disc. 4
cs61b sp22

# objects

classes, interfaces, etc.

slides
**bit.ly/abhi-disc**

attendance
**bit.ly/abhi-attendance**

# announcements

# announcements

1. Project 0 due 2/11 (friday)

# announcements

1. Project 0 due 2/11 (friday)
2. HW 2 due 2/8 (tomorrow)

# announcements

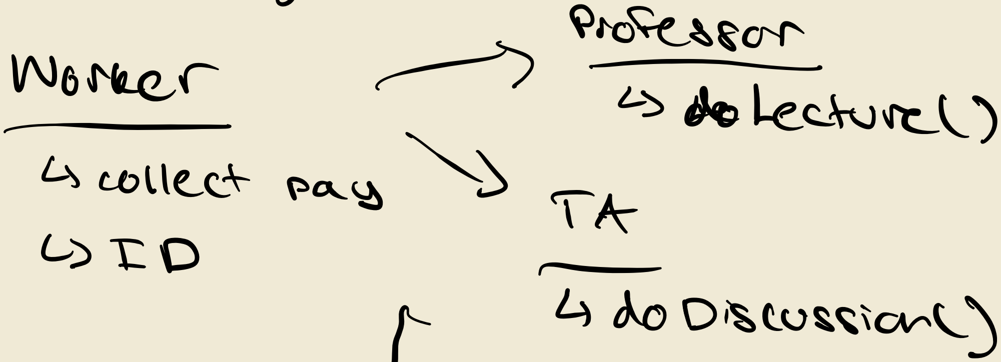
1. Project 0 due 2/11 (friday)
2. HW 2 due 2/8 (tomorrow)
3. Labs this week are Project 0 Office Hours

# announcements

1. Project 0 due 2/11 (friday)
2. HW 2 due 2/8 (tomorrow)
3. Labs this week are Project 0 Office Hours
4. Weekly survey due today and worth points!

# ~~subclasses/child classes~~

Static v. dynamic



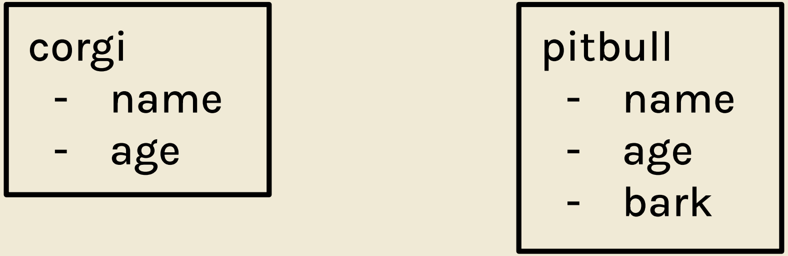
```
Worker abhi { new TA("abhi");
abhi.collectPay() ?
```

