# Computer Basics Assignment

# Solution

Question 1: What is a computer?

Answer 1: A computer is a programmable electronic device that accepts raw data and as input and processes it with a set of instruction (a program) to produce the result as output.

- A computer is designed to execute applications and provides a variety of solutions through integrated hardware and software components. It works with the help of programs and represents the decimal numbers through a string of binary digits. It also has a memory that stores the data, programs, and result of processing. The components of a computer such as machinery that includes wires, transistors, circuits, hard disk are called hardware. Whereas, the programs and data are called software.

Question 2: What is RAM?

Answer 2: RAM (Random-access memory) is volatile memory. This means that information is kept in RAM while the computer is running, but it is erased when the machine is powered off.

- It is a hardware device generally located on the motherboard of a computer and acts as an internal memory of the CPU. It allows CPU store data, program, and program results when you switch on the computer. It is the read and write memory of a computer, which means the information can be written to it as well as read from it.
- RAM has no potential for storing permanent data due to its volatility. A hard drive can be compared to a person's long-term memory and RAM to their short-term memory. Short-term memory can only hold a limited number of facts in memory at any given time; however, it concentrates on immediate tasks. Facts kept in the brain's long-term memory can be used to replenish short-term memory.

Question 3: What is data stored in a computer?

Answer 3: Data stored in a computer refers to the information and instructions that are saved and kept in digital format within the computer's storage devices. This data can take various forms, including:

- **Text:** This includes documents, spreadsheets, and any form of written or typed information.

- **Images:** Digital photographs, graphics, and other visual data are stored in formats such as JPEG, PNG, or GIF.
- **Audio:** Music files, voice recordings, and other sound data are stored in formats like MP3, WAV, or FLAC.
- **Video:** Video files can be stored in formats like MP4, AVI, or MOV. These files contain moving images and often have accompanying audio.
- **Software:** Computer programs and applications are stored as data. This includes operating systems, word processors, video games, and more.
- **Databases:** Data stored in a structured manner, often used for organizing and retrieving information. These databases can contain customer records, inventory lists, and more.

Question 4: What is that input device used to type text and number on a document in the computer system?

Answer 4: The input device used to type text and numbers on a document in a computer system is called a "keyboard." A keyboard is a common and essential input device for computers and allows users to input alphanumeric characters, symbols, and various commands. Keyboards come in different layouts and designs, but most include a standard set of keys for letters, numbers, punctuation, and function keys.

- Keyboards are used for a wide range of tasks, including typing documents, entering data into spreadsheets, sending emails, programming, and interacting with various software applications. They are available in various forms, such as physical keyboards (wired or wireless) that connect to a computer via USB or Bluetooth, as well as virtual keyboards on touchscreens or virtual keyboard software on some devices.

Question 5: What are the output device?

Answer 5: An output device is any piece of computer hardware that converts information/Data into human perceptible form:

- **Monitor**
- **Printer**
- **Speaker**
- **Headphone**
- **Projector**
- **GPS device**
- **Optical mark reader**
- **3d painter**

Question 6: Which is the input device that allows a user to move the cursor or pointer on the screen?

Answer 6: The input device that allows a user to move the cursor or pointer on the screen are called **Pointing device**

- **Mouse:** A mouse is small handhold device that is moved across a flat surface to control the cursor on the screen
- **Track Pad:** A track pad is a flat touch-sensitive surface that is typically found on laptops. It is used to move cursor by sliding your fingers across it.
- **Track ball:** A track ball is a stationary pointing device that is rolled with the thumb to control the cursor on the screen
- **Joystick:** It is a pointing device that consists of a stick that can be moved in all directions. Joystick are often used for playing video games and controlling robots.
- **Light pen:** It is a pointing device that uses a light source to detect its position on the screen. Light pens are often used for drawing and edit images.

Question 7: Which language is directly understood by the computer without a translation program?

Answer 7: The language that is directly understood by the computer without a translation program is **Machine language**.

