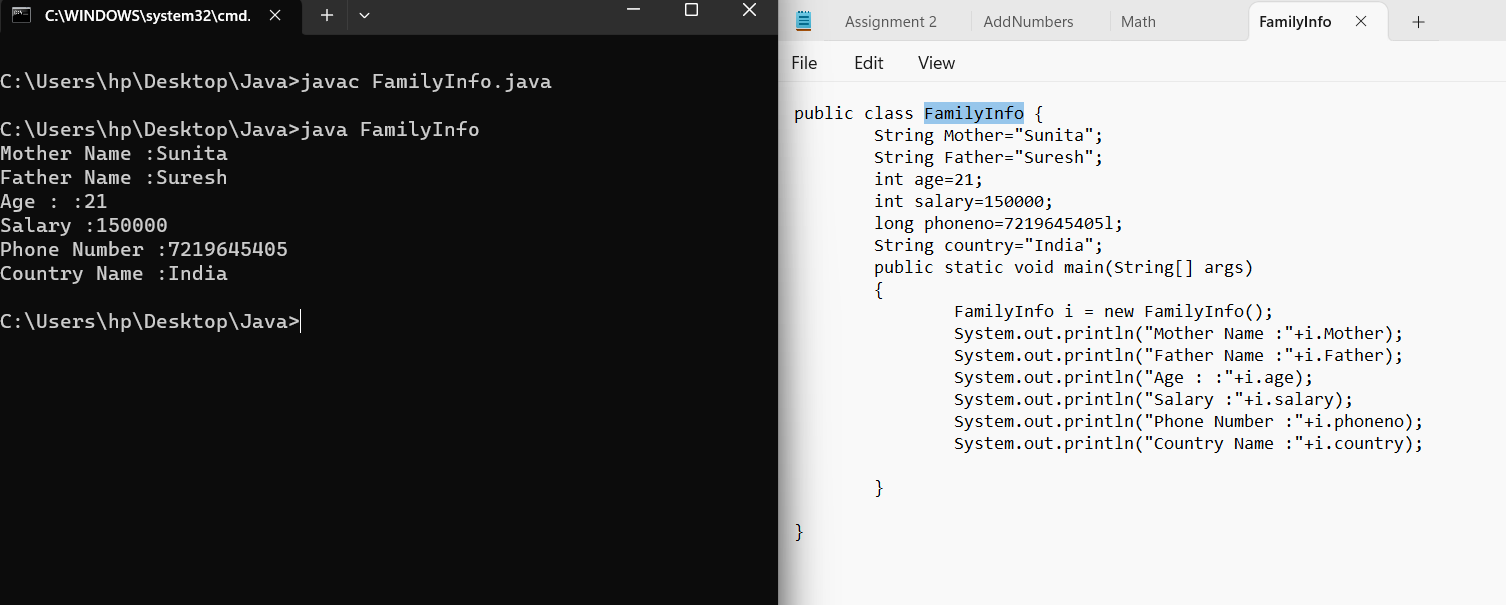
**Q1.** **Create a java program to print your mother and father name, age, salary, phone number, address like city state country.**