# subclasses/child classes

- classes that extend another class

# subclasses/child classes

- classes that extend another class



corgi

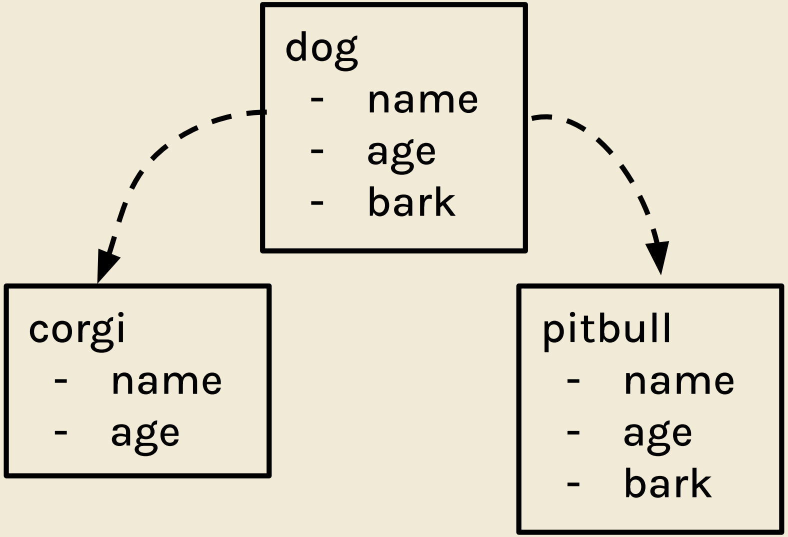
- name
- age

pitbull

- name
- age
- bark

# subclasses/child classes

- classes that extend another class



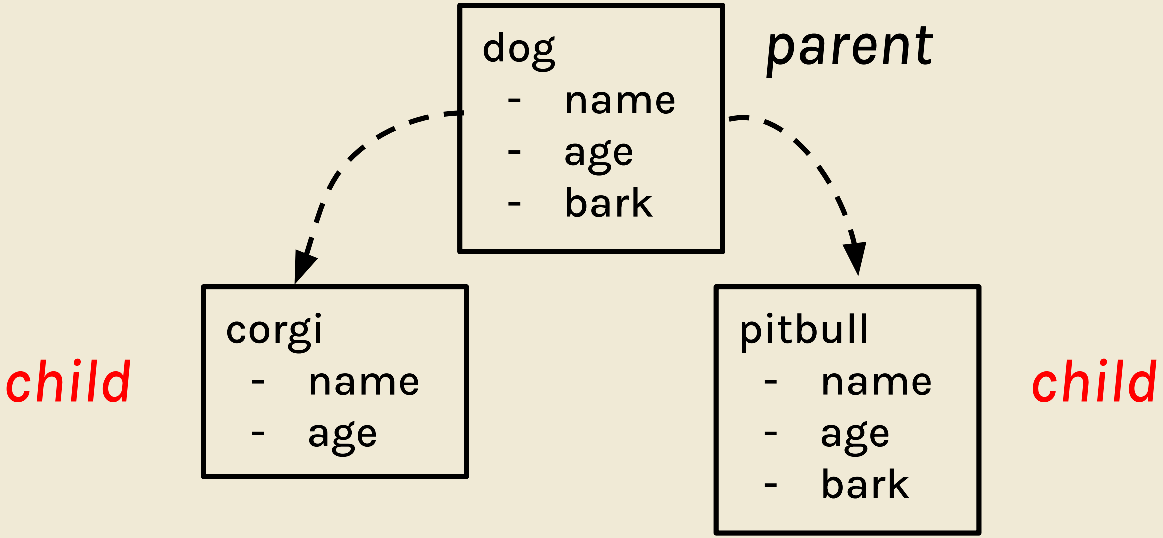
dog
- name
- age
- bark

corgi
- name
- age

pitbull
- name
- age
- bark

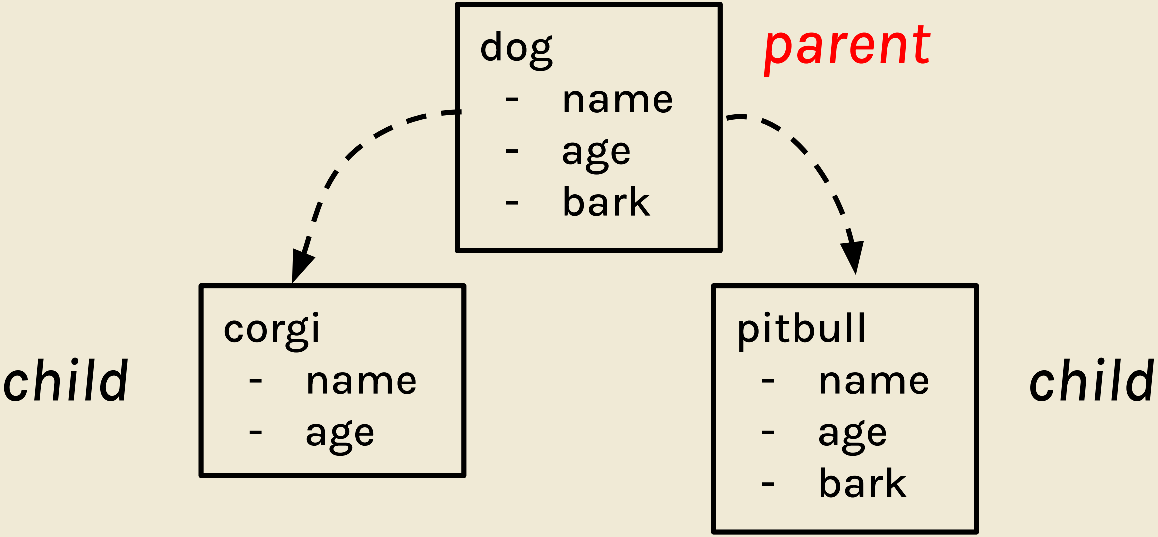
# subclasses/child classes

- classes that extend another class



# superclasses/parent classes

- classes that are extended by other classes



dog
- name
- age
- bark

parent

corgi
- name
- age

child

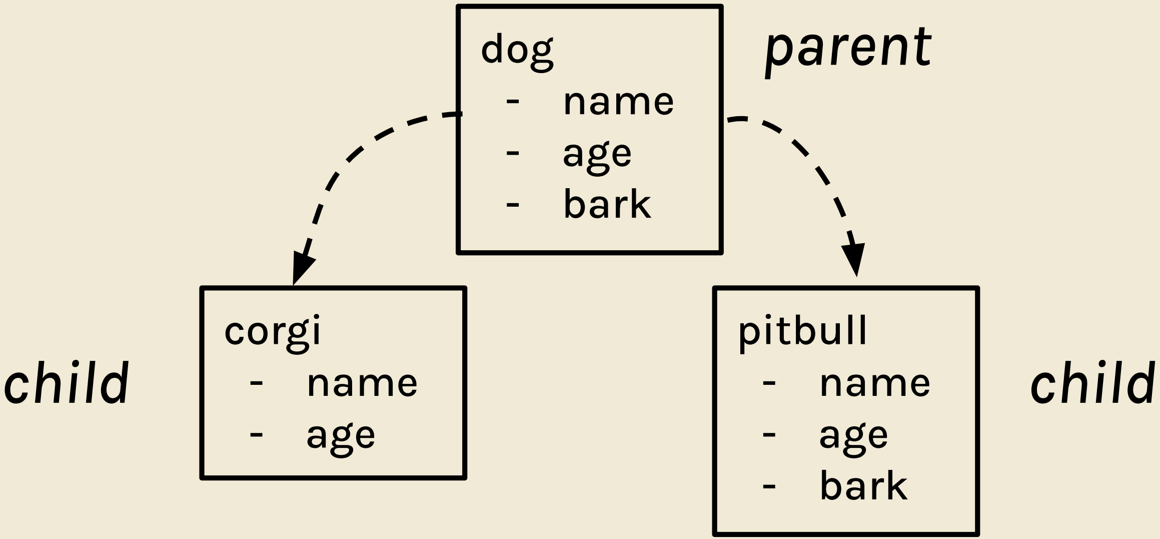
pitbull
- name
- age
- bark

child

# abstract classes

- cannot be directly referenced
    - must be extended by a **concrete class**
    - describe the functions that classes of this “type” should be able to do

