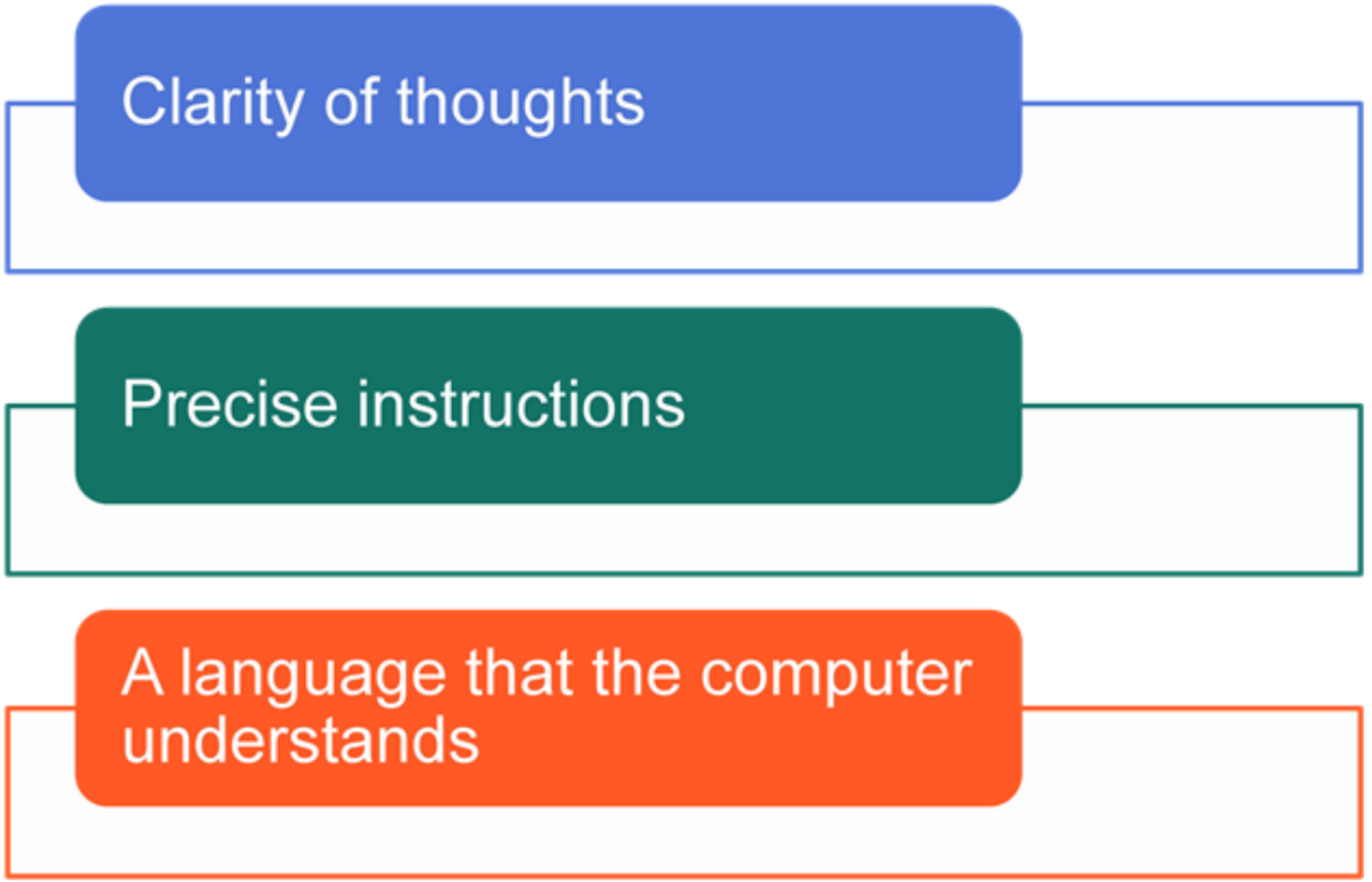
Learning Consolidation

Algorithmic Problem Solving

Learning Objectives

- Break down a complex problem into simple steps
- Use an algorithmic approach to solve problems
- Create pseudocode to represent algorithms
- Analyze the input and output requirements of a computer problem

How Can a Computer Be Programmed to Solve a Problem?



