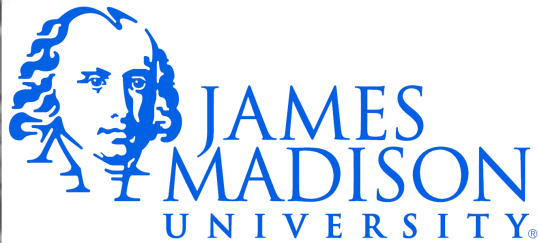




CS 149

Professor: Alvin Chao

CS149 – For Each and Reference Arrays





Looping Over the Contents of an Array

- We often use a for loop to access each element in an array:

```
for (int i = 0; i < names.length; i++) {  
    System.out.println("Hello " + names[i]);  
}
```
- If only there were a better way...



Enhanced For Loop (also called a *for each* loop)

- This loop does the same thing as the loop on the previous slide:

This variable will be assigned the elements from this array

```
for (String name : names) {  
    System.out.println("Hello " + name);  
}
```

- This code is shorter, easier to understand, less error-prone



When To Use an Enhanced For Loop

- Always
- Unless you can't:
 - Need to modify the array
 - Need to know the element index for some reason
 - Need to process the elements out of order
 - ...



Exercise #1

1) What will be printed by the following code?

2) Where is the style problem in this code?

```
String[] summer = {"June", "July", "August"};
```

```
String letters = "";
```

```
for (String i : summer) {  
    letters += i.charAt(0);  
}
```

```
System.out.println(letters);
```

Exercise #1

1) What will be printed by the following code?

1) JJA

2) Where is the style problem in this code?

```
String[] summer = {"June", "July", "August"};

String letters = "";

for (String i : summer) {
    letters += i.charAt(0);
}

System.out.println(letters);
```

This is *not* an index variable, it requires a meaningful name (like “month”).



Exercise #2

- Complete the following method using an enhanced for loop (reminder: use `.equals` to compare strings.)

```
/**  
 * This method counts the number of times a target word occurs in  
 * an array of words. Comparisons are case-sensitive.  
 *  
 * @param words - The array to search  
 * @param target - The word to search for  
 * @return The word count  
 */  
public static int countWord(String[] words, String target) {
```


Exercise #2

- Complete the following method using an enhanced for loop:

```
/**
 * This method counts the number of times a target word occurs in
 * an array of words. Comparisons are case-sensitive.
 *
 * @param words - The array to search
 * @param target - The word to search for
 * @return The word count
 */
public static int countWord(String[] words, String target) {

    int count = 0;

    for (String curWord : words) {
        if (curWord.equals(target)) {
            count++;
        }
    }

    return count;
}
```

Reference Arrays

```
→ Die[] dice;  
  
dice = new Die[4];  
  
dice[0] = new Die(6);  
dice[2] = new Die(5);  
  
for (Die curDie : dice) {  
    if (curDie != null) {  
        curDie.roll();  
    }  
}
```

