JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Course Objectives:

The objective of the course is to

- Introduce the basic concepts of programming with Java
- Understand the OOP concepts with respect to Java
- Understand Database connectivity with Java using JDBC
- Introduce GUI programming and Java Swing

Course Outcomes:

At the end of the course, the student will be able to

- Apply the knowledge of Java Programming concepts to solve different problems
- Implement OOP concepts and represent real world objects
- Establish connectivity between Java and databases
- Design and develop GUI Applications using Java Swing with Database

JAVA TECHNOLOGIES

Course Overview:

Java is a widely used programming language and it is one of the go-to object-oriented programming languages. It is very important for the students to understand various aspects of Java programming and appreciate the uniqueness of the language. The course focuses on getting the students familiarized with various aspects of Java programming. The course also explores OOP with Java, Database Connectivity, and GUI design and development using Java, making it an important course in the journey of being a developer.

JAVA TECHNOLOGIES

Course Content:

Unit 1: Java Basics

Introduction to Java, Timeline, Features, Advantages of Java, JDK, JVM, JRE, The First Java Program, Keywords, Variables, Datatypes, Wrapper Classes, Autoboxing, Unboxing, Typecasting, Operators, Overview of Branching Statements and Loops, Methods, Strings and Arrays in Java

Experiential Learning: Installation, Programs based on decision making and branching, Programs based on loops, String and Array Handling

JAVA TECHNOLOGIES

Course Content:

Unit 2: OOP with Java

Introduction to OOP: OOP Concepts, OOP vs. POP, Advantages, Java API's, Classes and Objects, Constructor, Inheritance, Types of Inheritance, Interfaces, Polymorphism, Method Overloading vs. Method Overriding, Exception Handling: Structure, Checked and Unchecked, Custom Exceptions, Finally Block,Threads and Runnable

Experiential Learning: Object-oriented programs in Java using various OOP concepts

JAVA TECHNOLOGIES

Course Content:

Unit 3: Java Database Connectivity (JDBC)

Annotations: Built-in, JDBC Drivers and Types, Communicating with Database using Java; Database Operations using Java: Understanding CRUD Operations, Classes and Interfaces of JDBC API, Creation and Retrieval of Data using JDBC

Experiential Learning: Developing programs using JDBC and communicate with MySQL Database

JAVA TECHNOLOGIES

PES UNIVERSITY
CELEBRATING 50 YEARS

Course Content:

Unit 4: GUI Programming using Swing

Introduction to GUI Programming, Components of GUI; Swing: Introduction, Features of Swing, Swing Containers, Swing Components, Mouse and Keyboard Events, A Simple Swing Application, Swing Application with JDBC

Experiential Learning: Developing GUI Application using Swing, Developing Database Application using Swing and MySQL

JAVA TECHNOLOGIES

Text Books:

1. H. Schildt, Java: The Complete Reference, 12 th Ed. McGraw Hill Education, 2021.
2. P. Deitel and H. Deitel, Java: How to Program – Early Objects, 11 th Ed. Pearson, 2021.

Reference Books:

1. H. Schildt, Java: A Beginner’s Guide, 8 th Ed. McGraw Hill Education, 2018
No latest edition/reprint available.

JAVA TECHNOLOGIES

History of JAVA

- Designed and developed in the mid-1990s by James A. Gosling a former computer scientist with Sun Microsystems.

- The purpose was to allow programmers to code and produce software for multiple platforms. The compiled code, bytecode, runs on most operating systems (OS), including Windows, Linux and Mac OS.

JAVA TECHNOLOGIES

- The history of java starts from Green Team. Java team members (also known as Green Team), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

- James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of sun engineers called **Green Team.**

- Originally designed for small, embedded systems in electronic appliances like settop boxes. Firstly, it was called "**Greentalk"** by James Gosling and file extension was .gt.

- After that, it was called **Oak** and was developed as a part of the Green project.

JAVA TECHNOLOGIES

- Later in 2009, **Oracle** Corp. acquired Java. During this period Java also changes its logo.The visual identity of the well-known programming language is fun and can be recognized in a flash.

- The cup of coffee in blue and red shading mix has been with the brand since its starting and was just upgraded once, in 2003, keeping the original sense and color shading range.

- The logo was a recognition for the Java engineers, who have a lot of coffee while developing the Java programming language. The coffee that they have consumed was **Java coffee beans.** It is a variety of coffee. Java coffee beans is a wet-processed (washed) coffee grown on the island of Java in **Indonesia.** So, the name of the programming language and the visible identity of the language were picked from the Java coffee beans by the founder of the Java programming language **James Arthur Gosling.**

JAVA TECHNOLOGIES

JAVA TIME LINE

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan, 1996)
3. JDK 1.1 (19th Feb, 1997)
4. J2SE 1.2 (8th Dec, 1998)
5. J2SE 1.3 (8th May, 2000)
6. J2SE 1.4 (6th Feb, 2002)
7. J2SE 5.0 (30th Sep, 2004)
8. Java SE 6 (11th Dec, 2006)
9. Java SE 7 (28th July, 2011)
10. Java SE 8 (18th March, 2014)

