Chapter One

Introduction to Object-Oriented Programming

Types of programming paradigms

- **Programming paradigm** is an approach to solve problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach.
- There are lots for programming language that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is **paradigms**.

- There are several kinds of major programming paradigms:
  - Imperative
  - Logical
  - Functional
  - Object-Oriented
- It can be shown that anything solvable using one of these paradigms can be solved using the others; however, certain types of problems lend themselves more naturally to specific paradigms.

1. Procedural Programming

- Procedural programming can also be referred to as **imperative programming.**
- It is a programming paradigm based upon the concept of procedure calls, in which statements are structured into procedures (also known as subroutines or functions).
- They are a list of instructions to tell the computer what to do step by step, Procedural programming languages are known as top-down languages.
- Most of the early programming languages are all procedural.
  - Examples of Fortran C and Cobol.

Features of Procedural Code

- Procedural Programming is excellent for general-purpose programming
- The source code is portable
- The code can be reused in different parts of the program, without the need to copy it
- The program flow can be tracked easily as it has a top-down approach.

2. Logical Programming

- Logical programming is a computer programming paradigm that has its foundations in mathematical logic in which program statements express facts and rules about problems within a system.
- Rules are written as logical clauses with a head and a body.
- They also follow a declarative rather than an imperative approach.

- To understand how a problem can be solved in logical programming, you need to know about the building blocks – Facts and Rules –
- In declarative approach, rather than providing a step by step instruction (imperative), you tell the system what you need and let it try to come up with a solution (declarative).
- Prolog follows the Logical paradigm and is probably the most famous language in the logical programming family.

Features of Logical Programming

- Logical programming can be used to express knowledge in a way that does not depend on the implementation, making programs more flexible, compressed and understandable.
- It enables knowledge to be separated from use, i.e. the machine architecture can be changed without changing programs or their underlying code.
- It can be altered and extended in natural ways to support special forms of knowledge, such as meta-level of higher-order knowledge.
- It can be used in non-computational disciplines relying on reasoning and precise means of expression.

3. Functional Programming

- Functional programming is a programming paradigm where you have a style of building the structure and elements of computer programs.
- Here you treat computation as an evaluation of mathematical functions and you avoid changing-state and mutable data.
- Functional programming consists only of PURE functions.
- Pure functions are those which take an argument list as an input and whose output is a return value.
  - ***if a function relies on the global variable or class member's data, then it is not pure.***
  - ***Has no side effect:*** *A side effect is a change in the state of an application that is observable outside the called function other than its return value.*

Features of Functional Paradigm

- **Pure functions** – In case of pure functions, the output depends only on the input.
- **Recursion-**A recursive function is a function that calls itself during its execution. This enables the function to repeat itself several times, the result being outputted at the end of each iteration.

- **Referential transparency-**An expression is said to be referentially transparent if it can be replaced with its corresponding value without changing the program's behavior.
- As a result, evaluating a referentially transparent function gives the same value for fixed arguments.
- In functional programming, only referentially transparent functions are considered.
- They are a lot easier to read and understand.

- **Functions are First-Class and can be Higher-Order-**
- A programming language **is** said to have First-class functions when functions in that language are treated like any other variable.
  - For example, in such a language, a function **can** be passed as an argument to other **functions**, **can** be returned by another function and **can** be assigned as a value to a variable.
- Programming Languages that support functional programming: Haskell, JavaScript, Scala, Lisp

- **Variables are Immutable-**In functional programming you cannot modify a variable after it has been initialized. You can create new variables and this helps to maintain state throughout the runtime of a program.

Object-Oriented Programming(OOP)

- In this framework, all real-world entities are represented by ***Classes***.
- ***Objects*** are instances of classes so each object encapsulates ***state*** and ***behavior***.
  - ***State*** implies the fields, attributes of the object and ***behavior*** is what you do with the state of the object and they are the methods.
- Objects interact with each other by passing messages.

Features of OOP

- Encapsulation
- Inheritance
- Abstraction
- Polymorphism

Overview of object oriented principles

- **Object-Oriented Programming (OOP)**
  - Object-oriented programming is the successor of procedural programming.
  - **Procedural programming** describes programs as groups of reusable code units (procedures) which define input and output parameters.
  - Procedural programs consist of **procedures**, which invoke each other.