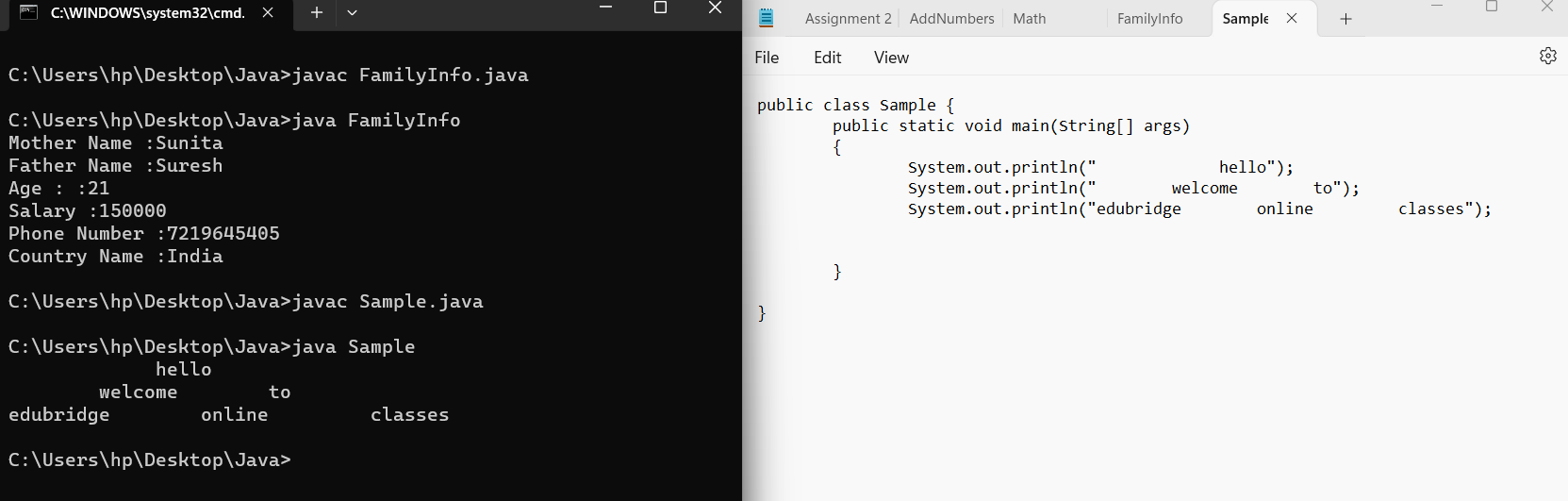
****

**Q2.** **Write a java program to print the below msg in the given format**

**hello**

**welcome to**

**edubridge online classes.**

****

**Q3.** **Scanner class is in which package**

The Scanner class in Java is located in the java.util package. The Scanner class is used to get user input. It was introduced in Java 1.5.

**Q.4 list the methods of Scanner class.**

The Scanner class in Java has the following methods:

1. next(): This method returns the next token from the input.
2. nextInt(): This method returns the next integer from the input.
3. nextLong(): This method returns the next long from the input.
4. nextDouble(): This method returns the next double from the input.
5. nextFloat(): This method returns the next float from the input.
6. nextChar(): This method returns the next character from the input.
7. nextLine(): This method returns the next line from the input.
8. hasNext(): This method returns true if there is another token in the input.
9. close(): This method closes the scanner.

**Q5.List the types of datatypes with size and give an example of each by doing declaring and initializating the variables.**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Default value** |
| Byte | 1 | 0 |
| Short | 2 | 0 |
| Int | 4 | 0 |
| Long | 8 | 0 |
| Flot | 4 | 0.0 |
| Double | 8 | 0.0 |
| Char | 2 | Empty |
| Boolean | 1 | False |

Byte:  
 byte a = 32;

Short:

short b = 1000;

Integer:

int c = 5;

Long:

long d = 123456789L;

Flot:

float e= 3.14f;

Double:

double f = 3.14159265359;