Algorithmic Thinking

- How can you solve a given problem?
  - Define the problem
  - Think of possible solutions
  - Evaluate and select the optimal solution
  - Implement the solution
- Why is problem-solving important for a person?
  - Gives the ability to think outside of the box
  - Helps to sharpen logical thinking
  - Lays a strong foundation in programming skills

Problem-Solving Techniques

Algorithms

Pseudocode

What Is an Algorithm?

- An algorithm is:
  - A set of instructions designed to perform a specific task.
  - A simple description of how the problem needs to be solved.
  - Baby steps towards writing programs in programming languages.
  - A simple set of English statements.
  - An algorithm can be handwritten on a piece of paper or on a text editor on a computer.

Characteristics of a Good Algorithm

- Precision: The steps are precisely stated or defined.
- Uniqueness: The results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- Finiteness: The algorithm always stops after a finite number of steps.
- Input: The algorithm receives some input.
- Output: The algorithm produces some output.

ATM Withdrawal

- How do you withdraw cash from an ATM machine?

```
Step 1: Start
Step 2: Insert your ATM card for the machine to check.
Step 3: Enter the PIN for the ATM to validate.
Step 4: Press the 'Cash with Receipt' button.
Step 5: Specify the amount you wish to withdraw.
Step 6: Collect your ATM card.
Step 7: Wait till the machine counts the cash.
Step 8: Collect the amount and statement from the  ATM slot
Step 9: End
```

How Are These Instructions Processed by Computer Systems?

- All modern computers function on the general model of input, process, and output.
- A computer system:
  - Accepts inputs from the user
  - Process the input
  - Generates the output

What is Pseudocode?

- Pseudocode is a plain language description of the steps in an algorithm.
- Pseudocode often uses structural conventions of a normal programming language.
- It is intended for human reading rather than machine reading.
- **Pseudocode helps in transitioning to writing programs easily.**

```
BEGIN
GET costPrice
GET sellingPrice
BEGIN IF
     IF SP > costPrice
        SET profit = sellingPrice - costPrice
        PRINT profit
     ELSE
        SET loss = costPrice - sellingPrice
        PRINT loss
END IF
END
```

Pseudocode

- Given the cost price and the selling price of a product, calculate the profit gained or the loss incurred on the purchase of the product.

Algorithm vs. Pseudocode

Algorithm	Pseudocode
An algorithm design technique is a general approach to solving problems algorithmically that applies to a variety of problems from different areas of computing.	Pseudocode is written in a format that is similar to the structure of a high-level programming language which can later be converted to a program code easily.
Algorithm uses English statements.	Pseudocode also uses English statements, but they are more relevant to a programming language.
The words used in an algorithm don't need to be very specific. For example: Accept a number can also be written as Get a number.	The words used are more specific. For example, GET number cannot be modified. Here GET is a keyword which must be used as is.
It is not mandatory to write start and end for an algorithm.	It is mandatory to write blocks of pseudocode between BEGIN and END.

Self Check

What are the three types of constructions available in pseudocode?

A. Input, Output, and Process
B. Input/Output, Decision, and Repeat
C. Sequence, Decision, and Repeat
D. Loop, Input/Output, and Process

Self Check: Solution

What are the three types of constructions available in pseudocode?

A. Input, Output, and Process
B. Input/Output, Decision, and Repeat
C. **Sequence, Decision, and Repeat**
D. Loop, Input/Output, and Process

