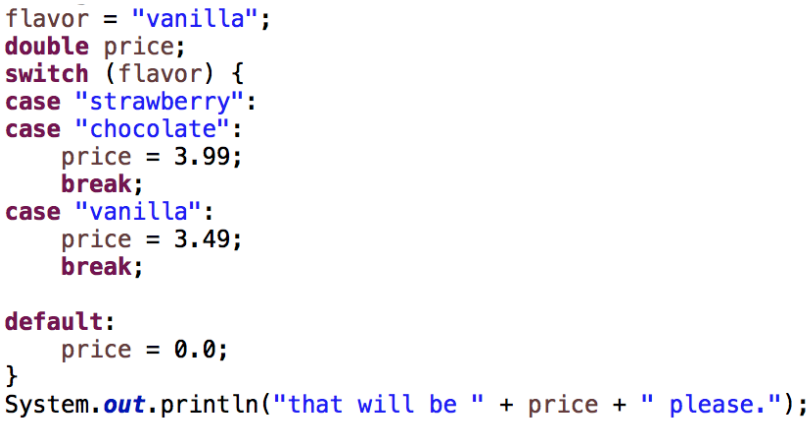
**Types**

1- Evaluate (Java):

| | |
|---|---|
| 5.0/2 = | 2.5 |
| 10 % 3 = | 1 |
| 3 / 3 + 5 * 3 = ? | 16 |
| 2/3 = ? | 0 |
| true \|\| false | true |
| “Comp” + ”Sci” | CompSci |
| “10” + ”-1” | 10-1 |
| “Belcalis Marlenis Almanzar”.charAt(2) | l |
| String elem = “hydrogen”;<br>elem.length(); | 8 |

**Conditions**

2- What does this code snippet print?



```
flavor = "vanilla";
double price;
switch (flavor) {
case "strawberry":
case "chocolate":
    price = 3.99;
    break;
case "vanilla":
    price = 3.49;
    break;

default:
    price = 0.0;
}
System.out.println("that will be " + price + " please.");
```

That will be 3.49 please.

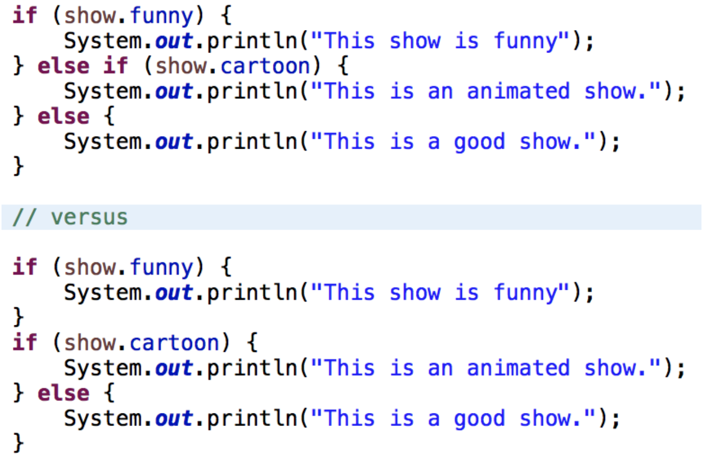
What would the price be if the flavor was strawberry? = 0.0

What is the price for pistachio? = Also 0.0

3- Write a condition that is only true if an integer, x, is divisible by 4 or 7.

return x % 4 == 0 || x % 7 == 0;

4- Are these two code snippets different?  Why or why not?



```java
if (show.funny) {
    System.out.println("This show is funny");
} else if (show.cartoon) {
    System.out.println("This is an animated show.");
} else {
    System.out.println("This is a good show.");
}

// versus

if (show.funny) {
    System.out.println("This show is funny");
}
if (show.cartoon) {
    System.out.println("This is an animated show.");
} else {
    System.out.println("This is a good show.");
}
```

-

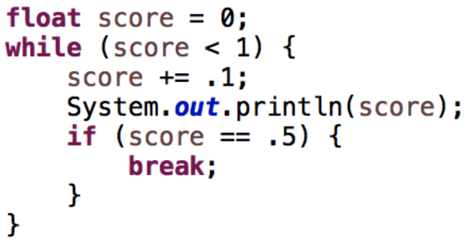
Yes theyre different. The first one always only prints 1 message while the second can print 2. For example if show.funny = true and show.cartoon = false then the first one will only print “This show is funny” while the second one will print that as well as “This is a good show”.

**Loops**

5- Create a for loop that prints the *cubes* *($x^3$)* of all integers from -10 to 10

for (int i = -10; i <= 10; i++){
System.out.println(i*i*i);
}

6- What is the output of this snippet?