JAVA TECHNOLOGIES

11. Java SE 9 – September 2017
12. Java SE 10 (18.3) – March 2018
13. **Java SE 11 (18.9 LTS) – September 2018**
14. Java SE 12 (19.3) – March 2019
15. Java SE 13 – September 2019
16. Java SE 14 – March 2020
17. Java SE 15 – September 2020
18. Java SE 16 – March 2021
19. **Java SE 17 (LTS) – September 2021**
20. Java SE 18 – March 2022
21. Java SE 19 – September 2022
22. Java SE 20 – March 2023
23. **Java SE 21 (LTS) – September 2023**
24. **Java 24** (March 2025)

JAVA TECHNOLOGIES

Open JDK and Oracle JDK

Oracle JDK

Oracle strongly recommends using the term JDK to refer to the Java SE (Standard Edition) Development Kit (there are also Enterprise Edition and Micro Edition platforms).

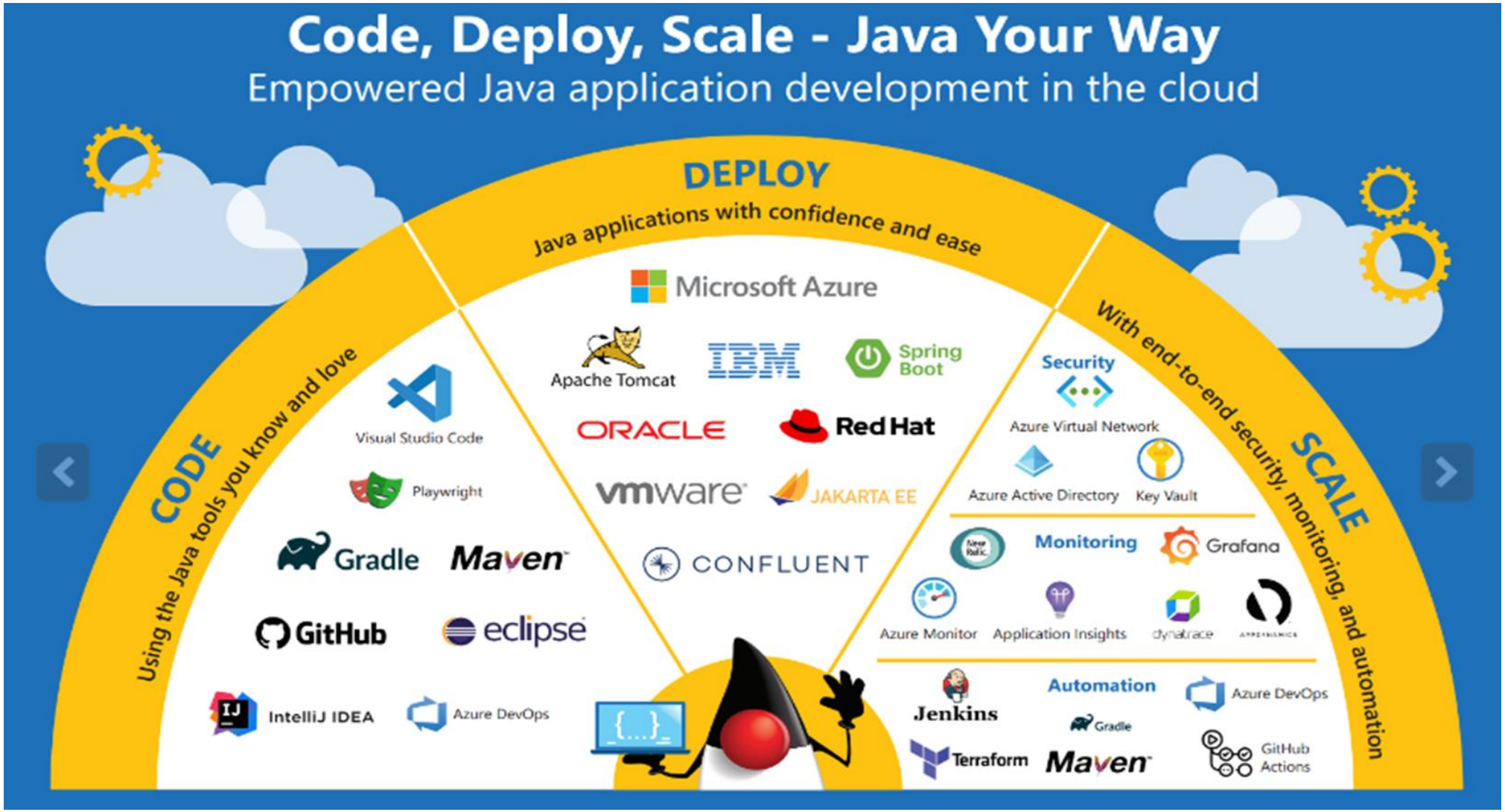
Oracle will deliver releases every three years

Open JDK

OpenJDK is a free and open-source implementation of the Java SE Platform Edition. It was initially released in 2007 as the result of the development that Sun Microsystems started in 2006.