dice



Reference Arrays

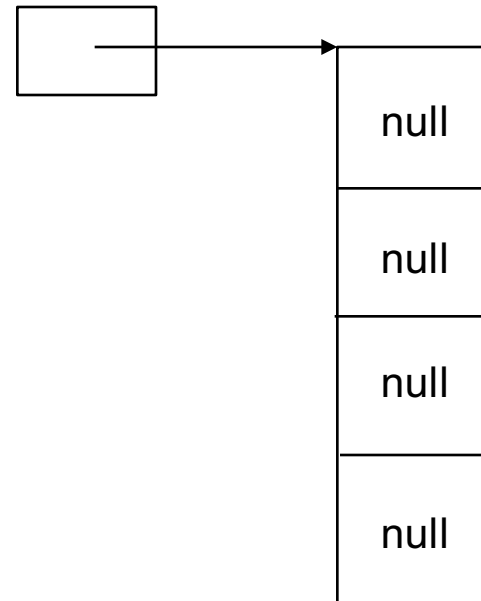
```
Die[] dice;
```

→ `dice = new Die[4];`

```
dice[0] = new Die(6);  
dice[2] = new Die(5);
```

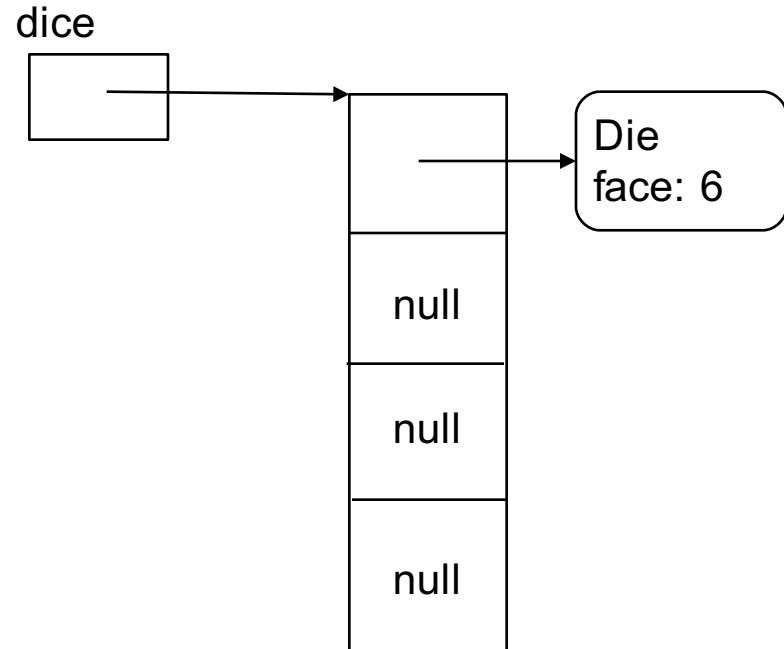
```
for (Die curDie : dice) {  
    if (curDie != null) {  
        curDie.roll();  
    }  
}
```

dice



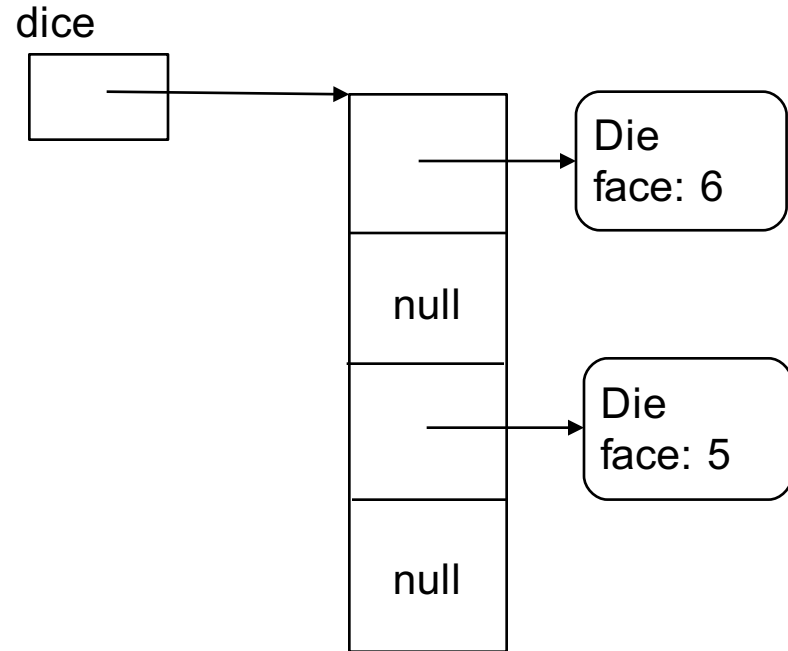
Reference Arrays

```
Die[] dice;  
  
dice = new Die[4];  
  
dice[0] = new Die(6);  
dice[2] = new Die(5);  
  
for (Die curDie : dice) {  
    if (curDie != null) {  
        curDie.roll();  
    }  
}
```



Reference Arrays

```
Die[] dice;  
  
dice = new Die[4];  
  
dice[0] = new Die(6);  
dice[2] = new Die(5);  
  
for (Die curDie : dice) {  
    if (curDie != null) {  
        curDie.roll();  
    }  
}
```



Exercise #3

- Draw the memory diagram and determine output.

```
Die single;  
Die[] dice;
```

```
dice = new Die[4];
```

```
single = new Die(1);
```

```
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}
```

```
dice[0].setFace(3);
```

```
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```

Exercise #3

- Draw the memory diagram and determine output.