- Machine-low level programming language that is specific to a particular computer architecture. It consists of binary instructions which are strings of 0's and 1's. the computer's CPU (central processing unit), can directly execute machine language instructions.
- Machine language is very complex and difficult to program in directly that is why high-level language are used for most programming tasks. However, there are some cases where it is necessary to program in machine language such as when writing device or O.S
- Example: simple machine language instruction.
    - 10000000 00000000 00000001 00100011
    - This instruction tells the CPU to load the value 11 into register 1.

Question 8: What are the input devices?

Answer 8: An input device is a place of hardware a computer you to enter data into a computer system. Some common input devices are:

- **Keyboard**

- **Mouse**
- **Scanner**
- **Microphone**
- **Webcam**

# *Fundamental of Java Assignment Solution*

Question 1: What is Programming Language?

Answer 1: A programming language is a computer language that is used by programmers (developers) to communicate with computer. It is a set of instructions written in any specific language (C, C++, JAVA, Python) to perform a specific task.

- A programming language is mainly use to develop desktop applications, websites and mobile applications.
- Type of programming language:
    - Low level programming language
    - High level programming language
    - Middle level programming language
- **Low level programming language**: It is machine-dependent (0's & 1's) programming language. The processor runs low level programs directly without the need of a compiler or interpreter. So, the programs written in low level language can be run very fast
- **High level programing language**: It is designed for developing user-friendly software programs and websites. This programming requires a complier or interpreter to translate the program into machine language (execute the program). The main advantage of a high-level language is that it is easy to read, write and maintain. High-level programming level includes (Python, Java, JS, PHP, C#, C++, C, and more…).
- **Middle level language**: It lies between the low-level programming language and high-level programming. It is also known as the intermediate programming language and pseudo language. A middle-level programming language advantage are that it supports the feature of high-level programming, it is a user-friendly language and closely related to machine language and human language.

Question 2: Why do we need a programming language?

Answer2: Computers are incredibly powerful machines but they can only understand and follow instructions that are written in specific language programming language allow us to write these instructions in away that computers can understood.

- Programming language are also used in other fields, such as science, engineering, and mathematics.
- Example: programming language can be used to: -

    - Develop algorithms for solving compare problem.
    - Automate task.
    - Control robots and other machines.
- Programming language are an essential tool in the machine world. They are used by people of all ages and back-ground to create new things and solve problems. In short, we need programming language because they allow us to harness the power of computer to sole problem and create new things.

Question 3: What are the features of java?

Answer 3:  The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java buzzwords.

A list of the most important features of the Java language is given below.

- **Simple**
- **Object-Oriented**
- **Portable or Portability**
- **Platform independent**
- **Secured or Security**
- **Robust and Reliable**
- **Architecture neutral**
- **Interpreted**
- **High Performance**
- **Multithreaded**
- **Distributed**
- **Dynamic**
    - **Platform Independence:** Java is designed to be platform-independent, which means that Java programs can run on any platform with a Java Virtual Machine (JVM). This "Write Once, Run Anywhere" feature is achieved through bytecode compilation.
    - **Object-Oriented:** Java is an object-oriented programming (OOP) language, which encourages the use of classes and objects for organizing and structuring code.
    - **Simple and Easy to Learn:** Java was designed to be simple and easy to understand, with a clear and straightforward syntax that makes it accessible for beginners.

- **Robust and Reliable:** Java includes features like strong typing, automatic memory management (garbage collection), and exception handling, which contribute to its robustness and reliability.
- **Multi-threading:** Java provides built-in support for multithreading, allowing developers to create applications that can perform multiple tasks concurrently.
- **Security:** Java has a strong security model with features like a bytecode verifier, runtime security checks, and a security manager, making it suitable for building secure applications.
- **Portability:** Java applications can run on various platforms with minimal modifications, which is crucial for cross-platform development.
- **High Performance:** Java has evolved over the years to offer high-performance execution through Just-In-Time (JIT) compilation and optimization by the JVM.
- **Automatic Memory Management:** Java uses a garbage collector to automatically manage memory, which reduces the likelihood of memory leaks and makes memory management less error-prone.
- **Dynamic:** Java supports dynamic loading of classes, which allows the addition of new classes and resources at runtime.
- **Community and Ecosystem:** Java has a large and active community of developers, along with a vast ecosystem of libraries, frameworks, and tools that support various development needs.
- **Integration:** Java can be easily integrated with other languages, platforms, and technologies through its various APIs and interfaces.
- **Open Source:** While Java itself is governed by Oracle, many components of the Java ecosystem are open source, and open-source Java development frameworks and libraries are widely available.
- **Multi-paradigm:** Although primarily an object-oriented language, Java supports multiple programming paradigms, including procedural and functional programming.
- **Scalability:** Java is used in a wide range of applications, from small mobile apps to large enterprise systems, making it suitable for projects of different scales.

