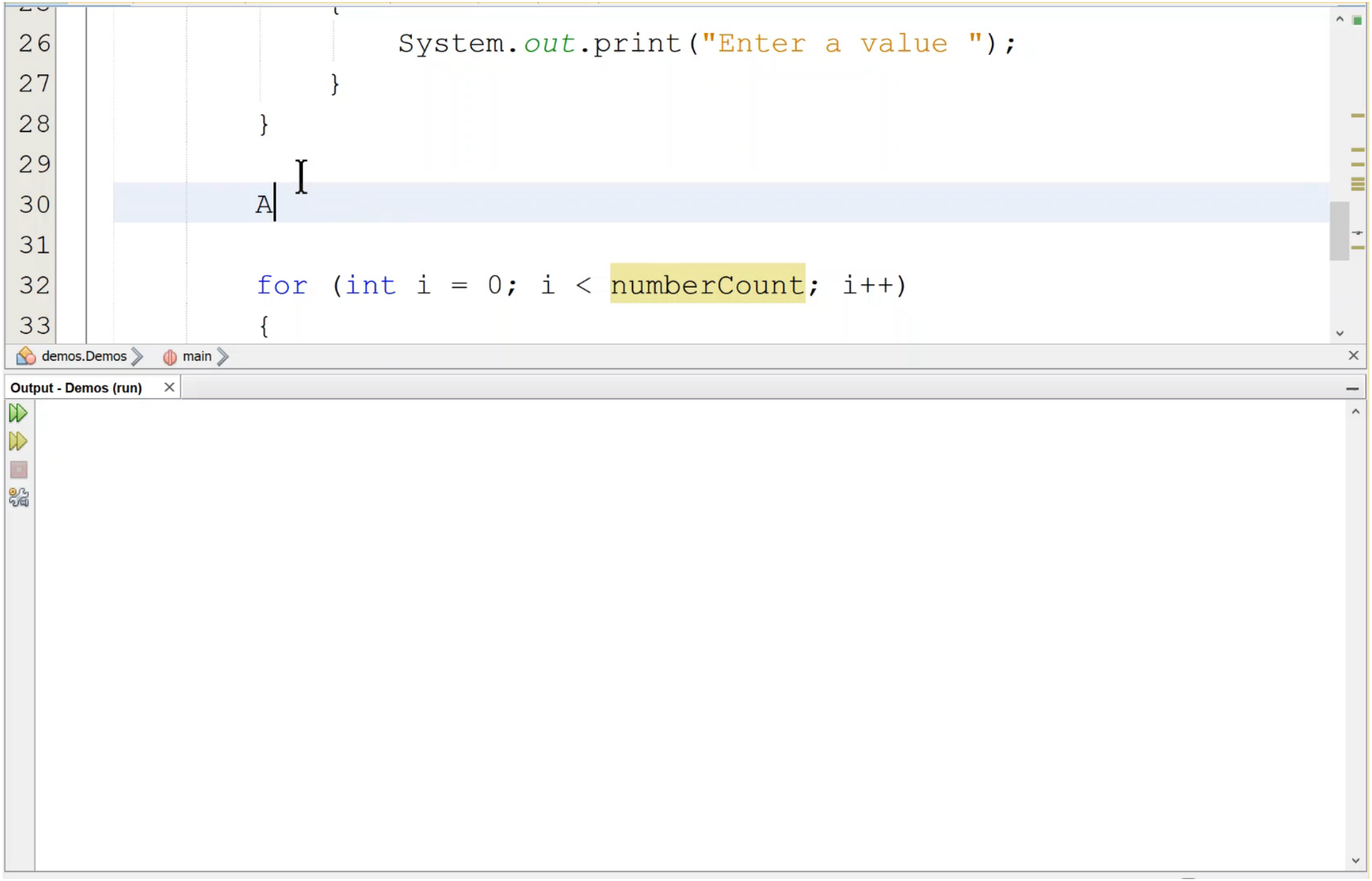

Source
History
21 System.out.printf("Storing %d in myNumbers[%d]\n",
22 myNumbers[numberCount], numberCount);
23 numberCount++;
24 if (numberCount < myNumbers.length)
25 {
26 System.out.print("Enter a value ");
27 }
28 }
demos.Demos
main
while (numberCount < myNumbers.length && in.hasNextInt())
Output - Demos (run)