```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```

single



Exercise #3

- Draw the memory diagram and determine output.

```
Die single;  
Die[] dice;
```

```
dice = new Die[4];
```

```
single = new Die(1);
```

```
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}
```

```
dice[0].setFace(3);
```

```
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```

single



dice



Exercise #3

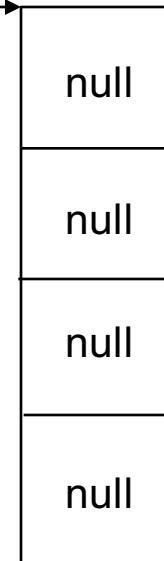
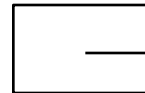
- Draw the memory diagram and determine output.

```
Die single;  
Die[] dice;  
→ dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```

single



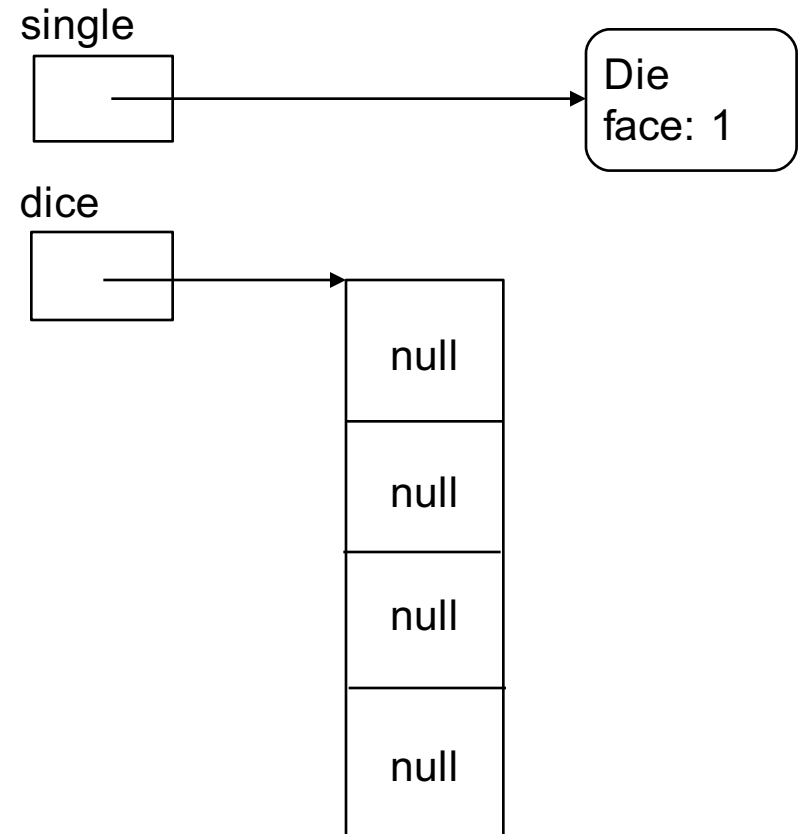
dice



Exercise #3

- Draw the memory diagram and determine output.

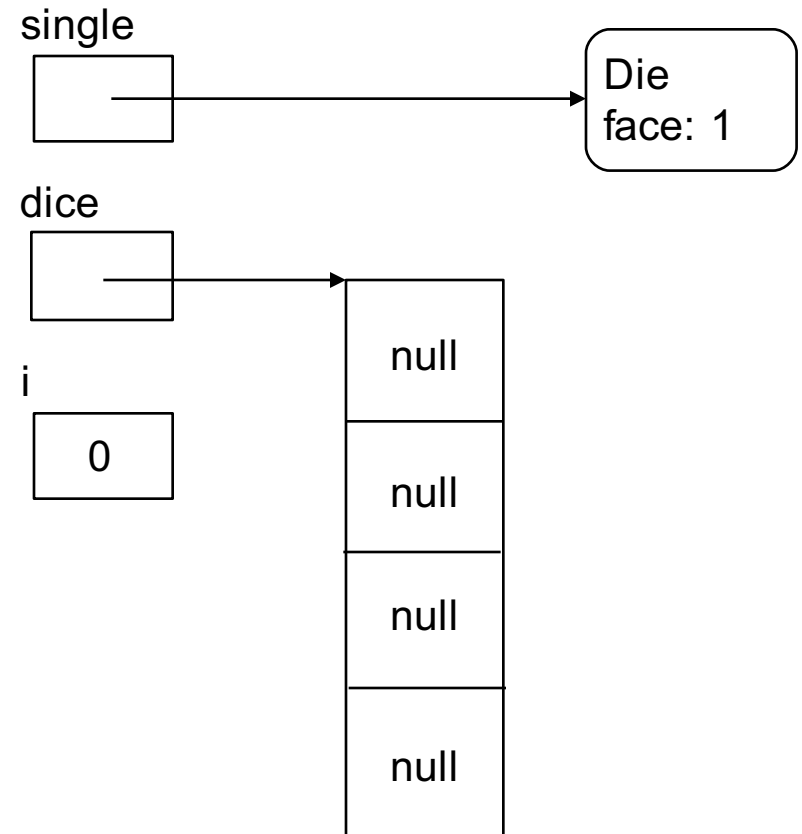
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
→ single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

