# *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.