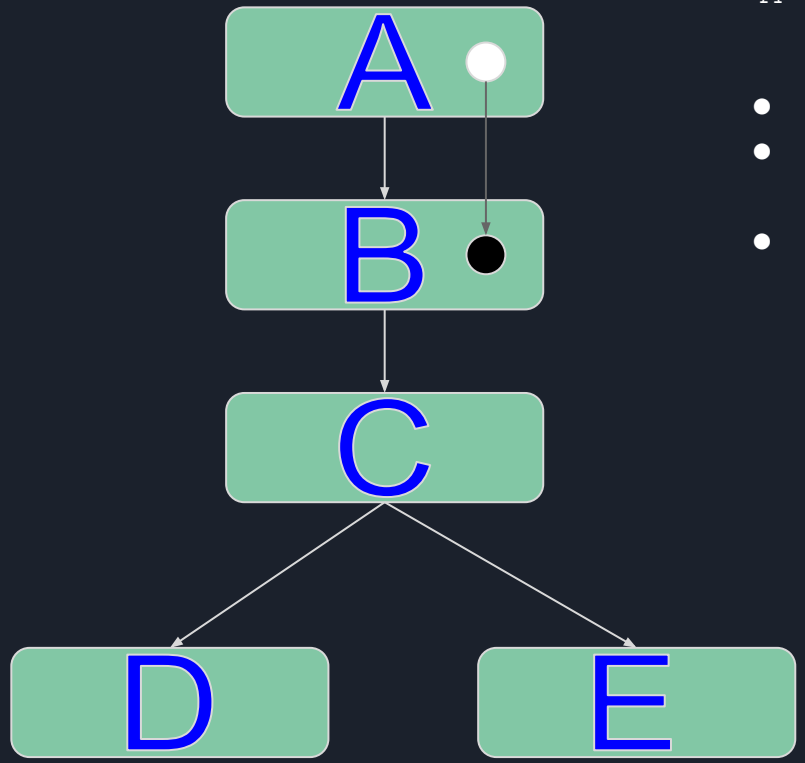
CS 314 Discussion

Problems

- Main Problem
  - Polymorphism
- Extra Problem
  - Baby Names!

Polymorphism

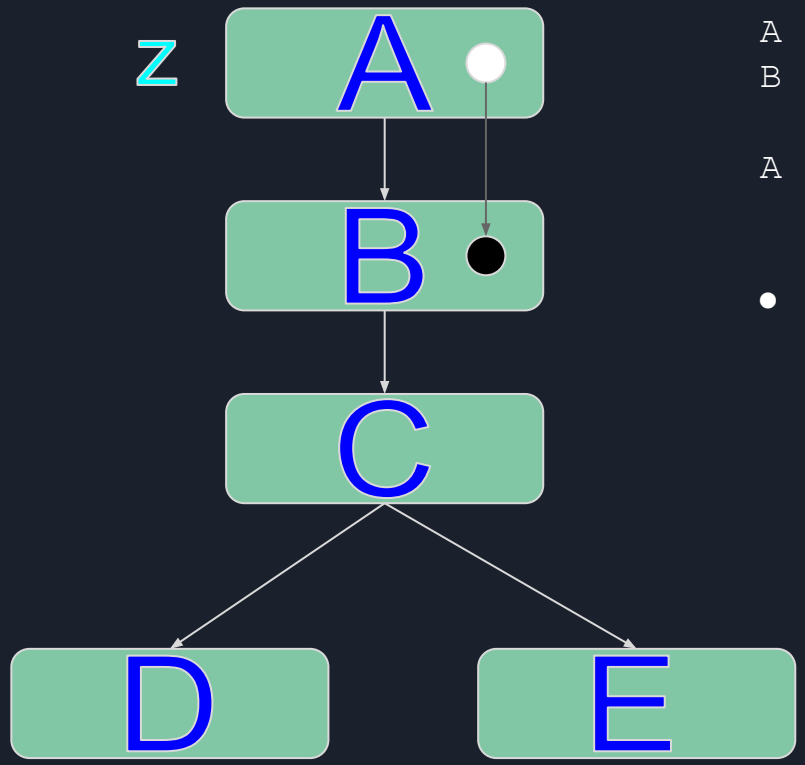



```
A obj = new B();
```

- A is the static type
- B is the dynamic type

- In order for the expression to be valid, there has to be a path from A directly equal/down to B

