**What is java?**

Java is a high-level, object-oriented programming language developed by Sun Microsystems in the mid-1990s, now owned by Oracle Corporation.

Java is a platform because java has a runtime environment (JRE) and API.

**Who invented Java?**

James Gosling, Mike Sheridan,Patrick Naughton initiated Java language project in June1991.

JDK 1.0 was released on January 23, 1996.

Before Java, its name was Oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

**What are java features?**

Java is strong object-oriented programming language.

Java is platform independent write once, run anywhere.

Java is multi-threaded because it deals with many tasks at once.

Java is robust. It has exception handling and the type checking mechanism.

**What is JDK?**

The JDK is a software development kit that provides tools needed to develop Java applications, including the Java compiler (javac), the Java Runtime Environment (JRE), and various development tools like the debugger and profiler.

**What is JRE?**

The JRE is a subset of the JDK and includes the JVM and standard libraries. It provides the runtime environment necessary to execute Java applications.

**What is JVM?**

JVM executes Java bytecode, which is generated by the Java compiler from source code. Bytecode is platform-independent, meaning it can run on any system if it has a JVM.

The JVM manages memory. It does garbage collection. It automatically handles the allocation and deallocation of memory.

JVM provides security. The class loader loads classes into the JVM, and the bytecode verifier checks the code for illegal code that could violate access rights to objects.

JVM Interprets bytecode one instruction at a time by using interpreter.

JVM converts bytecode into native machine code at runtime for better performance by using JIT compiler.

**Explain JVM memory structure?**

Method area stores class-level data, information about methods, Static fields of classes, literals, bytecode, access flags, etc.

Heap area stores instances of classes(objects), arrays, primitive types and reference types.

Stack memory area stores local Variables defined within the method and intermediate calculations and results of operations.

Program counter register stores the address of the current instruction being executed by the thread. Each thread has its own PC register, which points to the next instruction to be executed in the thread’s method.

Native method stack used for the execution of native (non-Java) methods.

**Explain Garbage collection?**

Garbage collection is the process of reclaims memory by identifying and disposing of objects that are no longer referenced. It is done by JVM.

Marking, sweeping and compacting are involved in Garbage collection.

Garbage collection is the process by which the JVM reclaims memory by identifying and disposing of objects that are no longer referenced. The garbage collection process typically involves:

Marking is the process of identifying which objects are still in use.

Sweeping is reclaiming memory occupied by objects that are no longer in use.

Compacting is reorganizing the remaining objects to reduce fragmentation.

**What happens at compile time?**

At compile time, the Java file is compiled by Java Compiler (It does not interact with OS) and converts the Java code into bytecode. It is done under JVM.

**What happens at runtime?**

Class loader is the subsystem of JVM that is used to load class files.

The Bytecode Verifier Checks the code fragments for illegal code that can violate access rights to objects.

The Interpreter reads bytecode stream then execute the instructions by using hardware.

**What is data?**

Data refers to information that is stored and manipulated within a computer system. In programming data can be numeric, character, string or Boolean.

**Explain literals, variables and constants?**

A literal is a fixed value that appears directly in the source code.

Literals represent constant values of primitive data types or strings.

We have integer literals (10,123,456 etc.), Floating-point literals (3.14, 1.5e2 etc.), Character literals ('A', '\n' etc.), String literals (Hello, World etc.), Boolean literals (true, false etc.), Null literal(null).

A variable is a named memory location that holds a value that can change during the execution of a program.

Variables must be declared with a specific data type ( int, double, char, String).

Variable names should be meaningful and follow camelCase naming convention.

Variables can have different scopes (local, instance, class).

There are 3 types of variables like Local Variables, Instance Variables, Class Variables (Static Variables).

Local variables are declared inside a method and accessible only within that method.

Instance variables are declared inside a class but outside any method and accessible by all methods in the class.

Class Variables (Static Variables) are declared with the static keyword and shared among all instances of the class.

A constant in Java is a variable whose value cannot change once it has been assigned. Constants are typically declared using the final keyword.

Constants are immutable once a constant is assigned a value, it cannot be changed.

By convention, constants are named using uppercase letters with underscores separating words.

**What are data types?**

Data type specifies the size and type of value stored in the variable.

Data types are divided into two main categories, primitive data types and reference data types.