# abstract classes



dog
- name
- age
- bark

*parent*

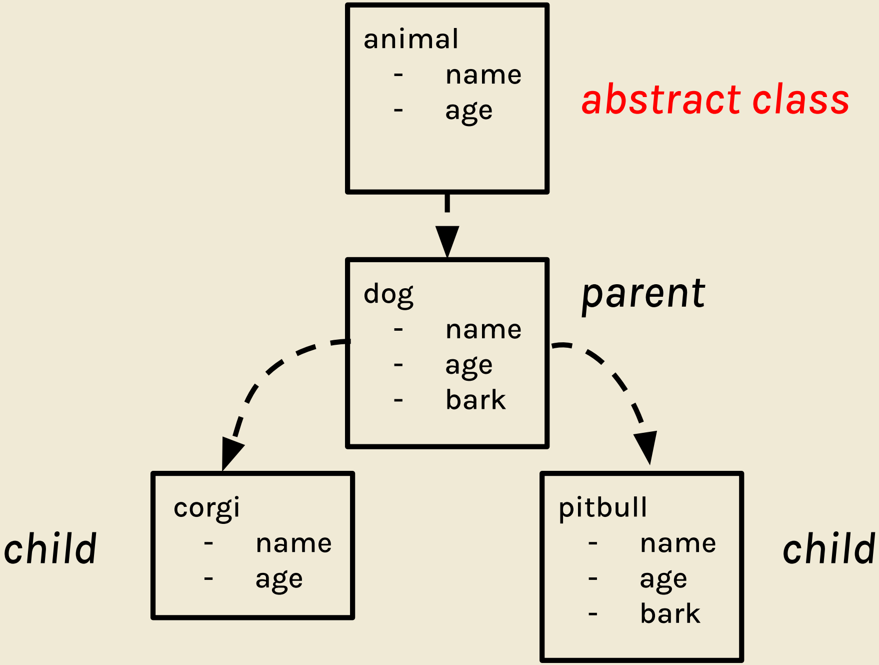
corgi
- name
- age

*child*

pitbull
- name
- age
- bark

*child*

# abstract classes



animal
- name
- age

*abstract class*

dog
- name
- age
- bark

*parent*

corgi
- name
- age

*child*

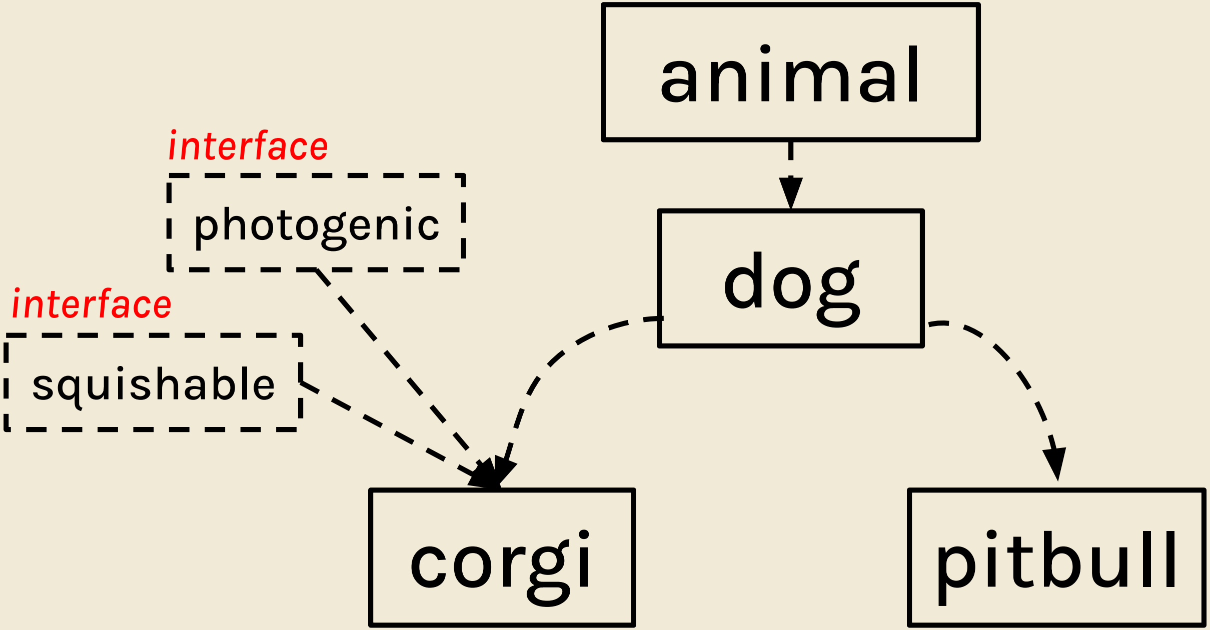
pitbull
- name
- age
- bark

*child*

# interfaces

- **implemented** by classes
- specify methods that describe an ability
  - e.g., Comparable, List
  - these methods aren’t usually “filled out”—they’re just blueprints for the “implementing” class

# interfaces


animal
interface
photogenic
dog
interface
squishable
corgi
pitbull

# using classes and interfaces

```
abstract class Animal {...}

interface Squishable {...}

interface Photogenic {...}

class Dog extends Animal {...}

class Pitbull extends Dog {...}

class Corgi extends Dog implements Squishable, Photogenic {...}
```

List l1 =

# worksheet

(on 61B website)

# 1A Objects Review

```java
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```

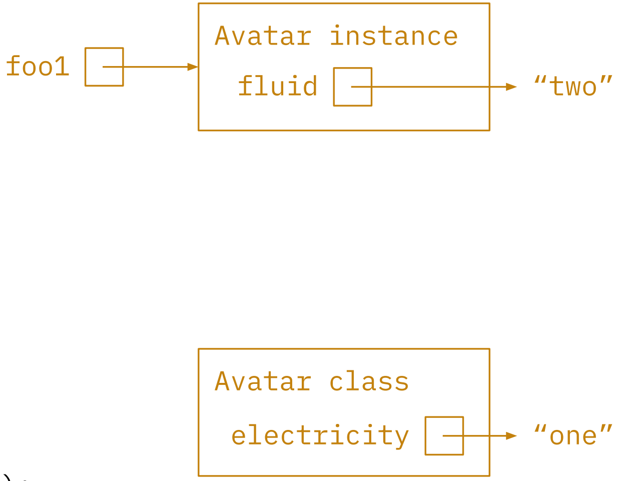
**What would be printed after executing the main method?**

# 1A Objects Review

