CSCI 132:
Basic Data Structures and Algorithms

Arrays

Reese Pearsall

Spring 2025

Announcements

Lab 4 due **tomorrow** at 11:59PM

Program 1 due next Wednesday

```
Roses are Red,
Violets are Blue.

Unexpected ‘{’ on line 32.
```

What do you need to dig a hole?

What do you need to dig a hole?

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
		
		
		

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
		
		

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
		

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
	• Really good at digging	• Takes up a lot of garage space

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
	• Really good at digging	• Takes up a lot of garage space

Best tool for the job?

Burying your pet goldfish

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
	• Really good at digging	• Takes up a lot of garage space

Best tool for the job?

Building Express tunnel to Bridger Bowl

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
	• Really good at digging	• Takes up a lot of garage space

Best tool for the job?

Creating the foundation for a house

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
	• Really good at digging	• Takes up a lot of garage space

Best tool for the job?

Digging a Well for water

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

Best tool for the job?

Digging a Well for water

What do you need to dig a hole?

We can't use the best tool for the job unless we know that tool exists!

Best tool for the job?

Digging a Well for water

What do you need to dig a hole?

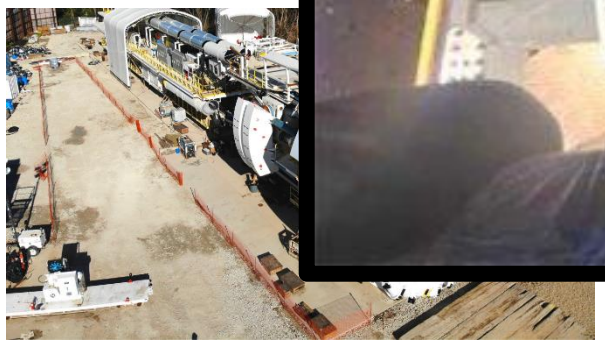
	Pros	Cons
	• Cheap • Precise • No Training • Availability	• Slow • Labor
	• Fast • Labor	• Expensive • Training
	• Really good at digging	• Takes up a lot of garage space

Best tool for the job?

Creating the foundation for a house

Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

	Pros	Cons
		
		ve
		a lot of ace

Best tool for the job?

Creating the foundation for a house

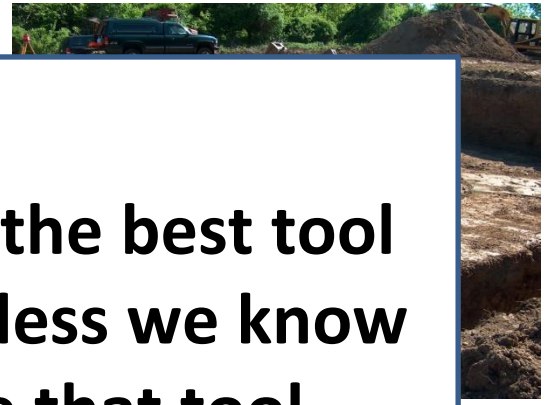
Thank you to Sean Yaw (the goat) for the analogy

What do you need to dig a hole?

Pros	Cons
	• Slow • Labor

Best tool for the job?

