Question 1:

Interface: The publicly accessible methods and operations of a black box that define its inputs, outputs, and behavior.

Implementation: The internal logic and code that makes a black box function as intended.

Question 2:

The contract of a subroutine: Specifications and obligations that the subroutine guarantees to fulfill.

Importance of understanding the contract: Ensures correct usage and behavior of the subroutine.

Syntactic aspect: Rules and requirements for correct usage and structure of the subroutine.

Semantic aspect: Intended meaning and behavior of the subroutine.

Question 3:

Subroutines in top-down design: Enable modularization, abstraction, and easier problem-solving by breaking down complex problems into smaller, manageable subproblems.

Question 4:

Parameters: Variables used to pass data between different parts of a program or subroutines.

Formal parameters: Placeholders defined in the subroutine's declaration or signature.

Actual parameters: Values or variables passed to the subroutine during its execution.

Question 5:

Reasons for using named constants: Enhance readability, maintainability, and code flexibility. Simplify value modifications and promote code reuse.

Question 6:

API: Application Programming Interface. It defines rules, protocols, and tools for software applications to interact and communicate with each other. Example: Java API provides a set of classes and methods for Java programming language.

Question 7:

The expression "(a,b) -> aa + bb + 1" represents a lambda expression in a program. It defines an anonymous function with two parameters 'a' and 'b'. The function calculates the square of 'a', the square of 'b', and adds 1. The result of this expression is the sum of the squares of 'a' and 'b' plus 1.

Question 8:

Lambda expression to get a random integer in the range 1 to 6 inclusive:

SupplyInt randomInt = () -> (int) (Math.random() \* 6) + 1;

Lambda expression to get an integer from the user:

SupplyInt userInput = () -> {

System.out.print("Enter an integer: ");

return TextIO.getInt();

};

Question 9:

public static void stars(int numberOfStars) {

for (int i = 0; i < numberOfStars; i++) {

System.out.print("\*");

}

System.out.println();

}

Question 10:

public static void main(String[] args) {

for (int i = 1; i <= 10; i++) {

stars(i);

}

}

Question 11:

public static int countChars(String str, char ch) {

int count = 0;

for (int i = 0; i < str.length(); i++) {

if (str.charAt(i) == ch) {

count++;

}

}

return count;

}

Question 12:

public static int findSmallest(int a, int b, int c) {

int smallest = a;

if (b < smallest) {

smallest = b;

}

if (c < smallest) {

smallest = c;

}

return smallest;

}

Question 13:

public static double findAverage(double[] array, int N) {

if (N <= 0 || N > array.length) {

return 0.0; // Handle invalid input

}

double sum = 0.0;

for (int i = 0; i < N; i++) {

sum += array[i];

}

return sum / N;

}

Question 14:

The purpose of the stripZeros function is to remove all occurrences of the number zero from an input array and return a new array without the zeros. The function works by iterating over the input array in two passes. In the first pass, it counts the number of non-zero elements in the array to determine the size of the new array (count). It then creates a new array, newList, with the appropriate size. In the second pass, it copies the non-zero elements from the input array to the new array, newList. The variable j keeps track of the index in newList, ensuring that the elements are correctly assigned. Finally, the function returns the new array newList, which contains only the non-zero elements from the original array.