# General Notes

* **Syntax**
	+ **Not** is written as “!”
		- (not(true))
		- !(true)
	+ **//** One line comment
	+ **/\* \*/** Multi line comment
		- /\* multi- line
		 \* comment \*/
	+ **||** is OR in Java
	+ **&&** is AND in Java
		- **Ex:**
			* Boolean isHappy = true;
			* Boolean youKnowIt = true;
			* Boolean clapYourHands = (isHappy && youKnowIt)
				+ Would be true in this case
	+ **Types:**
		- In Java you actually need to say what type you’re using
		- **int** is Java’s version of a number
			* **Integer**also works. Integer is an object in Java that has a few extra built-in methods
		- **String** is a string
		- **boolean** is a Boolean (so, true or false)
* **Classes**
	+ Names are written in **CamelCase**
	+ Are constructs that define new types of data
	+ “Blueprint” for data
	+ Method that operates on a class goes in a class
		- Method to check if a Dillo is dead goes in Dillo class
* **Methods**
	+ Mixed case, with first letter being lowercase
	+ Body of method goes inside of **{}** (curly braces)
	+ Method bodies don’t go in interfaces, just method names and their return types
		- More about this probably tomorrow in class
	+ All take current object as an implicit argument named **this**
		- Basically, every method knows that “this” exists and “this” is the object that you’re operating on (a dillo, a boa, etc)
* **Objects**
	+ Piece of data with concrete values for fields required by a class
		- A specific dillo you made, with a name, age, etc
	+ Must have concrete values, or it’s not an object
	+ A piece of data that holds both data and functions

# Some Examples

**Side Notes:**

/\* You don’t need to assign the values you passed into the constructor for a class in the order they were passed in. You should (for clarity’s sake), but you don’t need to. What matters is that the correct variable in the constructor is assigned to the correct variable within an object. This is also functional but isn’t the best practice: \*/
Person(int aHeight, int anAge, String aName, int aWeight){
this.weight = aWeight;
this.height = aHeight;
this.age = anAge;
this.name = aName;
}
// See how the order we passed them in was “height, age, name, weight,” but then we assigned them in the order “weight, height, age, name”?

/\* Similarly, the names of the variables in the constructor don’t need to match the names of the variables within the class. They SHOULD match, or at least be very close (like “name” vs “aName”), but they don’t need to. So, this is ALSO functional, but **horrendous** practice: \*/
Person(int abcdefgh, int aNumber, String xyz, int x){
this.height = abcdefgh;
this.age = aNumber;
this.weight = x;
this.name = xyz;
}

class Person {
int age;
int height;
int weight;
String name;

Person(int age, int height,
 int weight, String name){
 this.age = age;
 this.height = height;
 this.weight = weight;
 this.name = name;
}

/\* Produces a string saying what this
 \* person eats \*/
String showMealType () {
 if (this.age <= 1) {
 return “Baby food”;
 } else if (this.age < 13) {
 return “Chicken fingers, mac ‘n cheese, and pudding”;
 } else if (this.age < 18) {
 return “Broccoli, carrots, chicken”;
 } else {
 return “Whatever I want because I’m an adult and I make my own decisions”;
 }
}

**class** Examples {

Examples () {}

Person ChristinaAiello = **new** Person(22, 5, 130, "Christina Aiello");

Person AshKetchum = **new** Person(10, 5, 95, "Ash Ketchum");

Person EdwardElric = **new** Person(16, 5, 135, "Edward Elric");

**boolean** testChristinasMealType(Tester t) {

 **return** t.checkExpect(ChristinaAiello.showMealType(), "Whatever I want because I’m an adult and I make my own decisions");

 }

**boolean** testAshsMealType(Tester t) {

 **return** t.checkExpect(AshKetchum.showMealType(), "Chicken fingers, mac ‘n cheese, and pudding");

 }

**boolean** testEdwardsMealType(Tester t) {

 **return** t.checkExpect(EdwardElric.showMealType(), "Broccoli, carrots, chicken");

 }

}