Creating the foundation for a house

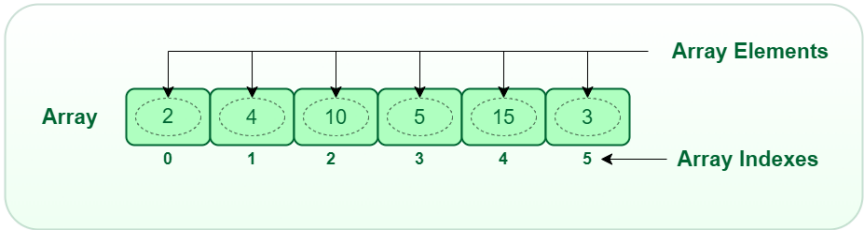


We can't use the best tool for the job unless we know how to use that tool

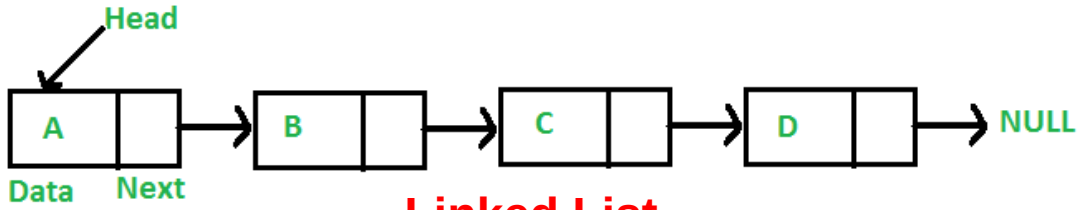
garage space

Thank you to Sean Yaw (the goat) for the analogy

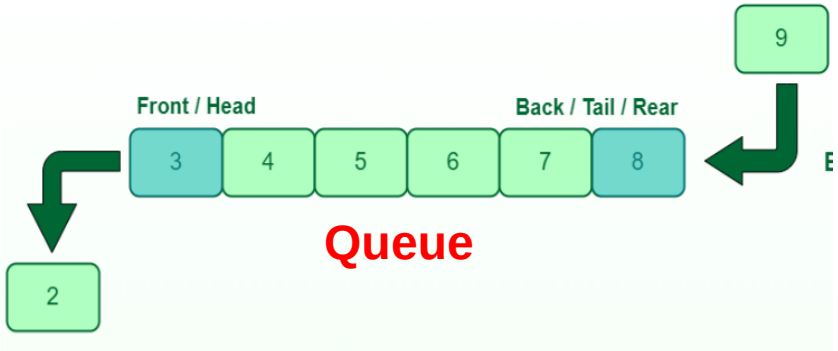
A **data structure** is a mechanism for storing and organizing data