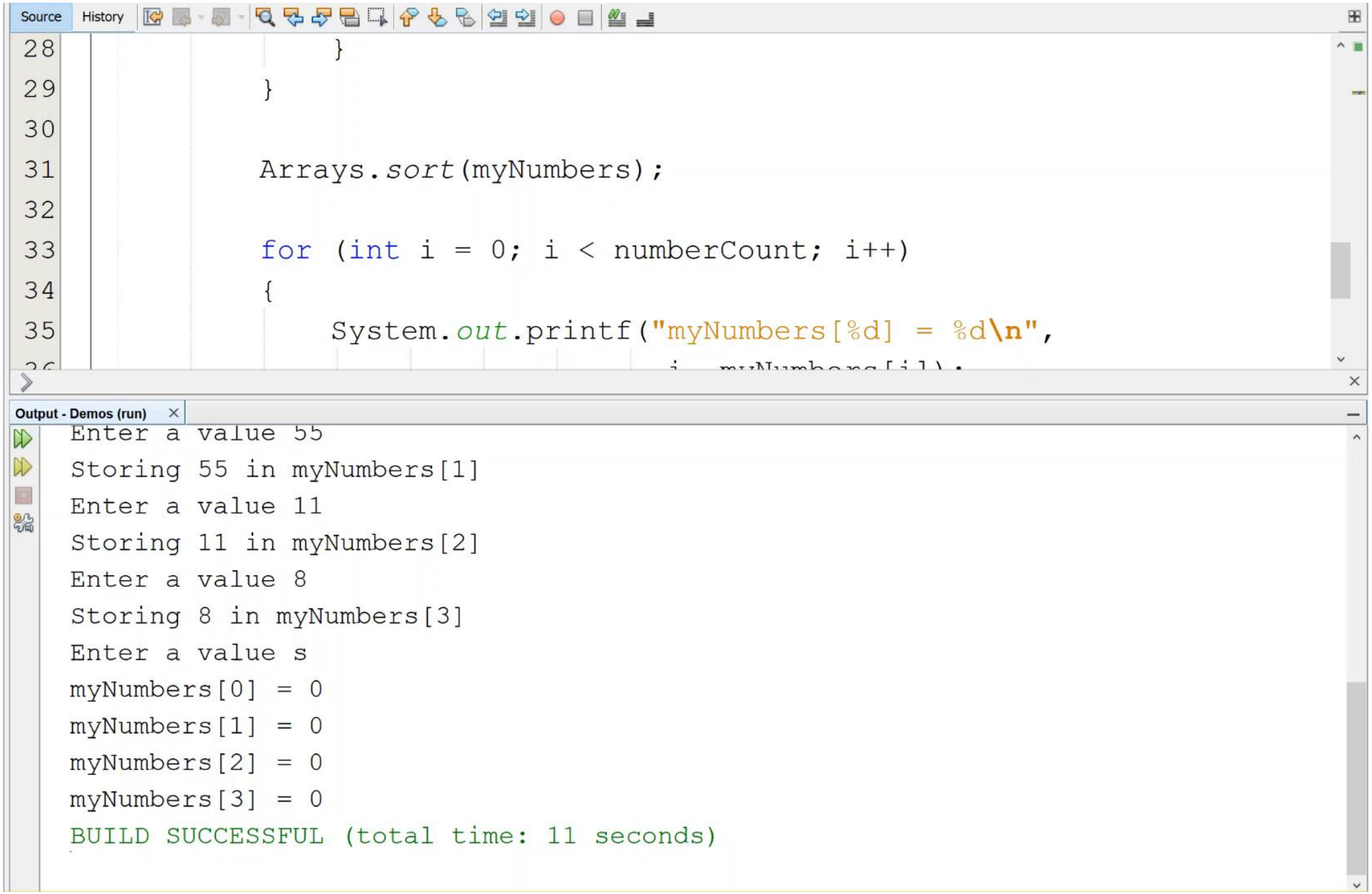
Partial Arrays

What if we have a partial array and we want to sort it?

```
Arrays.sort(myNumbers);
```

What happens to the unfilled array cells?


System.out.print("Enter a value ");
}
}
A
for (int i = 0; i < numberCount; i++)
{
demos.Demos
main
Output - Demos (run)


Source
History
28 }
29 }
30
31 Arrays.sort(myNumbers);
32
33 for (int i = 0; i < numberCount; i++)
34 {
35 System.out.printf("myNumbers[%d] = %d\n",
Output - Demos (run)
Enter a value 55
Storing 55 in myNumbers[1]
Enter a value 11
Storing 11 in myNumbers[2]
Enter a value 8
Storing 8 in myNumbers[3]
Enter a value s
myNumbers[0] = 0
myNumbers[1] = 0
myNumbers[2] = 0
myNumbers[3] = 0
BUILD SUCCESSFUL (total time: 11 seconds)

Partial Arrays

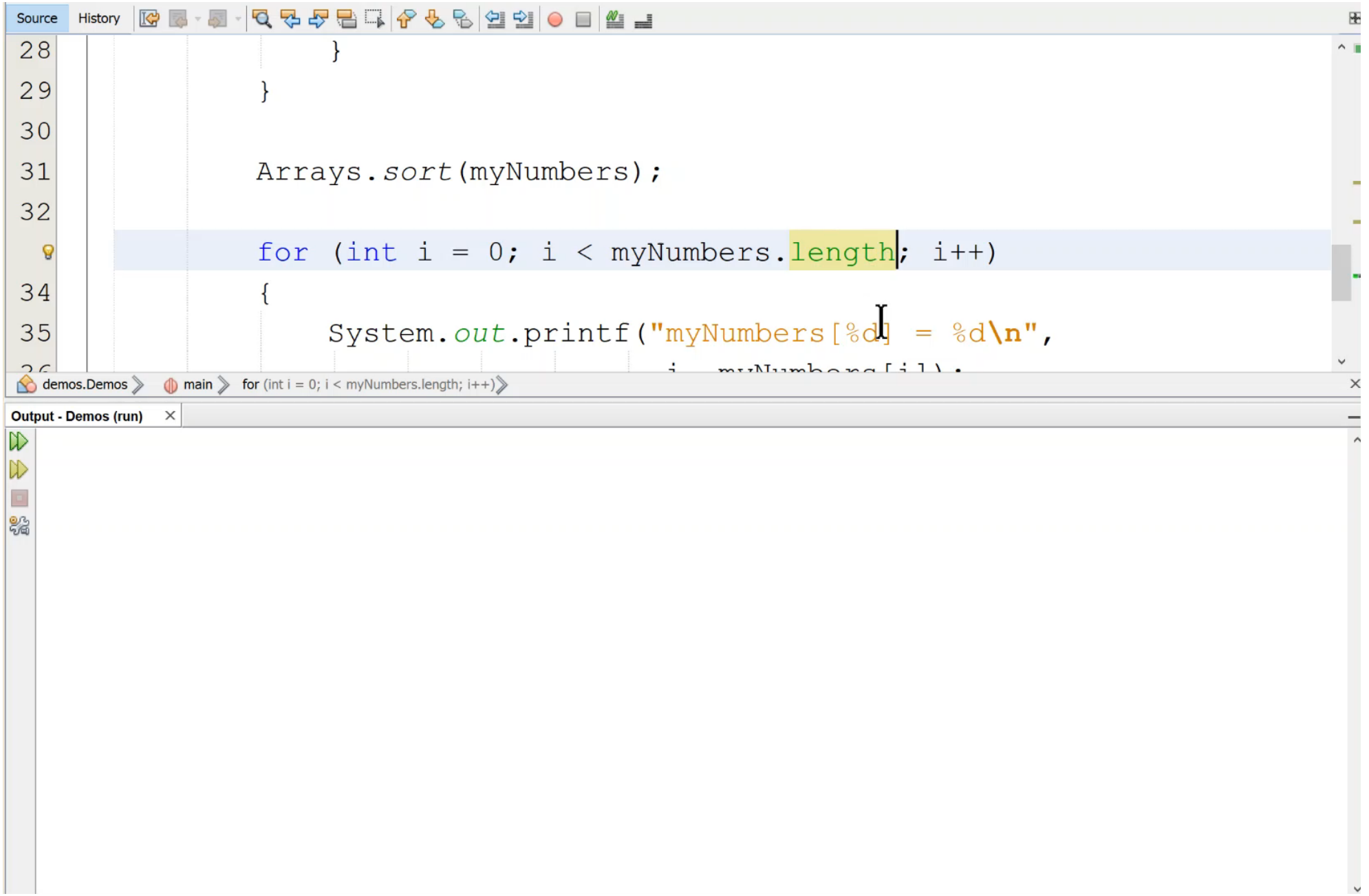
When we count how many elements were put into the array, then we can also control our sort and not have an issue with the 0's in the array sorting to the start of the array.