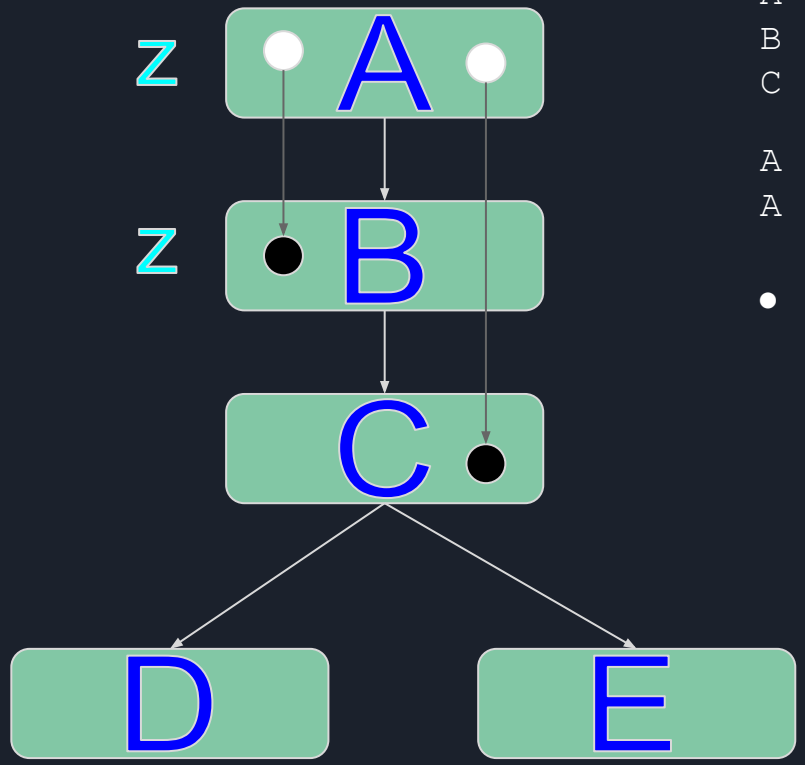
Polymorphism




```
A {z();}
B {}

A obj = new B();
```

- Rule: If obj.z(); a valid call:
  - z is a method in A (static type)