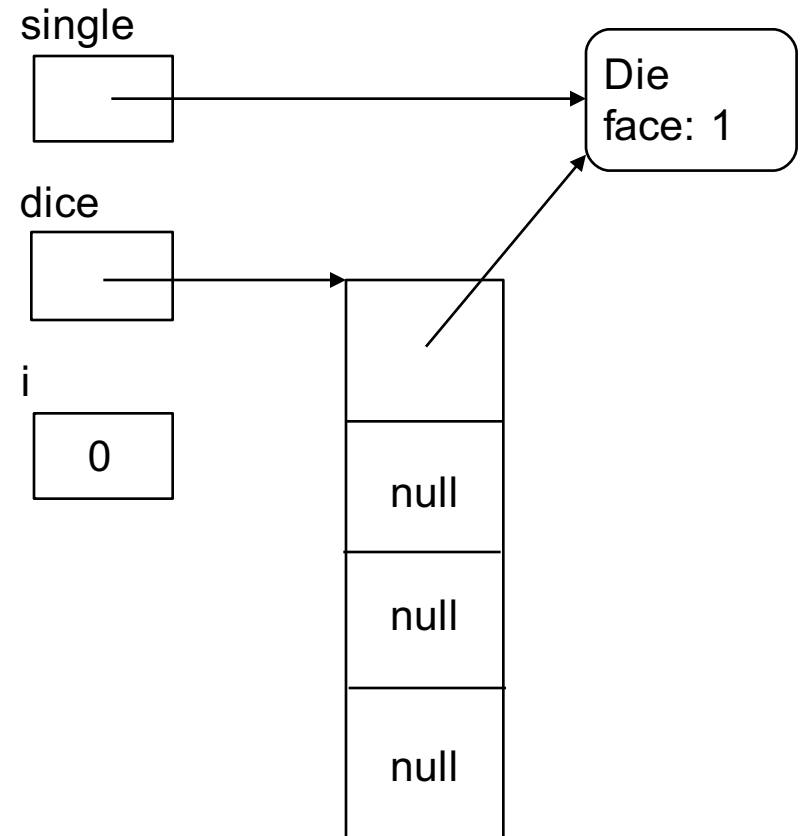
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
→ for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

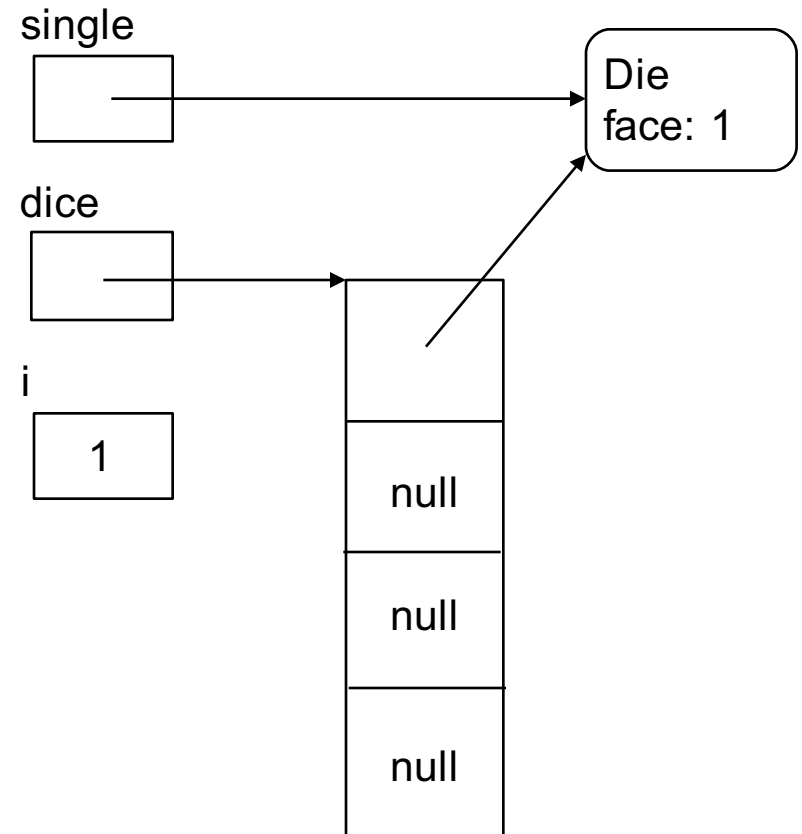
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

