Lecture 9 - 2/13/2024
----------------------
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 10 - 2/15/2024
----------------------
You took a deeper dive into how methods actually work in Java today. Specifically, you learned about method overloading and the applications it can have on a program.

Consider this standard method header that we are all familiar with by now as we have encountered numerous quantities of them:

```
public static void main(String[] args)
```

We know at this point that the above header consists of the following components:

1. Access/Visibility modifier - This can be, for purposes of this class right now, public or private
2. Additional Modifiers: keywords like static, final etc go here
3. Return type: can be a simple as void and as complex as a custom object that you've made
4. Method name: this is what we will use to reference and invoke the method
5. Parameter list: could have 0 or more declared values within it.

There comes some potential circumstances where you'd find it useful to reuse the same method name that would potentially have different functionality. The main purpose won't be revealed for a while but it is still useful to learn now.

To begin, methods are determined unique by their method signature which consists: the name, number of arguments, and the order of arguments. When you alter any of these, except the name, you will have an overloaded method.

From this we can conclude that solely altering the return type is not a valid way to overload a method.

Let's look at some valid ways to overload the following method:

```
public int computeExponent(int value) {}
```

The following are valid:

1. public int computeExponent(double value)
2. public int computeExponent(int value, String s)
3. public double computeExponent(double value)

The following is not valid:

1. public double computeExponent(int value)