Learning Consolidation

Introduction to Java, Variables, Datatypes, and Operators

Learning Objectives

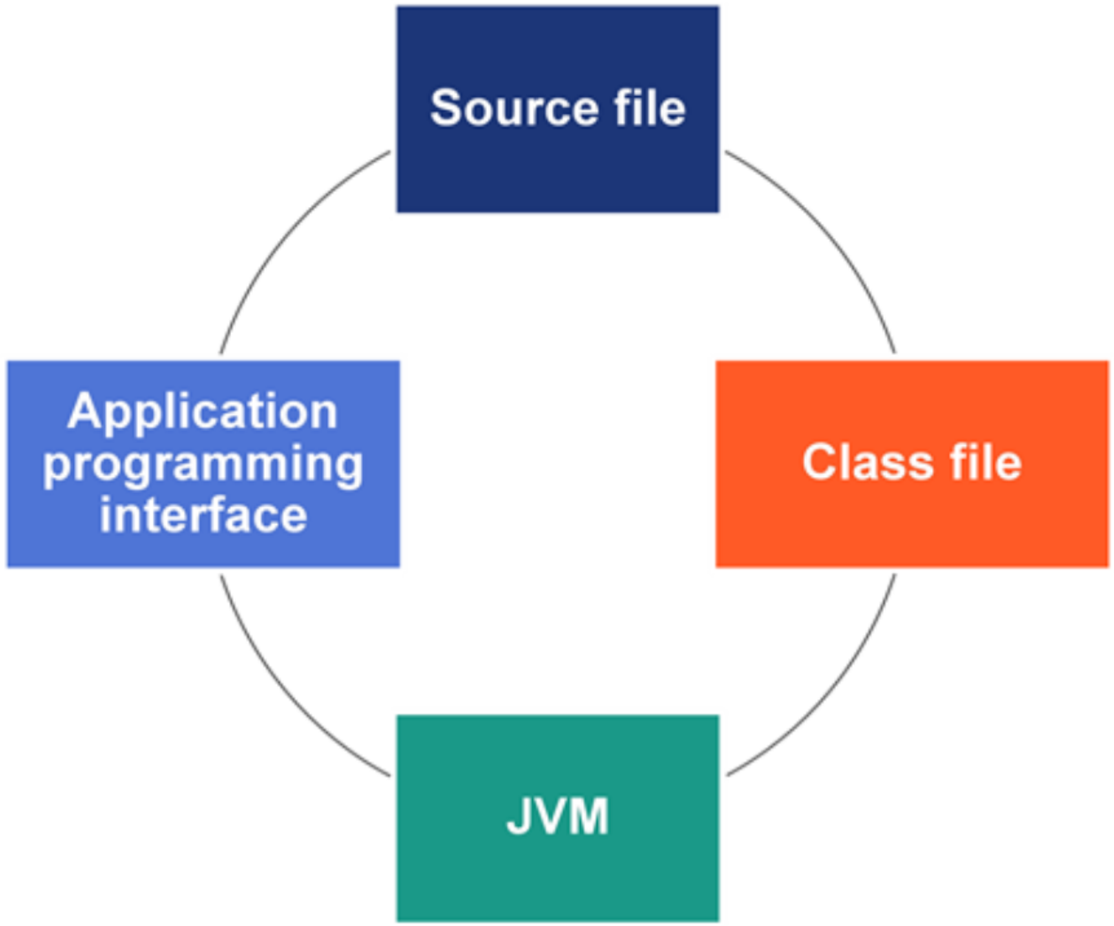
- Introduction to Java
- Define the Java Architecture
- Explore the components of the Java Architecture
- Explain Naming Conventions for classes and variables.
- Explore the Java Keywords
- Define variables
- Implement implicit widening of data types when using the arithmetic operator

Introduction to Java

- Java is an object-oriented general-purpose programming language designed for application development.
- The Java architecture defines the components that are essential to carry out the creation and execution of code written in the Java programming language.
- Before moving on to writing a Java program it is essential to understand the architecture and working of Java as a language. Below are some key terms:
  - Source file
  - Class file
  - JVM
  - JRE

