Week 2 - Wednesday

COMP 2100

Last time

- What did we talk about last time?
- Exceptions
- OOP
- Interfaces

Assignment 1

Project 1

Questions?

Generics

Generics

- Allow classes, interfaces, and methods to be written with a generic type parameter, then bound later
- Java does the type checking (e.g. making sure that you only put **`String`** objects into a **`List<String>`**)
- After type checking, it erases the generic type parameter
  - This works because all classes extend **`Object`** in Java
- Appears to function like templates in C++, but works very differently under the covers
- Most of the time you will **use** generics, not create them

Generic classes

- You can make a class using generics
- The most common use for these is container classes
  - For example, you want a **`List`** class that can be a list of anything
- The JCF is filled with such generics

Generic class example

```
public class Pair<T> {
   private T x;
   private T y;
   public Pair(T a, T b ) {
       x = a;
       y = b;
   }

   public T getX() { return x; }
   public T getY() { return y; }

   public void swap() {
       T temp = x;
       x = y;
       y = temp;
   }

   public String toString() {
       return "( " + x + ", " + y + " )";
   }
}
```

Generic class use

```
public class Test {
  public static void main(String[] args) {
     Pair<String> pair1 = new Pair<>("ham", "eggs");
     Pair<Integer> pair2 = new Pair<>(5, 7);
     pair1.swap();
     System.out.println(pair1);
     System.out.println(pair2);
  }
}
```

JCF

Java Collections Framework

Container interfaces

- `Collection`
- `Iterable`
- `List`
- `Queue`
- `Set`
  - `SortedSet`
- `Map`
  - `SortedMap`

Container classes

- `LinkedList`
- `ArrayList`
- `Stack`
- `Vector`
- `HashSet`
- `TreeSet`
- `HashMap`
- `TreeMap`

Tools

- Collections
  - sort()
  - max()
  - min()
  - replaceAll()
  - reverse()
- Arrays
  - binarySearch()
  - sort()

Computational Complexity

Running Time

- How do we measure the amount of time an algorithm takes?
- Surely sorting 10 numbers takes less time than sorting 1,000,000 numbers
- Sorting 1,000,000 numbers that are already sorted seems easier than sorting 1,000,000 unsorted numbers
- We use a **worst-case**, **asymptotic** function of input size called Big Oh notation

Big Oh Notation

- **Worst-case** because we care about how bad things could be
- **Asymptotic** because we ignore lower order terms and constants
- $15n^2 + 6n + 7\log(n) + 145$ is $O(n^2)$
- $2^n + 3906n^{10000} + 892214$ is $O(2^n)$
- If the function of n is polynomial, we say that it is *efficient* or *tractable*

Examples

- For running time functions $f(n)$ listed below, give the Big Oh
- $f(n) = 3n^2 + 4n + 100$
  - $O(n^2)$
- $f(n) = 15n^3 + n\log n + 100$
  - $O(n^3)$
- $f(n) = 1000n + 10000\log n$
  - $O(n)$
- $f(n) = 5050$
  - $O(1)$

Back to CS world

- We assume that all individual operations take the same amount of time
- So, we compute how many total operations we'll have for an input size of ***n*** (because we want to see how running time grows based on input size)
- Then, we find a function that describes that growth

Multiplication by hand

- How long does it take to do multiplication by hand?

```
  123
x 456
-----
  738
 615
492
-----
56088
```

- Let's assume that the length of the numbers is ***n*** digits
- (***n*** multiplications + ***n*** carries) x ***n*** digits + (***n*** + 1 digits) x ***n*** additions
- Running time: $O(n^2)$

Finding the largest element in an array

- How do we find the largest element in an array?

```
int largest = array[0];
for (int i = 1; i < length; ++i) {
  if (array[i] > largest) {
     largest = array[i];
  }
}
System.out.println("Largest: " + largest);
```

- Running time: ***O***(***n***) if ***n*** is the length of the array
- What if the array is sorted in ascending order?

```
System.out.println("Largest: " + array[length-1]);
```

- Running time: ***O***(1)

Multiplying matrices

- Given two n x n matrices A and B, the code to multiply them is:

```
double[][] c = new double[N][N];
for (int i = 0; i < N; ++i) {
  for (int j = 0; j < N; ++j) {
     c[i][j] = 0;
     for (int k = 0; k < N; ++k) {
           c[i][j] += a[i][k]*b[k][j];
     }
  }
}
```

- Running time: $O(n^3)$
- Is there a faster way to multiply matrices?
- Yes, but it's complicated and has other problems

Bubble sort

- Here is some code that sorts an array in ascending order
- What is its running time?

```
for (int i = 0; i < array.length - 1; ++i) {
  for (int j = 0; j < array.length - 1; ++j) {
     if (array[j] > array[j + 1]) {
           int temp = array[j];
           array[j] = array[j + 1];
           array[j + 1] = temp;
     }
  }
}
```

- Running time: $O(n^2)$

Printing a triangle

- Here is some code that prints out a triangular shaped set of stars
- What is its running time?

```
for (int i = 0; i < n; ++i) {
  for (int j = 0; j <= i; ++j) {
    System.out.print("*");
  }
  System.out.println();
}
```

- Running time: $O(n^2)$

Mathematical issues

- What's the running time to factor a large number ***N***?
- How many edges are in a completely connected graph?
- If you have a completely connected graph, how many possible tours are there (paths that start at a given node, visit all other nodes, and return to the beginning)?
- How many different ***n***-bit binary numbers are there?

Formal definition of Big Oh

- Let $f(n)$ and $g(n)$ be two functions over integers
- $f(n)$ is $O(g(n))$ if and only if
  - $f(n) \leq c \cdot g(n)$ for all $n > N$
  - for **some** positive real numbers c and N
- In other words, past some arbitrary point, with some arbitrary scaling factor, $g(n)$ is always bigger

Upcoming

Next time…

- Computing complexity
- Abstract data types (ADTs)

CAREER JUMPSTART EVENT

Engineering & Computer Science

THURSDAY, SEPTEMBER 12TH FROM 4:45PM-7PM

Otterbein University @ The Point

Come and network with alumni and recruitment partners and learn how to be successful with your field.

OTTERBEIN
UNIVERSITY

SCAN the QR CODE to REGISTER

Reminders

- Read section 1.2
- Finish Assignment 1
  - Due Friday by midnight
- Start Project 1
  - Due Friday, September 20 by midnight