Disclaimer:

The answers provided below are not entirely precise and should not be relied upon for learning purposes. They are part of an exercise to improve my understanding, explanation skills, and problem-solving techniques.
Thank you, Haim Levhar

What is hardware?

Hardware refers to the components of a computer system that support the computing process. If you can touch it and you can physically feel it, then it is computer hardware.

What happens to information when it's stored digitally?

The computer system breaks this information into base 2 (one and zero) and creates a unique series of these numbers for each item in the information. For example, a text that contains "Nice try", it divided this text into as many pieces as needed. In this case, 7.

The music on a CD is created using a sampling rate of 44,000 measurements per second. Each measurement is stored as a number that represents a specific voltage level. How many such numbers are used to store a three-minute-long song? How many such numbers does it take to represent one hour of music?

To store a three-minute-long song, we need the following formula:

[44000 (Measurements per second) \times 60 (Seconds in one minute) \times 3 (Minutes) = 7920000]

To represent one hour of music, we need the following formula:

[44000 (Measurements per second) \times 60 (Seconds in one minute) \times 60 (One hour) = 158400000]

How many unique items can be represented with the following?

- 2 bits? (2^2 = 4) items can be represented.
- 4 bits? (2^4 = 16) items can be represented.
- 5 bits? (2^5 = 32) items can be represented.
- 7 bits? (2^7 = 128) items can be represented.

Suppose you want to represent each of the 50 states of the United States using a unique permutation of bits. How many bits would be needed to store each state representation? Why?

We need to represent at least (2^6), which equals 64. This will be enough for our 50 states because we need only 50 unique items, and (2^6) provides more than 50 unique items.

What is the difference between the literals 4, 4.0, '4', and "4" in Java?

The first group consists of primitive data types, while the second is an object of the String class.

Explain the following programming statement in terms of objects and the services they provide:

Here's the statement:

```
System.out.println("I gotta be me!");
```

- `System` is a built-in class in Java.
- `out` is an object from the `System` class, used to produce output.
- `println` is a method of the class `System` that displays the string within the line.
- `""` To assign a String value, double-quotation marks are required at the beginning and end of the string.
- `;` is a statement terminator used to indicate the end of a statement to the compiler.

What output is produced by the following code fragment?

```
System.out.print("Here we go!");
System.out.println(12345);
System.out.print("Test this if you are not sure.");
System.out.print("Another");
System.out.println();
System.out.println("All done.");
```

Here is the output for this code segment with a brief explanation:

```
Here we go!12345
Test this if you are not sure.Another
All done.
```

It prints text and numbers using both `println` (which adds a new line) and `print`.

What is wrong with the following program? How can it be fixed?

```
System.out.println("To be or not to be,
that is the question");
```

```
Error, not a statement
```

It prints an error because the strings need to be concatenated with the `+` operator when you have two strings on different lines used in the `println` method.

What output is produced by the following statement? Explain.

```
System.out.println("50 plus as is" + 50 + 25);
```

Here is the output for this code segment with a brief explanation:

```
50 plus as is5025
```

When you concatenate several int/float types into a string type, the number is converted into a String. As a result, it is displayed as `5025` instead of `75`.

What is the output produced by the following JAVA statement? Explain.

```
System.out.println("He trusts his fists\n\t against" +
" the post \t and still insists\n\t he sees the \"ghost\".");
```

Here is the output for this code segment with a brief explanation:

```
He trusts his fists
     against the post
 and still insists
     he sees the "ghost".
```

In Java, escape sequences are used to indicate that the character or characters that follow should be interpreted specially. The `\n` escape sequence forces the output to a new line. The `\t` escape sequence represents the tab character. The `\"` escape sequence ensures that the quote character is treated as part of the string, not the termination of it.

What does the result of `19 % 5` when evaluated in a JAVA expression? Explain.

The `%` operator in Java represents the remainder operator. It returns the remainder after dividing the first operand by the second operand. In the case of `19 % 5`, the result will be `4`. Because 5 goes into 19 three times (giving 15) with a remainder of `4`.

Describe the hardware components of your PC or a computer in a lab to which you have access. Include the processor type and speed, storage capacities of main and secondary memory, and the type of I/O devices. Explain how you determined your answer.

To find information about the hardware components of a computer, you can use system information tools or check the specifications provided by the manufacturer.

How many unique items can be represented with each of the following?

- Bit? With a bit, you can represent (2^1 = 2) unique items.

- 3 bits? With 3 bits, you can represent (2^3 = 8) unique items.
- 6 bits? With 6 bits, you can represent (2^6 = 64) unique items.
- 8 bits? With 8 bits, you can represent (2^8 = 256) unique items.
- 10 bits? With 10 bits, you can represent (2^{10} = 1024) unique items.
- 16 bits? With 16 bits, you can represent (2^{16} = 65536) unique items.

If a picture is made up of 128 possible colors, how many bits would be needed to store each pixel of the picture? Why?

At least 7 bits are required because 7 bits represent (2^7 = 128) unique items, which is enough for the 128 possible colors.

If a language uses 240 unique letters and symbols, how many bits would be needed to store each character in a document? Why?

At least 8 bits are required because 8 bits represent (2^8 = 256) unique items, which is enough for the 240 unique letters and symbols.

