Java Programming

Entering Java World

Training Agenda

Day 01

- Introduction to Java and eco-system
- Data types, Variables & Operators
- OOPS – Classes and Objects
- Array
- String & DateTime

Day 02

- Inheritance
- Polymorphism
- Abstraction & Encapsulation
- Exception Handling
- File I/O
- Thread Programming
- SOLID Principles

Day 03

- Collections Framework
- Generics
- Lambda Expressions and Functional Interfaces
- Java Database Connectivity (JDBC)
- Junit With Mockito

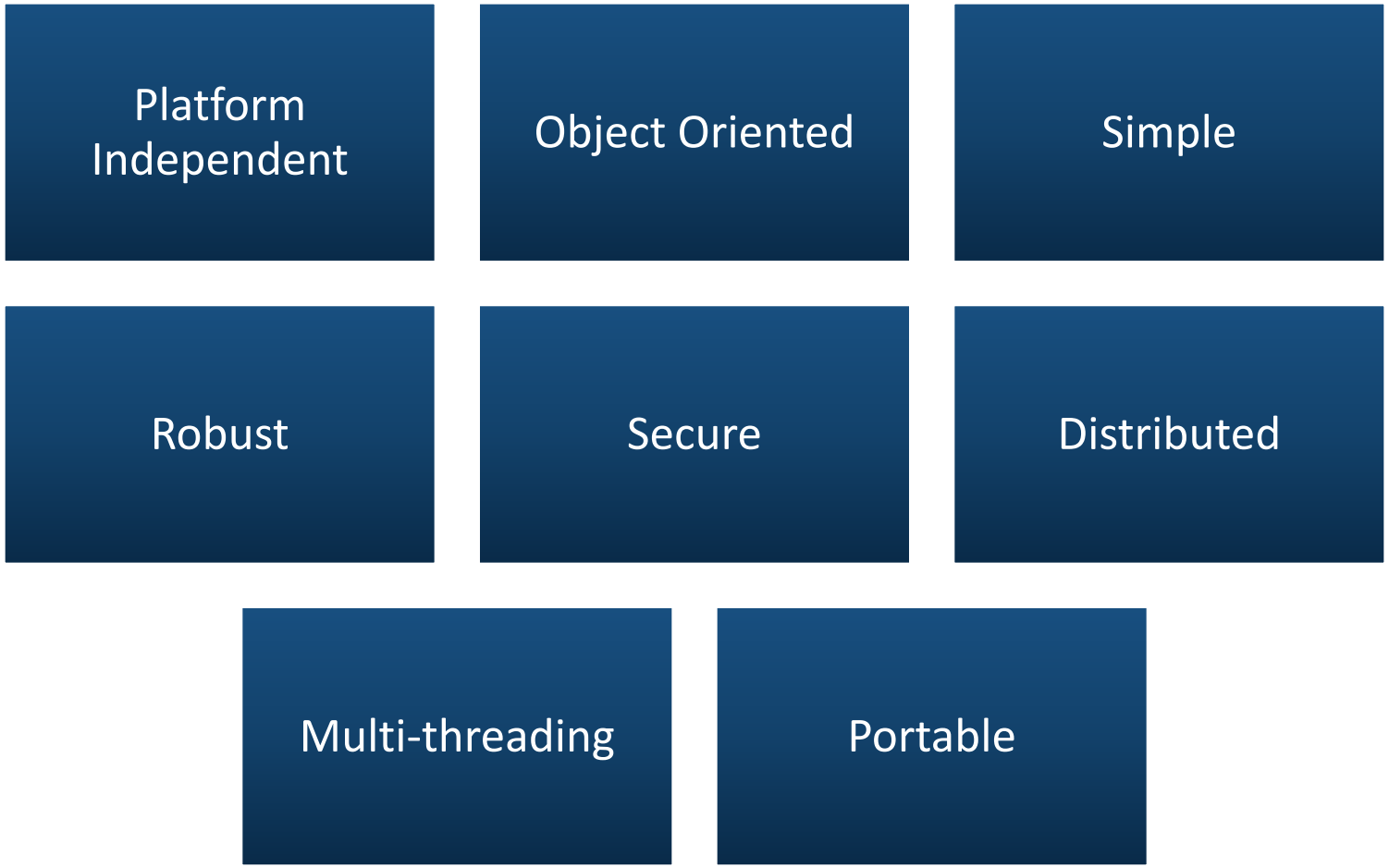
Day 01

- Introduction to Java and its Ecosystem
- Data types, Variables & Operators
- OOPS – Classes and Objects
- String & Date Time
- Array

Java : Introduction

Java is a class-based object-oriented, general-purpose, high-level, platform-independent programming language.