OpenJDK will be released every six months.

JAVA TECHNOLOGIES



JAVA TECHNOLOGIES

Introduction to Java

Java is a high-level, object-oriented, platform-independent programming language developed by Sun Microsystems and later maintained by Oracle. It is widely used for developing desktop applications, web applications, mobile apps, enterprise systems, and embedded systems.

JAVA TECHNOLOGIES

Object oriented programming Languages

Languages with OOP support include Ada, ActionScript, C++, Common Lisp, C#, Dart, Eiffel, Fortran 2003, Haxe, Java, JavaScript, Kotlin, Logo, MATLAB, Objective-C, Object Pascal, Perl, PHP, Python, R, Raku, Ruby, Scala, SIMSCRIPT, Simula, Smalltalk, Swift, Vala and Visual Basic (. NET).

Pure Object Oriented Programming Languages

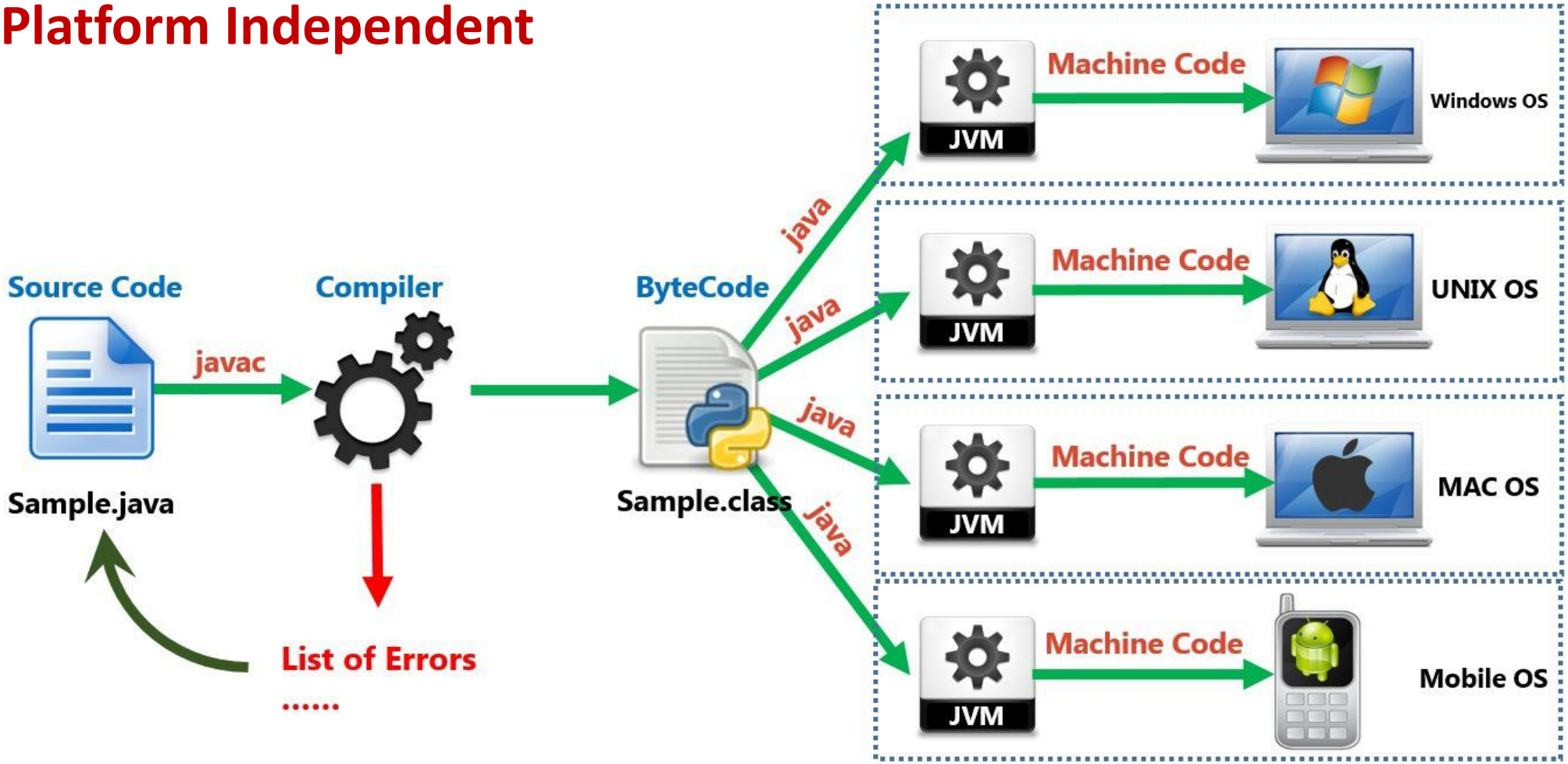
Smalltalk, Squeak, Pharo, Self, Erlang, and CLOS (Common Lisp Object System) are purely object-oriented languages.

