CS 61B Summer 2020 | Small Group Tutoring
Section 5 Exceptions, Iterators, and Iterables | Worksheet 5

1 Pusheen Exceptions

Below is a class that represents a Pusheen. Pusheen cares about two things: happiness and food. Her happiness is directly proportional to how much she is fed.

```
public class Pusheen {
    public int happiness;

    public Pusheen() {
        happiness = 0;
    }

    public void feed(int amount) {
        happiness = 14 * amount;
    }
}
```

Unfortunately, some Pusheen haters have decided to try and feed Pusheen a negative amount! Obviously, we must prevent this from happening.

Modify the `feed` method to throw an `InvalidPusheenException` if Pusheen is fed with a negative amount. Being fed a negative amount should **NOT** change Pusheen's happiness.

```
public void feed(int amount) throws InvalidPusheenException {









}
```

2 Exceptions

What does Java display when the main method of Test is run?

```
class Test
{
    String str = "a";

    public void A()
    {
        try
        {
            str +="b";
            B();
        }
        catch (Exception e)
        {
            str += "c";
        }
    }
    public void B() throws Exception
    {
        try
        {
            str += "d";
            C();
        }
        catch(Exception e)
        {
            throw new Exception();
        }
        finally
        {
            str += "e";
        }

        str += "f";

    }
    public void C() throws Exception
    {
        throw new Exception();
    }
    public void display()
    {
        System.out.println(str);
    }
    public static void main(String[] args)
    {
        Test object = new Test();
        object.A();
        object.display();
    }
}
```

3 Pizza Iterator

Artichoke's is overwhelmed by the number of hungry students in line at 12 AM. To make things more efficient, the owner has asked you to build a custom iterator that will aggregate all orders and print out the number of slices that should made for each kind of pizza.

The static menu array declared inside `PizzaIterator` contains the three types of pizza offered that night.

```
static String[] menu = {"Artichoke", "Margherita", "Meatball"};
```

The input array passed into the constructor contains the list of orders.

```
int [] orders = { 0 , 2 , 1 , 0 , 1 , 0 };
```

Each order is represented by an integer that corresponds to the pizza's index in the menu array. For example, 0 represents an order of Artichoke pizza.

Fill in the code for `MenuIterator`, an iterator that takes in an `int[]` array representing orders at the restaurant and iterates over the aggregated results.

Given the input above, calls to `next()` would eventually return "Artichoke 3","Margherita 2", "Meatball 1". Make sure your iterator adheres to standard iterator rules.

```
public class MenuIterator implements Iterator {
    private static String[] menu = {"Artichoke", "Margherita", "Meatball"};
    private int[] order_counts = new int[3];
    private int index;

    public MenuIterator(Integer[] orders){




    }

    public boolean hasNext() {


    }

    public String next() {
        //Should return a string in the format "Artichoke 3".




    }
}
```