CS 635 Advanced Object-Oriented Design & Programming
Fall Semester, 2019
Doc 6 Assignment 1 Comments
Sep 17, 2019

```
public class CircularQueue {

    Node firstElement; //First element of the Circular Queue
```

Duh

```
public class CircularQueue {

    // Fields
    Node firstElement;

    // Constructor
    public CircularQueue() {
        blah
    }
```

```
public class LinkedQueue<T> {

    //inserts into the queue
    public void insert(T toInsert) {
        blah
```

Duh

```
public class CircularQueue {

    public void add(String name, String owner, int pid, ....) {
        Node newNode = new Node(name, owner, pid, etc);
```

No code reuse

Only works for this assignment

```
public Process[] getProcessInfoSortBy(String sortBy) {
    blah
    blah

    switch (sortBy)  {
        case "Process Id":
            SortData.sort(processArray, new sortByProcessId());
            break;
        case "Process Name":
            blah
    }
}
```

Have to know exact strings used inside of the method

Information Hiding?

Information Hiding

Hiding of design decisions in a computer program

Hide decisions are most likely to change,
To protect other parts of the program

```
public static final String PROCESS_ID = "Process ID";
public static final String PROCESS_NAME = "Process Name";

public Process[] getProcessInfoSortBy(String sortBy) {
    blah
    blah

    switch (sortBy)  {
        case PROCESS_ID:
            SortData.sort(processArray, new sortByProcessId());
            break;
        case PROCESS_NAME:
            blah
    }
}
```

Now caller does not need to know the strings

But if we want to sort another way - Need to edit method

Easy to modify?

~~public static final String PROCESS_ID = "Process ID";~~
~~public static final String PROCESS_NAME = "Process Name";~~

```
public Process[] getProcessInfoSortBy(Comparator sortBy) {
    blah
    blah

    switch (sortBy) {
        case PROCESS_ID:
            SortData.sort(processArray, new sortByProcessId());
            break;
        case PROCESS_NAME:
            blah
    }
    SortData.sort(processArray, sortBy);
}
```

Now caller has freedom to determine sort order

```
public Process[] toArray(Comparator sortBy) {
    blah
    blah


    SortData.sort(processArray, sortBy);
}
```

Using Java name

```
public Object[] toArray() {
    blah
    return array without sorting
}
```

Is it Queue class responsibility to sort elements?

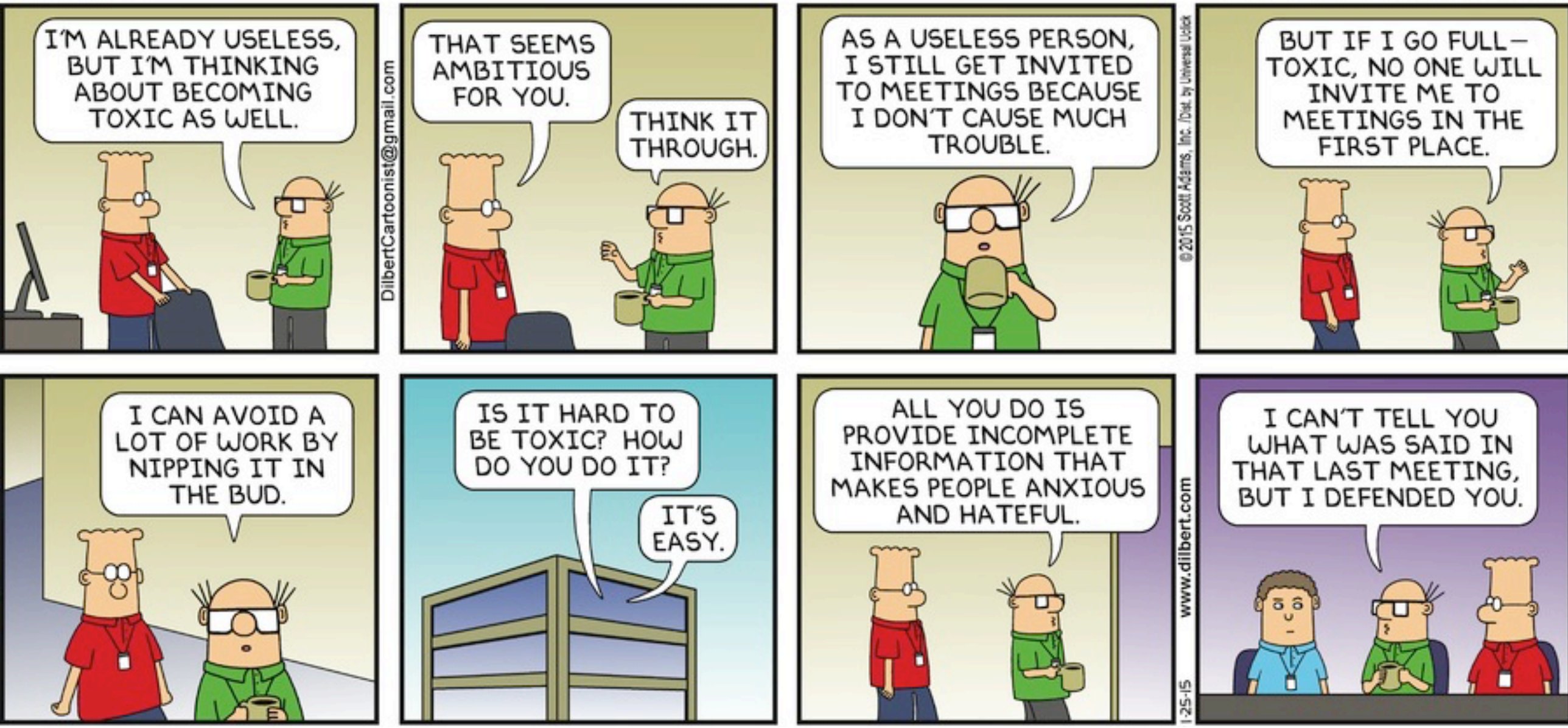
Java's collections
Return array of elements
Caller now can sort