Sort entire array including empty elements

```
Arrays.sort(myNumbers);
```

Sort array elements 0 through numberCount

```
Arrays.sort(myNumbers, 0, numberCount);
```


Source
History
28 }
29 }
30
31 Arrays.sort(myNumbers);
32
for (int i = 0; i < myNumbers.length; i++)
34 {
35 System.out.printf("myNumbers[%d] = %d\n",
demos.Demos
main
for (int i = 0; i < myNumbers.length; i++)
Output - Demos (run)

Declaring a Two Dimensional Array

To create a two dimensional (2D) array, we need the number of rows and columns.

We can use those two values to create the 2D array.

```
int my2D[][] = new int[numberOfRows][numberOfColumns];
```

Declaring a Two Dimensional Arrays

We can initialize it when we create it

```
int my2D [][] = {{0,1,2,3},
                 {4,5,6,7},
                 {8,9,10,11}};
```

0	1	2	3
4	5	6	7
8	9	10	11

This would create an array/matrix with 3 rows and 4 columns.

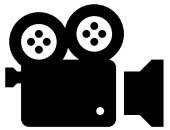
```
int my2D[][] = {{0,1,2,3},{4,5,6,7},{8,9,10,11}};
```

Source History

```
23          int row = 3, col = 4;
24
25          int[][] my2D = new int[row][col];
26
27          for (int i = 0; i < row; i++)
28          {
29              for (int j = 0; j < col; j++)
30              {
31                  System.out.printf("%d\t",my2D[i][j]);
32              }
33              System.out.println();
34          }
```

studentcode.StudentCode > main >

Output - StudentCode (run)



Initializing a 2D Array

If nested for loops can be used to display an array, then couldn't we use a similar construct to initialize/update an array?

```
for (int i = 0; i < row; i++)
{
    for (int j = 0; j < col; j++)
    {
        my2D[i][j] = i*2;
        System.out.printf("%d\t",my2D[i][j]);
    }
    System.out.println();
}
```

0	0	0	0
2	2	2	2
4	4	4	4

```
23          int row = 3, col = 4;
24
25          int[][] my2D = new int[row][col];
26
27          for (int i = 0; i < row; i++)
28          {
29              for (int j = 0; j < col; j++)
30              {
31                  my2D[i][j] = i * 2;
32                  System.out.printf("%d\t",my2D[i][j]);
33              }
34              System.out.println();
35          }
```

studentcode.StudentCode > main > for (int i = 0; i < row; i++) > for (int j = 0; j < col; j++)

Output - StudentCode (run)

Creating a method to print a 2D Array

```
public static void print2DArray(int[][] my2D, int row, int col)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            System.out.printf("%d\t",my2D[i][j]);
        }
        System.out.println();
    }
}
```

Creating a method to print a 2D Array

If we wanted to print the WHOLE array, then we don't need to pass in the row and column – an array knows that information.

```
public static void print2DArray(int[][] my2D)
{
    for (int i = 0; i < my2D.length; i++)
    {
        for (int j = 0; j < my2D[0].length; j++)
        {
            System.out.printf("%d\t",my2D[i][j]);
        }
        System.out.println();
    }
}
```

`my2D.length` is the number of rows

`my2D.length[0]` is the number of columns

```
public static void PrintMT(int [][] MT)
{
   System.out.print("    ");
   for(int i = 1; i <= MT.length; i++ )
   {
      System.out.printf("%5d",i);
   }
   System.out.println();
   System.out.print("   ");

   for (int i = 1; i <= MT.length ; i++)
   {
      System.out.print("-----");
   }
   System.out.println();

   for (int i = 0; i < MT.length; i++)
   {
      System.out.printf("%d |", i+1);
      for (int j = 0; j < MT[0].length; j++)
      {
         System.out.printf("%5d", MT[i][j]);
      }
      System.out.println();
   }
}
```

```
    1    2    3    4    5    6    7    8    9
 ------------------------------------------------
1 |   1    2    3    4    5    6    7    8    9
2 |   2    4    6    8   10   12   14   16   18
3 |   3    6    9   12   15   18   21   24   27
4 |   4    8   12   16   20   24   28   32   36
5 |   5   10   15   20   25   30   35   40   45
6 |   6   12   18   24   30   36   42   48   54
7 |   7   14   21   28   35   42   49   56   63
8 |   8   16   24   32   40   48   56   64   72
9 |   9   18   27   36   45   54   63   72   81
```

`Integer`

`Integer` is a class; therefore, can do things and knows things.

We have already seen this

`Integer.MAX_VALUE` `Integer.MIN_VALUE;`

`Integer` can also do things. `parseInt()` is one of `Integer`'s many methods.

Integer.parseInt()

Convert a string that only contains numbers into a number

Integer.parseInt

```
String intString = "123";
int intValue = Integer.parseInt(intString);

System.out.print(intValue);
```

```
123
```

Source History

```
 9      public static void main(String[] args)
10      {
11          Scanner in = new Scanner(System.in);
12
13          String Rope = "123";
14          int Lasso = Integer.parseInt(Rope);
15          System.out.printf("%s %d", Rope, Lasso);
16
```

Output

Code6_1000074079 (run) | parseInt (run)

```
ction.name=run run
init:
Deleting: C:\Users\frenc\Documents\NetBeansProjects\parseInt\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\frenc\Documents\NetBeansProjects\parseInt\build\built-jar.properties
Compiling 1 source file to C:\Users\frenc\Documents\NetBeansProjects\parseInt\build\classes
compile:
run:
123 123BUILD SUCCESSFUL (total time: 2 seconds)
```

Integer.parseInt()

You may think that you can just cast the variable to an integer using `(int)`

```
String intString = "123";
int intValue = int(intString);

System.out.print(intValue);
```

Source History

```
9       public static void main(String[] args)
10      {
11          Scanner in = new Scanner(System.in);
12
13          String Rope = "123";
            int Lasso = (int)Rope;
            System.out.printf("%s %d", Rope, Lasso);
```

Output

Code6_1000074079 (run) | parseInt (run)

```
Deleting: C:\Users\frenc\Documents\NetBeansProjects\parseInt\build\built-jar.p
roperties
deps-jar:
Updating property file: C:\Users\frenc\Documents\NetBeansProjects\parseInt\bui
ld\built-jar.properties
Compiling 1 source file to C:\Users\frenc\Documents\NetBeansProjects\parseInt\
build\classes
C:\Users\frenc\Documents\NetBeansProjects\parseInt\src\parseint\ParseInt.java:
14: error: incompatible types: String cannot be converted to int
        int Lasso = (int)Rope;
C:\Users\frenc\Documents\NetBeansProjects\parseInt\src\parseint\ParseInt.java:
20: error: variable Lasso is already defined in method main(String[])
        int Lasso = Integer.parseInt(Rope);
```

`Double`

`Double` is a class; therefore, can do things and knows things.