JAVA TECHNOLOGIES

Procedural Programming Language	Object Oriented Programming Language
1. Program is divided into functions.	1. Program is divide into classes and objects..
2. The emphasis is on doing things.	2. The emphasis on data.
3. Poor modeling to real world problems.	3. Strong modeling to real world problems.
4. It is not easy to maintain project if it is too complex.	4. It is easy to maintain project even if it is too complex.
5. Provides poor data security.	5. Provides strong data Security.
6. It is not extensible programming language.	6. It is highly extensible programming language.
7. Productivity is low.	7. Productivity is high.
8. Do not provide any support for new data types.	8. Provide support to new Data types.
9. Unit of programming is function.	9. Unit of programming is class.
10. Ex. Pascal , C , Basic , Fortran.	10. Ex. C++ , Java , Oracle.

JAVA TECHNOLOGIES

Platform Independent



JAVA TECHNOLOGIES

Features of Java

1. Simple – Easy to learn and use compared to C/C++
2. Object-Oriented – Uses concepts like class, object, inheritance, polymorphism
3. Platform Independent – Runs on any OS using JVM
4. Secure – No pointers, bytecode verification, security manager
5. Robust – Strong memory management and exception handling
6. Multithreaded – Supports concurrent execution
7. High Performance – Uses Just-In-Time (JIT) compiler
8. Distributed – Supports networking and distributed computing
9. Dynamic – Supports dynamic class loading

JAVA TECHNOLOGIES

Advantages of Java

Technical Advantages

- Platform independence
- Automatic garbage collection
- Strong standard library (API)
- High security
- Excellent exception handling

Development Advantages

- Large developer community
- Rich IDE support (Eclipse, IntelliJ, NetBeans)
- Easy maintenance and scalability

Usage Advantages

- Widely used in enterprise applications
- Popular in Android development
- Suitable for web, desktop, and cloud applications

JAVA TECHNOLOGIES

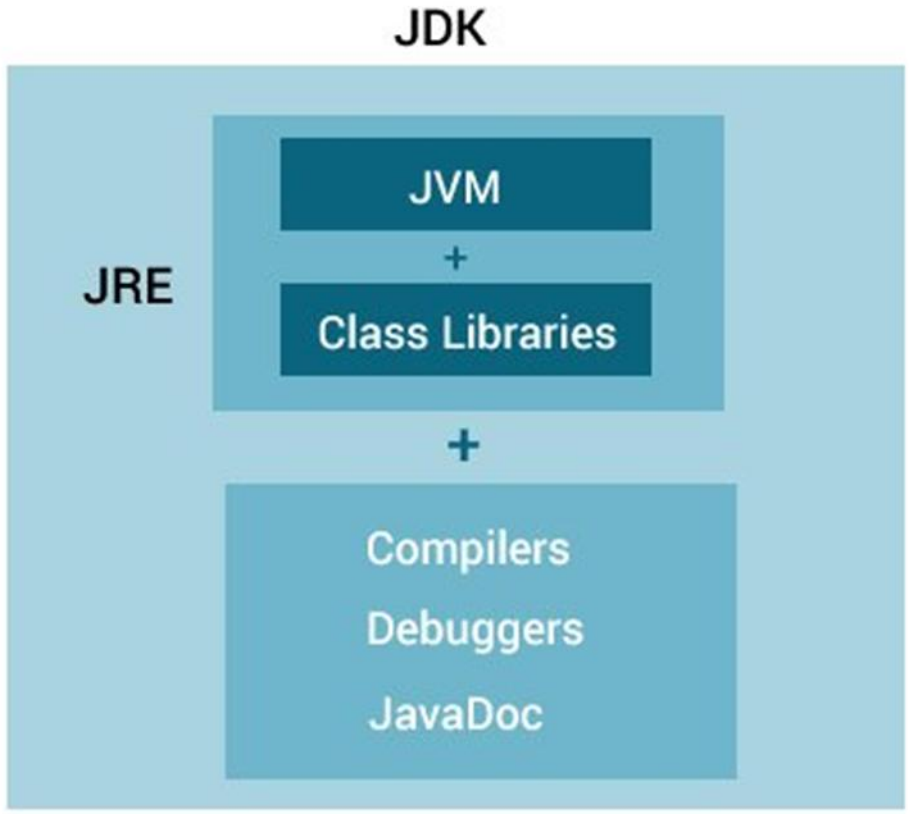
SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

JDK, JVM, JRE



- JDK is used to develop Java programs
- JRE is used to run Java programs
- JVM is used to execute Java bytecode

JAVA TECHNOLOGIES

JDK, JVM, JRE

Bootstrap: Loads std. classes like lang, util, io etc. (lib folder)