```
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```



foo1

Avatar instance

fluid

“two”

Avatar class

electricity

“one”

Console:

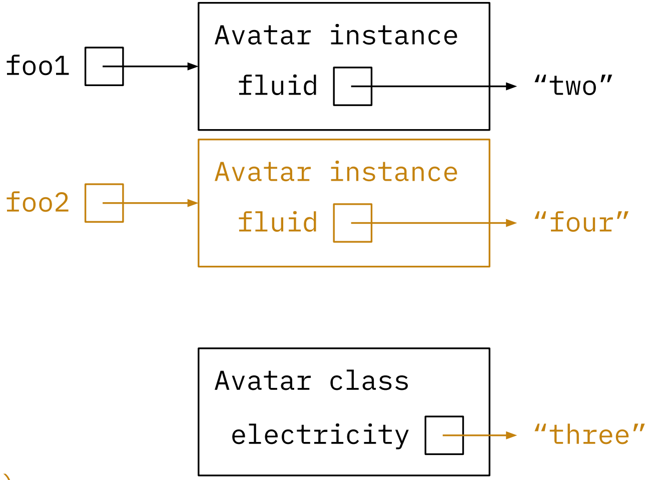
**What would be printed after executing the main method?**

# 1A Objects Review

```java
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```



Console:

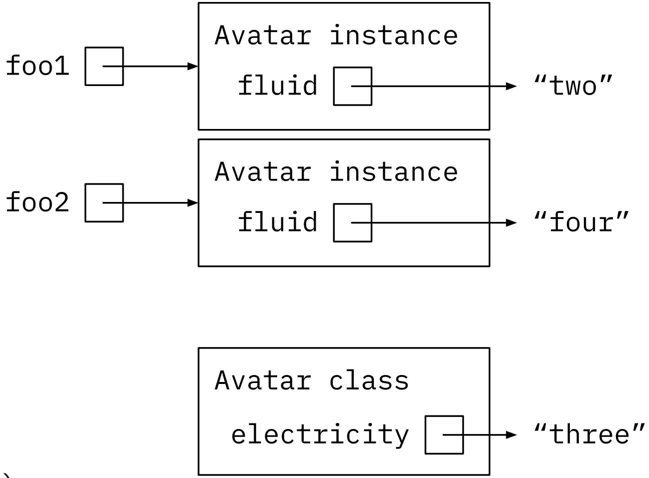
**What would be printed after executing the main method?**

# 1A Objects Review

```java
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```



Console:

```
three two
```

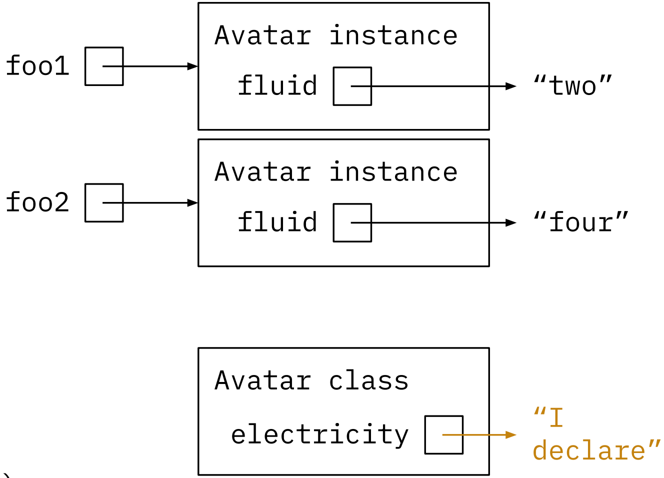
**What would be printed after executing the main method?**

# 1A Objects Review

```java
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```



```
Console:
three two
```

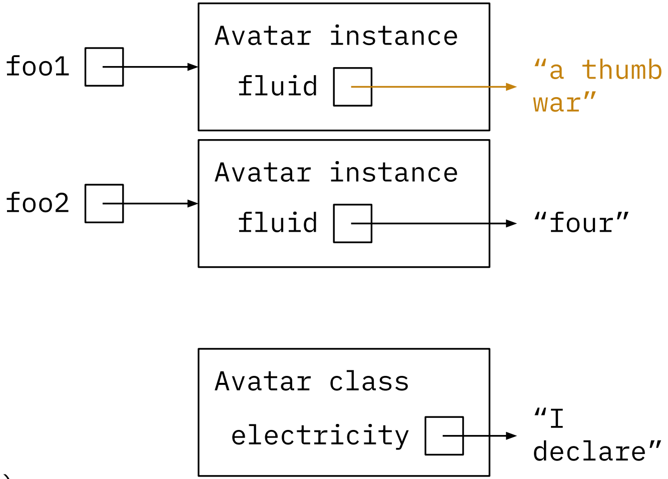
What would be printed after executing the main method?

# 1A Objects Review

```
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```



```
Console:
three two
```