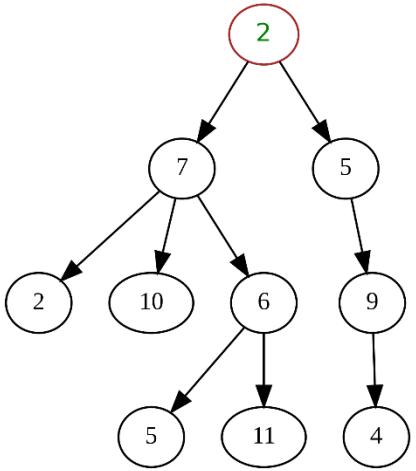
Arrays



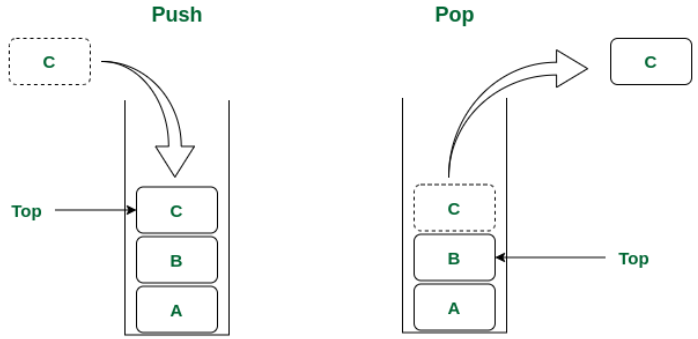
Linked List



Queue



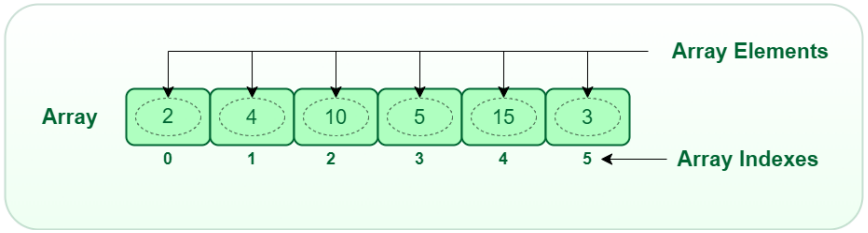
Trees



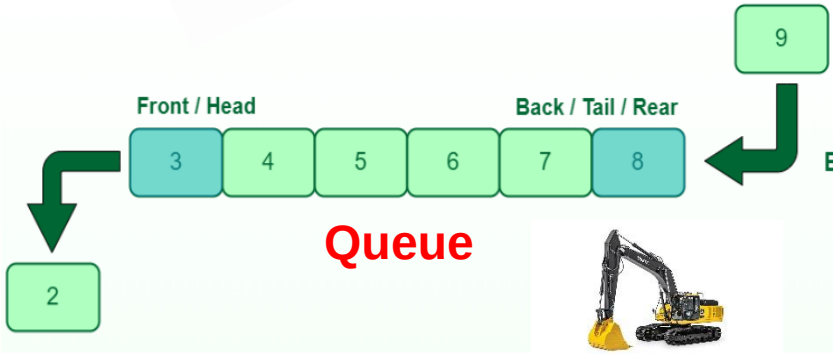
Stack

A **data structure** is a mechanism for storing and organizing data

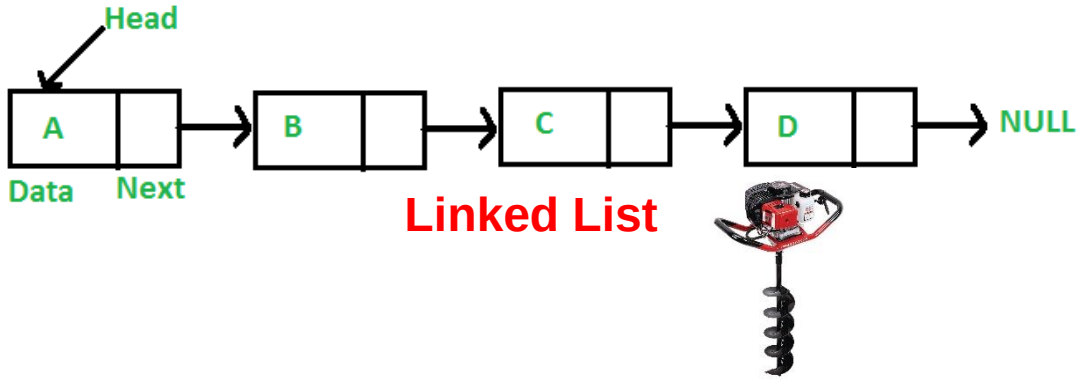
- We have structured ways of *accessing* and *managing* data