Programs that use your queue will want to display and print out the elements in the queue. They will need to display/print out the queue ordered by name, PID, CPU time used, percent of CPU time, total CPU time or by owner.

Heuristics

Keep related data and behavior in one place

A class should capture one and only one key abstraction


I'M ALREADY USELESS, BUT I'M THINKING ABOUT BECOMING TOXIC AS WELL.
DilbertCartoonist@gmail.com
THAT SEEMS AMBITIOUS FOR YOU.
THINK IT THROUGH.
AS A USELESS PERSON, I STILL GET INVITED TO MEETINGS BECAUSE I DON'T CAUSE MUCH TROUBLE.
BUT IF I GO FULL-TOXIC, NO ONE WILL INVITE ME TO MEETINGS IN THE FIRST PLACE.
© 2015 Scott Adams, Inc. /Dist. by Universal Uclick
I CAN AVOID A LOT OF WORK BY NIPPING IT IN THE BUD.
IS IT HARD TO BE TOXIC? HOW DO YOU DO IT?
IT'S EASY.
ALL YOU DO IS PROVIDE INCOMPLETE INFORMATION THAT MAKES PEOPLE ANXIOUS AND HATEFUL.
I CAN'T TELL YOU WHAT WAS SAID IN THAT LAST MEETING, BUT I DEFENDED YOU.
www.dilbert.com
1-25-15

1. Never do any work that you can get someone else to do for you

Excuse me Smithers. I need to know the total bills that have been paid so far this quarter. No, don't trouble yourself. If you'll just lend me the key to your filing cabinet I'll go through the records myself. I'm not that familiar with your filing system, but how complicated can it be? I'll try not to make too much of a mess.

Verses

SMITHERS! I need the total bills that have been paid since the beginning of the quarter. No, I'm not interested in the petty details of your filing system. I want that total, and I'll expect it on my desk within the next half millisecond.

Encapsulation & Responsibility

Encapsulation is about responsibility

Who does the work

Who should do the work

2. Avoid Responsibility

If you must accept a responsibility, keep it as vague as possible.

For any responsibility you accept, try to pass the real work off to somebody else.

```
class TernarySearchTree {
```

Have the Node do the work

```
public void insert(String word) {
   root.insert(new StringIterator(word));
}
```

Information Hiding

```
public class CircularQueue {

    public void enqueue(Process value) { blah }
    public Node dequeue() {  blah }
```

```
public class CircularQueue {

    public Node front;
    public Node end;
```

```
public T pop() {
    //Checks if queue is empty, returns null if so
    if (currentSize == 0)  {
        System.out.println("Queue is empty");
        return null;
    }
    blah
}
```

Who are you talking to?

```
public T pop() {
    if (isEmpty())  {
        return null;
    }
    blah
}
```

Java Queue Interface

public interface Queue<E> extends Collection<E>

	Throws exception	*Returns special value*
Insert	add(e)	offer(e)
Remove	remove()	poll()
Examine	element()	peek()

```
public class CircularQueue<E> {
    public boolean isQueueEmpty() { return blah;}

    public int getQueueSize() { return queueSize; }
```

```
public class CircularQueue<E> implements Queue<E> {
    public boolean isEmpty() { return blah;}

    public int size() { return queueSize; }
```