- The **problem** with procedural programming is that **code reusability is hard** and limited – only procedures can be reused and it is hard to make them generic and flexible.
- There is no easy way to work with abstract data structures with different implementations.

- The object-oriented approach relies on the paradigm that each and every program works with data that describes **entities** (objects or events) from real life.
  - For example: accounting software systems work with invoices, items, warehouses, availabilities, sale orders, etc.
- This is how objects came to be. They describe characteristics (properties) and behavior (methods) of such **real life entities**.

- **The main advantages and goals of OOP** are :
  - to make complex software faster to develop and easier to maintain.
  - OOP enables the easy reuse of code by applying simple and widely accepted rules (principles).
- The following are main principles and components of object oriented programming.

Class and objects

- Object is a thing, both tangible and intangible, that we can imagine.
- An object is comprised of data and operations that manipulate these data.
  - For example, a student object may consists of data such as name, gender, birth date, home address, phone number and age and operation for assigning and changing these data values.
- Inside a program we write instructions to create objects.
- For the computer to be able to create an object, we must provide a definition, called a class.
- A class is a kind of mold or template that dictates what can and cannot do.
- An object is called instance of a class.

Messages and methods

- A method is a sequence of instructions that a class or an object follows to perform a task.
- To instruct a class or an object to perform a task, we send a message to it.

Encapsulation

- Encapsulation is:
  - Binding the data with the code that manipulates it
  - It keeps the data and the code safe from external interference
- There are two important aspects of encapsulation:
  - Access restriction - preventing one object from accessing another's internal state, for example.
  - Namespaces/scopes - allowing the same name to have different meanings in different contexts.
- Objects are a fundamental encapsulation mechanism of object-oriented languages (OOLs).
  - objects - encapsulate data together with operations, called *methods* that process or act upon the data.
- On the Java platform, you can use *access modifiers* to vary the nature of object relationships from *public* to *private*.
- Public access is wide open, whereas private access means the object's attributes are accessible only within the object itself.

Polymorphism

- Polymorphism refers to the ability of different objects to respond to the same message in different ways.
- Polymorphism is essential for modeling our world including our social environment
- Polymorphism manifests itself by having multiple methods all with the same name, but slightly different functionality