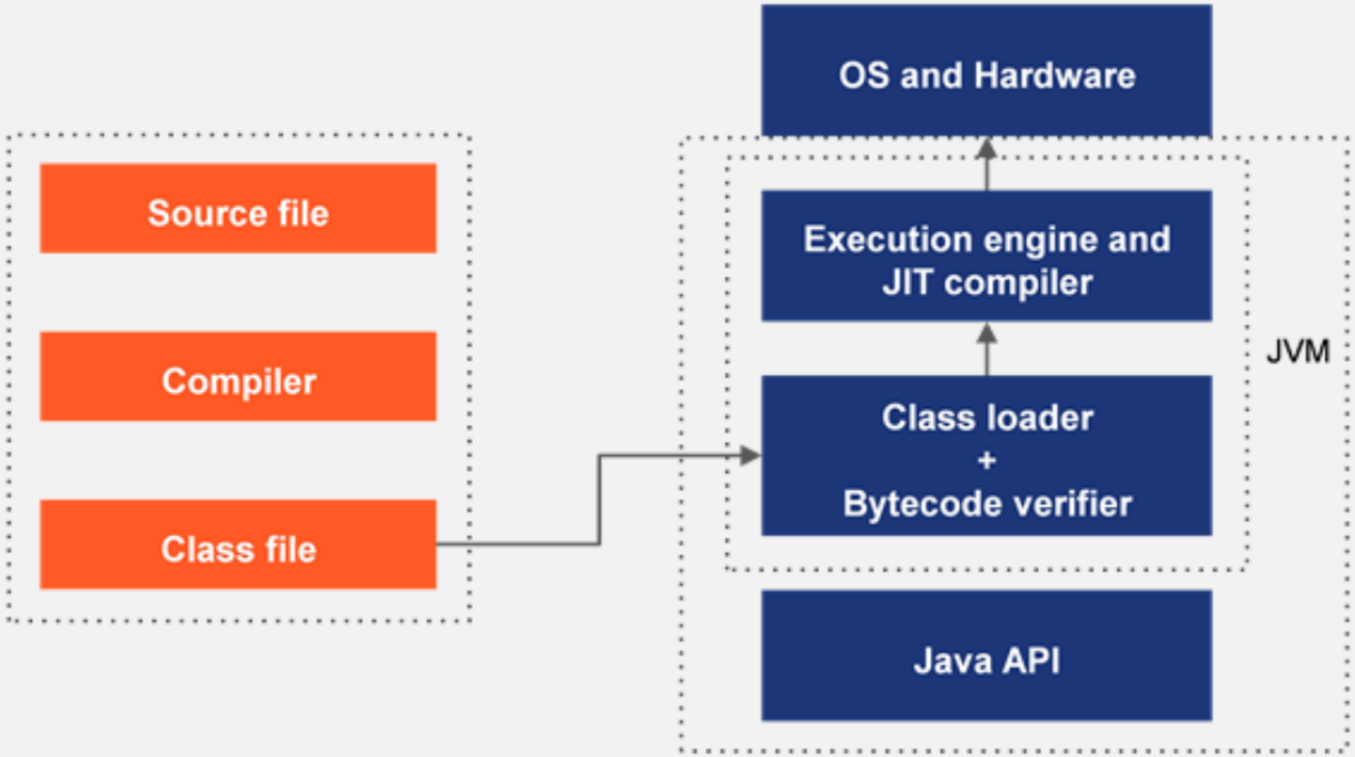
Java Architecture

- Java architecture defines the components that are essential to carry out the creation and execution of Java code.
- The Various components of Java Architecture are:



Architecture Components

- The file in which we write the Java program is called the **source file** and we save it with a .java extension.
- The source file is fed to the **compiler** that produces a **.class** file.
- The JVM is a specification that provides a run-time environment in which java bytecode can be executed. The JVM is an application that is responsible for executing Java programs on a computer.
- The JVM is responsible for executing the bytecode and generating the machine-specific code for the machine on which the Java program needs to be executed.



Naming Conventions for Java Classes and Variables

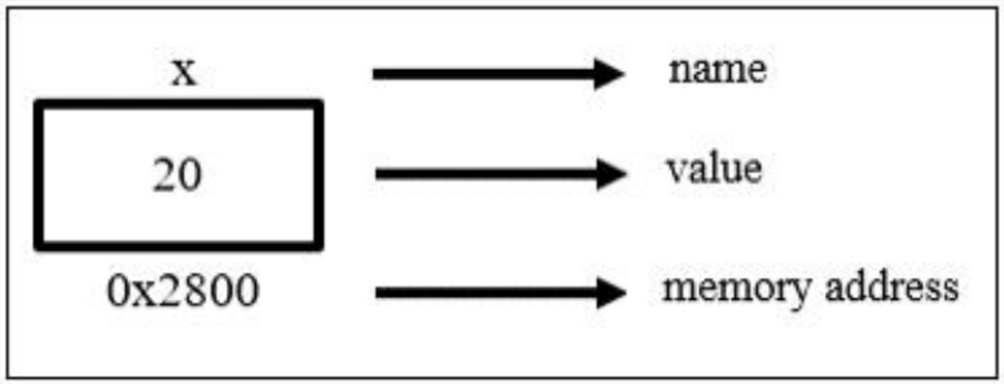
- The name of a class or variable should not contain any embedded space or symbol, such as ?, !, #, @, %, &, {}, [], :, ;, “, and /.
- A class or variable name must begin with a letter, an underscore (_), or the dollar symbol ($). Or, it must begin with an alphabet that can be followed by a sequence of letters or digits (0 to 9), ‘$’, or ‘_’.
- A class or variable name should not consist of a keyword.
- The first letter of the class name should be capitalized.
- If the class name consists of several words, the first letter of each word should be capitalized.
- If the variable name consists of only one word, choose to keep that word in all lowercase letters. However, if the name is a combination of two words, use the **camel case notation, f**or example, `topSpeed` and `mobileNumber`
- If the variable name is all uppercase characters, it is primarily identified as a constant variable. For example, `PI`, because it has a fixed value, which cannot be changed throughout the program.