We have already seen this with

`Double.MAX_VALUE` `Double.MIN_VALUE;`

`Double` can also do things. `parseDouble()` is one of `Double`'s many methods.

Double.parseDouble()

Convert a numeric string into a number

Double.parseDouble

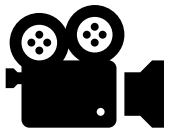
```
String doubleString = "123.45";
double doubleValue = Double.parseDouble(doubleString);

System.out.print(doubleValue);
```

```
123.45
```

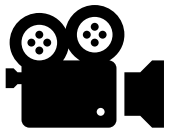
`Integer.parseInt()` and `Double.parseDouble()`

What happens if we try to parse a string containing a decimal number with `parseInt()`?

What happens if we try to parse a string containing an integer number with `parseDouble()`?



```
8   {
9       public static void main(String[] args)
10      {
11          Scanner in = new Scanner(System.in);
12
13          String Rope = "123.1";
14          int Lasso = Integer.parseInt(Rope);
15          System.out.printf("%s %d", Rope, Lasso);
16
17
18
19
20      }
21  }
22
```



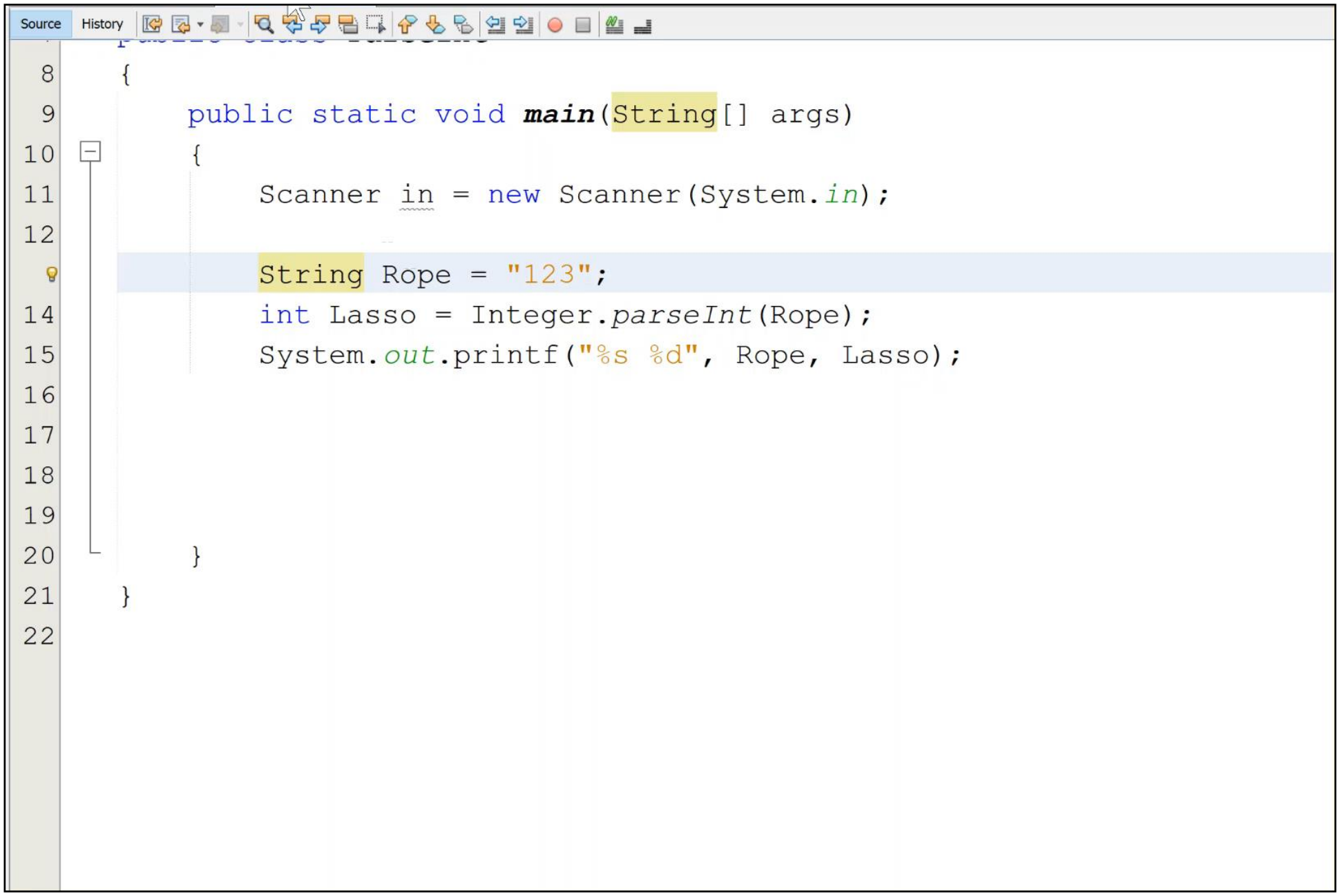
```
8   {
9       public static void main(String[] args)
10      {
11          Scanner in = new Scanner(System.in);
12
13          String Rope = "123";
14          Double Lasso = Double.parseDouble(Rope);
15          System.out.printf("%s %f", Rope, Lasso);
16
17
18
19
20      }
21  }
22
```