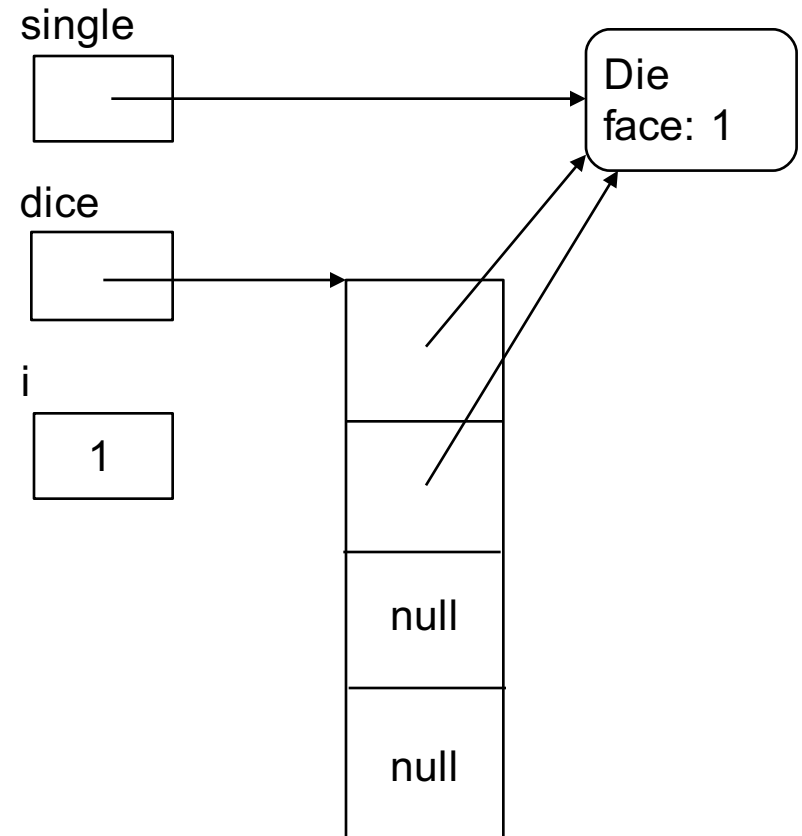
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
→ for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

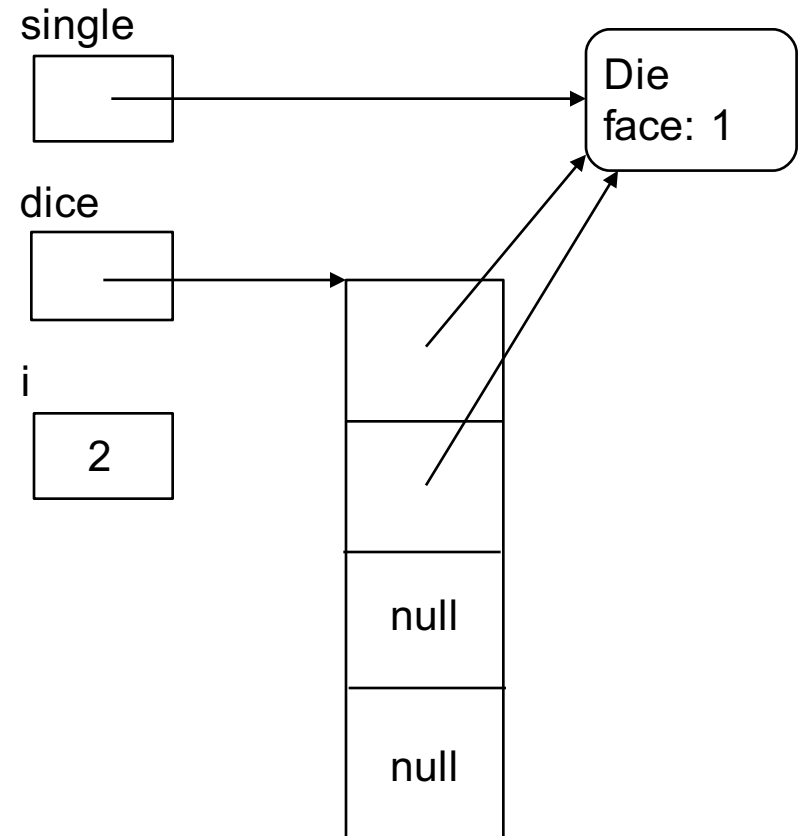
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