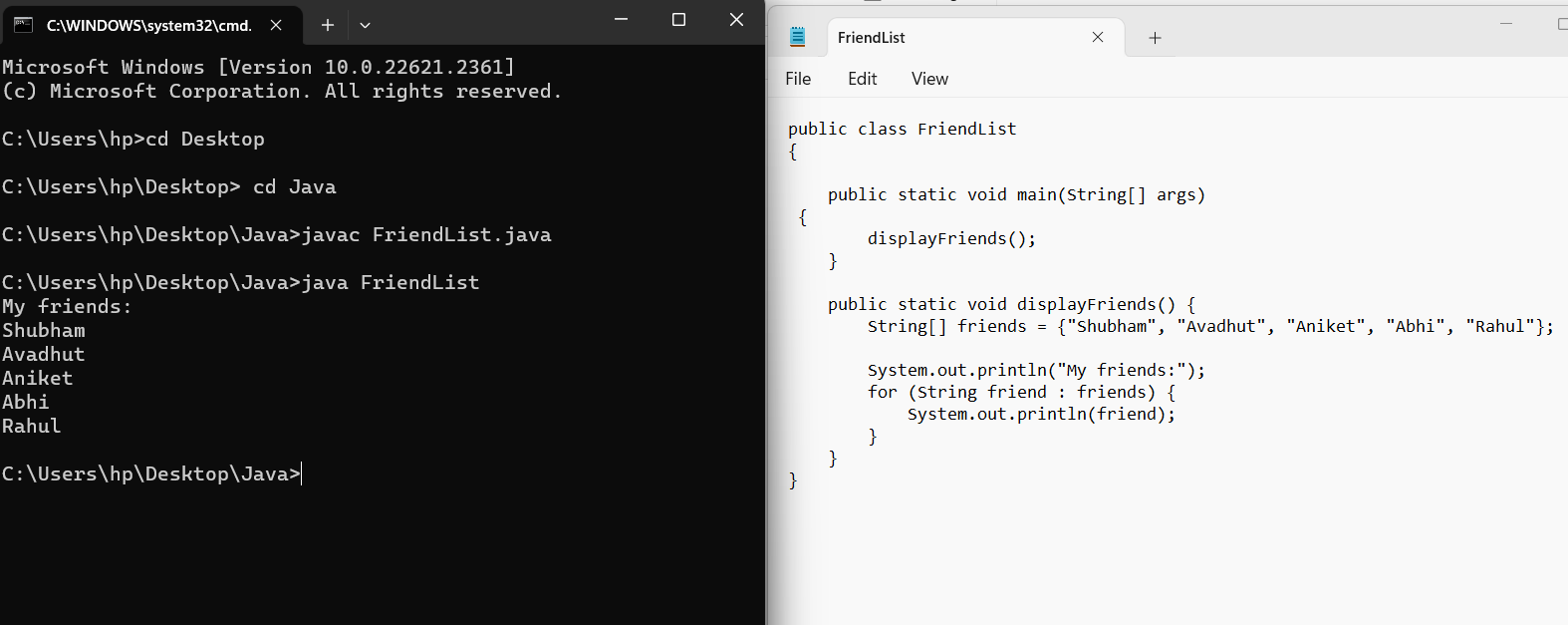
Char:

char g= 'A';

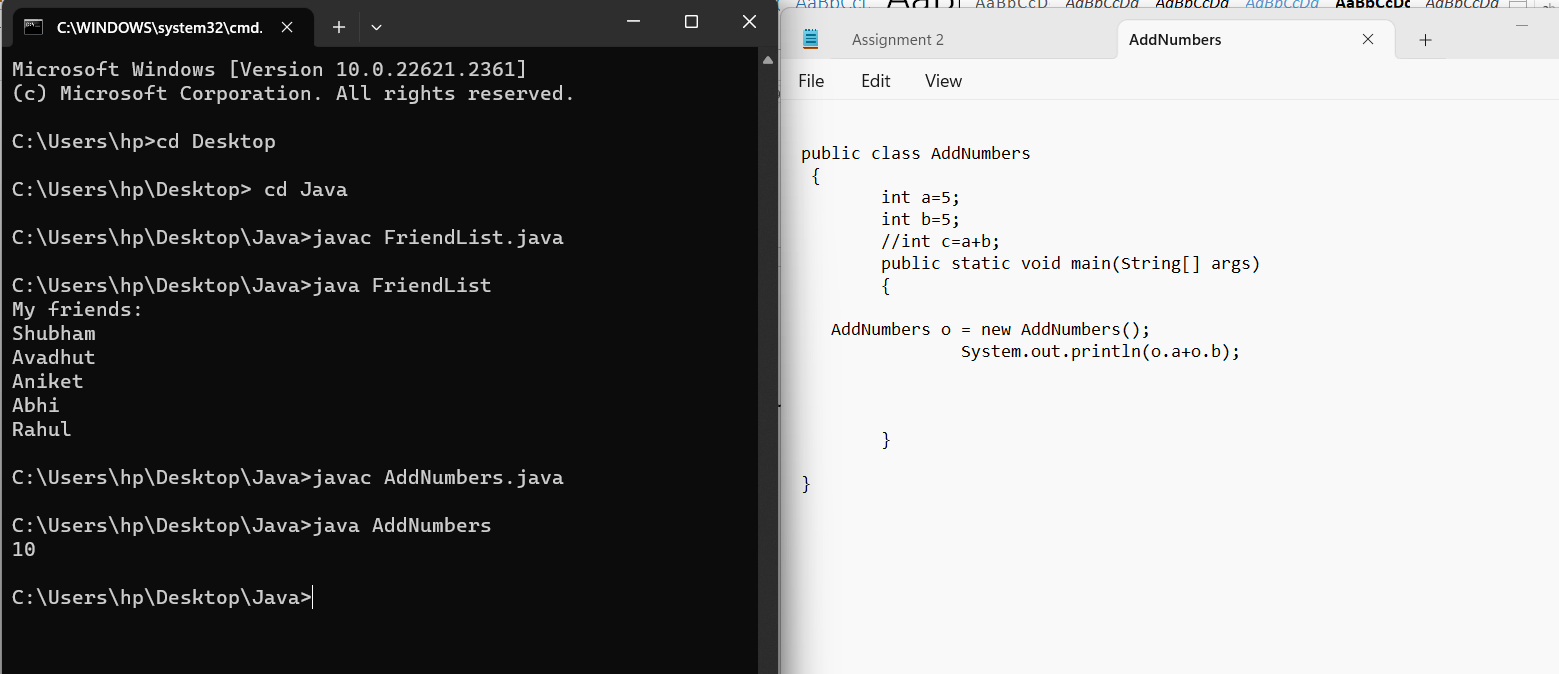
Boolean:

boolean h = true;

**Q6.** **Write a program to display the names of your friends by calling the static methods**

****

**Q7**.**Write a program to add two number by using static method**

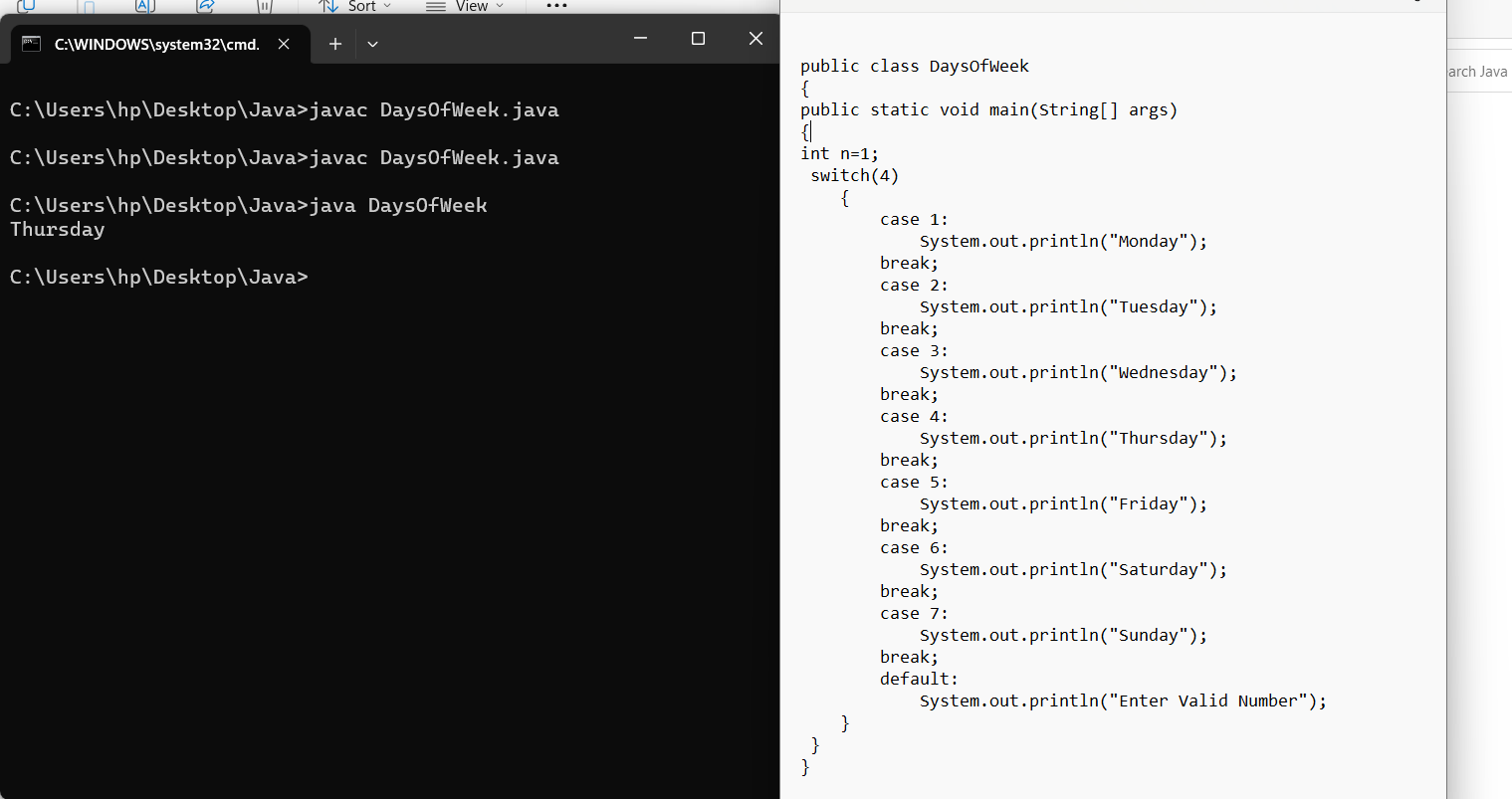
****

**Q8.** **Explain the public static void main(String[] args) in detail.**

In Java, the public static void main(String[] args) method is the entry point of a Java program. It's the method that's executed when the program is run. The method's signature has the following parts:

1. public: Indicates that the method is accessible to any class or object.
2. static: Means that the method is class related and not instance related.
3. void: Means that the method returns nothing.
4. main: The name of the method.
5. String[] args: An array of strings that receives any command-line arguments present when the program is execute

**Q9.** **Write a program to display names of days by using commandLine arguments**

****

**Q10 What are the rules of naming the java file**

1. File Name: The name of the Java file should match the name of the public class defined within the file For example, if your class is named `MyClass`, the file should be named `MyClassjava`
2. Valid Characters: The file name can only contain letters, digits, underscores, and dollar signs It must begin with a letter, an underscore, or a dollar sign It cannot begin with a digit
3. File Extension: Java files should have the `java` extension This extension is necessary for the Java compiler to recognize the file as a source code file
4. Case Sensitivity: File names are case-sensitive For example, `MyClassjava` and `myclassjava` are considered different files
5. Avoid Java Keywords: It's best to avoid using Java keywords as the names of your files or classes to prevent potential conflicts

**Q11 what are the rules of naming the class**

1. Class Name: The class name should be a noun and should start with an uppercase letter If the name contains multiple words, each word should start with an uppercase letter, known as camel case For example, `MyClass`, `MyFirstClass`, `StudentDetails`
2. Avoid Reserved Keywords: Avoid using Java reserved keywords as class names, as they have predefined meanings in Java and cannot be used for naming classes For example, you should not name your class `int`, `class`, or `public`, as these are reserved keywords in Java
3. Choose Descriptive Names: Choose meaningful and descriptive names that reflect the purpose and functionality of the class This helps make your code more understandable and maintainable by other developers
4. Use Singular Nouns: Class names should typically be singular nouns to represent an individual instance of the class For example, use `Car`, `Student`, or `Book` instead of `Cars`, `Students`, or `Books`