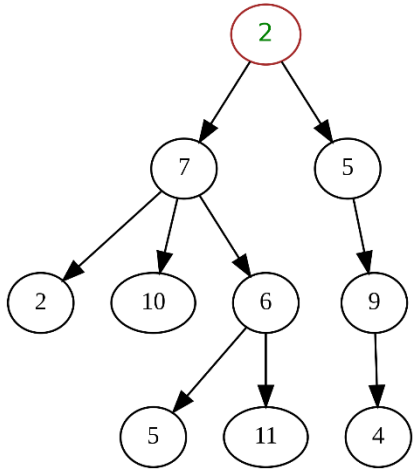
Arrays



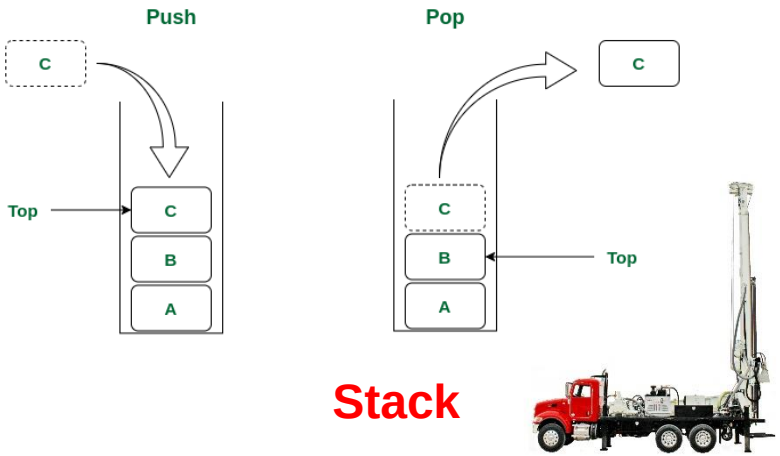
Queue



Linked List



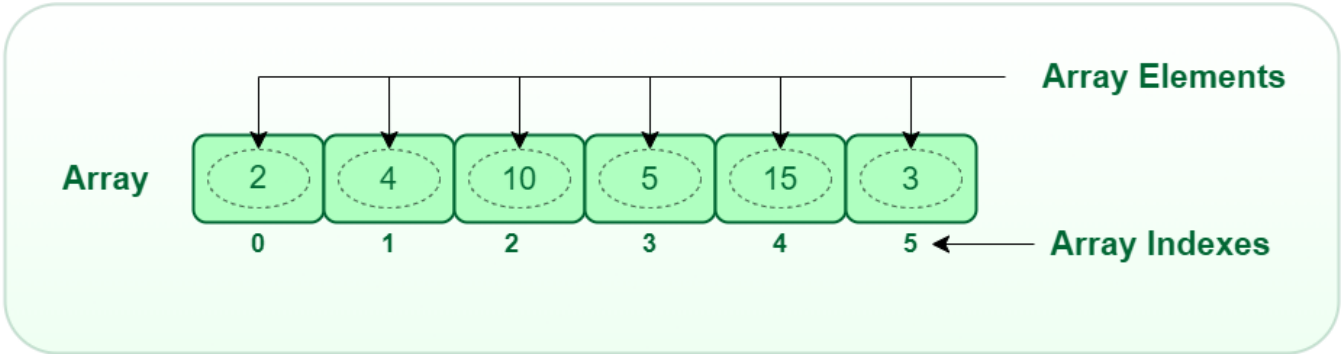
Trees



Stack

There are many types of data structure, and each data structure has its pros and cons

An **array** is a data structure that can hold multiple, similar values



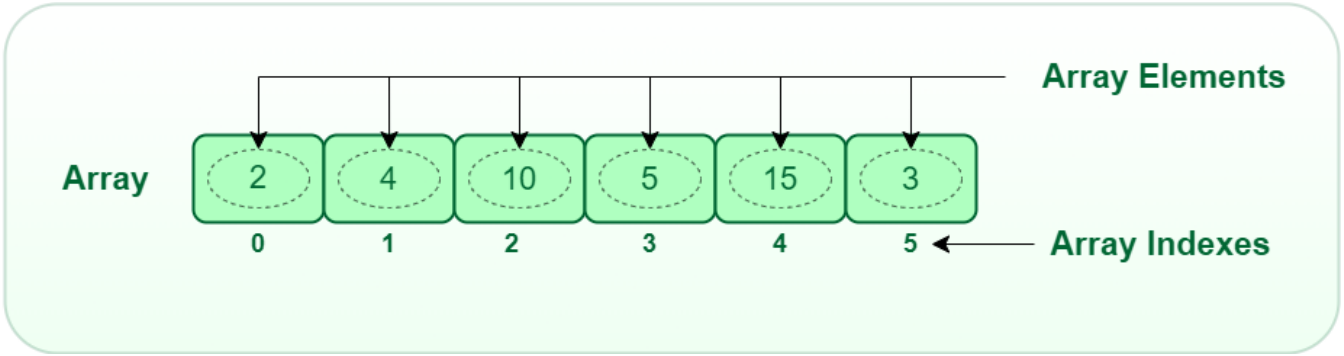

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

int[] myNum = {10, 20, 30, 40};
```

Pros

- Holds multiple pieces of information
- Information is ordered (by index)
- Can easily change what is stored in each slot
- Can store duplicate data
- Easy to iterate through

An **array** is a data structure that can hold multiple, similar values