Java Keywords

- Keywords are reserved words with a special meaning for a language, which express the language features.
- Keywords cannot be used to name variables or classes. Java is a case-sensitive language, and the keywords should be written in lowercase only.
- The table lists the Java keywords.

case	catch	char	do
const	continue	default	final
double	else	extends	goto
finally	float	for	instanceof
if	implements	import	native
int	interface	long	protected
new	package	private	static
public	return	short	synchronized
strictfp	super	switch	transient
this	throw	throws	while
try	void	volatile	boolean
enum	assert	byte	break
abstract	class		

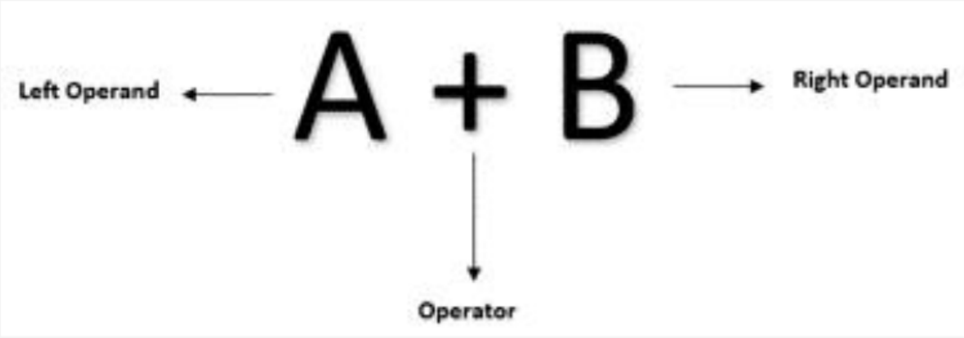
Variables



- A variable is a name given to a memory location that holds a data value in it.
- It is essentially a container that stores various kinds of data in it.
- A value in a variable can change many times during an execution of a program.
- Every variable, in any programming language, is associated with a data type.
- This data type decides the memory space and the kind of values the variable can have.

Java Operators

- Java supports the following types of operators:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Logical operators
  - Unary operators
  - Bitwise operators
  - Shift operators



Implicit Widening of Data Types in an Arithmetic Operation

- The arithmetic operations performed on the operands that hold values of the primitive data types such as byte, short, int, and long or character types such as char will result in a numeric type.
- For example, when an expression involves two primitive data types lower than int, the arithmetic operation will result in an int value. This is because the values of the lower data types such as byte, short, and char get automatically converted to an int value. This conversion is known as implicit **widening of data types**.

Implicit Widening of Data Types in an Arithmetic Operation (contd.)

- The following rules apply to operands that get converted using implicit widening:
  - If one of the two operands is of type double, the other operand will be converted to type double.
  - If one of the two operands is of type float, the other operand will be converted to type float.
  - If one of the two operands is of type long, the other operand will be converted to type long.
  - Otherwise, both the operands will be converted to type int.

Implicit Widening – An Example

The following code performs arithmetic operation on `byte` and `short` data types:

```
public class Add
{
     public static void main(String []args)
     {
        byte num1=10; // Line 1
        byte num2=12; // Line 2
        short result=num1+num2; // Line 3
        System.out.println("The sum of two
   numbers are: " + result);
     }
}
```

- The code generates a compile-time error since short has lower memory space allocated and cannot add two `byte` datatype variables.
- The result of adding the two operands `num1`, and `num2` will be automatically converted to `int`. Hence it cannot be stored in a `short` variable.