How many bits are there in each of the following? How many bytes are there in each?

12 KB? 12 * 1024 = 12888 bits
5 MG? 5 * 1048576 = 5242880 bits
3 GB? 3 * 1073744824 = 3221225472 bits
2 TB? 2 * 1099511627776 = 2199023255552 bits

Explain the differences between a local-area network (LAN) and a wide-area network (WAN), what is the relationship between them?

A **local-area network (LAN)** is used in small organizations to connect the computers to the same network. A **wide-area network (WAN)** is used worldwide for connecting LANs into one large network.

Explain the difference between the Internet and the World Wide Web (WWW).

They may seem equal in their features, but there are important differences between them. The **Internet** makes communication possible via computers worldwide, Whereas the **WWW** (i.e., web) makes that communication straightforward and enjoyable.

Give two examples of two types of comments in Java And explain the difference between them.

Java has two types of comments: the '//' comment type and the '/, /' *comment type. The first type, '//' is used to display information to the user about the program you coded in a single line (it cannot continue to the next line; if you do, you will get a compile-time error). The second type, '/, /' is used for multiple comments lines. You need to start the comment with the '/' sign to represent the beginning of a comment and to the end, you add* ''.

What is a Java package?

A Java package is a set of classes that supports the development of software programs.

What does the java.net package contain? The javax.swing package?

java.net facilitates network communication across a network. **javax.swing** creates graphical user interfaces (GUI) with components extending the AWT capabilities.

What package contains the Scanner class? The String class? The Random class? The Math class?

The **Scanner** and **Random** classes are part of the **java.util** package. The **String** and **Math** classes are part of the **java.lang** package.

What does an import statement accomplish?

To use classes from any other package, you can write an import statement that makes that package available in your program, including all classes and methods within it.

Why doesn't the String class have to be specifically imported into our program?

The String class does not have to be specifically imported into a Java program because it is a fundamental library and can be thought of as a basic extension of the language. Therefore, Java imports that class automatically each time we create a program.

Which of the following are not valid Java identifiers? Why?

a. **Factorial** - valid. b. **anExtremelyLongIdentifierIfYouAskMe** - valid but not recommended. c. **2ndLevel** - not valid. Identifiers cannot start with a number. d. **level2** - valid. e. **MAX_SIZE** - valid; recommended only for constants. f. **highest$** - valid. g. **hook&ladder** - not valid. You can't put the % sign in an identifier.

For the following pair, which represents a class and which represents an object of that class, Superhero, Superman?

Superman is an object of the **Superhero** class.

List some attributes and operations that might be defined for a class called 'PictureFrame' that represents a picture frame.

Attributes - Size, Color, Background color, Image. Operations - setColor(color), setImage(image), updateSize(newSize), addImage(newImage).

List some attributes and operations that might be defined for a class called 'Meeting' that represents a business meeting.

Attributes - Company, Names of attendees, Date of the meeting, Type of the meeting (e.g., board meeting, team meeting), Money earned during the meeting, Money lost during the meeting. Operations - setNames(attendees), updateDateMeeting(newDate), setTypeMeeting(newType), moneyEarned(amountEarned), moneyLost(amountLost).

List some attributes and operations that might be defined for a class called 'Course' that represents a college course (not a particular offering for a course, just the course in general).

Attributes - Name of the Course, Date of the course (start date, end date), Number of exams in the course, Names of students, Names of teachers, Cost of the course.

Operations - updateDate(newStartDate, newEndDate), updateNumberOfExams(newNumExams), setStudentName(newStudentName), setTeacherName(newTeacherName), updateCost(newCost).

What is meant by the flow of control through a program?

The flow of control through a program determines the program statements that will be executed on a given run by the program.

What type of conditions are conditional and loops based on?

Each condition or loop is based on a boolean condition that evaluates either true or false.

What are the equality operators? What are the relational operators? What are the logical operators?

The equality operators are the following:

- Equal to(==). For example: `5 == 5`
- Not equal to (!=). For example: `5 != 10`

The relational operators are the following:

- Greater than(>). For example: `10 > 5`
- Less than(<). For example: `5 < 10`
- Greater than or equal to(>=). For example: `10 >= 10 || 10 >= 5`
- Less than or equal to(<=). For example: `10 <= 10 || 5 <= 10`

The logical operators are the following:

- AND (&&). For example: `true && false == false`
- OR (||). For example: `true || false == true`
- NOT (!). For example: `!true == false`

Exercise: Analyzing Code Output.