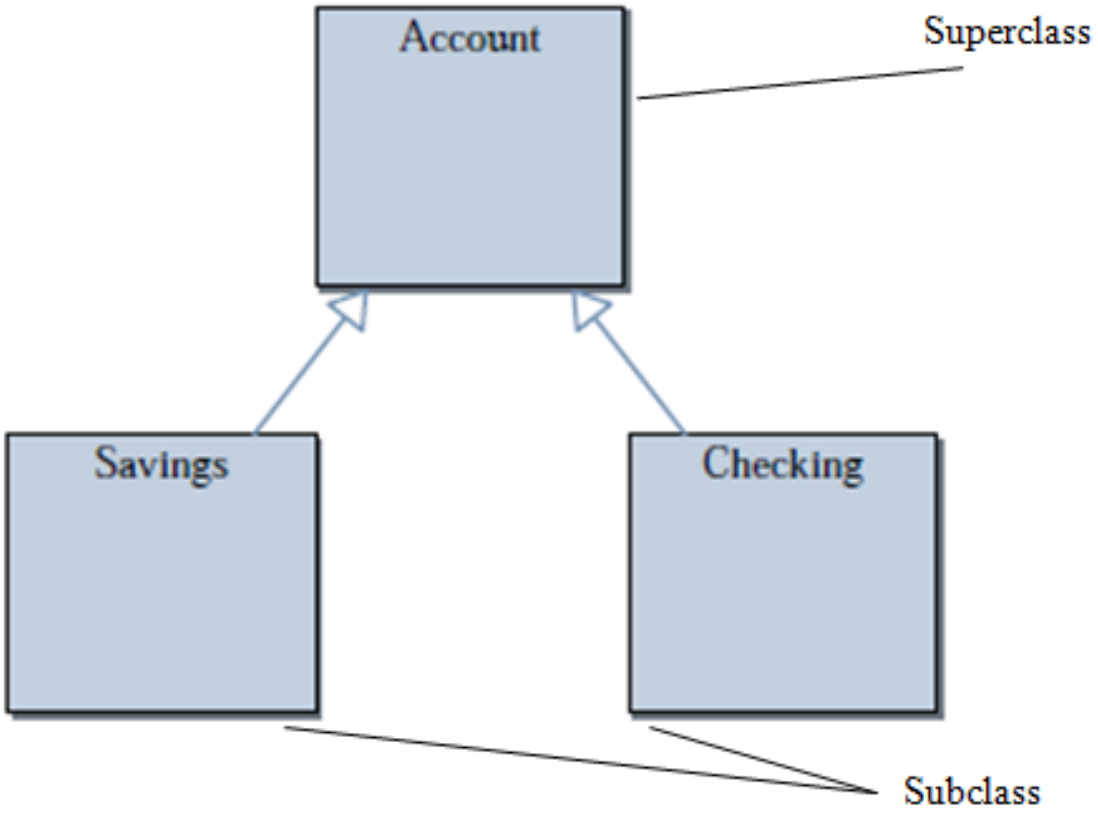
Abstraction

- Its main goal is to handle complexity by hiding unnecessary details from the user.
- Objects in an OOP language provide an abstraction that hides the internal implementation details.
- You just need to know which methods of the object are available to call and which input parameters are needed to trigger a specific operation.
- But you don’t need to understand how this method is implemented and which kinds of actions it has to perform to create the expected result.

Inheritance

- Inheritance can be defined as the process where one object acquires the properties of another.
- With the use of inheritance, the information is made manageable in a hierarchical order.
- When we talk about inheritance, the most commonly used keyword would be **extends** and **implements**.
- These words would determine whether one object IS-A type of another.
- By using these keywords we can make one object acquire the properties of another object.

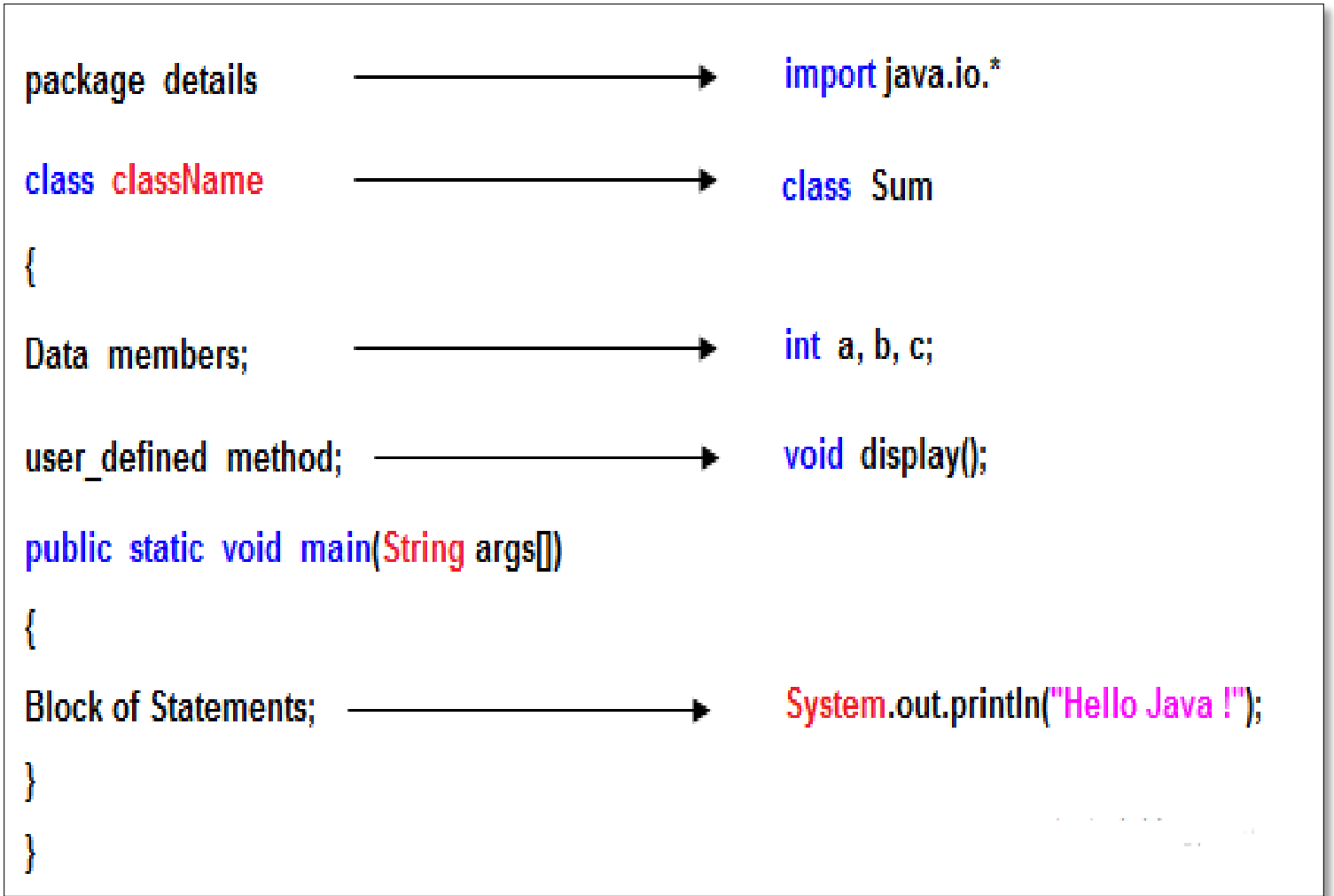
- Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.
- The idea behind inheritance in java is that you can create new classes that are built upon existing classes.
- When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.