**Q12 what are the rules for naming the methods**

1. Method Name: The method name should be a verb or a verb phrase that describes the action performed by the method It should start with a lowercase letter, and if the name contains multiple words, each subsequent word should start with an uppercase letter, following the camel case convention For example, `calculateTotal`, `getUserDetails`, `printInformation`
2. Use of Parentheses: Methods in Java must be followed by parentheses `()` even if they don't take any parameters For example, `void performAction() { // method body }`
3. Choose Descriptive Names: Use descriptive and meaningful names that clearly indicate the purpose or behavior of the method This helps other developers understand the functionality of the method without needing to examine the implementation details
4. Avoid Confusing Names: Avoid using names that are too similar to existing Java methods or commonly used methods, as this may lead to confusion It's essential to choose names that accurately represent the specific behavior of the method
5. Follow Naming Conventions: Adhere to the standard Java naming conventions to maintain consistency throughout your codebase Consistent naming makes your code more readable and understandable for other developers

**Q13 what are the rules for naming variables in java**

1. Variable Name: The variable name should be a meaningful and descriptive identifier that represents the purpose of the variable. It should start with a lowercase letter, and if the name contains multiple words, each subsequent word should start with an uppercase letter, following the camel case convention. For example, `myVariable`, `totalAmount`, `studentName`.
2. Avoid Reserved Keywords: Avoid using Java reserved keywords as variable names, as they have predefined meanings in Java and cannot be used as identifiers. For example, you should not name your variable `int`, `class`, or `public`, as these are reserved keywords in Java.
3. Choose Descriptive Names: Use descriptive and meaningful names that clearly indicate the purpose of the variable. This helps other developers understand the role and usage of the variable within the context of the code.
4. Avoid Single Letters: Try to avoid using single letters or abbreviations that may not be clear to others reading your code. Instead, opt for names that provide more context and clarity about the purpose of the variable.
5. Follow Naming Conventions: Adhere to the standard Java naming conventions to maintain consistency throughout your codebase. Consistent naming makes your code more readable and understandable for other developers.