Question 4: What is an object?

Answer 4: An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

An object has serval characteristics:

- **State:** An object's state is determined by the values of its fields at a particular point in time. The state can change as the object's methods are called and as it interacts with other objects.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.
- **Methods:** Objects also have methods, which are functions or procedures that define the behavior or actions that the object can perform. These methods can operate on the object's data and may interact with other objects.
- **Properties (Fields):** Objects have properties, also known as fields or attributes, which represent the data associated with the object. These fields can be variables of different data types, including primitives and other objects.

Question 5: What is class?

Answer 5: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- **Fields:** Classes can have fields, also known as properties or attributes, which represent the data associated with objects of that class. Fields are defined as variables and can have different data types
- **Methods:** Classes contain methods, which are functions or procedures that define the behavior and actions that objects of the class can perform. Methods can operate on the class's fields and may interact with other objects.
- **Constructors:** Classes often include one or more constructors. A constructor is a special method used for initializing objects when they are created. It sets the initial state of the object by assigning values to its fields.
- **Blocks:** class blocks are sections within a class where you can define various elements such as fields, methods, constructors, and static initialization blocks. These blocks help organize the code and define the behavior and structure of the class
- **Nested class and interface:** A nested class is a class that is defined within another class (the enclosing class). It can be of any access modifier, such as public, private, protected, or package-private (default). Nested classes are often used to logically group classes that are only used within the context of the enclosing class. A nested interface is an interface that is defined within another class or interface (the enclosing class or interface). It can

be declared as public, private, or package-private. Nested interfaces are typically used to declare a contract that is closely related to the enclosing class or interface.

- **Syntax to declare a class:**

```
class <class_name>{
field;
method;
}
```

**Question 6:** Explain about the main () method in java?

**Answer 6**: In Java programs, the point from where the program starts its execution or simply the entry point of Java programs is the main () method. Hence, it is one of the most important methods of Java, and having a proper understanding of it is very important.

- The Java compiler or JVM looks for the main method when it starts executing a Java program. The signature of the main method needs to be in a specific way for the JVM to recognize that method as its entry point. If we change the signature of the method, the program compiles but does not execute.
- The execution of the Java program, the java.exe is called. The Java.exe in turn makes Java Native Interface or JNI calls, and they load the JVM. The java.exe parses the command line, generates a new String array, and invokes the main () method. A daemon thread is attached to the main method, and this thread gets destroyed only when the Java program stops execution.
- Example:

```
// Java Program to demonstrate the
// syntax of the main() function
class Demo{
    public static void main(String[] args)
    {
        System.out.println("Hello world");
    }
}
```

- **Public**: It is an *Access modifier*, which specifies from where and who can access the method. Making the *main()* method public makes it globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class.
- **Static:** It is a keyword that is when associated with a method, making it a class-related method. The main() method is static so that JVM can invoke it without instantiating the class. This also saves the unnecessary wastage of memory which would have been used by the object declared only for calling the main() method by the JVM.
- **Void:** It is a keyword and is used to specify that a method doesn't return anything. As the main() method doesn't return anything, its return type is void. As soon as the main() method terminates, the java program terminates too. Hence, it doesn't make any sense to return from the main() method as JVM can't do anything with the return value of it.
- **main:** It is the name of the Java main method. It is the identifier that the JVM looks for as the starting point of the java program. It's not a keyword.