Graphical representation of inheritance

Overview of java programming and types of java program

- A typical java program has the following structure



- A package is a collection of classes, interfaces and sub-packages.
  - java.lang.*; package is imported by default and this package is known as default package.
- Class is keyword used for developing user defined data type and every java program must start with a concept of class.
- "ClassName" represent a java valid variable name treated as a name of the class each and every class name in java is treated as user-defined data type.

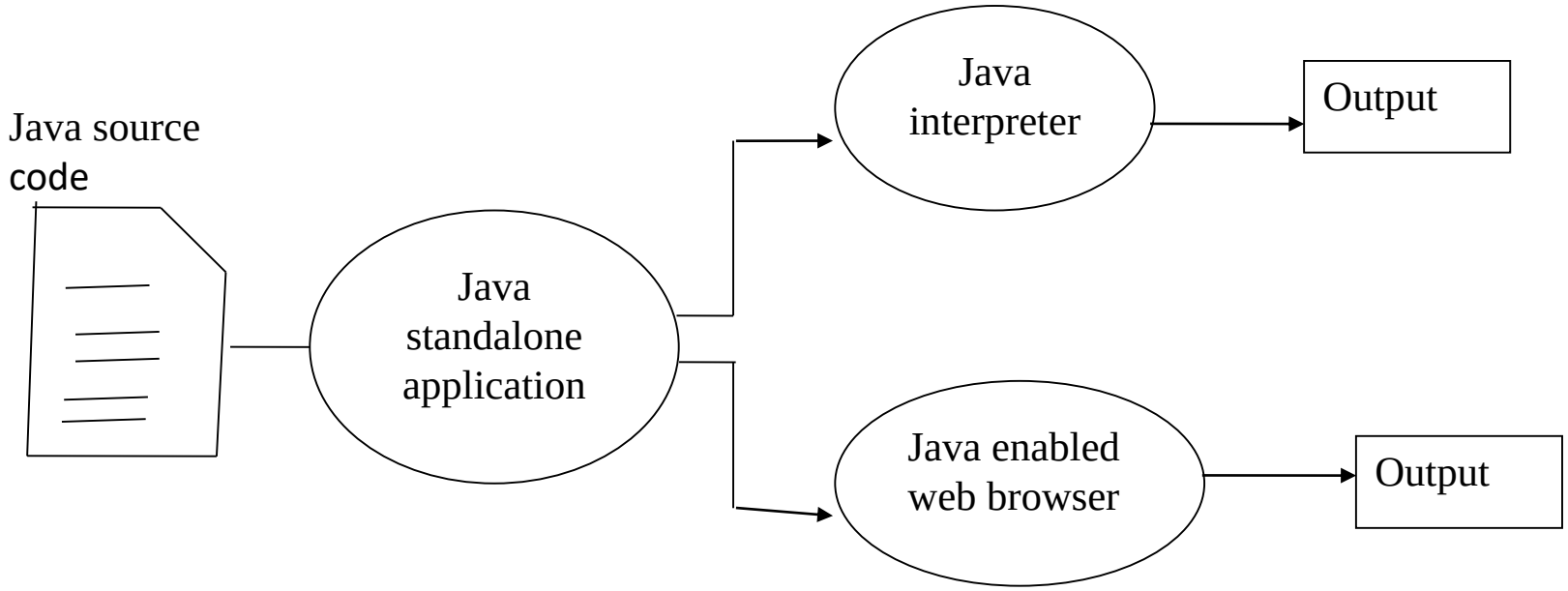
- Data member represents either instance or static they will be selected based on the name of the class.
- User-defined methods represents either instance or static they are meant for performing the operations either once or each and every time.