Java : Main Features



Java : Terminologies

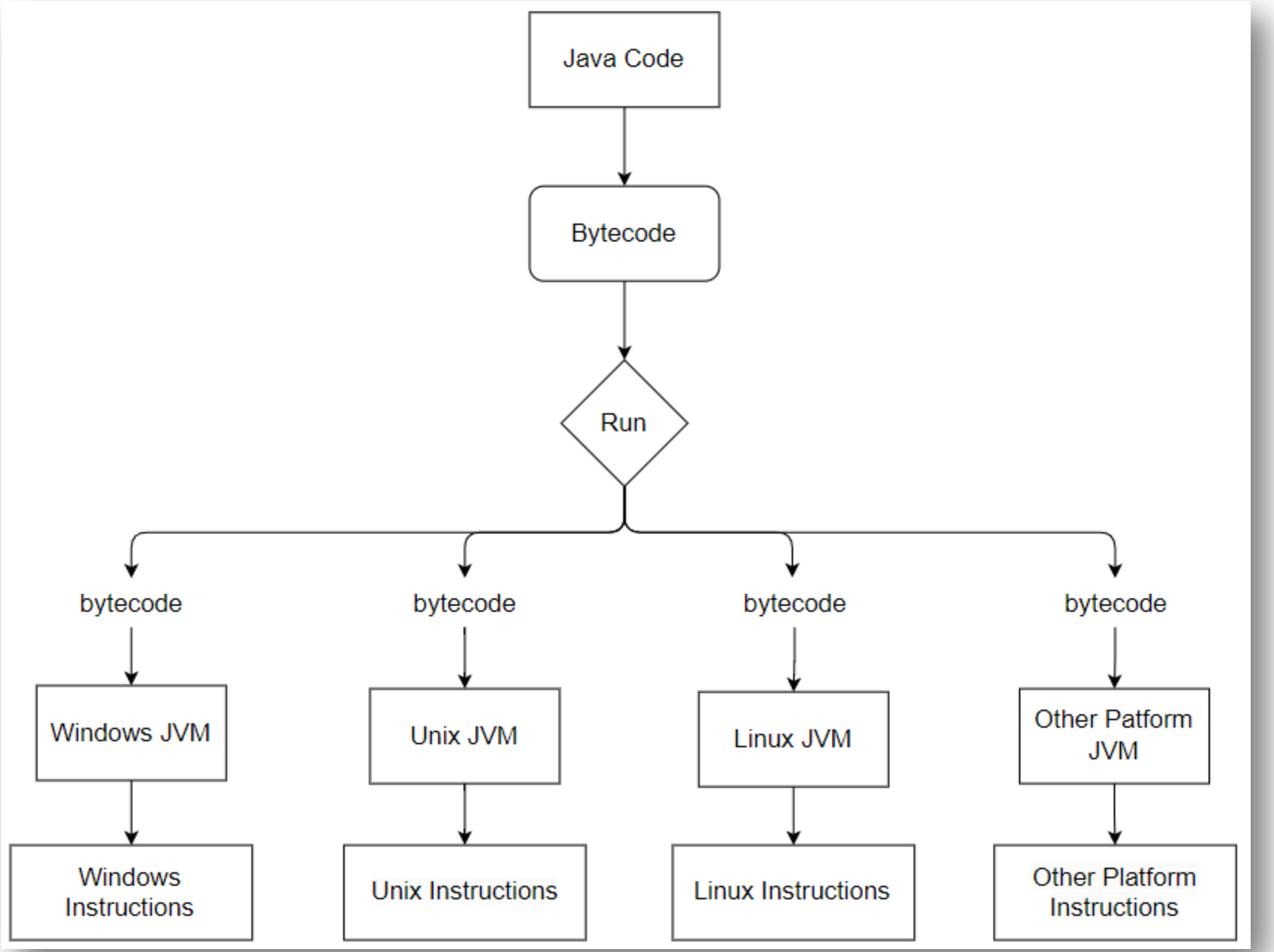
JRE = JVM + Libraries + Other Components

- JVM runs your program bytecode
- Libraries are built in Java utilities that can be used within any program you create. e.g. `System.out.println()` defined in `java.lang`
- Other components include tools for debugging and code profiling (for memory management and performance)

JDK = JRE + Compilers + Debuggers

- JDK refers to Java Development Kit. It's an acronym for the bundle needed to compile and run your Java program.

Terminologies (Contd)



JVM, JRE, JDK (Contd)

Points to Remember

- ✓ JDK is needed to compile and run Java programs
- ✓ JRE is needed to run Java Programs
- ✓ JVM is needed to run bytecode generated from Java Programs

Variables

The syntax rules for variable definition are quite strict. Although we have freedom with the way we name variables.