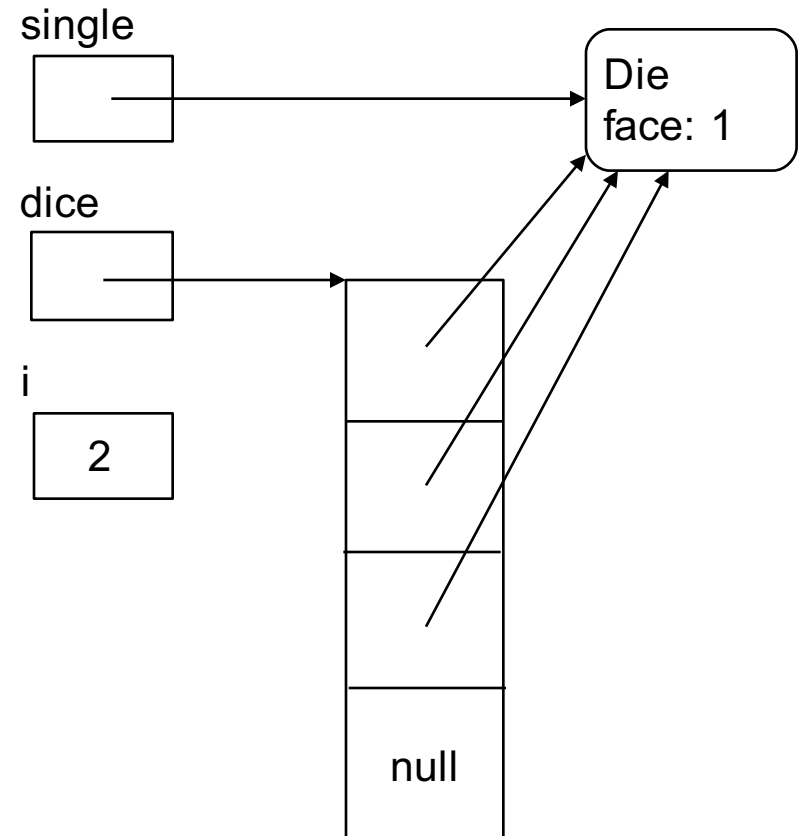
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

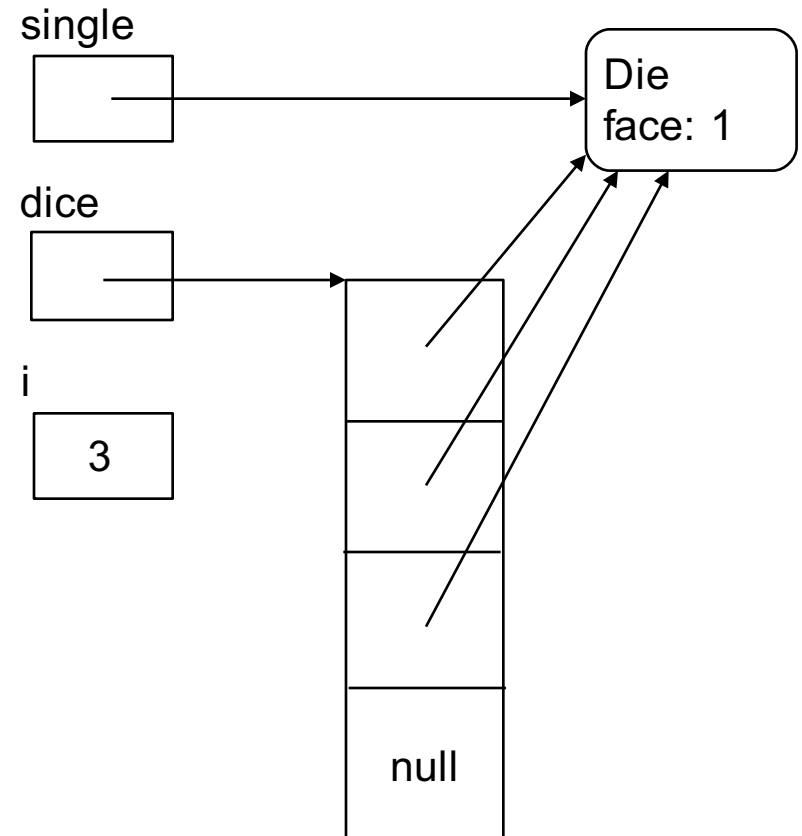
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