# *Java Variables and Data Types Assignment Solution*

**Question 1**: What is statically typed and dynamically typed programming language?

**Answer 1:** Statically typed and dynamically typed are two different approaches to type systems in programming languages. These approaches determine how and when data types are checked in a program. Here's an explanation of each:

**Statically Typed Programming Language:**

- In a statically typed language, the data types of variables are known at compile time. You must declare the type of a variable when you define it, and the type is fixed throughout the variable's lifetime.
- Type checking occurs at compile time, and any type errors are caught and reported by the compiler before the program is executed. This means that you need to ensure that all variables are used in a manner consistent with their declared types.
- Static typing can help catch type-related errors early in the development process, making programs more robust and less error-prone.
- Examples of statically typed languages include Java, C, C++, and Rust.

**Dynamically Typed Programming Language:**

- In a dynamically typed language, variable types are determined and checked at runtime. You do not need to declare the type of a variable explicitly when defining it, and a variable can change its type during the program's execution.
- Type checking occurs as the program runs, and type errors are only discovered when the code is executed. This flexibility allows you to write code more quickly but can lead to runtime errors if type mismatches occur.
- Dynamically typed languages provide more flexibility but may require extra testing to catch type-related issues.
- Examples of dynamically typed languages include Python, JavaScript, Ruby, and PHP.

**Question 2**: What is variable in Java?

**Answer 2**: In Java, a variable is a container or storage location that holds data, and it is used to store, manipulate, and retrieve values within a program. Variables are essential in programming because they allow you to work with data and perform operations on it. Here are some key aspects of variables in Java:

- Declaration: Before you can use a variable, you need to declare it. This involves specifying the data type of the variable and giving it a name. For example:
    - Example:

        - int age; // Declaring an integer variable named 'age'
- Data Type: Java is a statically typed language, which means you must specify the data type when declaring a variable. Common data types in Java include `int` (for integers), `double` (for floating-point numbers), `String` (for text), and many more.
- Initialization: Variables can be initialized when declared, which means you assign an initial value to the variable. If not initialized, variables have a default initial value.
    - Example:
        - int count = 5; // Initializing 'count' to 5
- Assignment: You can change the value of a variable by assigning it a new value using the assignment operator (`=`).
    - Example:
        - age = 30; // Assigning a new value to 'age'
- Scope: Variables have a scope, which defines where in the code the variable is accessible. In Java, variables can have block scope, method scope, or class scope, depending on where they are declared.
- Lifetime: The lifetime of a variable is determined by its scope. When a variable goes out of scope, its memory is reclaimed, and it cannot be accessed.
- Final Variables: You can declare a variable as `final` to make it a constant, meaning its value cannot be changed after initialization.
    - Example:
        - final double PI = 3.14159; // A constant variable
- Static Variables: Variables can be declared as `static` within a class. These variables belong to the class rather than to any particular instance of the class.
- Instance Variables: These are non-static variables that belong to a specific instance (object) of a class.
- Local Variables: Variables declared within a method or a block have local scope and are only accessible within that method or block.

**Question 3**: How to assign a value to variable?

**Answer 3**: In Java, you can assign a value to a variable using the assignment operator `=`. The assignment operator is used to store a value in a variable. Here's how you assign a value to a variable:

- Declaration and Initialization in One Step: To declare and initialize a variable in one step, you specify the data type, followed by the variable name, the assignment operator, and the initial value. For example:
    - Example:
        - int age = 30; // Declare 'age' as an integer and initialize it with the value 30.
- Initialization After Declaration: You can declare a variable first and then assign an initial value later in your code. For this, you declare the variable with the data type only, followed by the assignment statement at a later point in the code.