**Q14 which is the main entry point of java program**

The main entry point of a Java program is the `public static void main(String[] args)` method. This method serves as the starting point for the execution of a Java application. When you run a Java program, the Java Virtual Machine (JVM) looks for this method as the initial point of entry and starts the execution from this method.

**Q15 who is invoking the main method of java**

The `main` method in Java is invoked by the Java Virtual Machine (JVM) itself. When you run a Java program, the JVM is responsible for loading the class containing the `main` method and executing it. It serves as the entry point for the Java application, and the JVM is responsible for managing the program's execution.

**Q16 why main has to be static method**

The `main` method has to be declared as `static` in Java for a specific reason related to how it is called by the Java Virtual Machine (JVM) during the program's execution.

1. Static Context: The `main` method needs to be `static` so that it can be called without having to create an instance of the class. Since the `main` method serves as the entry point for a Java program and is called by the JVM before any objects are created, it must be accessible at the class level without requiring an instance of the class.
2. Standard Entry Point: By making the `main` method `static`, it becomes a standard entry point that can be invoked without the need for an object's instantiation. This ensures that the JVM can execute the `main` method directly when starting the program.
3. Simplified Invocation: Declaring the `main` method as `static` simplifies its invocation, allowing it to be accessed directly at the class level without the need for an object reference

**Q17 components of java program are?**

A basic Java program is composed of several components that work together to create a functional application. These components include:

1. Package Declaration: This is an optional statement that declares the package to which the Java file belongs. It helps organize and manage classes and interfaces into a directory structure.
2. Import Statements: These statements are used to bring specific classes or entire packages into the current file, enabling the use of their functionalities without fully qualifying their names.
3. Class Declaration: The class is a blueprint that defines the properties and behaviors of objects. It contains fields, methods, constructors, and other code blocks that define the structure and behavior of objects created from that class.
4. Methods: Methods are blocks of code that perform specific tasks. They are defined within a class and can be called to perform certain actions. The `main` method is the entry point of the Java program.
5. Variables: Variables are used to store data and provide a way to label data with a descriptive name. They can hold different types of data, such as integers, characters, and objects.
6. Comments: Comments provide explanations or annotations for the code and are ignored by the compiler. They help other developers understand the purpose of the code and its functionality.
7. Statements and Expressions: These are the building blocks of a Java program and are used to perform specific actions or calculations. Statements are complete lines of code, while expressions are building blocks of a statement and produce values.

**Q18 what is jvm,jre,jdk?**

* + 1. JVM (Java Virtual Machine): The JVM is an abstract machine that provides a runtime environment in which Java bytecode can be executed. It is responsible for interpreting the compiled Java bytecode and translating it into machine code or instructions that can be understood and executed by the underlying operating system. The JVM provides platform independence by allowing Java programs to run on any device or operating system that has a compatible JVM implementation.
    2. JRE (Java Runtime Environment): The JRE is a set of software tools used for developing and running Java applications. It includes the JVM, along with libraries, class files, and resources necessary for running Java applications. The JRE does not contain any development tools such as compilers or debuggers but is essential for running Java programs on a specific platform.
    3. JDK (Java Development Kit): The JDK is a software development kit that provides tools for developing, debugging, and monitoring Java applications. It includes the JRE, along with development tools such as the Java compiler (javac), the Java debugger, the Java application launcher (java), and other tools needed for developing Java applications. The JDK is required for writing and compiling Java code, as well as for creating Java applications and applets.

**Q19 Explain the components of compile time environments and run time environments.**

In the context of Java, there are two key environments: the compile-time environment and the run-time environment. Each environment consists of specific components that are crucial for the development and execution of Java programs.