- Each and every java program starts execution from the main() method.
- And hence main() method is known as program driver.
- Since main() method of java is not returning any value and hence its return type must be void.
- Since main() method of java executes only once throughout the java program execution and hence its nature must be static.
- Since main() method must be accessed by every java programmer and hence whose access specifier must be public.
- Each and every main() method of java must take array of objects of String.

- Block of statements represents set of executable statements which are in term calling user-defined methods are containing business-logic.
- The file naming conventionW in the java programming is that which-ever class is containing main() method, that class name must be given as a file name with an extension .java.

Types of java programs

- With Java API, many types of Java programs can be developed. These include
- **Java stand-alone applications:**
  - The Java stand-alone applications are the programs written in Java to carry out certain tasks.
  - These applications run directly by the Java interpreter.
  - A standalone application should be delivered and installed on each computer before it is run.
  - It can have either a Command-Line Interface (CLI) or Graphical User Interface (GUI).
  - You have already used a number of stand-alone applications such as text editors, word processors and games applications.


Java source code
Java standalone application
Java interpreter
Output
Java enabled web browser
Output

- **Java Applets**:
  - Java applets are the Java applications that are embedded inside HTML files and can be downloaded into a Java-capable browser (E.g. Netscape and Hot Java).
  - An applet can perform tasks and interact with users on their browser's WebPages without using resources from the web server after being downloaded.
  - Using applets, you can do anything ranging from animations to complete games and utilities that can be executed over the Internet.

- **Java Servlets**:
  - Java servlets are the Java programs that run on a web server to generate dynamic web contents.
- **Mobile Application:**
  - Mobile application is developed for run the mobile phones and tablets.

- Java is one Language with different capabilities and Editions. Just like a truck which can be used to carry all kinds of things.
  - The Java SE (Standard Edition) is used build desktop applications , console application with the Swing and the JavaFX
  - The Java EE (Enterprise Edition) is used to develop web applications using java. Comes with loads of technologies like Servlets, JSP and more…
  - The Java ME(Micro Edition) is used to develop lightweight mobile applications .