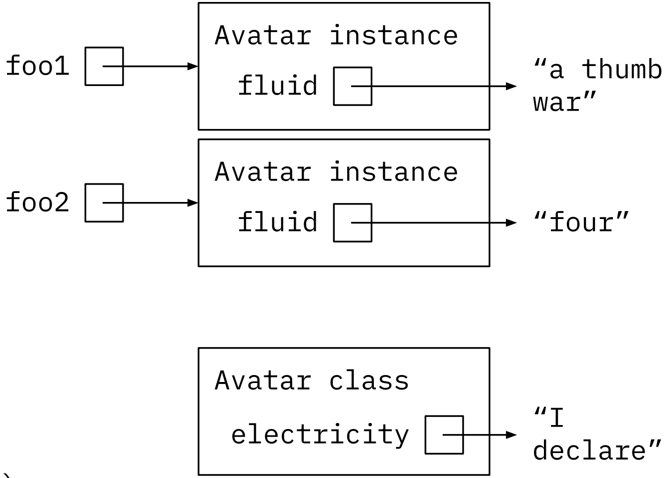
**What would be printed after executing the main method?**

# 1A Objects Review

```
public class Avatar {
    public static String electricity;
    public String fluid;

    public Avatar(String str1, String str2) {
        Avatar.electricity = str1;
        this.fluid = str2;
    }

    public static void main(String[] args) {
        Avatar foo1 = new Avatar(“one”, “two”);
        Avatar foo2 = new Avatar(“three”, “four”);
        System.out.println(foo1.electricity + foo1.fluid);
        foo1.electricity = “I declare ”;
        foo1.fluid = “a thumb war”;
        System.out.println(foo2.electricity + foo2.fluid);
    }
}
```



foo1

Avatar instance

fluid

“a thumb war”

foo2

Avatar instance

fluid

“four”

Avatar class

electricity

“I declare”

```
Console:
three two
I declare four
```

**What would be printed after executing the main method?**

# 1B Objects Review

```
 1  public class Avatar {
 2      public ~~static~~ String electricity;
 3      public static String fluid;
 4
 5      public Avatar(String str1, String str2) {
 6          Avatar.electricity = str1;
 7          this.fluid = str2;
 8      }
 9
10      public static void main(String[] args) {
11          Avatar foo1 = new Avatar(“one”, “two”);
12          Avatar foo2 = new Avatar(“three”, “four”);
13          System.out.println(foo1.electricity + foo1.fluid);
14          foo1.electricity = “I declare ”;
15          foo1.fluid = “a thumb war”;
16          System.out.println(foo2.electricity + foo2.fluid);
17      }
18  }
```

**Would this code compile if we changed lines 2 and 3?**

# 1B Objects Review

```
 1  public class Avatar {
 2      public ~~static~~ String electricity;
 3      public static String fluid;
 4
 5      public Avatar(String str1, String str2) {
 6          Avatar.electricity = str1; // Errors since it is now an instance variable
 7          this.fluid = str2; // This is still fine!
 8      }
 9
10      public static void main(String[] args) {
11          Avatar foo1 = new Avatar("one", "two");
12          Avatar foo2 = new Avatar("three", "four");
13          System.out.println(foo1.electricity + foo1.fluid);
14          foo1.electricity = "I declare ";
15          foo1.fluid = "a thumb war";
16          System.out.println(foo2.electricity + foo2.fluid);
17      }
18  }
```

**Would this code compile if we changed lines 2 and 3?**

# 1C Objects Review

```
public class Avatar {
    public static String electricity;
    public String fluid;

    ...

    public static String getFluid() {
        return fluid;
    }
}
```

**Would this code compile if we added this getFluid() function?**

# 1C Objects Review

```
public class Avatar {
    public static String electricity;
    public String fluid;

    ...

    public static String getFluid() { // Compile-time error
        return fluid; // Can’t access fluid from a static function
    }
}
```

**Would this code compile if we added this getFluid() function?**

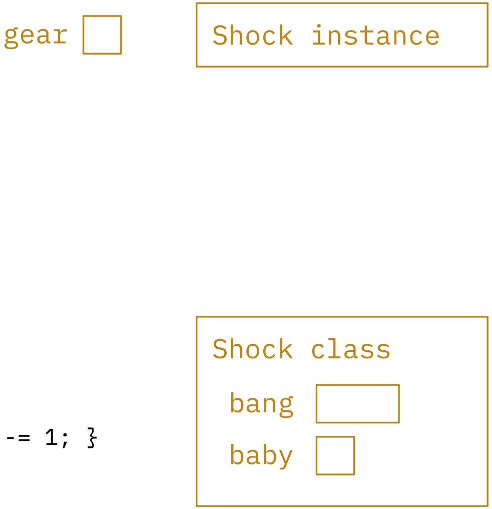
# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

**What will the main method print?**

# 2 Static Shock *Extra*

```
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```



gear

Shock instance

Shock class

bang

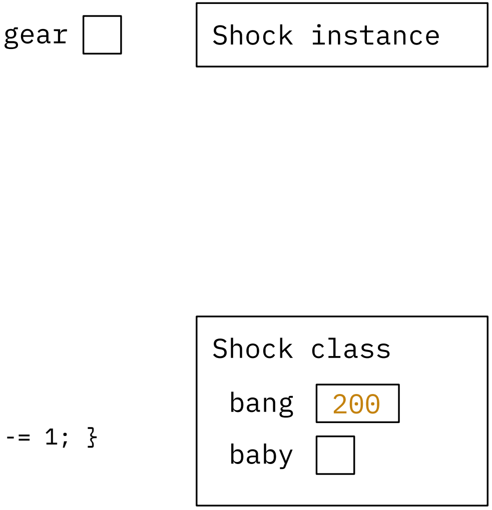
baby

Console:

**What will the main method print?**

# 2 Static Shock *Extra*