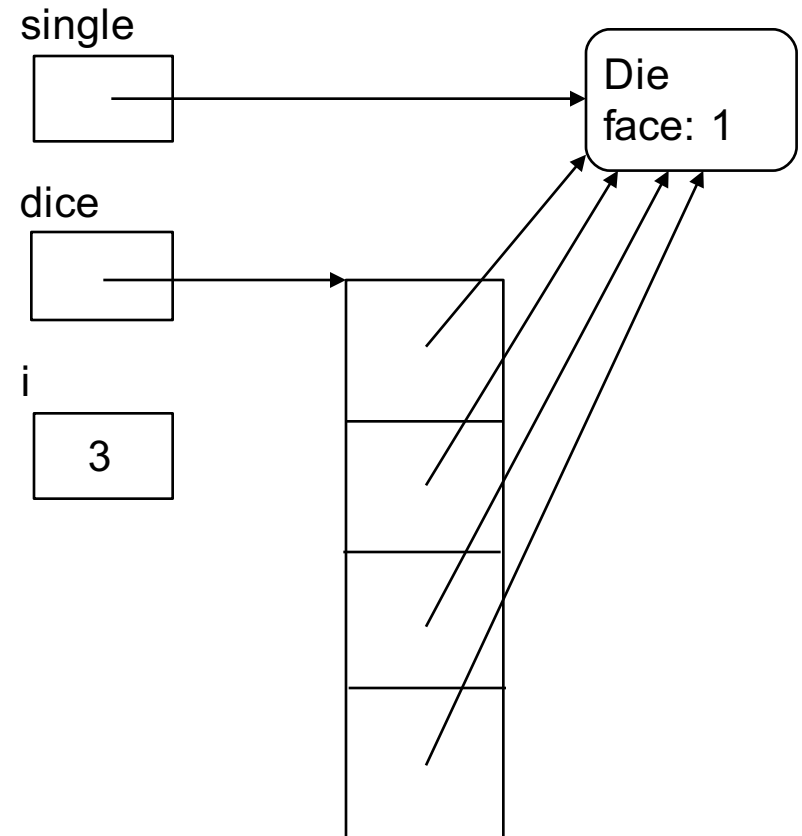
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

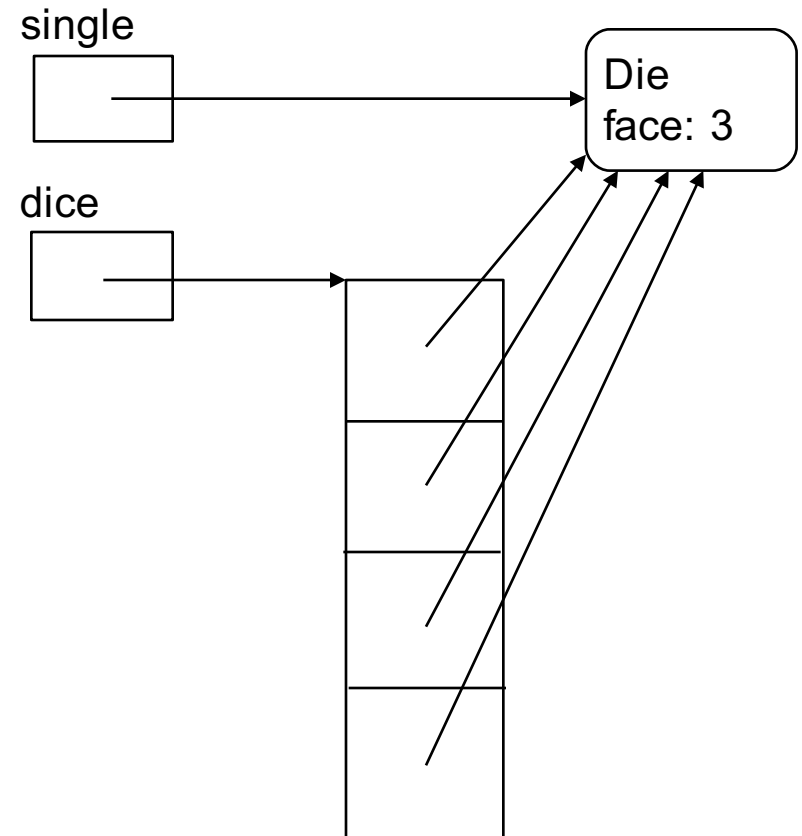
```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```



Exercise #3

- Draw the memory diagram and determine output.