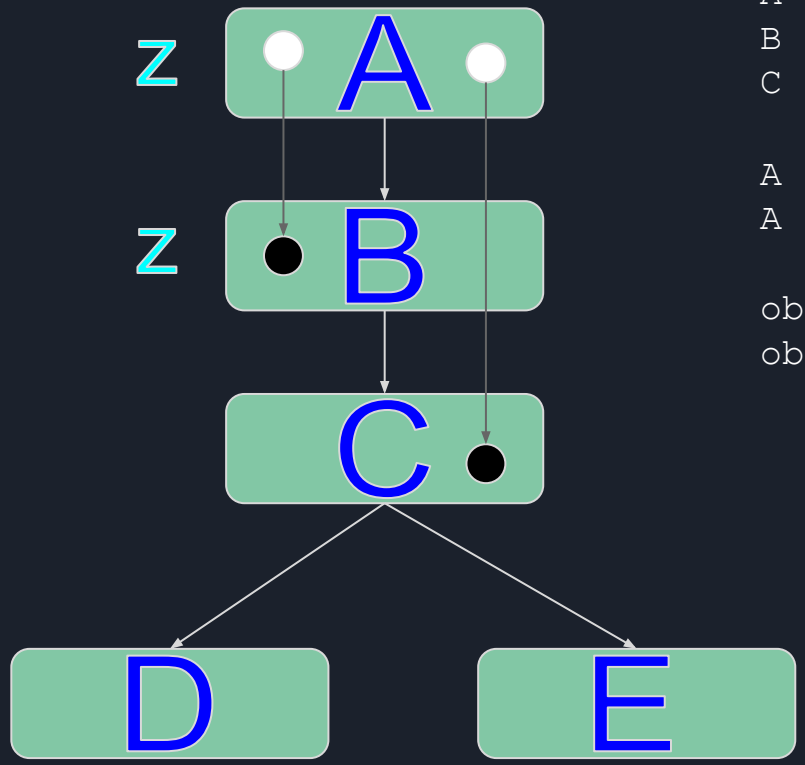
Polymorphism



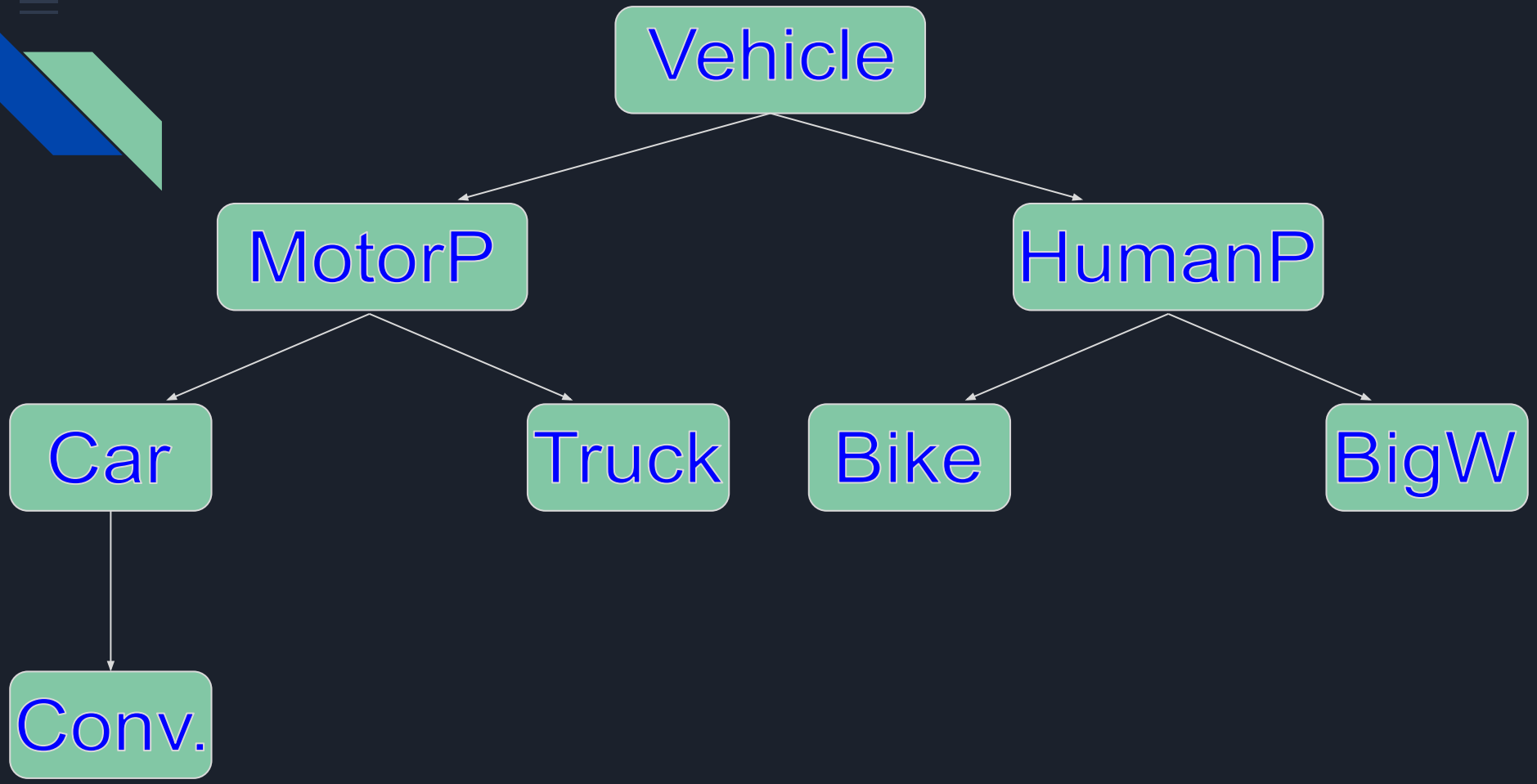

```
A {z();}
B {z();}
C {}

A obj1 = new C();
A obj2 = new B();
```