Consider the following Java code snippet. Predict the output based on the provided assumptions.

```
if (num1 < num2)
  System.out.println("red");
if ((num1 + 5) < num2)
  System.out.println("white");
else
  System.out.print("blue");
System.out.println("yellow");
```

Assumptions: `num1` is assumed to be 2, `num2` is assumed to be 10. Expected Output:

```
red
white
yellow
```

Assumptions: `num1` is assumed to be 10, `num2` is assumed to be 2. Expected Output:

```
blue
yellow
```

Assumptions: `num1` is assumed to be 2, `num2` is assumed to be 2. Expected Output:

```
blue
yellow
```

Consider the following Java code snippet. Predict the output based on the provided assumptions:

```
if (num1 >= num2)
{
  System.out.println(" red ");
  System.out.println(" orange ");
}
if ((num1 + 5) >= num2)
{
  System.out.println(" white ");
}

else
  if ((num1 + 10) >= num2)
  {
    System.out.println(" black ");
    System.out.println(" blue ");
  }
  else
    System.out.println(" yellow ");
System.out.println(" green ");
```

Assumptions: `num1` is assumed to be 5, `num2` is assumed to be 4. Expected output:

```
red
orange
white
green
```

Assumptions: `num1` is assumed to be 5, `num2` is assumed to be 12. Expected output:

```
black
blue
green
```

Assumptions: `num1` is assumed to be 5, `num2` is assumed to be 27. Expected output:

```
yellow
green
```

Write an expression that will print a message based on the value of the int variable named temperature. If temperature is equal to or less than 50, it prints "It is cold." on one line and "Dress warmly." on the next. If temperature is greater than 80, it prints "It is warm." on one line and "Dress coolly." on the next. If temperature is between 50 and 80, it prints "It is pleasant." on one line and "Dress pleasantly." on the next.

```
public class testing {
    public static void main(String[] args) {
      int temperature = 40;

      if (temperature <= 50) {
        System.out.println("It is cold");
        System.out.println("Dress warmly.");
      } else if (temperature > 80) {
        System.out.println("It is warm");
        System.out.println("Dress coolly");
      } else if (temperature > 50 && temperature <= 80) {
        System.out.println("It's pleasant.");
        System.out.println("Dress pleasantly.");
      }
    }
}
```

What type of elements does an ArrayList hold?

an ArrayList holds the object type that you determine by writing it within angle brackets located after the ArrayList declaration. an ArrayList holds a list of objects; it cannot hold primitive data types. If you want an ArrayList to hold primitive data types, use Wrapper classes that convert primitive data types into objects. For example, for an ArrayList to hold int data type, perform the following declaration: ArrayList listInt = new ArrayList();

Write a declaration for a variable named dice that is an ArrayList of Dice objects

Here is the declaration: ArrayList dice = new ArrayList();

What output is produced by the following code fragment?

```
ArrayList<String> names = new ArrayList<String>();
names.add("Andy");
names.add("Betty");
names.add(1, "Chet");
names.add(1, "Don");
names.remove(2);
System.out.println(names);
```

Here's the output:

```
Andy Don Betty
```

Put the following list of strings in lexicographic order as if determined by the `compareTo` String class' method. Consult the Unicode chart in Appendix C.

Here is the list: "fred", "Ethel", "?-?", "{([])}", "Lucy", "ricky", "book", "**", "12345", " ", "HEPHALUMP", "bookkeeper", "6789", ";+<?", "^^^^", "hephalump". Here is the corrected version list in lexicographic order: " ", "**", "12345", "6789", ";+<?", "?-?", "Ethel", "HEPHALUMP", "Lucy", "^^^^", "book", "bookkeeper", "fred", "hephalump", "ricky", "{([])}".

Transform the following nested if statement into an equivalent switch statement:

```
if (num == 1)
  myChar = 'a';
else
  if (num == 2)
    myChar = 'b';
  else
    if (num == 3)
      myChar = 'c';
    else
      myChar = 'd';
```

Here is the switch statement that provided the same result like the previous code fragment. option 1:

```
switch(num) {
  case 1:
    myChar = 'a';
    break;
  case 2:
    myChar = 'b';
    break;
  case 3:
    myChar = 'c';
    break;
```

```
    default:
      myChar = 'd';
  }
```

option 2:

```
switch(num) {
  case 1 -> myChar = 'A';
  case 2 -> myChar = 'B';
  case 3 -> myChar = 'C';
  default -> myChar = 'D';
}
```

What is the difference between a conditional operator and a conditional statement?

The distinction between the conditional operator and the conditional statement can be subtle but important. The ?: lets you form succinct logic that serves as an abbreviated if-else statement.

Write a declaration that initializes a char variable named id to 'A' if the boolean variable first is true and to 'B' otherwise.

```
// Declaration initializing char variable id to 'A' if first is true, 'B'
otherwise
char id = first ? 'A' : 'B';
```

succinctly express the following logic code using the conditional operator:

```
if (value <= 10)
  System.out.println("The value is not greater than 10");
else
  System.out.println("The value is greater than 10");
```

Here's the switch statement that provided the same result such as the previous code fragment:

```
// Express logic code succinctly using conditional operator
System.out.println(value <= 10 ? "The value is not greater than 10" : "The value
is greater than 10")
```

Write a do loop to obtain a sequence of positive integers from the user, using zero as a sentinel value. The program should output the sum of the numbers.

```
final int SENTINEL = 0;
int sum = 0;
int numUser = 0;

Scanner scanner = new Scanner(System.in);

do {
  System.out.println("Enter a positive integer value (Enter 0 to exit): ");
  numUser = scanner.nextInt();
  sum += numUser;
} while (numUser != SENTINEL);
  System.out.println("Sum of the numbers: " + sum);

  scanner.close();
```

How many iterators will the following for loops execute?

