Lecture 11 - 2/20/2024

Today in Lecture you talked about the representation of information. The short answer to this is really just Binary but it goes much deeper than this. Let's talk about it!

Earlier in the course we briefly discussed Binary and really it was simply used to describe the amount of memory allocated by the different primitive types in Java. Now we have to learn about binary in more detail.

Binary is a counting system, but unlike our traditional decimal system which uses Base 10 binary is Base 2, meaning we only use two symbols in this case a 0 and 1 to represent values. Take the following example:

$$10101101_2$$

Like in Base 10, in which each place value from right to left can be expressed as a power of 10: 10^0, 10^1, 10^2 … 10^n

The same can be done for binary with the powers of 2: 2^0,2^1,2^2…2^n So the number in base 2 above is actually 173_{10} From this you can derive how to convert between base 2 and base 10, for each position in which the digit is a 1, add that value’s base 10 equivalent to a sum once you are done you have your number in base 10, for the above example it would be $2^0+2^2+2^3+2^5+2^7 = 173_{10}$ If you wish to go backwards you would subtract powers of 2 from the base 10 value until you cannot. So 173-128=45-32=13-8=5-4=1-1=0 which gives us back 10101101_2

We can also perform addition and subtraction in Base 2 like we do in Base 10, I will leave this as an exercise to the reader to figure out but it is like base 10 accept you borrow groups of 2 not groups of 10, please also see the lecture slides on Courseworks for an example or two.

Finally we can acknowledge that Base 16 - also known as hexadecimal - Base 16 allows us to express Base 2 values with fewer place values; each hex digit can express the same information as 4 binary digits.

The symbols used to represent hex values are as follows: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F. So take the following hex value:

FA_{16}

F is equivalent to 15, so the value would be 15*16 + 10*1 = 250_{10}. Try converting this value to binary

Lecture 12 - 2/27/2024

You didn’t learn any new Java today, but you did learn how to more effectively write the Java you already know. You primarily learned about Top-Down Design or as Professor Cannon put it, “Wishful Programming”. So in this review, I will go over Top-Down design and also talk about Bottom-Up design as well so you will have a better understanding of both of these methods and choose which you think is the best for you when you write your programs.

With Top-Down Design we start with the main product that we want to run; From lecture this is the PigTest class, we can visualize it like a tree:

```
                             PigTest.java
```

From PigTest we derive another crucial element, we need a Game class so our tree begins to grow:

```
                             PigTest.java
                                   |
                                   |
                                   V
                              Game.java
```

Well then we start to think about the components of the Game class, well to play a Game you definitely need Players, and in the case of lecture we also needed a die. Our tree now looks as follows:

```
                             PigTest.java
                                   |
                                   |
                                   V
                              Game.java
                                  /\
                                 /  \
                                /    \
                              _/      \_
                              |        |
                              v        v
```

```
                      Player.java    Die.java
```

This could obviously keep going for a while, but in the case of lecture this is the final result. This process is the same for both Top-Down as well as Bottom-Up except the Class Tree is reversed in direction.

```
                            PigTest.java
                                 ^
                                 |
                                 |
                             Game.java
                                 ^
                                / \
                               /   \
                             _/     \_
                            |         |
                            |        |
                      Player.java    Die.java
```

(alternatively you could have wrote the arrows the same direction but have the bottom class on the top)

The direction of the arrow should key into the difference between these design techniques. With Top-Down design, you implement the class at the top and work your way down. With Bottom-Up design, you implement the classes on the bottom and work your way up to the main product. My preference is Bottom-Up design, I am not a “Wishful Programmer” like Professor Cannon.

That is pretty much it, you can learn more about the effectiveness of these techniques as well as when it may be better to use one or the other by googling these methods. Best of luck on the current homework!