```
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```



gear

Shock instance

Shock class

bang 200

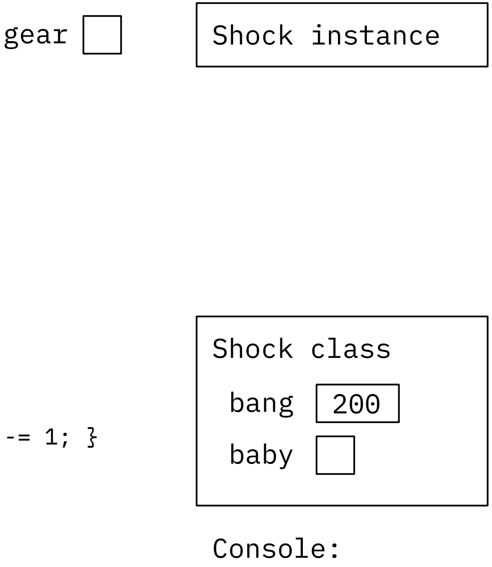
baby

Console:

**What will the main method print?**

# 2 Static Shock *Extra*

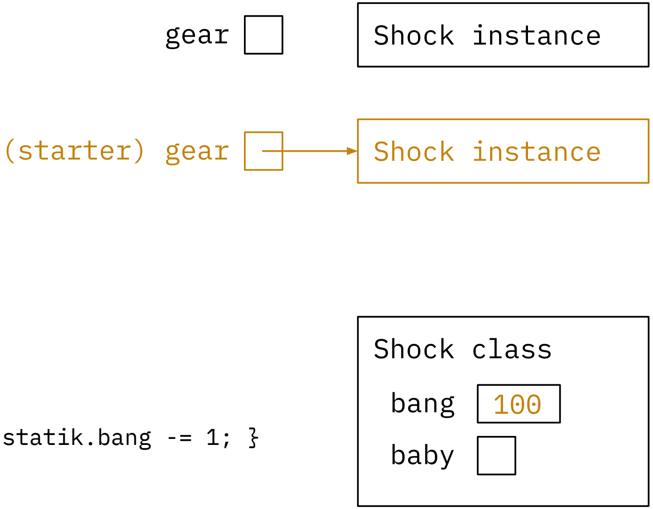
```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```



**What will the main method print?**

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

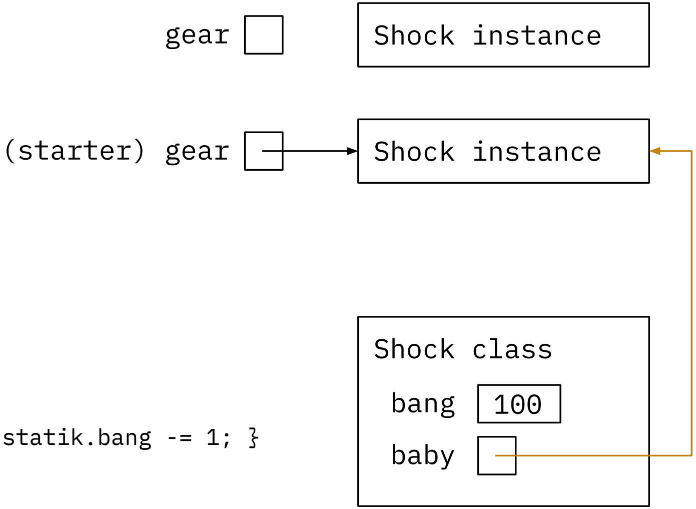


Console:

**What will the main method print?**

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

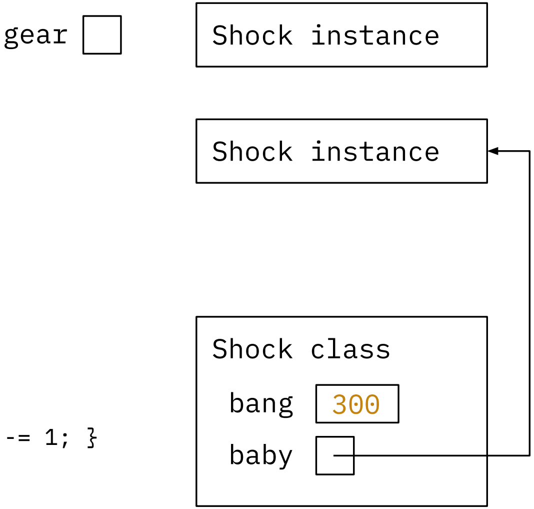


Console:

**What will the main method print?**

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

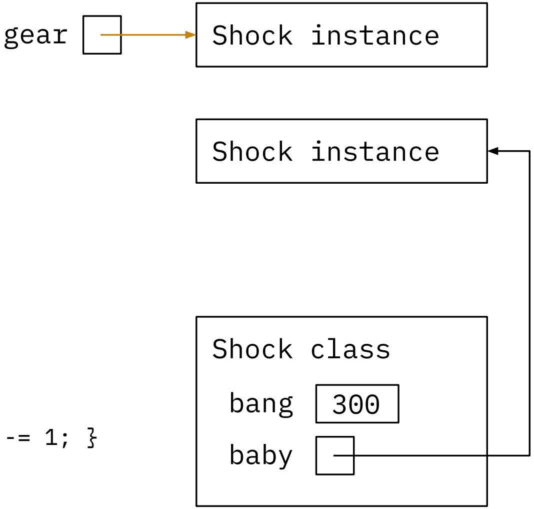


Console:

What will the main method print?

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

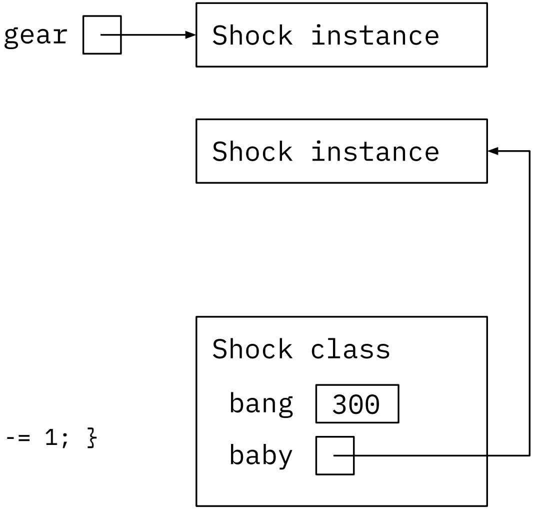


Console:

**What will the main method print?**

# 2 Static Shock *Extra*

