**1. Define the following in terms of Objects/classes in Java.**

**Stack:** A location in memory where local variables are stored

**blackbox testing:** A software testing method, where the internals of the procedure that's being tested is not known. There are different types of blackbox testing, such as General use (the way the program is supposed to work), Exhaustive Testing (checking each statement), and Unusual Cases (finding unusual/illegal inputs and how they react).

**whitebox testing:** A software testing method, where the internals of the procedure that's being tested is known to the tester. The different types of whitebox testing are, Full Coverage (executing each statement at least once), Full Path (testing the sequence of execution at least once), Corner Cases (Like Unusual Cases, finding where errors are to occur, such as edge cases), and Unit Testing (splitting the program up and testing the split parts).

**stack trace:** The location in execution and current calls when a run-time error occurs making finding errors easier.

**static binding:** binding of methods/statements which is determined at compile time by the compiler (such as overloading methods)

**Postcondition:** What a method is supposed to do after execution

**dynamic binding:** Binding that occurs during run time, unlike static binding, such as overriding methods

**Polymorphism:** Polymorphism is a concept of performing an action in different ways, such as overriding or overloading a method, it can represent a subclass <-> superclass relationship

**Assertion:** A condition that is held true by the programmer, helps inicat the logic of the program

**Precondition:** Explains the state of a method before execution, what is supposed to be true

**loop invariant:**  A type of assertion statement that stays true before and after a loop is executed

**Encapsulation:** wrapping code together in a single package to keep it hidden

**2. Examine the Java API and list three useful methods for each of these wrapper classes:**

**Integer:**

* String - toHexString(int): returns the String value of an unsigned hex integer in base 16.
* String - toString(): returns a String object representing the Integer’s value.
* int - parseInt(String): parses the String argument into an integer if possible (if not an exception is thrown)

**Double:**

* boolean - isNaN(): returns whether this Double is Not a Number
* double - parseDouble(String): parses the String argument into a Double if possible (if not an exception is thrown)
* boolean -isInfinite(): returns whether this Double is Infinity (based off of java 64 bit floating point)

**Char:**

* boolean - isLetter(char): returns whether a char is a letter from a/A-z/Z
* boolean - isDigit(char): returns whether a specified char is a digit.
* char - toUpperCase(char): returns a specified char as uppercase (based off of unicode file data)

**Boolean:**

* boolean - valueOf(String): returns a Boolean instance from a specified String input
* String - toString(): returns a String object representing the current boolean value
* String - parseBoolean(String): parses the string argument as a boolean.

**3. What is the difference between using the method length when accessing a String versus an int array?** In Arrays there is a final attribute given to that array known as the length, which coincidentally stores the length of the array. In Strings there is a method called length() where the amount of characters within a String can be accessed. When using the methods the difference is the method call vs accessing an attribute.

Ex. string.length() vs array.length.

**4. Math Classes - define the following methods by the following: purpose, input, return type, cautions:**

**abs() :**

* Purpose: Gives the absolute value of a floating point number or an int (ex. abs(-7) -> 7)
* Input : int, double, float, long
* Return type : int, double, float, long
* Cautions : if the input is Integer.MIN\_VALUE or Long.MIN\_VALUE, the result will be the same number (the lowest representable number of that data type, therefore negative)

**pow() :**

* Purpose: This method accepts two parameters and returns the value of the first parameter to the power of (raised) to the second.
* Input: double
* Return type: double
* Cautions: if the first/second parameter is a +/- 0 the return value will be 1.0, and if the second parameter is not a number the return value will be NaN.

**sqrt() :**

* Purpose: Takes a value and returns the square root of that value
* Input: double
* Return type: double
* Cautions: if NaN or a negative number is inputted, the return value is NaN, if the input is positive infinity or +/- 0 the same value will be returned.

**5. What does “NaN” stand for?**

NaN stands for “Not a Number” it is usually returned when a floating point operation’s input parameters have caused the operation to produce an undefined result. Ex 0.0/0.0

**6. Explain the difference between recursion, mutual recursion, and tail recursion.**

Recursive methods call upon themselves until a base case is met, they pile up in memory. Mutual Recursion is when 2 methods (can be multiple) call each other back and forth with at least one method decomposing the call so a base case can be met. In Tail Recursion a method calls itself using a subroutine as the final statement in the procedure, (each time the call is decomposed until a base case is met)