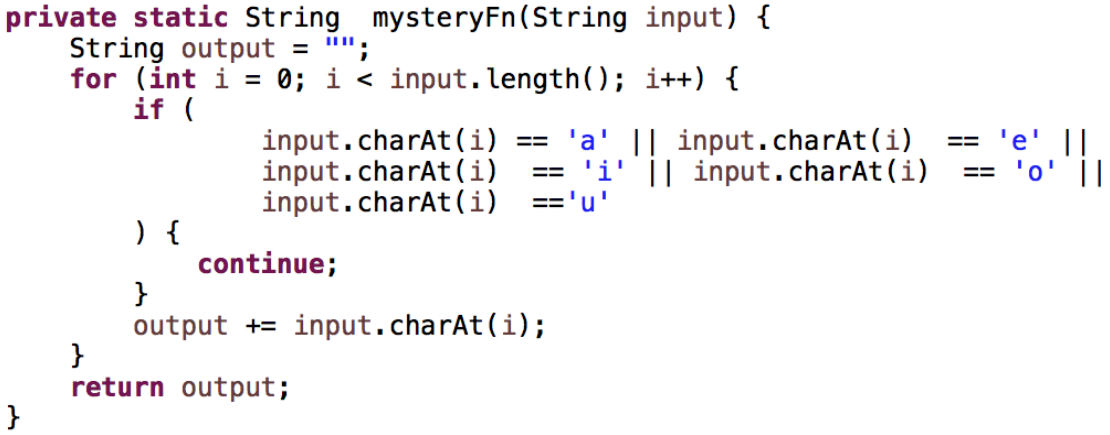


```java
float score = 0;
while (score < 1) {
    score += .1;
    System.out.println(score);
    if (score == .5) {
        break;
    }
}
```

0.1
0.2
0.3
0.4
0.5

**Functions**

7- Consider this function



```
private static String  mysteryFn(String input) {
    String output = "";
    for (int i = 0; i < input.length(); i++) {
        if (
                input.charAt(i) == 'a' || input.charAt(i)  == 'e' ||
                input.charAt(i)  == 'i' || input.charAt(i)  == 'o' ||
                input.charAt(i)  =='u'
        ) {
            continue;
        }
        output += input.charAt(i);
    }
    return output;
}
```

What is the return type? = String

What is the parameter type? = Also String

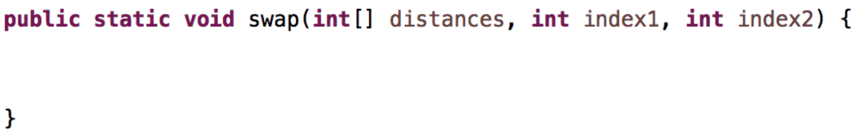
What does it do? = It removes any lowercase vowels in the string.

What is the result of mysteryFn(“Woah, we're half way there”) = "Wh, w'r hlf wy thr"

Is there a case that it does not solve correctly? = Yes, if a vowel is uppercase it doesn’t remove it.

**Arrays**

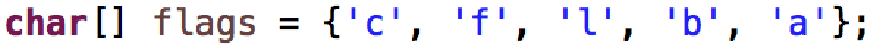
8- Write the implementation of a function that takes 2 arguments (an int array and two integer indices) and swaps the elements in the two indices of the array.



```
public static void swap(int[] distances, int index1, int index2) {


}
```

```
public static void swap(int[] distances, int index1, int index2) {
   int temporary = distances[index1];
   distances[index1] = distances[index2];
   distances[index2] = temporary;
}
```

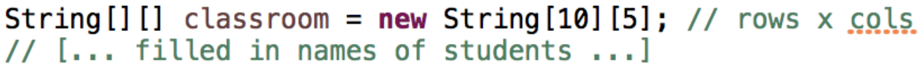
9- Write a code snippet that prints the values in an array **backwards**.



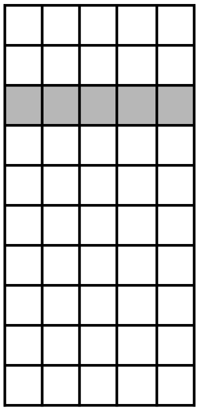
```
char[] flags = {'c', 'f', 'l', 'b', 'a'};
```

```
for (int i = flags.length-1; i >=0; i–){
         System.out.println(flags[i]);
         }
```