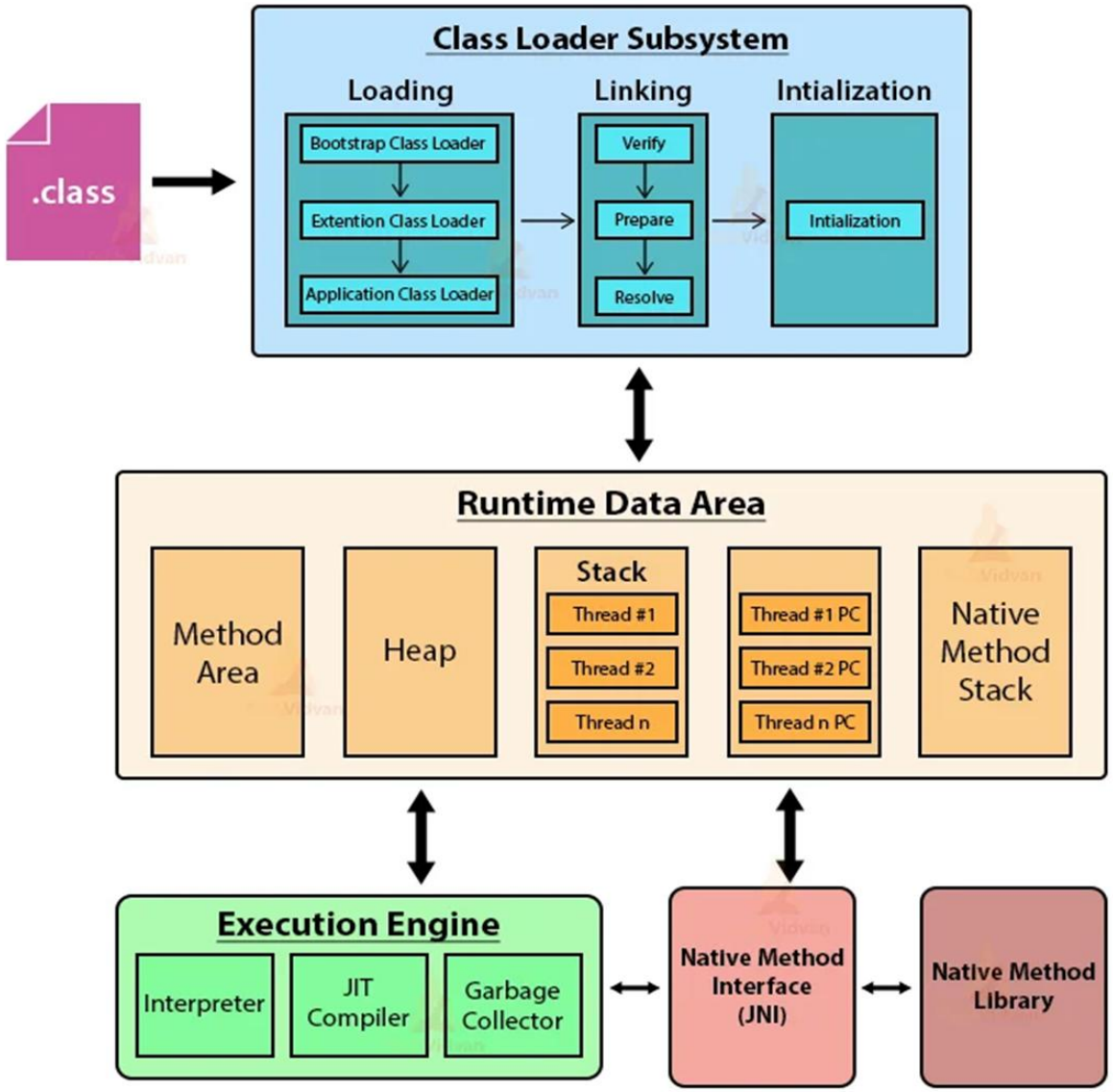
Extention: std. libraries (lib/ext folder)

Application: libraries from ClassPath-current dir

Verify: Java 11 on Java 8?

Prepare: init. Static variables

Resolve: Symbolic constants from other classes/references



JAVA TECHNOLOGIES

JDK (Java Development Kit)

Definition

The Java Development Kit (JDK) is a complete software package used to develop, compile, debug, and run Java programs.
It provides all the tools required by a Java developer.

- Without JDK, you cannot create Java applications.

Why JDK is Needed

- To write Java source code
- To compile .java files into .class files
- To run Java programs
- To debug and document Java applications

JVM (Java Virtual Machine)

Definition

The Java Virtual Machine (JVM) is an abstract machine that executes Java bytecode and enables platform independence.It acts as an intermediate layer between Java programs and the operating system.

Why JVM is Needed

- To run Java programs on any operating system
- To provide security
- To manage memory automatically
- To convert bytecode into machine-specific instructions

JRE (Java Runtime Environment)

Definition

The Java Runtime Environment (JRE) is a software package that provides the environment required to run Java applications. It is meant for executing Java programs, not for developing or compiling them.

Why JRE is Needed

- To run Java programs on any system
- To provide required libraries and runtime support
- To act as a bridge between Java applications and the operating system
- Without JRE, a Java program cannot run.

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Keywords in Java

Definition

Keywords are reserved words in Java that have predefined meanings and cannot be used as identifiers (variable names, class names, method names).