A variable can be a sequence, in any order, of

- Letters [A-Z, a-z]
- Numerical digits [0 - 9]
- The Special characters ‘$’ (dollar) and ‘_’ (underscore)

With the following exceptions -

- The name can not start with numerical digit [0 - 9]
- The name must NOT match with a predefined Java **keywords**

Primitive Types

Types of Values	Java Primitive Types	Size (in bits)	Range of Values	Example
Integral Values	byte	8	-128 to 127	byte b = 5;
Integral Values	short	16	-32768 to 32767	short s = 128;
Integral Values	int	32	-2,147,483,648 to 2,147,483,647	int i = 40000;
Integral Values	long	64	-9,223,372,036,854,775,808 to -9,223,372,036,854,775,807	long l = 2222222222222

Primitive Types

Types of Values	Java Primitive Types	Size (in bits)	Range of Values	Example
Floating-Point Values	float	32	Approximately ± 3.40282347E+38F. NOT very precise (avoid for financial/scientific Math)	float f = 4.0f;
Floating-Point Values	double	64	Approximately ± 1.7976931348623157E+308. NOT very precise, but better than Float (avoid for financial/scientific Math)	double d = 67.0;
Character Values	char	16	'\u0000 to '\uffff	char c = 'A'
Boolean Values	boolean	1	True or false	boolean isTrue = false;

Exercise : Choosing a Data Type

1	Goals in football match
2	Population of the world
3	Average rainfall in a month
4	Grades of a student
5	Is a number Odd or Even?

Expressions

Java provides a rich set of expressions:

Arithmetic	Relational	Logical
• Addition (+) • Subtraction (-) • Division (/) • Multiplication (*) • Modulus (%)	• Equivalent (==) • Not equivalent (!=) • Less than (<) • Greater that (>) • Less than or equal (<=) • Greater than or equal (>=)	• and (&&) • or (||) • not(!)

Conditionals and Loops

We use the term ‘control flow’ to refer to statement sequencing in the program.

if statement	switch…case statements
while loops	for loops

Exercise : The Menu

Ask the User for input:

- Enter two numbers
- Choose the arithmetic to do on those:
  - Add
  - Subtract
  - Multiply
  - Divide
- Perform the Operation
- Publish the Result

An example scenario could be:

Enter First Number
2

Enter Second Number
4

1 - Add
2 - Subtract
3 - Multiply
4 - Divide

Enter your choice of operation
3

The Result Is : 8

Procedural Programming

Human think in a step by step process. Let's say You have to take a flight from Delhi to Bengaluru-

• Take a cab to Delhi Airport	takeACabToDelhiAirport();
• Check-in	checkIn ()
• Pass security	passSecurity ()
• Board the flight	boardTheFlight ()
• Wish the hostess	wishTheHostess ()
• Take off	takeOff ()
• Cruise	cruise ()
• Land	land ()
• Get off the plane	getOffThePlane()
• Take a cab to destination in Bengaluru	…..

Object Oriented Programming (OOP)

Object oriented programming brings a new thought process. Thinking about various Actors and storing the related data besides them. Giving them responsibilities and let them do their own actions.

Person

- Name
- boardFlight(Plane flight), wishHostess(Hostess hostess), getOffThePlane(Plane flight)

Airplane

- altitude, pilot, speed, flightMode
- takeOff(), cruiseMode(), land()

Hostess

- welcome()

OOP : Terminologies

- ***Class*** – is a template
- ***Object*** – is an instance of Class
- ***Fields*** – are the elements that make up the object state
- ***Methods*** – implements object behaviour

```
class Planet
        name, location, distanceFromSun
        rotate(), revolve()

earth : new Planet
venus: new Planet
```

Exercise : OOP

In each of the following systems, identify the basic entities involved and organize them using object oriented terminology –

- Online Shopping System
- Person

```
Customer
        name, address
        login(), logout(), selectProduct()
ShoppingCart
        items
        addItem(), removeItem()
Product
        name, price, quantity
        order(), changePrice()
```

```
Person
        name, address, hobbies, work
        walk(), run(), sleep(), eat(), drink()
```

Wrapper Classes

Each primitive type in Java has a corresponding built-in wrapper class.