**Primitive Data Types:**

Primitive data types are fundamental data types which are not derived from others. byte, short, int, long, float, double, char, boolean are primitive.

byte data type size 8 bit, range -128 to 127 used for small integers.

short data type size 16 bit, range -32,768 to 32,767 can hold medium integers.

int data type size 32 bit, range -2^31 to 2^31-1 can hold large integers.

long data type size 64 bit, range -2^63 to 2^63-1 can hold wider range integers.

float data type size 32 bit, range 6 to 7 decimal digits and can hold decimals.

double data type size 64 bit, range 15 decimal digits, can hold large decimals.

boolean data type size 1 bit, can hold either true or false used for flags.

Char data type size 16 bit used to store single character.

**Reference Data Types:**

Reference data types in Java include classes, interfaces, arrays, and enums.

Unlike primitive types, reference types store references to the actual data.

**What is type casting in java?**

Type casting refers to the process of converting a variable from one data type to another.

There are two types of type casting in Java Implicit Type Casting (Widening) and Explicit Type Casting (Narrowing).

Implicit type casting, also known as widening or automatic type conversion, occurs when Java automatically converts a smaller data type to a larger data type. This conversion does not result in loss of data.

Java automatically promotes smaller data types (e.g., int) to larger ones (e.g., long, double) to avoid loss of data.

When converting larger data types to smaller ones, explicit casting is required. It may result in data loss due to truncation or overflow.

Explicit type casting, also known as narrowing, requires a manual intervention where the programmer explicitly specifies the target data type. This is necessary when converting a larger data type to a smaller data type, potentially resulting in loss of data.

Syntax: dataType variableName = (dataType) expression;

Not all data types can be cast into each other.

**What are operators in java?**

Arithmetic operators are used to perform basic mathematical operations.

Addition (+): Adds two operands.

Subtraction (-): Subtracts the second operand from the first.

Multiplication (\*): Multiplies two operands.

Division (/): Divides the first operand by the second.

Modulus (%): Returns the remainder of the division of the first operand by the second.

Assignment operators are used to assign values to variables.

Assignment (=): Assigns the value on the right to the variable on the left.

Compound Assignment (+=, -=, \*=, /=, %=): Performs operation and assigns result to left operand.

Comparison operators are used to compare two values.

Equal to (==): Checks if two operands are equal.

Not equal to (!=): Checks if two operands are not equal.

Greater than (>): Checks if the left operand is greater than the right.

Less than (<): Checks if the left operand is less than the right.

Greater than or equal to (>=): Checks if the left operand is greater than or equal to the right.

Less than or equal to (<=): Checks if the left operand is less than or equal to the right.

Logical operators are used to perform logical operations.

Logical AND (&&): Returns true if both operands are true.

Logical OR (||): Returns true if at least one operand is true.

Logical NOT (!): Reverses the logical state of its operand.

Bitwise operators perform operations at the bit level.

Bitwise AND (&): Performs bitwise AND operation.

Bitwise OR (|): Performs bitwise OR operation.

Bitwise XOR (^): Performs bitwise XOR (exclusive OR) operation.

Bitwise NOT (~): Inverts all the bits of the operand.

Left Shift (<<): Shifts bits to the left by specified number of positions.

Right Shift (>>): Shifts bits to the right by specified number of positions.

Unary operators operate on a single operand.

Unary Plus (+): Indicates positive value (usually redundant).

Unary Minus (-): Negates the value of its operand.

Increment (++): Increases the value of operand by 1.

Decrement (--): Decreases the value of operand by 1.

The ternary operator (?:) is a shorthand for if-else statement.

**Explain java statements?**

Statements are individual instructions that together form a complete unit of execution within a program.

Java programs are composed of one or more statements, which are executed sequentially unless control flow statements (like loops or conditionals) alter the normal flow.

Expressions can be variable assignments, method calls, or any combination of operands and operators that result in a value.

Declarations introduce new variables into the program, specifying their name and type. Declaration statements declare a variable and optionally assign it an initial value using the = operator.

Control flow statements alter the normal flow of program execution based on certain conditions.

Control flow statements are classified into conditional statements, switch statements, loop statements, break and continue statements.

Conditional Statements (if, else, else if) executes a block of code based on a condition.

Switch Statement evaluates an expression against multiple possible cases and executes the corresponding block of code.

Loop Statements (for, while, do-while) repeats a block of code multiple times until a condition is met.