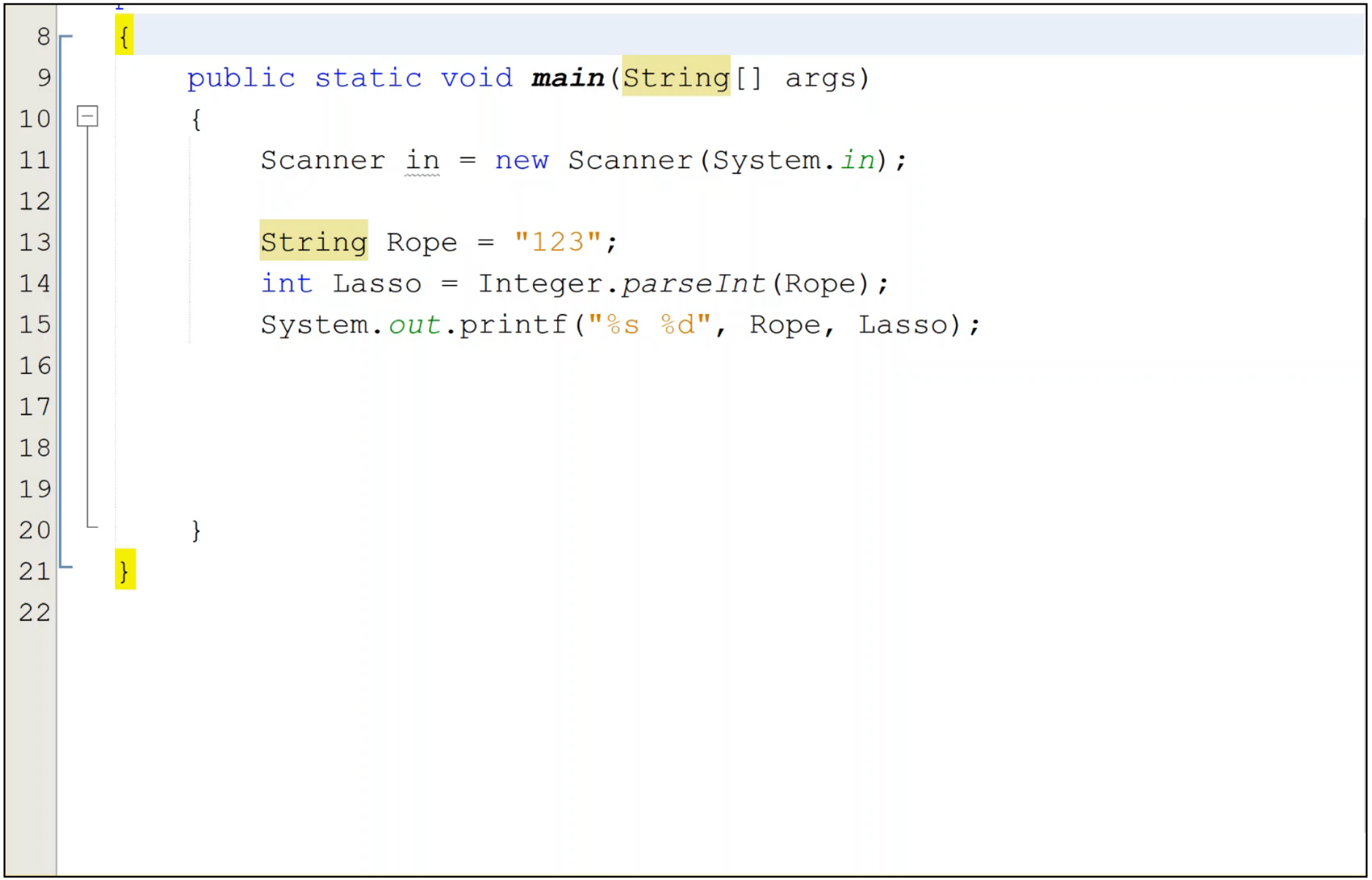
Integer.parseInt()

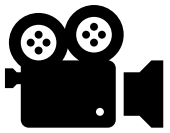
`parseInt()` will throw an exception if the value passed to it is not something that can be converted to an integer.

Whether that value is a decimal or a string that is not all integers.

```
String Rope = "123A";
int Lasso = Integer.parseInt(Rope);

String Rope = "A123";
int Lasso = Integer.parseInt(Rope);
```


Source
History
{
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
String Rope = "123";
int Lasso = Integer.parseInt(Rope);
System.out.printf("%s %d", Rope, Lasso);
}
}


{
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
String Rope = "123";
int Lasso = Integer.parseInt(Rope);
System.out.printf("%s %d", Rope, Lasso);
}
}



Source History

```
7   public class ParseInt
8   {
        public static void main(String[] args)
10      {
            Scanner in = new Scanner(System.in);
12
13          System.out.print("Enter a movie seat ");
14          String MovieSeat = in.next();
15          MovieSeat = MovieSeat.toUpperCase();
16
17          char Row = MovieSeat.charAt(0);
18          int Seat = Integer.parseInt(MovieSeat.substring(1));
19
20          System.out.printf("Your seat is in Row %c and Seat %d\n",
21                  Row, Seat);
22      }
23  }
24
```

```
Scanner in = new Scanner(System.in);

System.out.print("Enter a movie seat ");

String MovieSeat = in.next();
MovieSeat = MovieSeat.toUpperCase();

char Row = MovieSeat.charAt(0);
int Seat = Integer.parseInt(MovieSeat.substring(1));

System.out.printf("Your seat is in Row %c and Seat %d\n",
                  Row, Seat);
```

`Math`

`Math` is a class; therefore, can do things and knows things.

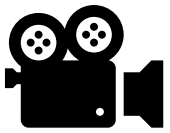
We have already seen this with

```
Math.PI
```

`Math` can also do things. We have already used several of `Math`'s methods

Common Mathematical Methods

`Math.pow(base, exp)`	Returns `base` raised to the power of `exp`.
`Math.ceil(n)`	Rounds `n` up to the nearest integer
`Math.floor(n)`	Rounds `n` down to the nearest integer
`Math.round(n)`	Adds `0.5` to `n` and then applies `Math.floor()`
`Math.abs(n)`	Returns the absolute value of `n`
`Math.min(a,b)`	Returns the minimum of `a` or `b`
`Math.max(a,b)`	Returns the maximum of `a` or `b`



```
6
7   public class MathDemo
8   {
9       public static void main(String[] args)
10      {
11          Scanner in = new Scanner(System.in);
12
13          System.out.print("Enter two numbers ");
14          int N1 = in.nextInt();
15          int N2 = in.nextInt();
16          System.out.printf("%d is the max\n", Math.max(N1, N2));
17          System.out.printf("%d is the min\n", Math.min(N1, N2));
18      }
19  }
```

```
int n = -192;
System.out.println(n);
System.out.println(Math.abs(n));
```

```
-192
192
```

```
double n = -1.4;
System.out.println(n);
System.out.println(Math.abs(n));
```

```
-1.4
1.4
```

```
int a = 4, b = 10;
System.out.println(Math.min(a,b));
System.out.println(Math.max(a,b));
```

```
4
10
```

```
double a = 4.1234, b = 4.1235;
System.out.println(Math.min(a,b));
System.out.println(Math.max(a,b));
```

```
4.1234
4.1235
```

`Character`

`Character` is a class; therefore, can do things and knows things.