Wrapper classes are also immutable (as well as final)

The main incentives of using such wrappers in your code, are:

- Accessing type information about the corresponding primitive type
- Auto-Boxing feature, where a primitive data is automatically promoted to an object reference type
- Moving primitive type data around data structures (called collections), using their wrapper-style counterparts

Wrapper Classes (Contd)

Java also has a wrapper class corresponding to each of them, which make their primitive versions more versatile -

- Knowing the risk, storing large data into smaller bins is called "**explicit cast**". *The compiler is not responsible for the type-safety.*

- Operations in the other direction (storing a smaller data value in a larger bin), is a piece of cake for the compiler. Such a conversion is called an "**implicit cast"**

Following is a list of built-in Java wrapper classes and their corresponding primitive types:

- byte : Byte
- short : Short
- int : Integer
- long : Long
- float : Float
- double : Double
- char : Character
- boolean : Boolean

Exercise : Wrapper Classes

Create a Java class *`BiNumber`* that stores a pair of integers, and has the following functionality:

- Can be created by passing its initial two numbers to store
- Must support Addition and Multiplication operations on the stored integers
- An operation to double the values of both numbers
- Operations to access each number individually

BigDecimal : Another built-in data type

double and *float* are not very precise representations of floating-point numbers.

In fact, they are not used in computations that require high degrees for accuracy, such as scientific experiments and financial applications.

Try adding 34.56789876 + 34.2234

Accuracy of *BigDecimal* representation is retained only when literals are used to build it.

Exercise : BigDecimal

Write a Program that does a Simple Interest computation for a Principal amount. Recall that the formula for such a calculation is:

```
Total amount (TA) = Principal Amount (PA) + ( PA * Simple Interest (SI)
Rate * Duration In Years (N))
```

Hint -

```
SimpleInterestCalculator calculator = new SimpleInterestCalculator("4500.00", "7.5");
BigDecimal totalValue = calculator.calculateTotalValue(5); //5 year duration
System.out.println(totalValue);
```

String Class

A sequence of characters, such as “hello” “qwerty”, and ”PDF” is very different from other pieces of data.

- In Java, a sequence of characters is typically represented by ***String Class*** .
- String provides several built-in utility methods. e.g length(), charAt(), substring() etc
- String objects are *immutable*. You cannot change their value after they are created.
- **+** can also be used as a String concatenation operator.

Few String Utility Methods

indexOf()	lastIndexOf()	startsWith()	endsWith()
isEmpty()	equals()	equalsIgnoreCase()	Substring()
charAt()	length()	trim()	concat()

Storing Mutable Text

StringBuffer and *StringBuilder* allow you to modify a string literal in-place.

- StringBuffer is thread safe. If you are writing a multi threaded program, use StringBuffer.
- Otherwise, use StringBuilder, since it offers better performance in both execution-time and memory usage.

```
StringBuffer sb = new StringBuffer("TEst");

sb.append(" 123");

sb.setCharAt(1, 'e');

StringBuilder sbldr = new StringBuilder("TEst");
```

The Java Date API

From Java 8, Java provided an API for classes based on the Joda Date Framework.

The three most significant classes within this framework are -

LocalDate, LocalTime and LocalDateTime.

- LocalDate.now() : Returns the current date value in readable format
- LocalTime.now() : Returns the current time value in a readable format
- LocalDateTime.now() : Returns a combination of the current date and time values, in a readable format

Let’s Play With LocalDate

LocalDate provides a number of methods to retrieve Date and Date Meta information -

getYear()

getDayOfWeek ()

getDayOfMonth ()

getDayOfYear ()

getMonth ()

getMonthValue ()

isLeapYear ()

lengthOfYear ()

lengthOfMonth ()

plusDays ()

plusMonths()

plusYears()

minusYears()

withYear(2016)

withDayOfMonth(20)

withMonth(3)

withDayOfYear(3)

isBefore(yesterday)

isAfter(yesterday)

Java Array

Array in Java, is an object which contains fixed number of elements of a similar data type.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces.

```
int[] marks = {1, 2, 3 };
int[] marks2 = new int[5];
marks2[0] = 10;
```

Java Array (Contd)

Advantages

- Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.
- Random access: We can get any data located at an index position.

Disadvantages

- Size Limit: We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

Exercise : Array

1. Write a program that creates an array *marks* to store 8 **int** values, and code to iterate through *marks* using a for loop, printing out its values.

1. We would like to model a student report card in a Java program, which allows the user to do stuff such as:

```
Student student - new Student(name, list-of-marks);
int number = student.getNumberOfmarks();
int sum = student.getTotalSumOfMarks();
int maximumMark = student.getMaximumMark();
int minimumMark = student.getMinimumMark();
BigDecimal average = student.getAverageMarks();
student.addMark(35); student.removeMarkAtIndex(5);
```

Day 01

Conclusion

- Java Introduction & Ecosystem(JVM, JRE, JDK)
- Variables & Data Types
- Conditionals and Loops
- Wrapper Classes
- Reference Types
- Working with Strings & Date
- Arrays in Java