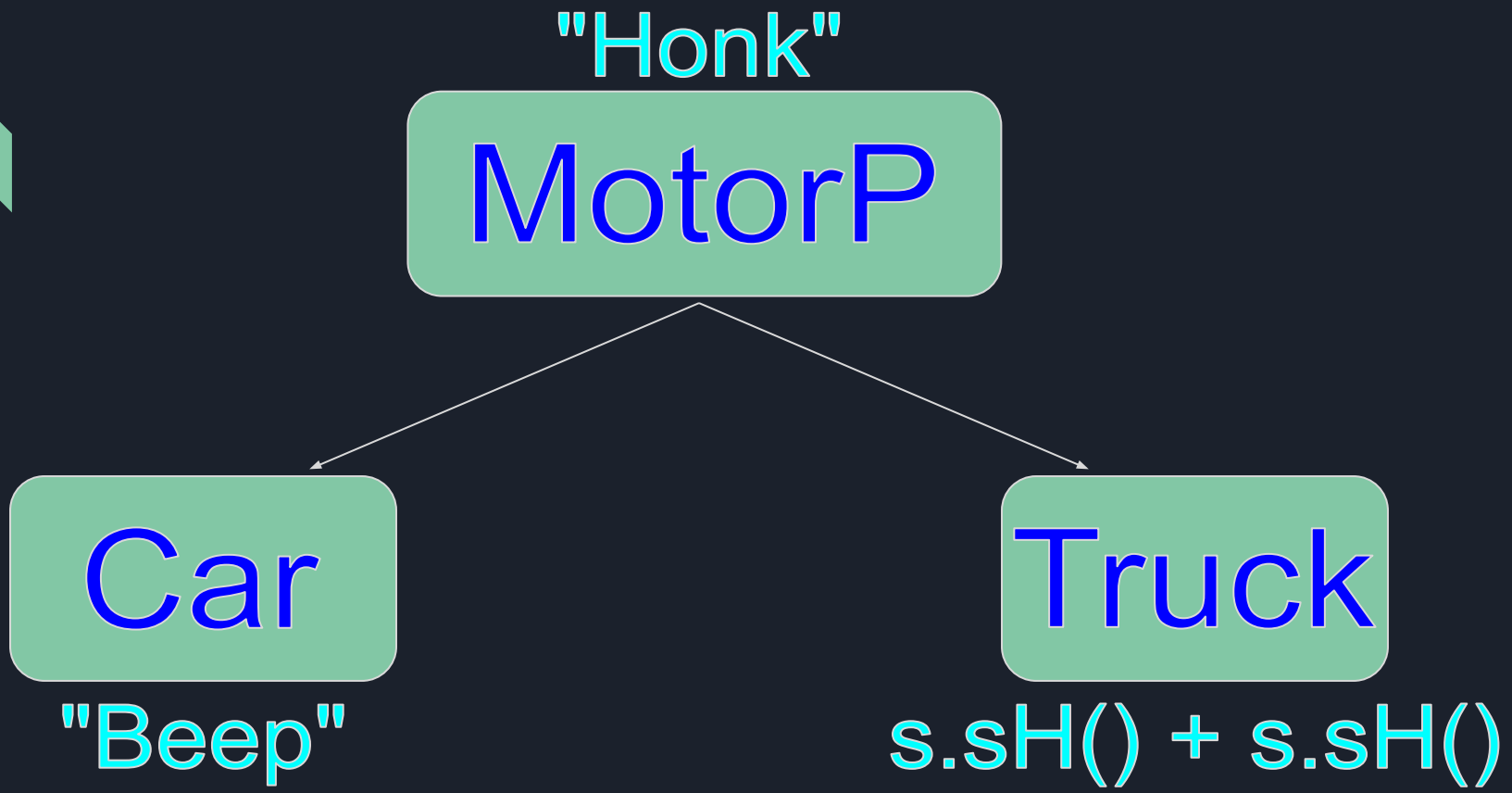
- Rule: If obj.z(); a valid call, the behavior is:
  - If B has its own implementation of z, then this will be used
  - If B doesn't have its own implementation of z, then the first class "above" C implementing z will be used