Characteristics

- Always written in lowercase
- Have fixed functionality
- Cannot be redefined by the programmer

JAVA TECHNOLOGIES

Categories of Java Keywords

a) Data Type Keywords

int, float, double, char, boolean, byte, short, long, void

b) Control Flow Keywords

if, else, switch, case, for, while, do, break, continue, return

c) Access Modifiers

public, private, protected, default

d) Class & Object Related

class, interface, extends, implements, new, this, super, abstract, final, static

e) Exception Handling

try, catch, finally, throw, throws

f) Other Important Keywords

package, import, instanceof, enum, assert, synchronized, volatile, transient

Variables in Java

Definition

A variable is a named memory location used to store data values that can change during program execution.

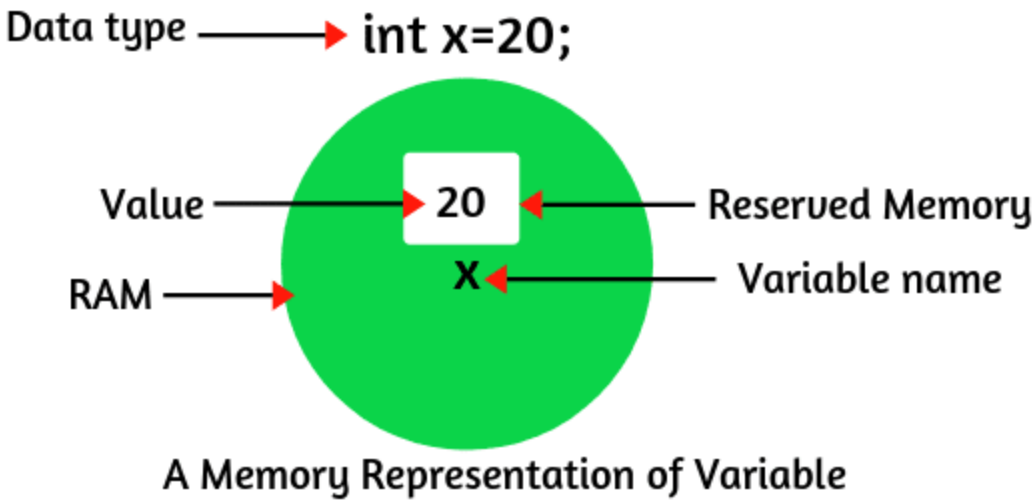
Syntax:

datatype variableName = value;

Example:

int age = 20;

float salary = 35000.50f;



A Memory Representation of Variable

Variables in Java

Java is a statically and strongly typed language, the type of every variable is determined at compile time, which helps catch many errors before runtime.

- The term "**Statically typed**" means the type of variable is known and checked by the Java compiler before the execution of program.
- The term "**Strongly typed**" means Java enforces strict type rules, preventing operations between incompatible types unless explicit type conversion is performed.

Types of Variables

a) Local Variables

- Declared inside methods/blocks
- Must be initialized before use

```
void display() {
   int x = 10;
}
```

Types of Variables

b) Instance Variables

- Declared inside a class but outside methods
- Belong to an object

```
class Student {
   int id;
}
```

Types of Variables

c) Static Variables

- Declared using static
- Shared among all objects

```
class College {
   static String name = "ABC College";
}
```

Types of Variables and Their Memory Storage

Type of Variable	Declared	Stored In	Lifetime
Local Variable	Inside method / block	Stack Memory	Until method execution ends
Instance Variable	Inside class, outside methods	Heap Memory (inside object)	As long as object exists
Static Variable	Inside class using static	Method Area / Class Area	Till program ends

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Data Types in Java

Definition

A **data type** specifies:

- Type of data a variable can hold
- Amount of memory allocated
- Valid operations on the data

Types of Data Types in Java

Java data types are broadly classified into two types:

1. Primitive Data Types
2. Non-Primitive (Reference) Data Types

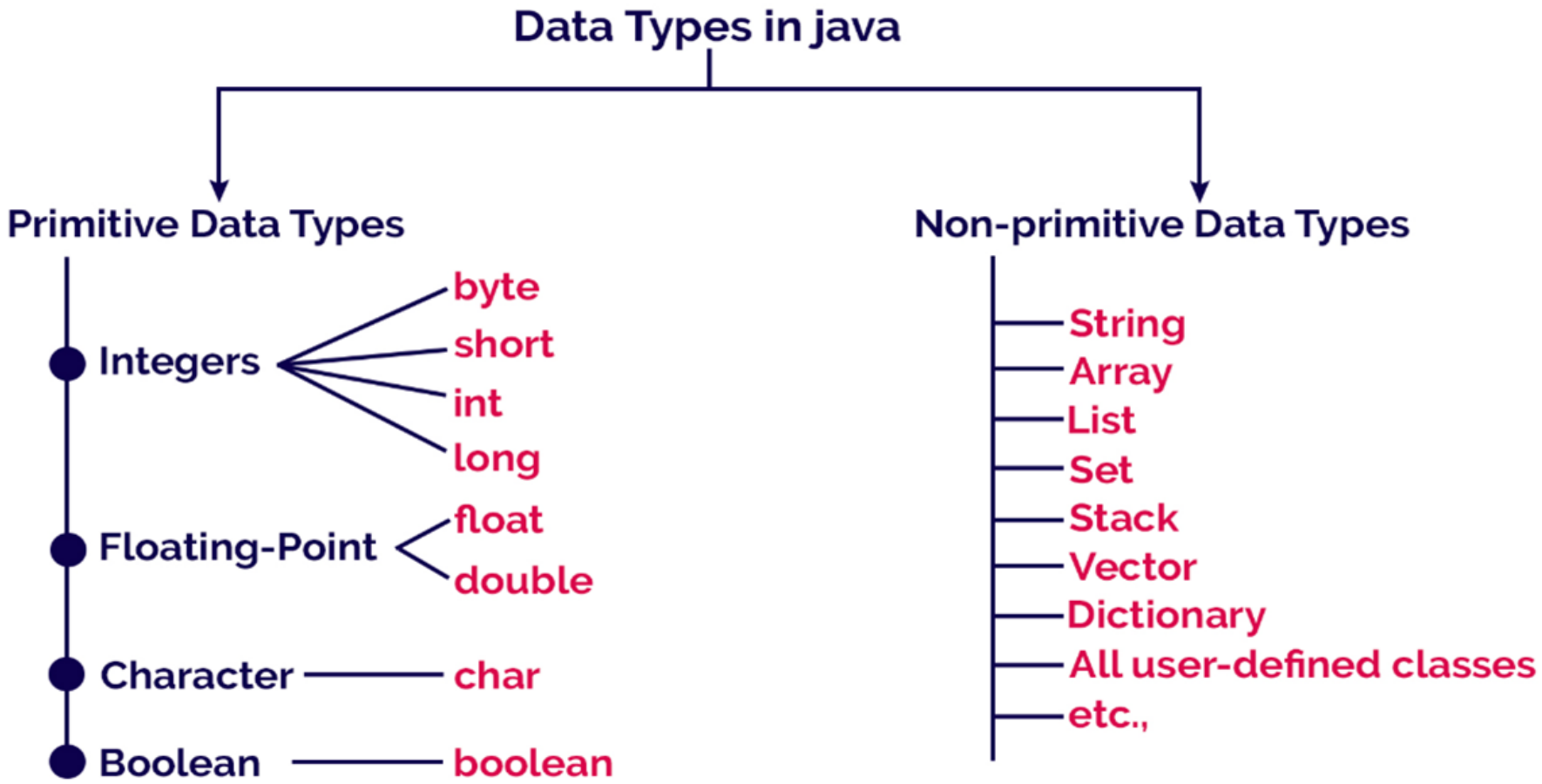
1. Primitive Data Types

Primitive data types are basic built-in data types provided by Java. They store simple values and are not objects.

a) Integer Data Types
b) Floating-Point Data Types
c) Character Data Type
d) Boolean Data Type

JAVA TECHNOLOGIES

Data Types in Java



PES UNIVERSITY
CELEBRATING 50 YEARS

Data Types in Java

Table: Java's built-in data types

Type	Contains	Default	Size	Range
boolean	True or false	False	1 bit	NA
byte	Signed integer	0	8bits	-128 to 127
char	Unicode Character	'\u0000'	16 bits	\u0000 to \uFFFF
double	IEEE 754 floating point	0.0d	64 bits	±4.9E –324 to ±1.7976931348623157E+308
float	IEEE 754 floating point	0.0f	32 bits	±1.4E – 45 to ± 3.4028235E+38
int	Signed Integer	0	32 bits	-2147483648 to 2147483647
long	Signed Integer	0L	64 bits	-9223372036854775808 to 9223372036854775807
short	Signed Integer	0	16 bits	-32768 to 32767

Note: Java defines four integer types: **byte, short, int,** and **long**

2. Non-Primitive Data Types:

Non-primitive data types are user-defined or predefined data types that store references to objects.

a) String

- Used to store sequence of characters
- String is an object in Java

Example:

String name = “Priya";

b) Array

- Used to store multiple values of same data type
- Size is fixed

Example:

int marks[] = {80, 85, 90};

Wrapper Class

- Since the Primitive Data Types cannot be directly used as objects that's why Wrapper classes come into picture.
- Generic classes work with objects and don't support Primitives. As a result, Wrapper classes are needed as they convert primitive data types into objects
- Java wrapper classes wraps or represents the values of primitive data types as an object. When an object is created using a wrapper class, it contains a field which can store the primitive data types.
- Each primitive type (int, char, double, etc.) has a corresponding wrapper class in java.lang package

Wrapper Class

Primitive Data Type	Size	Wrapper Class	Example (Primitive)	Example (Wrapper)
byte	1 byte	Byte	byte b = 10;	Byte bObj = 10;
short	2 bytes	Short	short s = 100;	Short sObj = 100;
int	4 bytes	Integer	int i = 25;	Integer iObj = 25;
long	8 bytes	Long	long l = 5000L;	Long lObj = 5000L;
float	4 bytes	Float	float f = 5.5f;	Float fObj = 5.5f;
double	8 bytes	Double	double d = 45.75;	Double dObj = 45.75;
char	2 bytes	Character	char c = 'A';	Character cObj = 'A';
boolean	JVM dependent	Boolean	boolean flag = true;	Boolean flagObj = true;