1. ```
   for (int i = 0; i < 20; i++) {}
   ```
2. ```
   for (int i = 1; i <= 20; i++) {}
   ```
3. ```
   for (int i = 5; i < 20; i++) {}
   ```
4. ```
   for (int i = 20; i > 0; i--) {}
   ```
5. ```
   for (int i = 1; i < 20; i = i + 2) {}
   ```
6. ```
   for (int i = 1; i < 20; i *= 2) {}
   ```

Answers for each for loop in the provided list:

1. 20 The loop will iterate from 0 to 19 inclusive (20 iterations).
2. 20 The loop will iterate from 1 to 20 inclusive (20 iterations).
3. 15 The loop will iterate from 5 to 19 inclusive (15 iterations).
4. 20 The loop will iterate from 20 to 1 inclusive (20 iterations).
5. 10 The loop will iterate from 1 to 19 only with the odd numbers (10 iterations).

6. 5 The loop will iterate from 1 to 19, doubling the value of 'i' in each iteration (5 iterations).

What output is produced by the following code fragment?

```
for (int num = 0; num <= 200; num += 2) {
  System.out.println("The current num is: " + num);
}
```

Here's the explanation and the output of the provided code fragment:

Explanation: The loop will iterate from 0 to 200 inclusive, incrementing the value of `num` by 2 in each iteration.

SOutput:

```
The current num is: 0
The current num is: 2
The current num is: 4
The current num is: 6
...
The current num is: 198
The current num is: 200
```

What output is produced by the following code fragment?

```
for (int value = 200; value >= 0; value -= 1)
  if (value % 4 != 0)
    System.out.println("The value is: " + value);
```

Here's the explanation of the provided code fragment:

Explanation: The loop will iterate from 200 to 0 inclusive, decrementing the value of `value` by 1 in every iteration. It will then check if the current value is not divisible by 4 without a remainder. If this is true, it prints the current value.

Transform the following `while` loop into an equivalent `do` and `for` loops (make sure it produces the same result).

```
int num = 1;
while (num < 20) {
  num++;
  System.out.println(num);
}
```

Here is the equivalent `do` loop that does the same result such as the provided `while` loop.

```
int num = 1;
do {
  num++;
  System.out.println(num);
} while (num < 20);
```

Here is the equivalent `for` loop that does the same result such as the provided `while` loop.

```
for (int num = 2; num <= 20; num++)
  System.out.println(num);
```

What is an array?

An array is a powerful programming language structure used to group and organize data.

How is each element of an array referenced?

An index is used to indicate the position of an element in an array (starting from 0). To reference a particular element in an array, use the array name and insert the index of the desired element in the array within square brackets. For example: names[2]. This notation represents the element at index 2.

Based on the array shown below,

```
int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

What are each of the following?

```
numbers[1] // 2
numbers[2] + numbers[5] //
numbers[2 + 5] // 8
/* The values stored at index 8 */ // 9
/* The fourth value */ // 4
numbers.length() // 12
```

What is an array's element type?

An array's element type is the type that an array can hold. The type of elements inside the array must be consistent with the type declared before the array name. For example, if we want to declare an array will hold integer values. This array contains the numbers from 1 through 10. The following code fragment accomplishes this:

```
int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

In this statement, the elements' type is integer, and the elements are the numbers from 1 to 10. Note that an array can hold only one element type. This means that all the elements in this array must be of the same object type. Attempting to change this will result in an error.

Write an array declaration to represent the ages of all 100 children attending a summer camp.

We need the `int` type to represent the children's ages and assign an appropriate name to the variable called `AgesChildren`. The following fragment accomplishes that.

```
int[] AgesChildren = new int[100];
```

First, it creates a new array with 100 index positions for each child's age inside square brackets using the Java reserved word `new`, and finally, it adds a semicolon.

Write an array declaration to represent counts of how many times each face appeared when a standard Six-sided die is rolled.

It needs an `int` type to represent how many times each particular face appears when the die is rolled. Assign an appropriate name to a variable called `faceCounts`. The following fragment accomplished that. First,

```
int[] faceCounts = new int[6];
```

it creates a new array with 6 index positions for each particular face, inside square brackets using the Java reserve word `new`. Finally, it adds a semicolon.

Write code that increments(by one) Each element of an array of integers named `values`.

The following fragment accomplished that,

```
int[] values = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

for (int i = 0; i < values.length; i++) {
  values[i] += 1;
}
```

First, it goes through the array, processing each element represented as `i`, and increments `i` (i.e., The current element) by one.

Write code that competes and prints the sum of the elements of an array of integers named `values`.

```
int[] values = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int sum = 0;

for (int i = 0; i < values.length; i++) {
  sum += values[i];
}
```

Suppose a team is an array of strings meant to hold the names of the six players on the Volleyball team: Amanda, Clare, Emily, Julie, Katie, and Maria. A- Write an array declaration for the team. B- Show how to both declare and populate the team using an initializer list

Answer for A:

```
String[] team = new String[6];
```

Answer for B:

```
// Declare the team array
String[] team = new String[6];

// Using an initializer list
String[] team = {"Amanda", "Clare", "Emily", "Julie", "Katie", "Maria"};
```

Assume Book is a class whose objects represent books. Assume a Constructor of the book class accepts two parameters - The name of the book and the number of the pages. A- Write a declaration for a variable named `library` That is an array of ten 10 books. B- Write a `new` statement that sets the first book in the library array to "Starship Troopers", which is 208 pages long.