Polymorphism




```
A {z();}
B {z();}
C {}

A obj1 = new B();
A obj2 = new C();

obj1.z();
obj2.z();
```


Vehicle
MotorP
HumanP
Car
Truck
Bike
BigW
Conv.


"Honk"
MotorP
Car
Truck
"Beep"
s.sH() + s.sH()

```java
int newSize = 0;
for (int i = 0; i < size; i++) {
    int index = 0;
    boolean unique = true;
    while (index < newSize && unique) {
        unique = !con[i].equals(con[index]);
        index++;
    }
    if (unique) {
        con[newSize] = con[i];
        newSize++;
    }
}
for (int i = newSize; i < size; i++) {
    con[i] = null;
}
    int numRemoved = size - newSize;
    size = newSize;
    return numRemoved;
}
```

```java
public ArrayList<NameRecord> remove(int cutoff) {
    ArrayList<NameRecord> result = new ArrayList<NameRecord>();
    ArrayList<NameRecord> newInstanceVar = new ArrayList<NameRecord>();
    for (int i = 0; i < records.size(); i++) {
        NameRecord temp = records.get(i);
        if (temp.anyRanksGreater(cutoff)) {
            result.add(temp);
        } else {
            newInstanceVar.add(temp);
        }
    }
    records = newInstanceVar;
    return result;
}
```

Assignment Grading

- Don’t email me for:
  - Disagreeing on taking of a point for something you did
- *Please* email me for:
  - Mistake with your correctness
  - Mistake with adding up grade
  - Inconsistent deduction w/ past assignment
  - Inconsistency w/ assignment page
  - I took off for something you didn’t do

Assignment Grading

- I can only regrade for the five days after I release grades
- Don’t get stressed about small **style** deductions:
  - A single exam coding Q is a little less than an entire assignment
  - Assignments only make up 22% of your grade
  - Y’all get 40 slack points + 10 for extra credit
  - I lost 10 points on my first 3 assignments and had slack points to spare (we had ~25)

Common Style Issues

- Spacing on operators (***AUTO FORMATTER!!!!!!!***):
  - 3+3 -> 3 + 3
  - if(...) -> if (...)
  - public int method (){ -> method() {
  - //test -> // test
- Lines should be 100 long (set a vertical line)
- Private instance variables
- Checking preconditions

Common Style Issues

- ***USE AN AUTO FORMATTER!!!!!!!***
- ***USE AN AUTO FORMATTER!!!!!!!***
- ***USE AN AUTO FORMATTER!!!!!!!***
- ***USE AN AUTO FORMATTER!!!!!!!***
- ***USE AN AUTO FORMATTER!!!!!!!***

Common Style Issues

- Magic numbers:
  - BAD:
    - `if (year < 10)`
  - GOOD:
    - `final int PERIOD_LENGTH = 10;`
    - `if (year < PERIOD_LENGTH)`
  - If you're using a magic number in multiple methods, declare it at the top of your class.