* 1. **Compile-Time Environment:**
     1. Java Source Code: This is the human-readable code written by the programmer in the Java programming language. It contains classes, methods, and other necessary components of the program.
     2. Java Compiler: The Java compiler (javac) is responsible for converting the human-readable Java source code into platform-independent bytecode. It performs syntax checking, optimization, and generates class files that contain the bytecode instructions.
     3. Development Tools: These tools are used for editing, compiling, and debugging Java source code. They include integrated development environments (IDEs) such as Eclipse, IntelliJ IDEA, and NetBeans, which provide a comprehensive set of tools for Java development.
  2. **Run-Time Environment:**
     1. Java Virtual Machine (JVM): The JVM is responsible for executing Java bytecode. It provides a run-time environment for Java applications, ensuring platform independence by interpreting the bytecode and translating it into native machine code.
     2. Java Runtime Environment (JRE): The JRE includes the JVM along with libraries and resources necessary for running Java applications. It provides the necessary runtime libraries and components required to execute Java applications.
     3. Dynamic Linking: During runtime, the JVM dynamically links the necessary libraries and class files to execute the Java program. It loads the required classes, verifies them, and executes the program.

**Q20 what is JIT?**

JIT stands for Just-In-Time compilation. In Java, the JIT compiler is a part of the Java Virtual Machine (JVM) that compiles bytecode into native machine code at run time. The purpose of JIT compilation is to improve the performance of Java applications by dynamically compiling frequently executed bytecode into native machine code.

**Q21 different types of memory in jvm.**

* 1. Heap Memory: This is the runtime data area where objects are allocated. It is the memory used by the JVM for dynamic memory allocation for objects and JRE classes. The heap is divided into two main regions: the young generation and the old generation. The young generation is further divided into the Eden space and two survivor spaces.
  2. Method Area (Non-Heap Memory): Also known as the Permanent Generation, it stores class structures and method data. This area is shared among all threads and is used to store per-class structures such as runtime constant pool, field, and method data.
  3. JVM Stacks (Non-Heap Memory): Each Java thread has a private JVM stack, created at the same time as the thread. It holds the state of Java method invocations, including parameters, local variables, and return values.
  4. PC Registers (Non-Heap Memory): Each Java thread has its own program counter (PC) register, which keeps track of the current execution address. It stores the address of the Java virtual machine instruction currently being executed.
  5. Native Method Stacks (Non-Heap Memory): These are used to store native method information and are similar to JVM stacks, but they store native method information instead of Java method information.
  6. Direct Memory (Non-Heap Memory): This area is used to store direct byte buffers, which can be used to interact with native libraries outside of the JVM.

**Q22 In which area .class is stored?**

The `.class` files, which contain the bytecode of compiled Java classes, are stored in the method area (also known as the Non-Heap Memory or the Permanent Generation) of the Java Virtual Machine (JVM). The method area is a part of the JVM's memory where the JVM stores the metadata about classes that have been loaded into memory. This includes information such as class names, method names, method data, and constant pool information.

**Q23 In which area object are stored?**

Objects in Java are stored in the heap memory. The heap is a runtime data area within the Java Virtual Machine (JVM) where objects are allocated. It is the memory area where the JVM dynamically allocates memory for objects during the runtime of a Java application. The heap is shared among all threads, and objects are created and reside in this memory area until they are no longer referenced and become eligible for garbage collection.

**Q24 Why do we call as java simple?**

Java is often considered simple due to several design choices and features that contribute to its ease of use and understanding. Some key reasons why Java is considered a simple programming language are:

* 1. Easy to Learn: Java has a straightforward syntax and a well-designed, readable code structure, making it relatively easy for programmers to learn and understand.
  2. Platform Independence: Java programs can be written once and run on any platform with a compatible Java Virtual Machine (JVM), making it a highly portable language.
  3. Automatic Memory Management: Java's automatic garbage collection system manages memory allocation and deallocation, eliminating the need for manual memory management and reducing the likelihood of memory-related errors such as memory leaks and dangling pointers.
  4. Rich Standard Library: Java provides a rich set of libraries and APIs that offer a wide range of functionalities, making it easier for developers to perform various tasks without having to write complex code from scratch.
  5. Object-Oriented Approach: Java is an object-oriented programming language, which promotes the use of classes and objects, facilitating the organization and management of complex code structures.
  6. Strong Community Support: Java has a large and active community of developers, which provides ample resources, documentation, and support for programmers of all levels.