Editing, Compiling and Interpreting

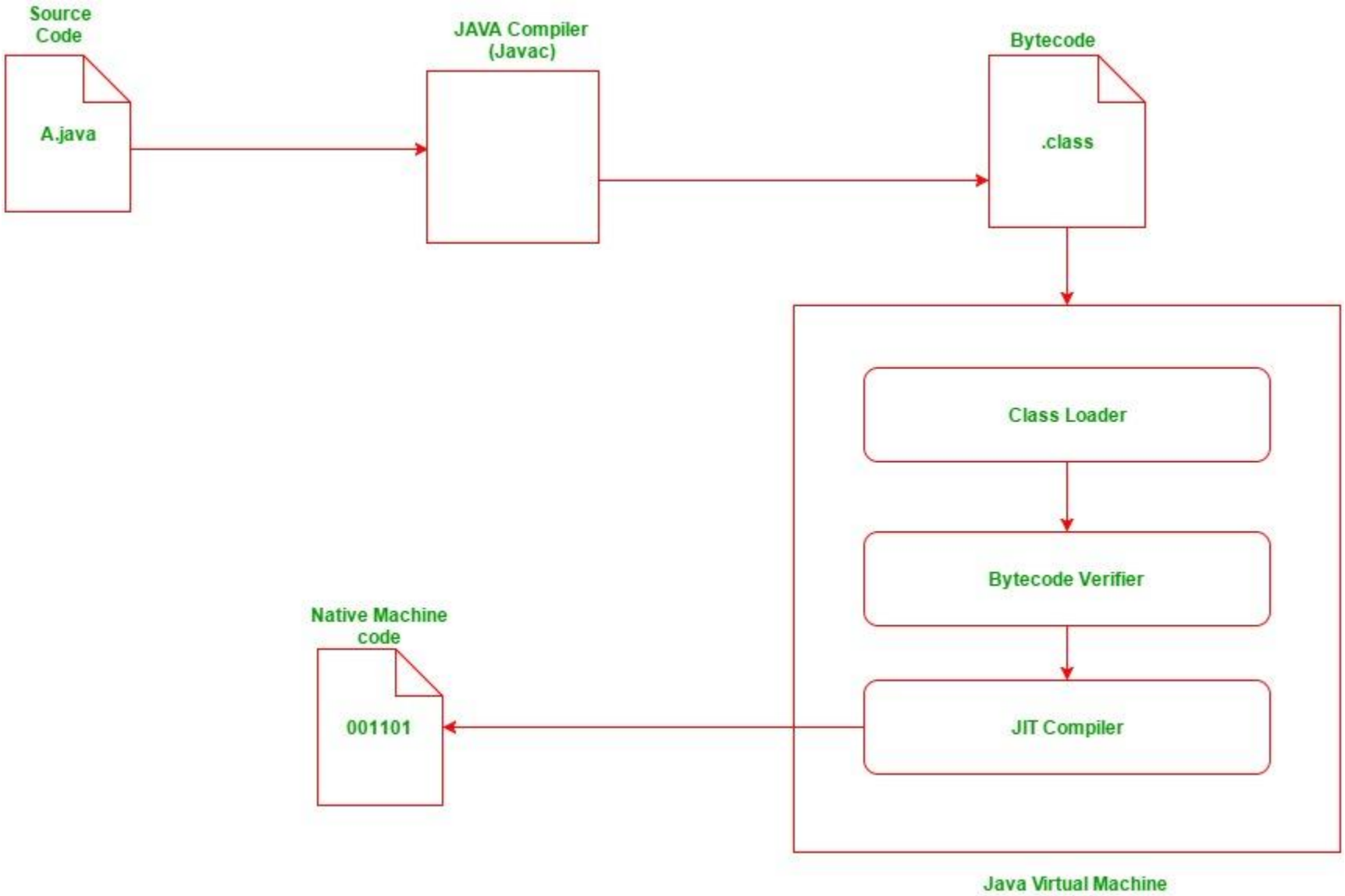
- First, the source ‘.java’ file is passed through the compiler, which then encodes the source code into a machine-independent encoding, known as Bytecode.
- The content of each class contained in the source file is stored in a separate ‘.class’ file

- The class files generated by the compiler are independent of the machine or the OS, which allows them to be run on any system.
- To run, the main class file (the class that contains the method main) is passed to the JVM and then goes through three main stages before the final machine code is executed
- These states do include:
  - ClassLoader
  - Bytecode Verifier
  - Just-In-Time Compiler

- Class Loader
  - The main class is loaded into the memory bypassing its '.class' file to the JVM, through invoking the latter. All the other classes referenced in the program are loaded through the class loader.

- Bytecode Verifier
  - After the bytecode of a class is loaded by the class loader, it has to be inspected by the bytecode verifier, whose job is to check that the instructions don't perform damaging actions. The following are some of the checks carried out:
    - Variables are initialized before they are used.
    - Method calls match the types of object references.
    - Rules for accessing private data and methods are not violated.
    - Local variable accesses fall within the runtime stack.
    - The run-time stack does not overflow.
    - If any of the above checks fail, the verifier doesn't allow the class to be loaded.

- Just-In-Time Compiler
  - This is the final stage encountered by the java program, and its job is to convert the loaded bytecode into machine code.


Source Code
A.java
JAVA Compiler (Javac)
Bytecode
.class
Class Loader
Bytecode Verifier
JIT Compiler
Native Machine code
001101
Java Virtual Machine

Java development kit (JDK)