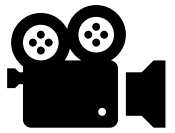
We haven't seen any of those yet so let's look at some…

Methods in the **`Character`** Class

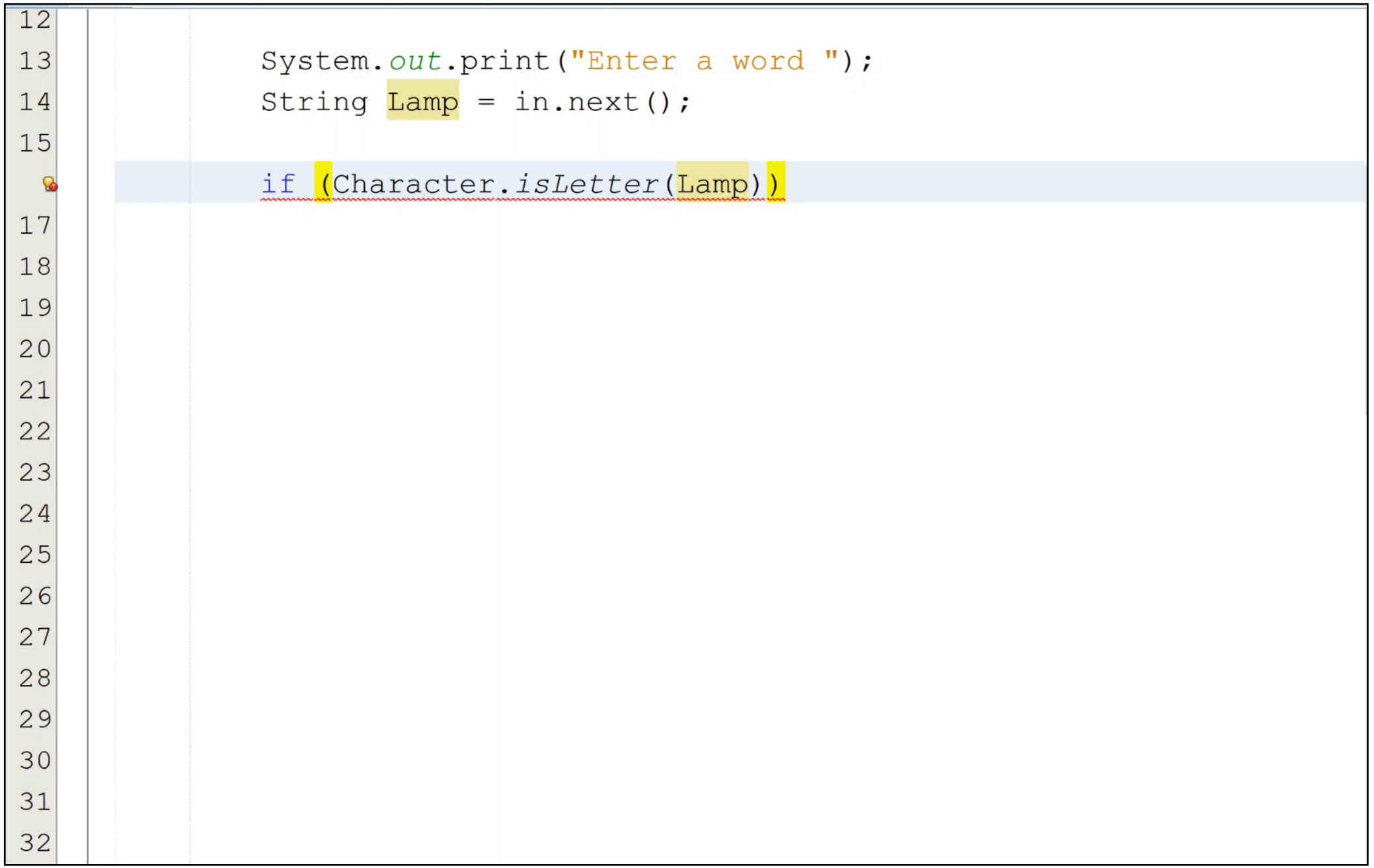
`isdigit(ch)`	Returns true if the specified character is a digit
`isLetter(ch)`	Returns true if the specified character is a letter
`isLetterOrDigit(ch)`	Return true if the specified character is a digit or letter
`isWhiteSpace(ch)`	Return true if the specified character is a space
`isLowerCase(ch)`	Returns true if the specified character is lowercase
`isUpperCase(ch)`	Returns true if the specified character is uppercase
`toLowerCase(ch)`	Returns the lowercase of the specified character
`toUpperCase(ch)`	Returns the uppercase of the specified character

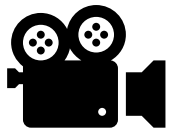
All of these should be prefixed with `Character.` in order to call them.

```
System.out.print("Enter a character ");
char ch = in.next().charAt(0);

if (Character.isLetterOrDigit(ch))
    System.out.println("We have a letter or digit!!");

if (Character.isDigit(ch))
    System.out.println("We have a digit!");
else if (Character.isLetter(ch))
{
    System.out.printf("We have a letter!\nand it is ");
    if (Character.isUpperCase(ch))
    {
        System.out.printf("UPPERCASE\nat least it was until...");
        ch = Character.toLowerCase(ch);
    }
    else
    {
        System.out.printf("lowercase\nat least is was until...");
        ch = Character.toUpperCase(ch);
    }
    System.out.printf("\nwe changed it!\t%c\n", ch);
}
else
    System.out.printf("Don't know what that is!\n");
```



```
9       public static void main(String[] args)
10      {
11          Scanner in = new Scanner(System.in);
12
13          System.out.print("Enter a character ");
14          char ch = in.next().charAt(0);
15
16          if (Character.isLetterOrDigit(ch))
17              System.out.println("We have a letter or digit!!");
18
19          if (Character.isDigit(ch))
20              System.out.println("We have a digit!");
21          else if (Character.isLetter(ch))
22          {
23              System.out.printf("We have a letter!\nand it is ");
24              if (Character.isUpperCase(ch))
25              {
26                  System.out.printf("UPPERCASE\nat least it was until..
27                  ch = Character.toLowerCase(ch);
28              }
29              else
```


12
13 System.out.print("Enter a word ");
14 String Lamp = in.next();
15
if (Character.isLetter(Lamp))
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32



Source History

```
12
13          System.out.print("Enter a number ");
14          int Lamp = in.nextInt();
15
16          if (Character.isLetter(Lamp))
17              System.out.printf("%d is a letter\n", Lamp);
18          else
19              System.out.printf("%d is not a letter\n", Lamp);
20
21
22
23
24
25
26
27
28
29
30
31
32
```

`String`

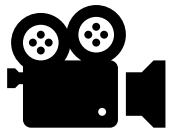
`String` is a class; therefore, can do things and knows things.