    - Example:
        - int height; // Declare 'height' as an integer
        - height = 175; // Assign the value 175 to 'height' later in the code.
- Reassignment: You can change the value of a variable by reassigning it with a new value using the assignment operator. This is common when you need to update the value of a variable during the execution of your program.
    - Example:
        - int count = 5; // Initialize 'count' with 5
        - count = 10; // Reassign 'count' with the value 10
- Initialization with Expressions: You can initialize a variable with the result of an expression or computation.
    - Example:
        - int sum = 2 + 3; // Initialize 'sum' with the result of the expression 2 + 3.
- Constants: If you want to create a constant variable whose value should not change, you can use the `final` keyword to declare it as a constant. The value of a constant variable is typically assigned when it is declared and cannot be changed thereafter.
    - Example:
        - final double PI = 3.14159; // Declare 'PI' as a constant with an initial value.
- It's important to declare a variable before you can assign a value to it, and the data type of the variable should match the type of the value you are assigning. This ensures that you are working with compatible data types.
- Here's an example of variable assignment:

```
public class VariableAssignmentExample {
    public static void main(String[] args) {
        int x; // Declaration
        x = 5; // Assignment
        System.out.println("x: " + x); // Output: x: 5
        // Reassignment
        x = 10;
        System.out.println("x: " + x); // Output: x: 10
    }
}
```

- In this example, we declare the variable `x`, assign it the value 5, and then reassign it the value 10, printing the result to the console.

**Question 4**: What are primitive data types in java?

**Answer 4:** The primitive data types are the predefined data types of Java. They specify the size and type of any standard values. Java has 8 primitive data types namely byte, short, int, long, float, double, char and Boolean. When a primitive data type is stored, it is the stack that the

values will be assigned. When a variable is copied then another copy of the variable is created and changes made to the copied variable will not reflect changes in the original variable. Here is a Java program to demonstrate all the primitive data types in Java.

- Integer: This group includes byte, short, int, long
    - byte: It is 1 byte(8-bits) integer data type. Value ranges from -128 to 127. Default value zero. example: byte b=10;
    - short: It is 2 bytes(16-bits) integer data type. Value ranges from -32768 to 32767. Default value zero. example: short s=11;
    - int: It is 4 bytes(32-bits) integer data type. Value ranges from -2147483648 to 2147483647. Default value zero. example: int i=10;
    - long: It is 8 bytes(64-bits) integer data type. Value ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. example: long l=100012;
- Floating-Point Number: This group includes float, double
    - float: It is 4 bytes(32-bits) float data type. Default value 0.0f. example: float ff=10.3f;
    - double: It is 8 bytes(64-bits) float data type. Default value 0.0d. example: double db =11.123;
- Characters: This group represent char, which represent symbols in a character set, like letters and numbers.
    - char: It is 2 bytes(16-bits) unsigned Unicode character. Range 0 to 65,535.
        - example: char c='a';
- Boolean: Boolean type is used when we want to test a particular condition during the execution of the program. There are only two values that a Boolean type can take: true or false.
    - Remember, both these words have been declared as keyword. Boolean type is denoted by the keyword Boolean and uses only 1 bit of storage.

**Question 5**: What are the identifiers in java?

**Answer 5:** In Java, an identifier is a name given to various program elements, such as classes, methods, variables, and packages. Identifiers are used to uniquely identify and reference these elements within a program. Here are the rules and conventions for identifiers in Java:

- Rules for Java Identifiers:
    - Character Set: Identifiers must start with a letter (A to Z or a to z), a dollar sign (`$`), or an underscore (`_`). Subsequent characters can be letters, digits (0 to 9), dollar signs, or underscores.
    - Length: Identifiers can be of any length, but it's a good practice to keep them meaningful and reasonably short.
    - Case Sensitivity: Java is case-sensitive, so `myVar`, `myvar`, and `MyVar` are considered distinct identifiers.