Jump statements transfer control to another part of the program. Break, Continue, Return and Throw are treated as jump statements.

Break statement exits the current loop or switch statement.

Continue statement skips the current iteration of a loop and proceeds with the next iteration.

Return statement exits from the current method and optionally returns a value to the caller.

Throw statement used to throw an exception explicitly within a method.

Block statements (or compound statements) are enclosed within curly braces {} and contain a group of zero or more statements. They are used to group statements together, such as in methods, loops, conditional statements, and static initializers.

**Explain arrays in java?**

Arrays are data structures that allow you to store multiple values of the same type in a single variable.

Arrays can also hold objects. For example, an array of String objects.

Array elements are accessed using their index the first element is at index 0.

The length of an array can be obtained using the length property.

An array declaration specifies the data type of the elements that the array w.ill hold and the array name. The declaration does not allocate memory for the array. Syntax is: dataType[] arrayName;

To allocate memory for an array, you use the new keyword, specifying the size (number of elements). Syntax is: arrayName = new dataType[size];

Alternatively, you can declare, create, and initialize an array in one statement.

Example: int[] myArray = new int[5];

You can also use an array initializer to create and initialize an array.

int[] myArray = {1, 2, 3, 4, 5};

You can use loops to iterate over array elements.

Java supports multidimensional arrays, which are arrays of arrays.

You can use System.arraycopy() to copy elements from one array to another.

The Arrays.sort() method can be used to sort array elements.

The Arrays.binarySearch() method can be used to search for elements in a sorted array.

**Explain java strings?**

In Java, strings are objects that represent sequences of characters.

The String class is used to create and manipulate strings.

Strings in Java are immutable, meaning once a String object is created, its value cannot be changed.

Once a String object is created, its value cannot be modified. Any modification to a string creates a new String object.

Java maintains a pool of strings for optimization. When a string literal is created, the JVM checks if it already exists in the pool. If it does, a reference to the existing string is returned; otherwise, a new string is added to the pool.

There are several ways to create strings in Java. The most common way to create a string is String literal. For example, String str1 = "Hello, World!";

String literals are stored in the string pool.

Using the new Keyword we create a new String object in memory, not in the string pool. For example, String str2 = new String("Hello, World!");

The String class provides a number of methods to manipulate and inspect strings.

String methods:

Length: Returns the number of characters in the string.

Char At: Returns the character at the specified index.

Substring: Extracts a substring from the string.

Contains: Checks if the string contains a specified sequence of characters.

Index Of: Returns the index of the first occurrence of the specified character or substring.

To Upper/Lower Case: Converts all characters in the string to upper or lower case.

Trim: Removes whitespace from both ends of the string.

Replace: Replaces occurrences of a character or substring with another character or substring.

Equals: Checks if two strings are equal.

Equals Ignore Case: Checks if two strings are equal, ignoring case differences.

Compare To: Compares two strings lexicographically.

Strings can be concatenated using the + operator or the concat method.

**What is string pool?**

Java maintains a pool of strings for optimization. When a string literal is created, the JVM checks if it already exists in the pool. If it does, a reference to the existing string is returned; otherwise, a new string is added to the pool.

**Explain String builder and String buffer?**

For mutable strings, Java provides the StringBuilder and StringBuffer classes.

They allow you to create and manipulate strings without creating new objects for every modification.

StringBuilder is not thread-safe but faster.

StringBuffer is Thread-safe but slower due to synchronization.

**Explain java enum?**

In Java, an enum (short for "enumeration") is a special data type that enables a variable to be a set of predefined constants.

It is used to define collections of constants that can be assigned to a variable, making your code more readable and reducing the chance of errors caused by invalid values.

Enums are particularly useful for representing fixed sets of related items, such as days of the week, states in a process, or types of user roles.

Define an enum using the enum keyword followed by the name of the enum and a list of constants.

Example:

public enum Day {

SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY

}

Once an enum is defined, you can create variables of that type and assign one of the predefined constants to it.

Enums can have fields, methods, and constructors, making them more powerful and flexible. The constructor for an enum is implicitly private.

Enums can also define abstract methods, which each enum constant must implement.

Enums come with several useful methods:

values(): Returns an array of all enum constants.

valueOf(String name): Returns the enum constant with the specified name.

ordinal(): Returns the position of the enum constant in its declaration, starting from 0.