Common Style Issues

- Returning early:

```
// BAD:
int sum = 0;
for (int i = 0; i < a.length; i++) {
    if (a[i] == 0) {
        sum += 1;
    }
}
return sum == 0;
```

```
// GOOD:
for (int i = 0; i < a.length; i++) {
    if (a[i] == 0) {
        return false;
    }
}
return true;
```

Common Style Issues

- Boolean zen (part 1):

```
// BAD:
if (a == 0) {
    return true;
} else {
    return false;
}
```

```
// GOOD:
return a == 0;
```

Common Style Issues

- Boolean zen (part 2):

```
// BAD:
if (val == true) {
    ...
}
```

```
// GOOD:
if (val) {
    ...
}
```

Common Style Issues

- Preferred method header comments:

```
// Calculates the amount of birds in my yard at a given time
// pre: bar != null, t >= 0
// post: returns birds at time t
public int foo(int[] bar, int t) {

// Prints the amount of snails on my desk
// pre: none (For this example, bar handles null vals)
// post: none
public void bar(String desk) {
```

Style Preferences

- I can't take off for this, but I'd prefer:

```
public void foo(String desk) {
    if (a) {

    }
}
```

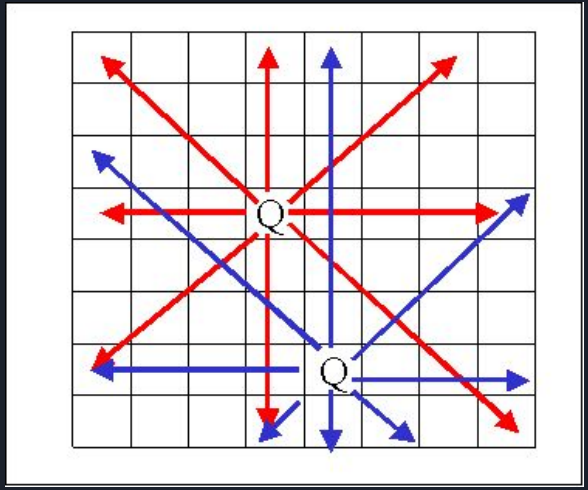
```
// rather than
public void foo(String desk)
{
    if (a)
    {

    }
}
```

Common Baby Names Issues

- Style:
  - Magic Numbers:
    - We should not have constants like 10 or 1000 in our code/names
    - Don’t have a constant for like VALUE_ZERO = 0;
  - Re-manipulating the same String repeatedly
  - Repetitive sorting
  - Check preconditions