Beware of classes that have many accessor methods defined in their public interface

```
public class ProcessNode {
    Process value;
    ProcessNode next;
    ProcessNode previous;

    public void setValue(Process value) {
        this.value = value;
    }

    public Process getValue() { return value; }

    public void setNext(ProcessNode next) { blah }
    public ProcessNode getNext() { blah }

    public void setPrevious(ProcessNode next) { blah }
    public ProcessNode getPrevious() { blah }

    public toString() { blah }
```

struct pretending to be a class

```
public class Queue {

    private ProcessNode addNodeToQueue(ProcessNode currentNode) {
        ProcessNode newNode = new ProcessNode();
        newNode.setNext(queueStart);
        newNode.setPrevious(currentNode);
        queueStart.setPrevious(newNode);
        currentNode.setNext(newNode);
        return newNode;
    }
```

Bug alert

- Assumes that currentNode is the last node in the queue
- If not nodes from currentNode to the end are dropped

```
public class ProcessNode {
    Process value;
    ProcessNode next;
    ProcessNode previous;

    ProcessNode(Process value) {
        setValue(value);
    }
    public void setValue(Process value) { this.value = value;}
    public Process getValue() { return value; }

    public void add(Process value) { add(new ProcessNode(value));}

    public void add(ProcessNode newNode) {
        newNode.next = next;
        next.previous = newNode;
        next = newNode;
        newNode.previous = this;
    }

    public ProcessNode getNext() { blah }
    public ProcessNode getPrevious() { blah }
    public toString() { blah }
```

Prefer append over add

Java collection classes use add

```
public class Node<E> {
    E value;
    ProcessNode next;
    ProcessNode previous;

    ProcessNode(E value) {
        setValue(value);
    }
    public void setValue(E value) { this.value = value;}
    public E getValue() { return value; }

    public void add(E value) { add(new Node(value));}

    public void add(Node newNode) {
        newNode.next = next;
        next.previous = newNode;
        next = newNode;
        newNode.previous = this;
    }

    public E getNext() { blah }
    public E getPrevious() { blah }
    public toString() { blah }
```

Much more useful

```
public class Queue {

    private ProcessNode addNodeToQueue(ProcessNode currentNode) {
        ProcessNode newNode = new ProcessNode();
        newNode.setNext(queueStart);
        newNode,setPrevious(currentNode);
        queueStart.setPrevious(newNode);
        currentNode.setNext(newNode);
        return newNode;
    }
```

queueStart.previous.add(new Node())

Printing

```
public class Queue {

   public void enqueue(Process newProcess) {
       System.out.println("Adding Element to the Queue"):
       checkAndCreateNodes();
       blah
   }

    public Process remove() {
if(isEmpty())  {
   System.out.println("Queue is empty");
   return null;
} else {
   // if only one element
   if(queueElements.getFront() == queueElements.getRear()) {
        processReturned = queueElements.getFront();
        queueElements.setFront(null);
        queueElemetns.setRear(null):
    } else {
        blah
    }
```

How are you talking to?

Program using Queue can not see this

```
public class Queue {

    public Process remove() {
        if (isEmpty())  {
            return null;
        }

        if (size == 1) {
    processReturned = queueElements.getFront();
    queueElements.setFront(null);
    queueElements.setRear(null):
    size = 0;
    return processReturned;
}

do normal case here
```

```
public void enqueue(Process newProcess) {
    System.out.println("Adding Element to the Queue"):
    checkAndCreateNodes();
    blah
}

public void enqueue(Process newProcess) {
    ensureCapacity();
    blah
}
```

```
public void add(Process newProcess) {
    ensureCapacity();
    blah
}

public bool add(Process newProcess) {
    ensureCapacity();
    blah
}
```

```
public class Queue {

    public bool add(Process newProcess) {
        ensureCapacity();
        blah
    }
```

```
  public class Queue<E> {

      public bool add(E value) {
          ensureCapacity();
          blah
      }
```

```
public class Queue<E> implements java.util.Queue {

    public bool add(E value) {
        ensureCapacity();
        blah
    }
```

```
public class Queue<E> implements java.util.AbstractQueue {

    public bool add(E value) {
        ensureCapacity();
        blah
    }
```

```
public class Process {
    private String name;
    private String owner;
    private int threads;
    ...

    private Process previous;
    private Process next;
}
```

Two classes merged into one

Only Process can be used in Queue

Who should be responsible for links?

```
public class Queue {
    private Queue front;
    private Queue end
    ...

    private Queue previous;
    private Queue next;
}
```

Two classes merged into one

```
public class Queue {

    private static Node front;
    private static Node rear;

    blah
}
```

```
Queue a = new Queue();
Queue b = new Queue();

a.add(new Process());
a.size()                    //1

b.add(new Process());
b.size()                    //2

a.size()                    //2
```

```
class LinkedQueue(object):
    def addProcess(self, newProcess):
        blah
```

```
class LinkedQueue(object):
    def addprocess(self, newProcess):
        blah
```

```
class LinkedQueue(object):
    def add_process(self, newProcess):
        blah
```

```
class LinkedQueue(object):
    def append(self, newProcess):
        blah
```