Answer for A:

```
Book[] library = new Book[10];
```

Answered for B:

```
Book[] library = new Book[10];

library[0] = new Book("Starship Troopers", 208);
```

Write a main method for a program that outputs the sum of the string lengths of its first 2 command line arguments.

```
public static void main(String[] args) {
  if (args.length < 2) {
    System.out.println("Please provide at least two command-line arguments. ");
    return;
  }

  int sumLengthStrings = args[0].length() + args[1].length();
  System.out.println("Sum lengths of the provided strings: " sum);
}
```

Write a main method for a program that outputs the sum of the values of its first two command line arguments, which are integers.

```
public static void main(String[] args) {
  if (args.length < 2) {
    System.out.println("Please provide at least two command-line arguments. ");
    return;
  }

  int num1 = Integer.parseInt(args[0]);
  int num2 = Integer.parseInt(args[1]);
  int sum = num1 + num2;
  System.out.println("Sum of the provided integers: " + sum);
}
```

Write a method called distance that accepts a multiple number of integer parameters (Each of which represents the distance of one leg of a journey), Then returns the total distance of the trip.

```
public static int distance(int ... leg) {
  int totalDistance = 0;

  for (int i : leg) {
    totalDistance += i;
  }
  return totalDistance;
}

System.out.println(distance(1, 2, 3, 5, 6, 79, 0));
```

Write a method called travelTime that accepts an integer parameter indicating average speed followed by a multiple number of integer parameters (each of which represents the distance of a leg of a journey) and returns the total time of the trip.

```
public static double travelTime(int averageSpeed, int ... leg) {
  int sumLegs = 0;

  for (int i : leg) {
    sumLegs += i;
  }

  return (double) sumLegs / averageSpeed;
}
System.out.println("Hours of the trip: " + travelTime(50, 100, 80, 70));
```

Two-dimensional array named scores holds the test scores for a class of students for a semester. int[][] scores = {{85, 90, 92}, {78, 85, 88}, {72, 68, 74}}; Write code that prints out a single value that represents the range of scores held in the array. It prints out the value of the highest score minus the value of the lowest score. score represented as an integer.

```
// Initialize the scores of the students
int[][] scores = {{85, 90, 92},
                  {78, 85, 88},
                  {72, 68, 74}};

int maxScore = Integer.MIN_VALUE;
int minScore = Integer.MAX_VALUE;

// Passing on every student and his scores
for (int i = 0; i < scores.length; i++) {
    for (int j = 0; j < scores[i].length; j++) {
      if (scores[i][j] > maxScore) {
        maxScore = scores[i][j];
      } if (scores[i][j] < minScore) {
        minScore = scores[i][j];
      }
    }
  }
// Expression to calculate the range of the scores
int scoresRange = maxScore - minScore;

// Prints a message accordingly
System.out.println("Range between the highest score - lowest score: " +
scoresRange);
```

Which of the following are valid declarations? Which instantiate an array object?

```
// Invalid declaration. To instantiate an array object,
// square brackets are needed after the type or name of the array.
int primes = {2, 3, 5, 7, 11};
```

```
// Valid declaration. Instantiates an array object,
// with the instantiation call after the variable name.
float elapsedTime[] = {11.47, 12.04, 11.72, 13.88};

// Invalid declaration. To instantiate an array object,
// the 'new' keyword must be used with square brackets specifying the size.
int[] scores = int[30];

// Invalid declaration. The type of the array elements should be specified after
the 'new' keyword.
int[] primes = new {2, 3, 5, 7, 11};

// Valid declaration. Instantiates an array object with a specified size.
int[] scores = new int[30];

// Valid declaration. Instantiates an array object with specified elements.
char grades[] = {'a', 'b', 'c', 'd', 'f'};

// Invalid declaration. Misses the size parameter for the array instantiation.
char[] grades = new char[];
```

Define a class `SchoolBook2` that extends `Book2` to indicate an attribute indicating the age (4 through 6) that a book targets. The constructor accepts the `age` as a parameter. The class also provides a `level` method that returns a string as follows: "Pre-school" if the target age is 4 through 6, "Early" if the target age is 7 through 9, "Middle" if the target age is 10 through 12, and "Upper" if the target age is 13 through 16.

Book2.java

```
public class Book2 {
  protected int pages;

  public Book2(int numPages) {
    pages = numPages;
  }

  public void setPages(int newNumPages) {
    pages = newNumPages;
  }

  public int getPages() {
    return pages;
  }
}
```

SchoolBook2.java

```
public class SchoolBook2 extends Book2 {
  private int ageLevel;

  public SchoolBook2(int numPages, int age) {
    super(numPages);
    this.ageLevel = age;
  }

  public String level() {
    if (ageLevel >= 4 && ageLevel <= 6) {
      return "Pre-school.";
    } else if (ageLevel >= 7 && ageLevel <= 9) {
      return "Early.";
    } else if (ageLevel >= 10 && ageLevel <= 12) {
      return "Middle.";
    } else if (ageLevel >= 13 && ageLevel <= 16) {
      return "Upper.";
    } else {
      throw new IllegalArgumentException("Invalid age level: " + ageLevel);
    }
  }

    public static void main(String[] args) {
      SchoolBook2[] schoolBooks = {new SchoolBook2(800, 5),
                                   new SchoolBook2(800, 8),
                                   new SchoolBook2(800, 11),
                                   new SchoolBook2(800, 14),
                                   new SchoolBook2(800, 3)};

      for (SchoolBook2 ageLevel : schoolBooks) {
        System.out.println(ageLevel.level());
      }
  }
}
```

True or False? Explain.