10- Write code that prints the values in the 3rd row of this 2D array



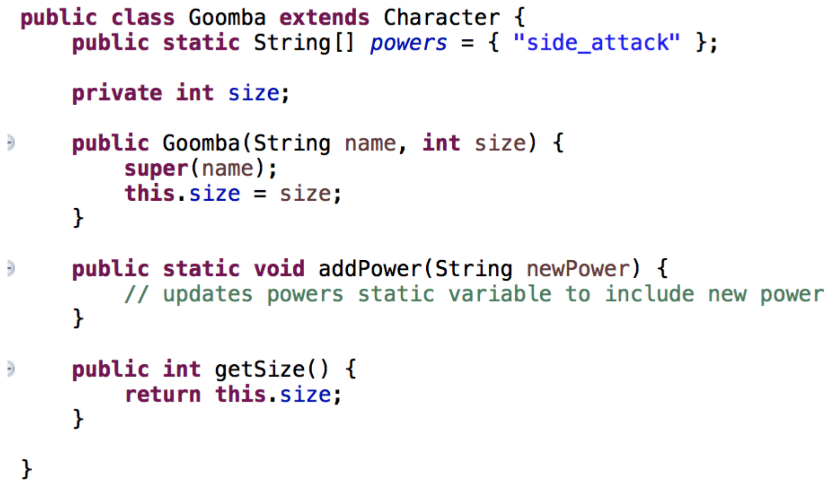
```
String[][] classroom = new String[10][5]; // rows x cols
// [... filled in names of students ...]
```



```
for (int column = 0; column < classroom[2].length; column++) {
   System.out.println(classroom[2][column]);
}
```

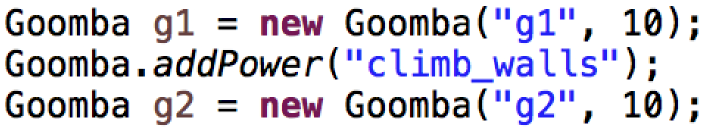
## Objects and Polymorphism

11- Consider this code and the following snippet;
Code:



```
public class Goomba extends Character {
    public static String[] powers = { "side_attack" };

    private int size;

    public Goomba(String name, int size) {
        super(name);
        this.size = size;
    }

    public static void addPower(String newPower) {
        // updates powers static variable to include new power
    }

    public int getSize() {
        return this.size;
    }

}
```

Snippet:



```
Goomba g1 = new Goomba("g1", 10);
Goomba.addPower("climb_walls");
Goomba g2 = new Goomba("g2", 10);
```

What does the **super** call do? = it calls the constructor from the Character class.

What does this.size = size do? = It assigns size to the this.size of the current goomba object.

What powers does g1 have at the end of the snippet? = It has side_attack, and climb_walls.

What powers does g2 have at the end of the snippet? = I think it has the same as g1 so side_attack and climb_walls

System.***out***.println(Goomba.*powers*) prints: [Ljava.lang.String;@6aaa5eb0
What is this value? = I believe that is the reference to the array, so basically the memory address.

12-
What is an interface? = Its a set of abstract methods that a class has to create if they want to implement the interface. Its basically a guide to help you figure out every method you need to make.

What is an abstract class? = a class that can have some methods actually implemented and others not(like in an interface).

Why would you use one or the other? = An interface is when you want to have multiple classes that arent related to have the same methods. An abstract class is when you want subclasses that are related to share variables and logic.

**I/O**

13- Write a snippet that will read from input continuously until the user types ‘q’.
(refer to the attached Scanner API if you need to)

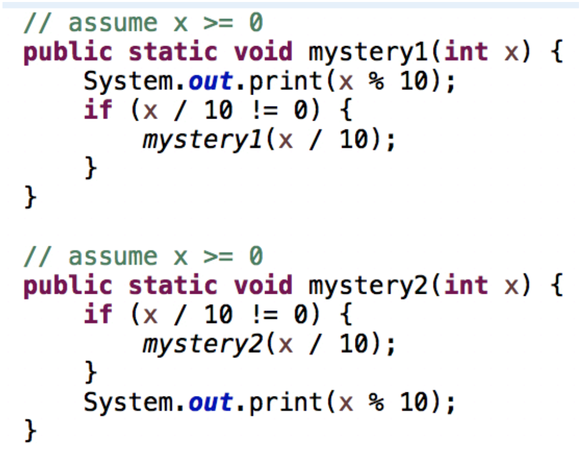
```
Scanner scanner = new Scanner(System.in);
    String input;

    while (true) {
      input = scanner.nextLine();
      if (input.equals("q")) {
        break;
      }
    scanner.close();
  }
```

**Math Foundations**

14- Evaluate or Approximate
If you approximate, indicate if the actual value is greater/less than your approximation.
*Example: 1000/999 = little more than 1*
25/3 = a little more than 8
10^-1 = 0.1
2^5 = 32
3^4 = 81
5^2 = 25
82^(.5) = a little more than 9
6! = 720
log(100) = 2
log(1000) = 3
log(log(10000000000)) = slightly less than 1

**Recursion**

15-



```
// assume x >= 0
public static void mystery1(int x) {
    System.out.print(x % 10);
    if (x / 10 != 0) {
        mystery1(x / 10);
    }
}

// assume x >= 0
public static void mystery2(int x) {
    if (x / 10 != 0) {
        mystery2(x / 10);
    }
    System.out.print(x % 10);
}
```

What is the value of:

*mystery1*(5678)? = 8765

*mystery2*(5678)? = 5678