    - Reserved Words: You cannot use reserved words (also known as keywords) as identifiers. Examples of reserved words in Java include `int`, `public`, `class`, and `if`. You can't use these words as variable or class names.
- Conventions for Java Identifiers: While the Java language does not enforce specific naming conventions for identifiers, there are widely accepted conventions to make code more readable and maintainable. These conventions are not required but are considered good practice:
    - Camel Case: Use camel case for variable and method names. In camel case, the first letter of the identifier starts with a lowercase letter, and each subsequent word within the identifier starts with an uppercase letter. For example: `myVariable`, `calculateInterestRate()`
    - Pascal Case: Use Pascal case for class and interface names. In Pascal case, the first letter of each word within the identifier starts with an uppercase letter. For example: `MyClass`, `MyInterface`
    - UPPER_CASE_WITH_UNDERSCORES: Use all uppercase letters with underscores to represent constants. For example: `MAX_VALUE`, `PI`, `HTTP_NOT_FOUND`
    - Packages: Package names are typically in lowercase. For example: `com.example.myproject`
    - Avoid Single-Letter Names: Avoid using single-letter variable names (e.g., `int x`) unless the variable's scope is very limited and its purpose is immediately obvious.
    - Descriptive Names: Choose meaningful and descriptive names for variables, methods, classes, and packages. This makes your code more self-documenting.
- Here are examples of valid identifiers:

```
int myVariable;
String firstName;
MyClass myObject;
MAX_SIZE
calculateTotalAmount
```

- And here are examples of invalid identifiers:

```
3rdPlace // Starts with a digit
$amount // Starts with a special character
class // A reserved word
My Variable // Contains a space
my-variable // Contains a hyphen (hyphens are not allowed)
```

- Using meaningful and consistent naming conventions for identifiers is essential for writing clean, readable, and maintainable code. It helps you and other developers understand the purpose of each identifier and makes your code more robust and less error-prone.

**Question 6:** List of operators in java?

**Answer 6:** Operators in Java are the symbols used for performing specific operations in Java. Operators make tasks like addition, multiplication, etc which look easy although the implementation of these tasks is quite complex. Types of Operators in Java

- There are multiple types of operators in Java all are mentioned below:
    - Arithmetic Operators
    - Unary Operators
    - Assignment Operator
    - Relational Operators
    - Logical Operators
    - Ternary Operator
    - Bitwise Operators
    - Shift Operators
    - instance of operator
- Arithmetic Operators: They are used to perform simple arithmetic operations on primitive data types.
    - * : Multiplication
    - / : Division
    - % : Modulo
    - + : Addition
    - – : Subtraction
- Unary operators : Unary operators need only one operand. They are used to increment, decrement, or negate a value.
    - – : Unary minus, used for negating the values.
    - + : Unary plus indicates the positive value (numbers are positive without this, however). It performs an automatic conversion to int when the type of its operand is the byte, char, or short. This is called unary numeric promotion.
    - ++ : Increment operator, used for incrementing the value by 1. There are two varieties of increment operators.
        - Post-Increment: Value is first used for computing the result and then incremented.
        - Pre-Increment: Value is incremented first, and then the result is computed.
    - –– : Decrement operator, used for decrementing the value by 1. There are two varieties of decrement operators.
        - Post-decrement: Value is first used for computing the result and then decremented.
        - Pre-Decrement: The value is decremented first, and then the result is computed.
    - ! : Logical not operator, used for inverting a Boolean value.

- Assignment Operator: '=' Assignment operator is used to assign a value to any variable. It has right-to-left associativity, i.e., value given on the right-hand side of the operator is assigned to the variable on the left, and therefore right-hand side value must be declared before using it or should be a constant.
    - +=, for adding the left operand with the right operand and then assigning it to the variable on the left.
    - -=, for subtracting the right operand from the left operand and then assigning it to the variable on the left.
    - *=, for multiplying the left operand with the right operand and then assigning it to the variable on the left.
    - /=, for dividing the left operand by the right operand and then assigning it to the variable on the left.
    - %=, for assigning the modulo of the left operand by the right operand and then assigning it to the variable on the left.
- Relational Operators: These operators are used to check for relations like equality, greater than, and less than. They return Boolean results after the comparison and are extensively used in looping statements as well as conditional if-else statements.
    - Some of the relational operators are-
        - ==, Equal to returns true if the left-hand side is equal to the right-hand side.
        - !=, Not Equal to returns true if the left-hand side is not equal to the right-hand side.
        - <, less than: returns true if the left-hand side is less than the right-hand side.
        - <=, less than or equal to returns true if the left-hand side is less than or equal to the right-hand side.
        - >, Greater than: returns true if the left-hand side is greater than the right-hand side.
        - >=, Greater than or equal to returns true if the left-hand side is greater than or equal to the right-hand side.
- Logical Operators: These operators are used to perform "logical AND" and "logical OR" operations, i.e., a function similar to AND gate and OR gate in digital electronics. One thing to keep in mind is the second condition is not evaluated if the first one is false, i.e., it has a short-circuiting effect. Used extensively to test for several conditions for making a decision. Java also has "Logical NOT", which returns true when the condition is false and vice-versa
    - Conditional operators are:
        - &&, Logical AND: returns true when both conditions are true.
        - ||, Logical OR: returns true if at least one condition is true.
        - !, Logical NOT: returns true when a condition is false and vice-versa
- Ternary operator: The ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name Ternary.