A child class may define a method with the same name as the method in the parent class.

True. Explanation: A child class can define a method with the same name as a method in the parent class by overriding it to provide its definitions. The `@Override` annotation above the method signature indicates to the compiler that this method overrides the method in the parent class.

A child class can override the constructor of the parent class.

False. Explanation: A child class can't override the constructor of the parent class. The constructor is a special method of the parent class that cannot be overridden. This design prevents unnecessary problems from arising.

A child class can override a `final` method of the parent class.

False. Explanation: A child class can't override a final method of the parent class. This restriction is by design, indicated by the `final` keyword.

It is considered poor design when a child class overrides a method from the parent class.

False. Explanation: It is not considered poor design when a child class overrides a method from the parent class. This is a common technique used in inheritance to extend and specialize behavior, promoting code reuse.

A child class may define a variable with the same name as a variable in the parent class.

True. Explanation: A child class can define a variable with the same name as a variable in the parent class. This is known as a "shadow variable." However, it's generally not recommended due to the potential for confusion and errors.

Experiment with a simple derivation relationship between two classes. Put `println` statement in constructors of both the parent and child classes. Do not explicitly call the constructor of the parent. Now what happens?

```
public class testing {
    public testing() {
        System.out.println("Hello from parent class.");
    }

    public static void main(String[] args) {
        testingTheTesting test1 = new testingTheTesting();
    }
}

class testingTheTesting extends testing {
    public testingTheTesting() {
        System.out.println("Hello from the child class.");
    }
}

// output:
// Hello from the parent class
// Hello from the child class
// First the parent's constructor is executed and after that, the child's
constructor is executed.
```

Write a recursive method that returns the value of 5 * n, where n > 0. Explain why you would not normally use recursion to solve this problem.

Here is the code that solves this problem:

```
public static int fiveMultiByNum(int num) {
  if (num <= 0) {
```

```
    return 0;
  } else if (num == 1) {
    return 5;
  } else {
    return 5 + (5 * fiveMultiByNum(num - 1));
  }
}
```

Here is the explanation: it is better that you would not use this recursion to solve such a problem. Recursion is the most elegant and appropriate way to solve some problems. However, for others, it is less intuitive than an iterative solution.

Describe what is returned by the following recursive method.

```
private static int unknown(int num) {
  if (num < 0) {
    return -1;
  } else if (num < 10) {
    return 1;
  } else {
    return 1 + unknown(num/10);
  }
}
```

This code returns how many times the provided `num` parameter is divided by 10

Illustrating the sequence of alterations implemented by the selection sort algorithm on the given list of numbers: 5 7 1 8 2 4 3.

The procedural steps of the selection sort algorithm for arranging an array of elements are as follows:

1. Original Array: 5 7 1 8 2 4 3
2. First Iteration: 1 7 5 8 2 4 3. The selection sort begins by identifying the smallest value in the array (in this case, '1') and exchanging it with the value at the initial position of the list. Thus, the numbers 1 and 5 are swapped.
3. Second Iteration: 1 2 5 8 7 4 3. Again, the algorithm identifies the smallest value among the remaining elements and swaps it with the value at the second position of the list (following the one already sorted). The exchanged values are 2 and 8.
4. Subsequent Iterations: Continue this process for all elements within the array.
5. Resultant Array after Iterations:
    - 1 2 3 8 7 4 5
    - 1 2 3 4 7 8 5
    - 1 2 3 4 5 8 7
    - 1 2 3 4 5 7 8 These steps demonstrate the progressive refinement of the array until it is fully sorted in ascending order.

Given the following list of numbers: 15 21 4 17 8 27 1 22 43 57 25 7 53 12 16. how many elements of the list would be examined by the liner search algorithm if each of the indicated target elements were on the list? 17, 15, 16, 45

If the target element is 17, the liner search algorithm will reach position 4, where the element in this position is the number 17. That is, The liner search algorithm will have examined 4 elements. If the target element is 15, the liner search algorithm will reach position 1, where the element in this position is the number 15.That is, The liner search algorithm will have examined 1 element. If the target element is 16, the liner search algorithm will reach position 15, where the element in this position is the number 16, which is the length of the list. That is, The liner search algorithm will have examined 15 elements which is the length of the list If the target element is 45, the liner search algorithm will reach the end of the list without finding the elements. That is, the linear search algorithm did not find the number 45.The liner search algorithm will have examined all elements in the list (In our case, 15 elements)

Explain in detail about the Binary search algorithm.

Binary search is an algorithm designed to help us search for an element in a list. The binary search algorithm must be applied to an ordered list in a consistent order. Steps at the binary search algorithm.