We have already seen this with

```
charAt()
toUpperCase()
toLowerCase()
```

Simple Methods for `String` Objects

`length()`	Returns the number of characters in this string
`charAt(index)`	Returns the character at the specified index from this string
`concat(s1)`	Returns a new string that concatenates this string with string s1
`toUpperCase()`	Returns a new string with all letters in uppercase
`toLowerCase()`	Returns a new string with all letters in lowercase
`trim()`	Returns a new string with whitespace characters trimmed on both sides



Source History

```
    public static void main(String[] args)
    {
        String Mom = "Duchess";
        String [] Kindle = {"Toulouse","Berlioz", "Marie"};

        int StringLength = Mom.length();
        int ArrayLength = Kindle.length;

        System.out.printf("Length of String is %d\n", StringLength);
        System.out.printf("Length of Array is %d\n", ArrayLength);


    }
}
```

```
10      public static void main(String[] args)
11      {
12          Scanner in = new Scanner(System.in);
13          String Phrase[] = new String[5];
            String Word = "";
15
16          System.out.printf("Enter a 5 word phrase  ");
17
18          for (int i = 0; i < 5; i++)
19          {
20              Phrase[i] = in.next();
21          }
22
23          System.out.print("Enter a word ");
24          Word = in.next();
25
26          System.out.printf("Word.length() = %d\n", Word.length());
27          System.out.printf("Phrase.length = %d\n", Phrase.length);
28      }
29  }
```

```
run:
Enter a 5 word phrase  red orange yellow blue green
Enter a word rainbow
Word.length() = 7
Phrase.length = 5
BUILD SUCCESSFUL (total time: 19 seconds)
```

```
String s1 = "String1 ";
String s2 = "String2 ";
String s3 = "String3 ";
String s4 = s1.concat(s2);


System.out.println(s1+s2+s3);
System.out.println(s4);

s1 = s1.toUpperCase();

System.out.println(s1);

System.out.println(s1.trim()+s2.trim()+s3.trim());

System.out.println(s1+s2+s3);

System.out.println(s1.toLowerCase()+s2.toUpperCase());
```

```
String1 String2 String3

String1 String2

STRING1

STRING1String2String3

STRING1 String2 String3

string1 STRING2
```

Comparison Methods for `String` Objects

`equals(s1)`	Returns true if this string is equal to string s1
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string s1; it is case insensitive
`compareTo(s1)`	Returns an integer great than 0 or less than 0 to indicate whether this string is greater than, equal to or less than s1
`compareToIgnoreCase`	Same as `compareTo` except that the comparison is case insensitive
`startsWith(prefix)`	Returns true if this string starts with the specified prefix
`endsWith(suffix)`	Returns true if this string ends with the specified suffix
`contains(s1)`	Returns true if s1 is a substring of this string

```
String s1 = "String1 ";
String s2 = "Strong2 ";
String s3 = "String3 ";
String s4 = "String";

if (s1.startsWith(s4))
   System.out.printf("s1 \"%s\" starts with \"%s\"\n", s1, s4);
else
   System.out.printf("s1 \"%s\" does not start with \"%s\"\n", s1, s4);

if (s2.startsWith(s4))
   System.out.printf("s2 \"%s\" starts with \"%s\"\n", s2, s4);
else
   System.out.printf("s2 \"%s\" does not start with \"%s\"\n", s2, s4);
```

```
s1 "String1 " starts with "String"
s2 "Strong2 " does not start with "String
```

```
String s1 = "String";
String s2 = "Strong";
String s3 = "String3 ";
String s4 = "ing";

if (s1.endsWith(s4))
   System.out.printf("s1 \"%s\" ends with \"%s\"\n", s1, s4);
else
   System.out.printf("s1 \"%s\" does not end with \"%s\"\n", s1, s4);

if (s2.endsWith(s4))
   System.out.printf("s2 \"%s\" ends with \"%s\"\n", s2, s4);
else
   System.out.printf("s2 \"%s\" does not end with \"%s\"\n", s2, s4);


s1 "String" ends with "ing"
s2 "Strong" does not end with "ing"
```

```
String s1 = "String";
String s2 = "Strong";
String s3 = "String3 ";
String s4 = "tr";

if (s1.contains(s4))
    System.out.printf("s1 \"%s\" contains \"%s\"\n", s1, s4);
else
    System.out.printf("s1 \"%s\" does not contain \"%s\"\n", s1, s4);

if (s2.contains(s4))
    System.out.printf("s2 \"%s\" contains \"%s\"\n", s2, s4);
else
    System.out.printf("s2 \"%s\" does not contain \"%s\"\n", s2, s4);
```

```
s1 "String" contains "tr"

s2 "Strong" contains "tr"
```

Finding a Character in a String

`indexOf(ch)`

Returns the index of the first occurrence of `ch` in the string.
Returns -1 if not matched.

`indexOf(ch, fromIndex)`

Return the index of the first occurrence of ch after `fromIndex` in the string.
Returns -1 if not matched.

s	u	p	e	r	c	a	l	i	f	r	a	g	i	l	i	s	t	i	c	e	x	p	e	d	i	a	l	i	d	o	c	i	o	u	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

```
String s1 = "supercalifragilisticexpedialidocious";
char c1 = 'l';

System.out.printf("%c is at index %d\n", c1, s1.indexOf(c1));

System.out.printf("%s starts at index %d\n", c1, s1.indexOf(c1,10));

System.out.printf("%s starts at index %d\n", c1, s1.indexOf(c1,20));
```

```
l is at index 7

l starts at index 14

l starts at index 27
```

Finding a Substring in a String

`indexOf(s)`

Returns the index of the first occurrence of string `s` in the string.