```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
→ dice[0].setFace(3);  
  
for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```

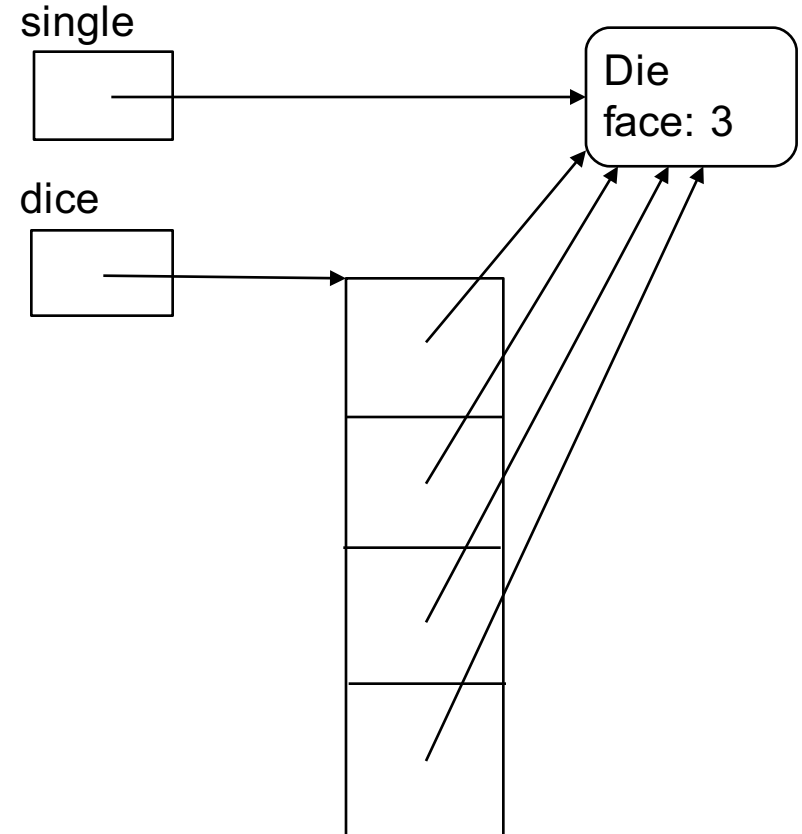


Exercise #3

- Draw the memory diagram and determine output.

```
Die single;  
Die[] dice;  
  
dice = new Die[4];  
  
single = new Die(1);  
  
for (int i = 0; i < dice.length; i++) {  
    dice[i] = single;  
}  
  
dice[0].setFace(3);  
  
→ for (Die curDie : dice) {  
    System.out.println(curDie.getFace());  
}
```

```
Output    3  
          :    3  
          :    3  
          :    3  
          :    3
```





Exercise #4

- Complete the following method

```
/**  
 * This method creates a Die array, and populates it with  
 * Die objects. Each Die object will be initialized with  
 * a random face value (using the zero argument constructor).  
 *  
 * @param numDice - The number of Die objects in the new array  
 * @return The array of Die objects  
 */  
public static Die[] createDice(int numDice) {
```



Exercise #4

- Complete the following method

```
/**
 * This method creates a Die array, and populates it with
 * Die objects. Each Die object will be initialized with
 * a random face value (using the zero argument constructor).
 *
 * @param numDice - The number of Die objects in the new array
 * @return The array of Die objects
 */
public static Die[] createDice(int numDice) {
    Die[] dice = new Die[numDice];

    for (int i = 0; i < dice.length; i++) {
        dice[i] = new Die();
    }

    return dice;
}
```



- **Acknowledgements**

Parts of this activity are based on materials developed by Chris Mayfield and Nathan Sprague.

</end>