Start at the Middle: The algorithm begins at the middle of the list. **Comparison**: If the target element is not found, it compares the target element with the middle element. If the target element is greater then the middle element, the algorithm creates a sub-list containing the elements from the middle element to the right of the current list. If the target element is less than the middle element, the algorithm creates a sub-list that contains the remaining elements from the middle element to the left of the current list. **Continuation**: The algorithm continues this process until the target element is found. **Return**: If the target element is not in the list, the algorithm returns 0, indicating that the target element was not found.

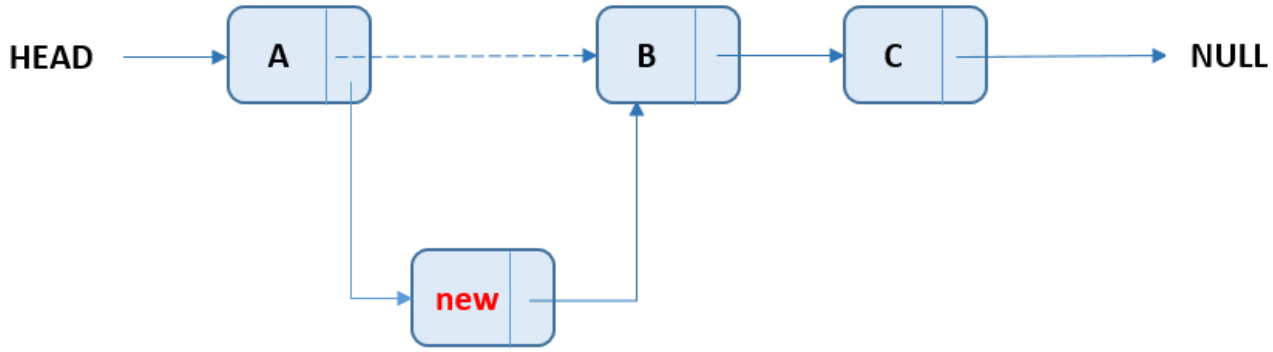
Given the following list of numbers: 1 4 7 8 12 15 16 17 21 22 25 27 43 52 53 57. How many elements of the list would be examined by the binary search algorithm to determine if each of the indicated target elements was on the list? 17, 15, 57, 45.

If the target element is 17, The binary search algorithm will have examined it only once. If the target element is 15, The binary search algorithm will have examined it 3 times. If the target element is 57, The binary search algorithm will have examined it 4 times. If the target element is 45, The binary search algorithm will have examined it only once.

What is a Dynamic Data Structure?

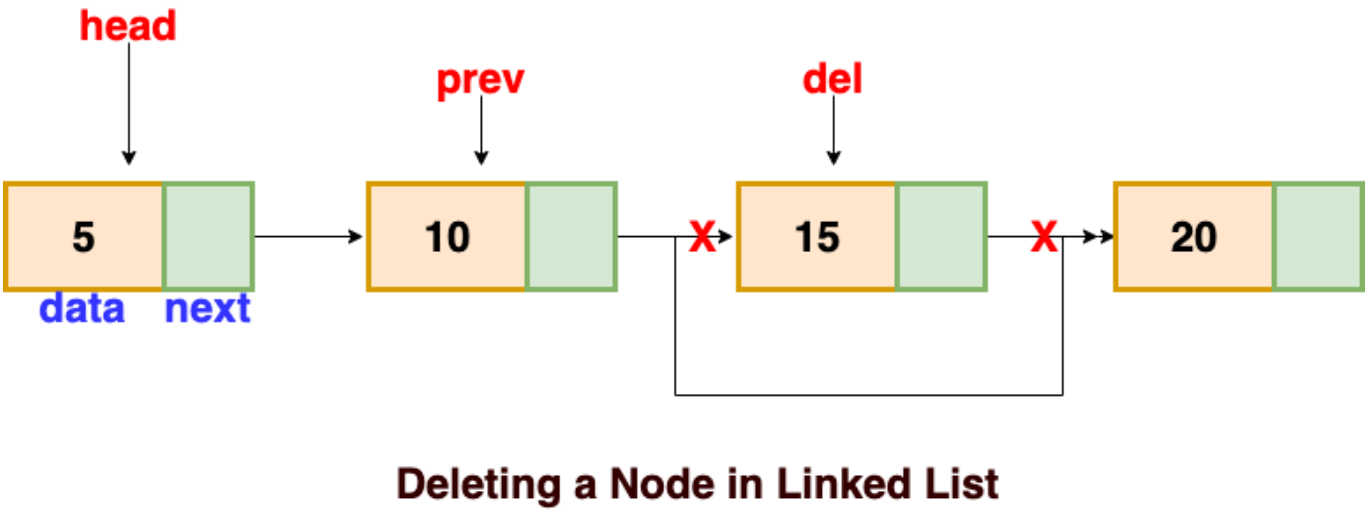
A dynamic data structure is a type of collection that can grow and shrink as needed. When you add a new object, the dynamic data structure doesn't behave like an array that copies everything into a new array whenever necessary. Instead, the dynamic data structure adjusts the collection accommodating the new objects without creating a new list. Another advantage of a dynamic data structure has over an array is that an array has a fixed size. This means that once an array is instantiated with a certain number of slots to hold objects, it cannot be changed later. In contrast, a dynamic data structure can be adjusted as needed.

