CS Notes 5-21

|  |  |
| --- | --- |
| 0 | Double Linked List   * Shouldn’t take longer than a week * Finding the answer online will not be useful to your education. * This will appear on the written part of the exam w/o resources. * Resisting being given the answer will be important moving forward. |
| 10 | Review 🡪 Static   * Pretty simple concept. * Any instance field that does not contain the word static is only on that class. * Public static void main(String[] args) * Java file, java.class, undecipherable instructions for the java virtual machine. * The JVM is a computer which runs on any other computer * This computer can be examined. * Books that talk about computer architecture refer to the JVM. * Easier to make changes to JVM than a computer. * Static objects always exist, whereas instance objects are tied directly to their class. * Static methods cannot interact with non-static variables. * Static methods are like food that’s already ready to eat. * Static methods cannot be run until the instance object is created. * Static methods can call static variables without issue. * If we name static methods, they can be called immediately without making the class. * If we make an actual object of the class, then it exists and can interact with static. * Static variables in class fields are shared by all classes. |
| 30 | **Shallow copies vs deep copies**   * Java’s update cycler has been becoming more and more frequent. * Var list1 = new ArrayList<Integer>(); * For local variables only, variable type doesn’t have to be defined on the left if var is listed on the right. * Local variables are not initialized until they’re called. * Instance fields would be initialized. * ArrayLIst<> = variable, does not make a new copy * It just points them both to the same space in memory. * **Shallow copies** are copies in name only but have the same address in the memory. Because of this, any change to one will change the other. * **Deep Copies** are copies are copies that have different addresses. Changes to one will not be reflected in the other. |
|  | **Mutable vs. Immutable**   * **Mutable** objects have methods which change the value of the variables inside. * **Accessors** allow other items to get things from inside the method * If an object is **immutable** then **it does not have mutators and thus cannot be changed.** * If there aren’t new objects being created, you’ve made a shallow copy. |
|  | Casting   * Using parenthesis in front of something let’s the compiler know that it’s an object of whatever type was casted. * May require additional parenthesis to force order of operations. |
|  |  |
|  |  |
|  |  |