Returns -1 if not matched.

`indexOf(s, fromIndex)`

Return the index of the first occurrence of string `s` after `fromIndex` in the string.

Returns -1 if not matched.

e	n	c	y	c	l	o	p	e	d	i	a
0	1	2	3	4	5	6	7	8	9	10	11

```
String s1 = "encyclopedia";
String s2 = "clo";
String s3 = "ped";

int index = s1.indexOf(s2);

System.out.printf("%s starts at index %d\n", s2, index);

System.out.printf("%s starts at index %d\n", s3, s1.indexOf(s3));
```

```
clo starts at index 4

ped starts at index 7
```

s	u	p	e	r	c	a	l	i	f	r	a	g	i	l	i	s	t	i	c	e	x	p	e	d	i	a	l	i	d	o	c	i	o	u	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

```
String s1 = "supercalifragilisticexpedialidocious";

String s2 = "li";

System.out.printf("%s starts at index %d\n", s2, s1.indexOf(s2));

System.out.printf("%s starts at index %d\n", s2, s1.indexOf(s2,10));

System.out.printf("%s starts at index %d\n", s2, s1.indexOf(s2,20));
```

```
li starts at index 7

li starts at index 14

li starts at index 27
```

Finding a Character in a String

`lastIndexOf(ch)`

Returns the index of the last occurrence of `ch` in the string.

Returns -1 if not matched.

`lastIndexOf(ch, fromIndex)`

Return the index of the last occurrence of `ch` working backwards from `fromIndex` in the string.

Returns -1 if not matched.

s	u	p	e	r	c	a	l	i	f	r	a	g	i	l	i	s	t	i	c	e	x	p	e	d	i	a	l	i	d	o	c	i	o	u	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

```
String s1 = "supercalifragilisticexpedialidocious";
char c1 = 'e';

System.out.printf("%c is at index %d\n", c1, s1.lastIndexOf(c1));

System.out.printf("%s starts at index %d\n", c1, s1.lastIndexOf(c1,22));

System.out.printf("%s starts at index %d\n", c1, s1.lastIndexOf(c1,10));
```

```
e is at index 23

e starts at index 20

e starts at index 3
```

Finding a Substring in a String

`lastIndexOf(s)`

Returns the index of the last occurrence of `s` in the string.

Returns -1 if not matched.

`lastIndexOf(s, fromIndex)`

Return the index of the last occurrence of `s` working backwards from `fromIndex` in the string.

Returns -1 if not matched.

s	u	p	e	r	c	a	l	i	f	r	a	g	i	l	i	s	t	i	c	e	x	p	e	d	i	a	l	i	d	o	c	i	o	u	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

```
String s1 = "supercalifragilisticexpedialidocious";
String s2 = "li";

System.out.printf("%s is at index %d\n", s2, s1.lastIndexOf(s2));
System.out.printf("%s is at index %d\n", s2, s1.indexOf(s2));
System.out.printf("%s starts at index %d\n", s2, s1.lastIndexOf(s2,22));
System.out.printf("%s starts at index %d\n", s2, s1.lastIndexOf(s2,10));
System.out.printf("%s starts at index %d\n", s2, s1.lastIndexOf(s2,5));

li is at index 27
li is at index 7
li starts at index 14
li starts at index 7
li starts at index -1
```

`split`

`split` is used to split a string into tokens based on a delimiter or set of delimiters

If we have the string

apple,banana,coconut,dragonfruit

We can use method `split()` to separate this string into tokens (pieces) based on the comma delimiter

Method `split()` will put the pieces/tokens into an array where each element of the array is one of the pieces/tokens.

split

`split` is used to split a string into tokens based on a delimiter or set of delimiters

```
String MyArray[] = MyString.split("[delimiters]");
```

MyString is the string to be separated/split into tokens/pieces

delimiters is the characters used to separate/split **MyString**

MyArray is the array where `split()` stores the token/pieces. Its type must be `String` since `split()` only works on `String` input

```
String MyArray[] = MyString.split("[delimiters]");
```

```
String[] tokens = "Java is fun!".split("[ ]");

for(int i = 0; i < tokens.length; i++)
   System.out.println(tokens[i]);
```

```
Java
is
fun!
```

```
String[] tokens = "Java is fun!".split("[a]");

for(int i = 0; i < tokens.length; i++)
   System.out.println(tokens[i]);
```

```
J
v
 is fun!
```

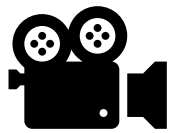
```
String[] tokens = "Java is fun!".split("[v]");

for(int i = 0; i < tokens.length; i++)
   System.out.println(tokens[i]);
```

```
Ja
a is fun!
```

```
System.out.print("Enter a sentence: ");
String sentence = in.nextLine();

String tokens[] = sentence.split("[ ]");

for(int i = 0; i < tokens.length; i++)
    System.out.println(tokens[i]);


Enter a sentence: Hello there! How are you?

Hello
there!
How
are
you?
```

```
System.out.print("Enter a sentence ");
String sentence = in.nextLine();

String tokens[] = sentence.split("[|]");

for(int i = 0; i < tokens.length; i++)
   System.out.println(tokens[i]);

Enter a sentence This|is|what|a|pipe|delimited|file|from|Excel|looks|like

This
is
what
a
pipe
delimited
file
from
Excel
looks
like
```

```
Scanner Zebra = new Scanner(System.in);

System.out.print("Enter a phrase to split ");
String Banana = Zebra.nextLine();

System.out.print("Enter the delimiters ");
String Delimiter = Zebra.nextLine();

String Tokens[] = Banana.split(Delimiter);

System.out.printf("Your input was divided into %d tokens\n", Tokens.length);

for (int i = 0; i < Tokens.length; i++)
      System.out.println(Tokens[i]);
```



```
8   {
9       public static void main(String[] args)
10      {
11          Scanner Zebra = new Scanner(System.in);
12
13          System.out.print("Enter a phrase to split ");
14          String Banana = Zebra.nextLine();
15
16          System.out.print("Enter the delimiters ");
17          String Delimiter = Zebra.nextLine();
18
19          String [] Tokens = Banana.split(Delimiter);
20
21          System.out.printf("Your input was divided into %d tokens\n",
22                  Tokens.length);
23
24          for (int i = 0; i < Tokens.length; i++)
25              System.out.println(Tokens[i]);
26      }
27  }
28
```