# MT 1 Notes

Wednesday, October 16, 2013 9:49 AM

**Types:**

- 0b = Binary: 2^n, n = location of 1 or 0 digits starting from left
  - To turn negative: start with 1
    - Invert all signs and add one, making sure last digit is 1
- 0x = Hexadecimal:
  - Every 4 digits of binary = new assignment:
    - 0-9
    - A = 10
    - B = 11
    - C = 12
    - D = 13
    - E = 14
    - F = 15
  - So 50 = 0011 0010 = 32

**Runtimes:**

1. Exponentiation
   a. Tree recursive (multiple calls per iteration/loop?)
2. Polynomial
3. Logarithmic
   a. Binary tree search (reduce 1/2 or more each time you perform the action)
4. Constant
   a. Inserting
   b. Failed while loop

- O(x) = upper bound
  - Take largest/highest value of expression
- $\Theta(x) = lower\ bound\ of\ upper\ bound\ (e.g. smallest\ upper - bound)$
- $\Omega(x) = lower\ bound$
- Amortization
  - If there is one expensive operation, averages out with "cheaper"/faster operations

**Exceptions:**

- Unchecked: bugs (programmer's fault)
- Checked: user's/UI's fault
- StackOverflow Error = never returning
- NullPointerException = didn't set a value

**Bitwise Operations:**

Basically 0 = false; 1 = 1 true

| Operator | Name | What it does |
|---|---|---|
| & | Mask/and | All become 0s unless both 1s |
| \| | Set/or | All become 1s unless both 0s |
| ^ | Flip/xor | Mismatches become 1, otherwise 0 |
| ~ | flip all | Flip everything |
| >> | | Move right |
| << | | Move left |

**:Java::**

- Fields:
  - A data member of a class. Unless specified otherwise, a field is not static.
  - generally a private variable on a instance class. It does not mean there is a getter and setter
- Constructor:
  - Constructs a value for a variable (e.g. in a constructor class)
- Property:
  - User-accessible vs. internal
  - Characteristics of an object that users can set, such as the color of a window.
  - the getter and setter combination
- Variable:
  - An item of data named by an identifier. Each variable has a type, such as int or Object, and a scope. See also class variable, instance variable, local variable.
- Static means doesn't have to be instantiated
  - Can call className.method() instead of class Name = new class()
  - Makes variable names can/cannot be called
  - From Discussion 3:
    - Static methods should be invoked with the class name, without the need for creating an instance of the class
    - You must instantiate an object to call an instance method of the class
    - Static methods cannot access instance variables or instance methods directly; they must use an object reference
    - Static methods cannot use the "this" keyword; only instances can.
- Final = cannot be changed
  - Public static final = basically global variables

```
public class Variables {
//Constant
    public final static String MyVariable = "that was a lot for a constant";
```

```
//Value
    final String dontChangeMeBro = "my god that is still long for a val";


//field
    protected String flipMe = "wee!!!";

//Property
    private String ifYouThoughtTheConstantWasVerboseHaHa;

//Still the propery
    public String getIfYouThoughtTheConstantWasVerboseHaHa() {
        return ifYouThoughtTheConstantWasVerboseHaHa;
    }
    //And now the setter
    public void setIfYouThoughtTheConstantWasVerboseHaHa(String ifYouThoughtTheConstantWasVerboseHaHa) {
        this.ifYouThoughtTheConstantWasVerboseHaHa = ifYouThoughtTheConstantWasVerboseHaHa;
    }

}
```

Pasted from <http://stackoverflow.com/questions/10115588/what-is-the-difference-between-field-variable-attribute-and-property-in-java>

**Static/Dynamic Type/Inheritance:**

1. asking for field or static method => use static type
2. asking for non-static method => use dynamic type
3. "this" within a [particular class] => static type becomes [particular class], dynamic type same
4. a method of dynamic type D and static type S is calling a non-static [method], then the [method] has to exist in S (not necessarily in D)
5. static methods must be overridden by static methods, non-static must be overridden by non-static

Pasted from <https://piazza.com/class/hiah3r2hg4e5aq?cid=755>

- Can only upcast
  - Animal C = new Chihuahua
  - Animal is static; Chihuahua is dynamic
- Variables always call static method (e.g. Animal)
- If non-static, and present in both super and subclass, call subclass (e.g. Chihuahua)
  - Else call superclass (e.g. Animal)
- Creating subclass: extends
  - Implementing interface: implements
- Non-static methods: calling method as defined by dynamic type
  - Static method: calling method as defined by static type
  - Static/non-static fields (variables): look at its static type

| Modifier | Class | Package | Subclass | World |
|---|---|---|---|---|
| Public | Y | Y | Y | Y |
| Protected | Y | Y | Y | N |
| No modifier/package private | Y | Y | N | N |
| Private | Y | N | N | N |