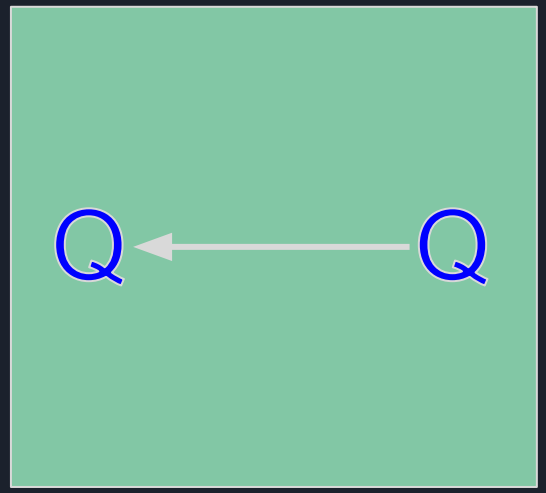
QueensAreSafe

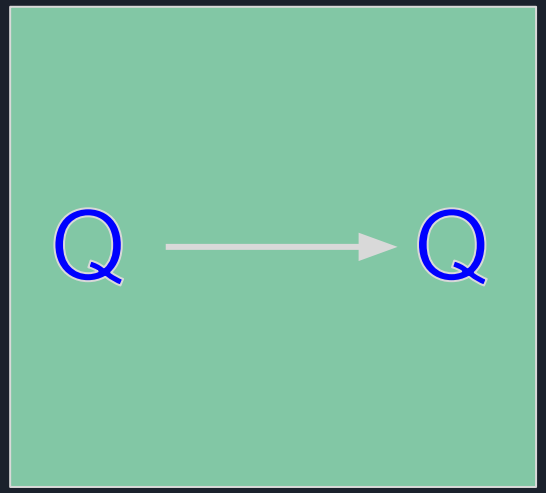
- Relevant Directions
- Parameterized Row/Col Solutions
- Slope Method

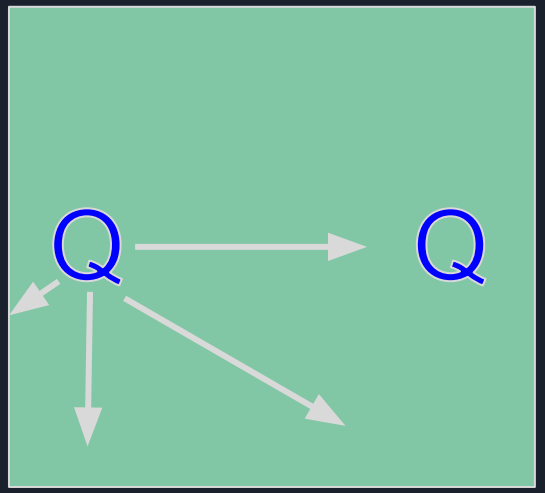


Directions

- Only need to check 4 directions

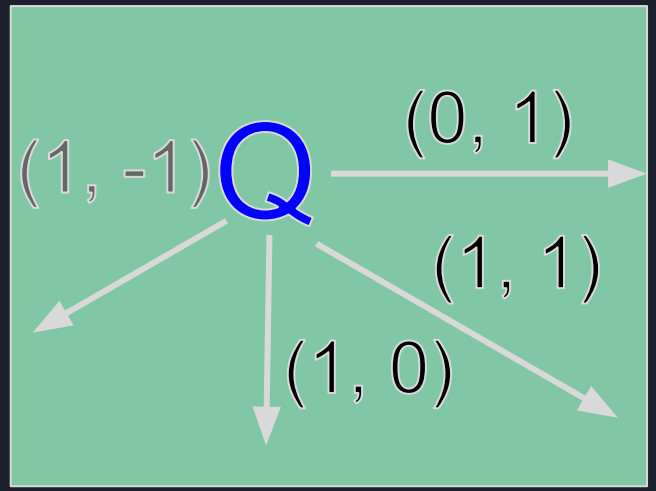






Directions

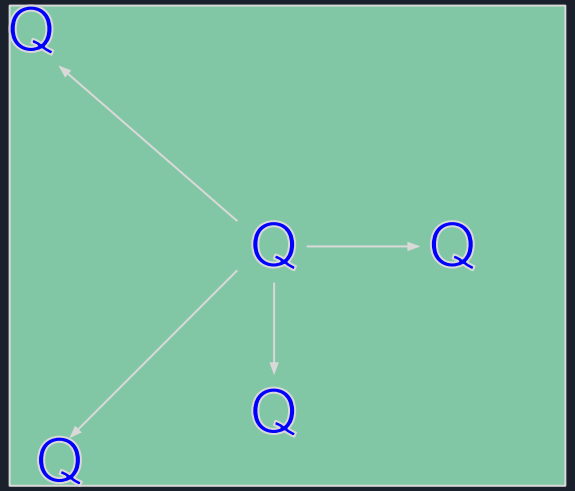
- You can use an array to store the different changes in rows and columns



rows = {0, 1, 1, 1}
cols = {1, 1, 0, -1}

Directions

- You can also use the slopes between queens to determine if they're in a line (if the slope is 0 or 1)



$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{rise}{run}$$