JAVA OOP ADVANCE

1. **Generics Classes**
   1. Defined with <Type Parameter 1, Type Parameter 2>
   2. Multiple Type Parameters
2. Type Parameter Scope
   1. You can use it anywhere inside the declaring class
3. Subclassing Generic Classes
   1. Can extend to a concrete class
4. Generic interfaces are similar to generic classes
5. Generic Methods
   1. Can take generic input and return generic output
6. Type Erasure
   1. Generics are compile time illusion
   2. Compiler deletes all angle bracket syntax
   3. Adds type casts for us (presented in byte - code)
7. Type Parameter Bounds
   1. <T extends Class> - specifies an “Upper bound”
8. Type Parameters Relationships
   1. Generics are invariant
9. Wildcards
   1. Wildcards introduce polymorphism to type parameters
   2. <?> - specifies a Type that can be any Type(i.e. extends Object)
10. Unbounded Wildcards
    1. List<?> can be a List<Integer>
    2. List<?> can be a List<Double >
    3. List<?> can be of any Type (<? Extends Object>)
11. **Iterators and Comparators**
12. Variable Arguments(varargs)
    1. Allows the method to accept zero or multiple arguments.
    2. There can be only one variable argument in the method.
    3. Variable argument must be the last argument.
13. Iterable<T> and Iterator<T>
    1. Root interface of the Java collection classes
14. A class that implements the Iterable<T> can be used with the new for loop
15. Iterable<T> Methods
    1. Abstract methods
       1. Iterator();
    2. Default methods
       1. forEach(Consumer<? Super T> action)
       2. spliterator() – used for parallel programming
16. Iterator<T>
    1. Enables you to cycle through a collection
    2. Nested class for Iterator<T>
    3. Don’t implement both Iterable<T> and Iterator<T>
17. Comparable<T> vs Comparator<T>
    1. Comparator provides a way for you to provide custom comparison logic for types that you have no control over
    2. Multiple sorting sequence
    3. Doesn’t affect the original class
    4. Compare() method
    5. Comparable allows you to specify how object that you are implementing get compared
    6. Single sorting sequence
    7. Affects the original class
    8. compareTo() method