```
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```



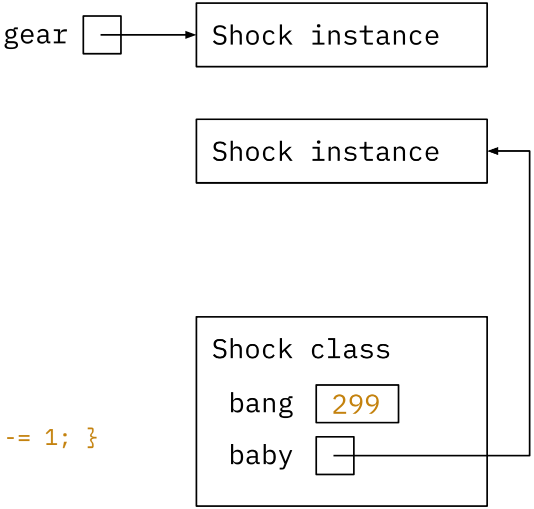
Console:

300

**What will the main method print?**

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

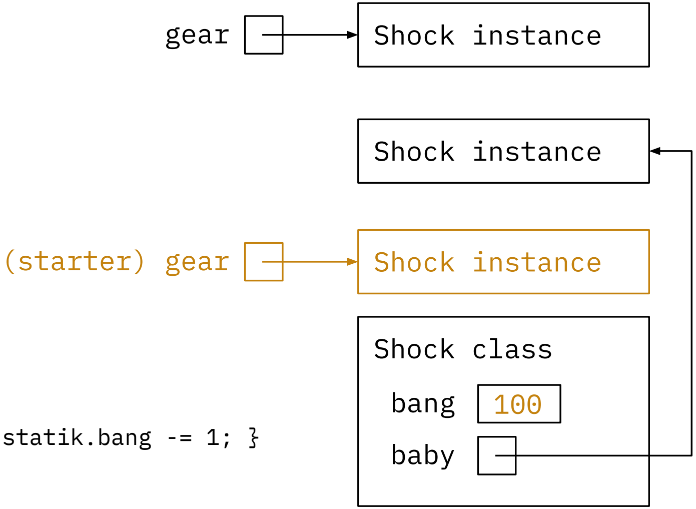


```
Console:
300
```

**What will the main method print?**

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```

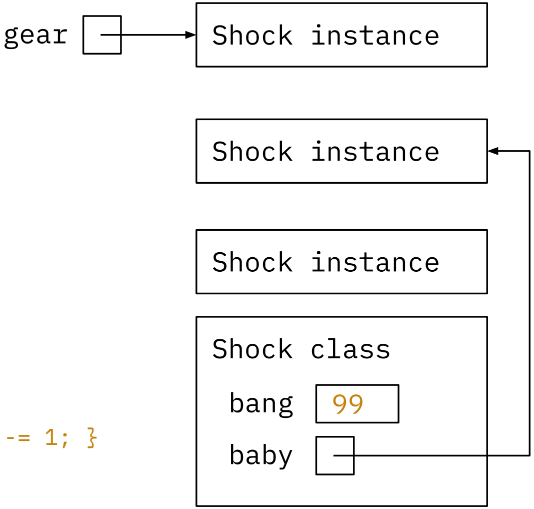


Console:
300

**What will the main method print?**

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```



gear

Shock instance

Shock instance

Shock instance

Shock class

bang 99

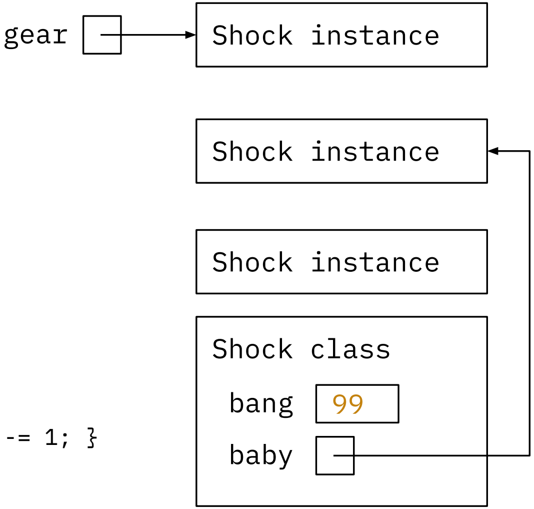
baby

```
Console:
300
```

What will the main method print?

# 2 Static Shock *Extra*

```java
public class Shock {
    public static int bang;
    public static Shock baby;
    public Shock() { this.bang = 100; }
    public Shock(int num) {
        this.bang = num;
        baby = starter();
        this.bang += num;
    }
    public static Shock starter() {
        Shock gear = new Shock();
        return gear;
    }
    public static void shrink(Shock statik) { statik.bang -= 1; }
    public static void main(String[] args) {
        Shock gear = new Shock(200);
        System.out.println(gear.bang);
        shrink(gear);
        shrink(starter());
        System.out.println(gear.bang);
    }
}
```



gear

Shock instance

Shock instance

Shock instance

Shock class

bang 99

baby

```
Console:
300
99
```

**What will the main method print?**

# 3A Reversing an Array

**Implement reverse such that it destructively reverses the elements of A.**

```
public static void reverse (int[] A) {




}
```

# 3A Reversing an Array

Implement reverse such that it destructively reverses the elements of A.

```
public static void reverse (int[] A) {





}
// First element needs to be the last element and vice versa
// Same pattern for the entire array
```