```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

int[] myNum = {10, 20, 30, 40};
```

Pros

- Holds multiple pieces of information
- Information is ordered (by index)
- Can easily change what is stored in each slot
- Can store duplicate data
- Easy to iterate through

Cons

- Can’t change the length
- Can only store one data type

Array Limitations

Cons

- **Can't change the length** *What can we do about this?*
- Can only store one data type

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));
```

What if we wanted to add 4 to the array?

Array Limitations

Cons

- **Can't change the length** *What can we do about this?*
- Can only store one data type

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));


int[] newArray = new int[myArray.length + 1];          // Create a new array that is one spot bigger
for(int i = 0; i < myArray.length; i++) {
        newArray[i] = myArray[i];                      // Fill new array with contents of old array
}
```

Array Limitations

Cons

- **Can't change the length** *What can we do about this?*
- Can only store one data type

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));
```

```
int[] newArray = new int[myArray.length + 1];          // Create a new array that is one spot bigger
for(int i = 0; i < myArray.length; i++) {
        newArray[i] = myArray[i];                      // Fill new array with contents of old array
}
```

```
int new_value = 4;
newArray[myArray.length] = new_value;                  // add new value to array
myArray = newArray;                                    // Update reference variable
```

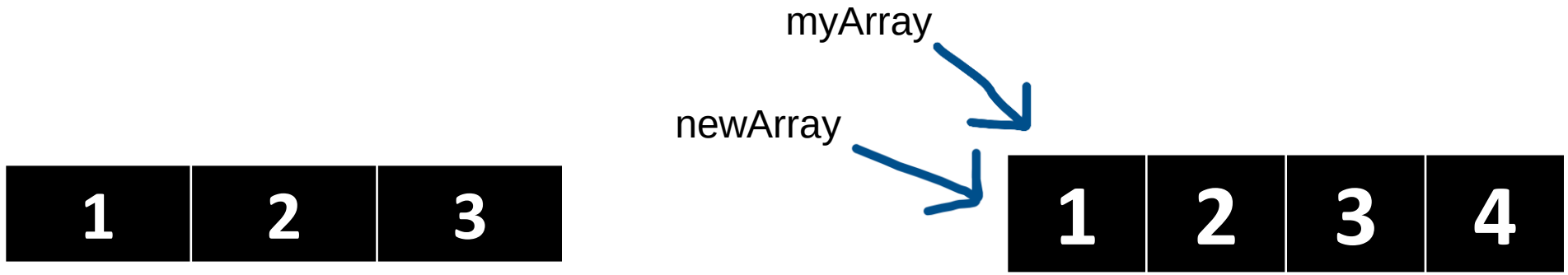
Array Limitations

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));

int[] newArray = new int[myArray.length + 1];
for(int i = 0; i < myArray.length; i++) {
            newArray[i] = myArray[i];
}

int new_value = 4;
newArray[myArray.length] = new_value;
myArray = newArray;
```

We updated our reference variable (`myArray`) to point to our new array with the new element



Array Limitations

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));

int[] newArray = new int[myArray.length + 1];
for(int i = 0; i < myArray.length; i++) {
            newArray[i] = myArray[i];
}

int new_value = 4;
newArray[myArray.length] = new_value;
myArray = newArray;
```

We updated our reference variable (`myArray`) to point to our new array with the new element