    - The general format is:
        - condition ? if true : if false
        - The above statement means that if the condition evaluates to true, then execute the statements after the '?' else execute the statements after the ':'.
- Bitwise Operators: These operators are used to perform the manipulation of individual bits of a number. They can be used with any of the integer types. They are used when performing update and query operations of the Binary indexed trees.
    - &, Bitwise AND operator: returns bit by bit AND of input values.
    - |, Bitwise OR operator: returns bit by bit OR of input values.
    - ^, Bitwise XOR operator: returns bit-by-bit XOR of input values.
    - ~, Bitwise Complement Operator: This is a unary operator which returns the one's complement representation of the input value, i.e., with all bits inverted
- Shift Operators: These operators are used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively. They can be used when we have to multiply or divide a number by two.
    - General format-
        - number shift_op number_of_places_to_shift;
    - <<, Left shift operator: shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as multiplying the number with some power of two.
    - >>, Signed Right shift operator: shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of the initial number. Similar effect to dividing the number with some power of two.
    - >>>, Unsigned Right shift operator: shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0
- instanceof operator: The instance of the operator is used for type checking. It can be used to test if an object is an instance of a class, a subclass, or an interface.
    - General format-
        - Object instance of class/subclass/interface

**Question 7:** Explain about increment and decrement operators and give an example.

**Answer 7:** In Java, the increment and decrement operators (`++` and `--`) are used to increase or decrease the value of a variable by 1. These operators are often used in loops, conditional statements, and other parts of your code to modify variables and control program flow. They come in two forms: prefix and postfix, and the choice between them can affect the order of execution. Here an explanation of both the increment and decrement operators and examples for each:

- Increment Operator (`++`):
    - The increment operator `++` is used to increase the value of a variable by 1.

    - It can be used in both prefix and postfix forms.
    - In the prefix form (`++variable`), the value is increased before the variable's current value is used in an expression.
    - In the postfix form (`variable++`), the value is increased after the current value is used in an expression.
- Decrement Operator (`--`):
    - The decrement operator `--` is used to decrease the value of a variable by 1.
    - Like the increment operator, it can also be used in both prefix and postfix forms.
- Examples:

```
int x = 5;
int y = 10;
// Prefix increment: Increase x by 1 and then use its new value.
int result1 = ++x; // result1 is 6, x is now 6
// Postfix increment: Use the current value of y in an expression and then increase it by 1.
int result2 = y++; // result2 is 10 (y's current value), y is now 11
System.out.println("Prefix Increment - x: " + x + ", result1: " + result1);
System.out.println("Postfix Increment - y: " + y + ", result2: " + result2);
// Decrement examples work similarly:
int a = 8;
int b = 15;
int result3 = --a; // Prefix decrement: a is now 7, result3 is 7
int result4 = b--; // Postfix decrement: result4 is 15 (b's current value), b is now 14
System.out.println("Prefix Decrement - a: " + a + ", result3: " + result3);
System.out.println("Postfix Decrement - b: " + b + ", result4: " + result4);
```

- In these examples, the increment and decrement operators are used to modify the values of variables `x`, `y`, `a`, and `b`. The choice between the prefix and postfix forms depends on whether you want the increment or decrement to occur before or after the current value is used in an expression.