# 3A Reversing an Array

Implement reverse such that it destructively reverses the elements of A.

```
public static void reverse (int[] A) {
    for (int i = 0; i < A.length / 2; i++) {
        // Only need to loop through half of the array
        // Other half gets solved as we swap

    }
}
```

# 3A Reversing an Array

Implement reverse such that it destructively reverses the elements of A.

```
public static void reverse (int[] A) {
    for (int i = 0; i < A.length / 2; i++) {
        int temp = A[A.length - i - 1]; // Now swap!
        A[A.length - i - 1] = A[i];
        A[i] = temp;
    }
}
```

# 3B Reversing an Array *Extra*

**Implement reverseDiagonal such that it destructively reverses the elements of B along  the diagonal.**

```
public static void reverseDiagonal (int[][] B, int diagonal) {





}
```

# 3B Reversing an Array *Extra*

Implement **reverseDiagonal** such that it destructively reverses the elements of **B** along the diagonal.

```
public static void reverseDiagonal (int[][] B, int diagonal) {







}
// Same idea as last problem except across the diagonal
// The nth diagonal has n+1 terms in it (check yourself!)
```

# 3B Reversing an Array *Extra*

**Implement reverseDiagonal such that it destructively reverses the elements of B along the diagonal.**

```
public static void reverseDiagonal (int[][] B, int diagonal) {
    for (int i = 0; i <= diagonal / 2; i++) { // Iterate through the diagonal




    }
}
```

# 3B Reversing an Array *Extra*

Implement reverseDiagonal such that it destructively reverses the elements of B along the diagonal.

```
public static void reverseDiagonal (int[][] B, int diagonal) {
    for (int i = 0; i <= diagonal / 2; i++) {
        int temp = B[diagonal - i][i]; // Diagonal so we need both coordinates



    }
}
```

# 3B Reversing an Array *Extra*

Implement **reverseDiagonal** such that it destructively reverses the elements of **B** along the diagonal.

```
public static void reverseDiagonal (int[][] B, int diagonal) {
    for (int i = 0; i <= diagonal / 2; i++) {
        int temp = B[diagonal - i][i];
        B[diagonal - i][i] = B[i][diagonal - i];
        B[i][diagonal - i] = temp; // Finish swapping
    }
}
```

# 4 Circular Buffer

**Implement `overflow` such that it non-destructively flattens the circular buffer.**

```
public static int[] overflow (int[] A, int i, int k) {




}
```

# 4 Circular Buffer

**Implement overflow such that it non-destructively flattens the circular buffer.**

```
public static int[] overflow (int[] A, int i, int k) {
    int[] B = new int[A.length + 1]; // Create new array that's one bigger




}
```

# 4 Circular Buffer

**Implement overflow such that it non-destructively flattens the circular buffer.**

```
public static int[] overflow (int[] A, int i, int k) {
    int[] B = new int[A.length + 1];
    System.arraycopy(A, i, B, 0, A.length - i); // Copying “beginning” of circular buffer



}
```

# 4 Circular Buffer

**Implement overflow such that it non-destructively flattens the circular buffer.**

```
public static int[] overflow (int[] A, int i, int k) {
    int[] B = new int[A.length + 1];
    System.arraycopy(A, i, B, 0, A.length - i);
    System.arraycopy(A, 0, B, A.length - i, i); // Copying “end” of circular buffer


}
```

# 4 Circular Buffer

**Implement overflow such that it non-destructively flattens the circular buffer.**

```
public static int[] overflow (int[] A, int i, int k) {
    int[] B = new int[A.length + 1];
    System.arraycopy(A, i, B, 0, A.length - i);
    System.arraycopy(A, 0, B, A.length - i, i);
    B[A.length] = k; // Insert new item

}
```

# 4 Circular Buffer

**Implement overflow such that it non-destructively flattens the circular buffer.**

```
public static int[] overflow (int[] A, int i, int k) {
    int[] B = new int[A.length + 1];
    System.arraycopy(A, i, B, 0, A.length - i);
    System.arraycopy(A, 0, B, A.length - i, i);
    B[A.length] = k;
    return B; // return our new array
}
```

# 5 Transposing a 2D Array *Extra*

Implement transpose such that it destructively transposes two dimensional array input.

```
public static void transpose (int[][] A) {







}
```

# 5 Transposing a 2D Array *Extra*

Implement transpose such that it destructively transposes two dimensional array input.

```
public static void transpose (int[][] A) {
    for (int i = 0; i < A.length; i++) { // Iterate through everything length-wise






    }
}
```

# 5 Transposing a 2D Array *Extra*

Implement transpose such that it destructively transposes two dimensional array input.

```
public static void transpose (int[][] A) {
    for (int i = 0; i < A.length; i++) {
        for (int j = i; j < A[i].length; j++) { // Iterate through everything height-wise



        }
    }
}
```

# 5 Transposing a 2D Array *Extra*

**Implement transpose such that it destructively transposes two dimensional array input.**

```
public static void transpose (int[][] A) {
    for (int i = 0; i < A.length; i++) {
        for (int j = i; j < A[i].length; j++) {
            int temp = A[j][i]; // Swap diagonally
            A[j][i] = A[i][j];
            A[i][j] = temp;
        }
    }
}
```

# 5 Transposing a 2D Array *Extra*

Implement **transpose** such that it destructively transposes two dimensional array input.

```
public static void transpose (int[][] A) {
    for (int i = 0; i < A.length; i++) {
        for (int j = i; j < A[i].length; j++) {
            int temp = A[j][i];
            A[j][i] = A[i][j];
            A[i][j] = temp;
        }
    }
}
```

attendance

**bit.ly/abhi-attendance**

feedback

**bit.ly/abhi-feedback**

slides: **bit.ly/abhi-disc**