Describe the Steps Depicted in the Image to Insert a Node into a List



The first step when inserting a node into a linked list is to insert it between two specific nodes (in most cases). Then, you update the next reference of the previous node to the newly node, and this newly node now references the next node that was previously referenced by the preceding node. Now, we have a sorted list of objects that form a linked list, where each node references to the next node, except for the last node, which references 'null' to indicate the end.

Describe the steps depicted in the image to delete a node from a list. What special cases exist?



Deleting a Node in Linked List

The initial step in deleting a node from a list is to locate the node immediately preceding the one to be deleted, as well as the node after the one to be deleted. Special cases occur when the node to be deleted is located the head or at the end of the list. In such cases, if the head of the list (node) is to be deleted, the reference to the next node will be updated. On the other hand, when the node at the end of the list is to be deleted, the reference to the rear node will be updated.

What is a doubly linked list?

A doubly linked list is a regular linked list with an additional reference in the object. Instead of having only one reference node to the next node, the object contains another reference to the previous node.

How is a queue different from a list?

The difference between a queue and a list is that a list can be managed from each index using methods provided by the Java API. In contrast, a queue is a list in approach of first in, first out(FIFO). This means that the first item put into the queue is the first to come out.

Show the contents of a queue after the following operations are performed. Assume the queue is initially empty. enqueue(5); enqueue(21); dequeue(); enqueue(72) enqueue(37); enqueue(15); dequeue();

The contents of the queue after the given operations are:

1. enqueue(5); 5
2. enqueue(21); 5, 21
3. dequeue(); 21
4. enqueue(72); 21, 72
5. enqueue(37); 21, 72, 37
6. enqueue(15); 21, 72, 37, 15
7. dequeue(); 72, 37, 15 Thus, the final contents of the queue are 72, 37, and 15.

What is a stack?

A stack is a linear data structure used to store and manage information in a last-in, first-out (LIFO) approach. This means that the last item put into the stack is the first item to come out. Imagine a stack as a high pile of books where each book is placed on top of the other. When you want to take out a book, you take the book from the top, and not from the bottom. This approach is called last-in, first-out (LIFO) in collections.

Show the contents of a stack after the following operations are performed. Assume a stack is initially empty. push(5); push(21); pop(); push(72); push(37); push(15); pop();

The contents of the stack after the given operations are:

1. push(5); 5
2. push(21); 5, 21
3. pop(); 5
4. push(72); 5, 72
5. push(37); 5, 72, 37
6. push(15); 5, 72, 37, 15
7. pop(); 5 ,72, 37 Thus, the final contents of the stack are 5, 72, 37

What do trees and graphs have in common?

Both trees and graphs are non-linear data structures, meaning that the data stored in them is not organized linearly. Trees create a hierarchy of nodes, while the nodes in a graph are connected using general edges.

Which structure (a tree or a graph) would be a good choice to represent each of the following: The directories and files on a computer system? Airplane route? An 'is a friend' relationship among a group of people? An 'is a buss' relationship in a company?

The appropriate structure (a tree || a graph) for each of the above are: **The directories and files on a computer system**- I think that the appropriate choice is a tree because each file is found on the path of the

primary directory (one only). **Airplane route**- A graph. **An 'is a friend' relationship among a group of people**- A graph. A person can be friends with a few people. A graph provides the opportunity to connect several edges (i.e., friends). **An 'is a buss' relationship in a company**- A tree. There is only one primary boss in a company. This boss has children (sub-bosses), who have their own children (employees), creating a hierarchy that corresponds to a tree more than a graph.

What is the Java collection API?

The Java Collection API is a standard class library that comprises several classes representing collections of various types. Examples of classes in this API include ArrayList, Set, and HashMap, among others. For further details, visit the official Java documentation.

Suppose `first` references to a `node` object, and that it refers to the first node in a linked list, Show, in pseudo-code, the steps that would count and return the number of nodes on the list.

```
/*
Set the `count` variable to count nodes
`current` is the first node

While `current` is not equal to null (the end of the list)
  Increment `count` by 1
  `Current` reference to the next node

return `count`
*/

set count = 0;
current = first;
while current != null;
  count++;
  current = current.next;
return count;
```

Suppose `current` is a reference to a `Node` object and that it currently refers to a specific node in a linked list. Show, in pseudo-code, the steps that would delete the node following `current` from the list. Carefully consider the cases in which `current` refers to the first and last nodes in the list.

```
/*
Set `nextItem` to the following reference after the `current` node

If the `nextItem` is not null
  If `nextItem` is not the last node
    Set `current.next` to `nextItem.next` to bypass `nextItem`
  Else
    Set `current.next` to null to indicate the end of the list
 */
if current.next != null {
  nextItem = current.next;
```

```
    if nextItem.next != null {
      current.next = nextItem.next;
    } else {
      current.next = null;
    }
  }
```

Modify your answer of the code fragment above and assume that the list was set up as a double-linked list. with both `next` and `prev` references.

```
if (current.next != null) {
  nextItem = current.next;
  list.delete(nextItem);

  // Adjust the `next` and `prev` references of neighboring nodes
  if (nextItem.next != null) {
    // If nextItem is not the last node
    nextItem.next.prev = current;
  }
  current.next = nextItem.next;
}
```