What happens to this array?
This is an unused object

Array Limitations

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));

int[] newArray = new int[myArray.length + 1];
for(int i = 0; i < myArray.length; i++) {
            newArray[i] = myArray[i];
}

int new_value = 4;
newArray[myArray.length] = new_value;
myArray = newArray;
```

We updated our reference variable (`myArray`) to point to our new array with the new element

Java has a mechanism called **Garbage Collection**, with deletes unused object to free up memory

(this runs automatically!)

What happens to this array?
This is an unused object

Array Limitations

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));

int[] newArray = new int[myArray.length + 1];
for(int i = 0; i < myArray.length; i++) {
            newArray[i] = myArray[i];
}

int new_value = 4;
newArray[myArray.length] = new_value;
myArray = newArray;
```

We updated our reference variable (`myArray`) to point to our new array with the new element

Java has a mechanism called **Garbage Collection**, with deletes unused object to free up memory

(this runs automatically!)

Java sees that we have an used/unreferenced object, so it will delete it!

Array Limitations

Cons

- **Can't change the length**

  Solution

  Create new array, copy everything over
  (this can be expensive ☹)

```
int[] myArray = {1, 2, 3};
System.out.println(Arrays.toString(myArray));


int[] newArray = new int[myArray.length + 1];
for(int i = 0; i < myArray.length; i++) {
            newArray[i] = myArray[i];
}


int new_value = 4;
newArray[myArray.length] = new_value;
myArray = newArray;
```

- **Can only store one data type**

  Solution

  Store an object, use two separate arrays, use a different data structure

We are going to write our own dynamic array data structure

Users should be able to:

1. Print the array
2. Add a new element to the array
3. Get an element at a particular index
4. Find the index of a particular element
5. Remove an element

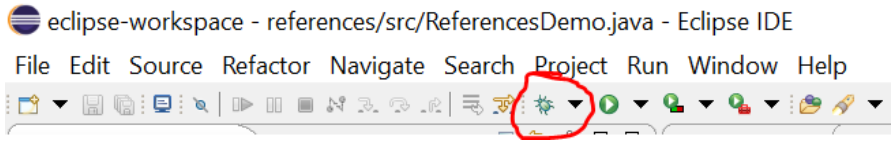
Debugging Code

```
2  public class ReferencesDemo {
3
4      public static void main(String[] args) {
5
6          String s1 = "reese";
7          String s2 = "reese";
8
9          System.out.println(s1 == s2);
10
11         String o1 = new String("reese");
12         String o2 = new String("reese");
13
14         System.out.println(o1 == o2);
15
16         System.out.println(o1.equals(o2));
17
18     }
19
20 }
21
```

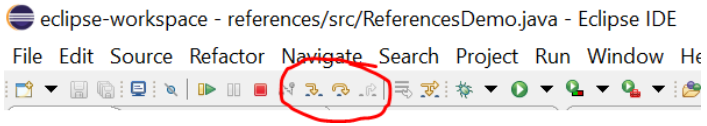
Our IDE has a super nifty **debugger**, which allows us to pause our code, and then step through each line in the control flow.

The first thing to do is to place a **breakpoint,** which is where execution will pause at, and debugging will begin

- Usually you try to place the breakpoint where you think things are going wrong



Then, press the little green bug icon next to the play button, which will run the debugger and stop at your breakpoint



Use the “step into” and “step over” buttons to start walking through your code

Debugging Code

Our IDE has a super slick debugger built in to it. I highly recommend learning how to use the debugger tool (see lecture)

Rubber Duck Debugging

Many programmers have had the experience of explaining a problem to someone else, possibly even to someone who knows nothing about programming, and then hitting upon the solution in the process of explaining the problem. In describing what the code is supposed to do and observing what it actually does, any incongruity between these two becomes apparent.[2] More generally, teaching a subject forces its evaluation from different perspectives and can provide a deeper understanding.[3] By using an inanimate object, the programmer can try to accomplish this without having to interrupt anyone else, and with better results than have been observed from merely thinking aloud without an audience.

(From Wikipedia)