Wrapper Class

Example 1: Wrapper Class Boxing and UnBoxing

```
class s {
public static void main(String args[])  {
int a = 10;
Integer obj = Integer.valueOf(a); // Boxing
System.out.println(obj);

int b = obj.intValue();     // Unboxing
System.out.println(b);
}
}
```

Example 2: Wrapper Class Auto Boxing and UnBoxing

```
class s {
public static void main(String args[])  {
int num=2;
Integer num1=num; // Auto boxing
System.out.println(num1);
int num2=num1; // Auto Unboxing
System.out.println(num2);
String str="11";
int num3=Integer.parseInt(str);
System.out.println(num3);
}  }
```

JAVA TECHNOLOGIES

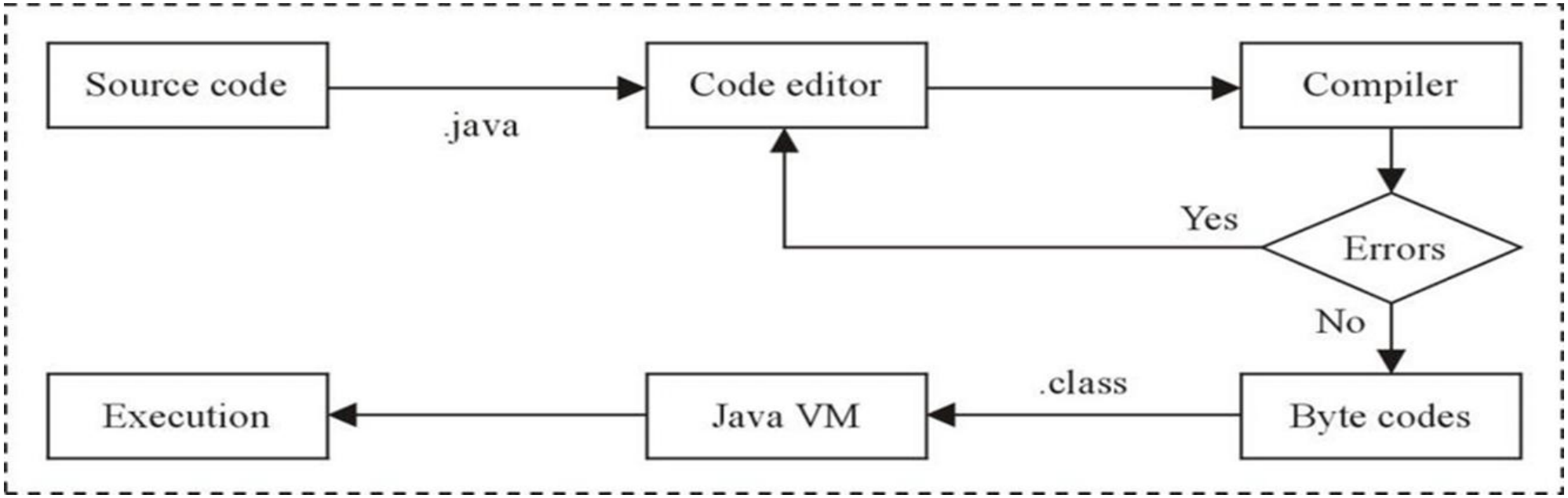
SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Java Program Execution Flow



Java follows the principle of"Write Once, Run Anywhere **(WORA)",**meaning a Java program compiled on one platform can run on any platform that has a Java Virtual Machine (JVM).

Simple Java Program

```
class HelloWorld {
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

Output: Hello World

Explanation

class HelloWorld

Defines a class.

Class name must match the file name (HelloWorld.java).

- public static void main(String[] args)

Main method: execution starts here.

- public → accessible to JVM
- static → no object needed
- void → no return value
- String[] args → command-line arguments
- { } Curly braces define the block of code.

System.out.println("Hello, World!");

- Prints output to the console.

JAVA TECHNOLOGIES

Experiential Learning:1

Software Installation :

Refer the Files for installation of following software.

1. **JDK Installation (latest Version)**
2. **VS Code**

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Experiential Learning:2

1. Write Java Program to add two numbers
2. Write Java Program to calculate simple interest
3. Write a Java program to declare and display values of different primitive data types such as int, float, double, char, and boolean.
4. Write Java Program to Convert Celsius to Fahrenheit.
5. Write a Java program to swap two numbers using a temporary variable and appropriate data types.

Experiential Learning:

1. **Write Java Program to add two numbers**

```
class AddTwoNumbers {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int sum = a + b;

        System.out.println("Sum = " + sum);
    }
}
```

2. **Write Java Program to calculate simple interest**

```
class SimpleInterest {
    public static void main(String[] args) {
        float principal = 1000;
        float rate = 5;
        float time = 2;

        float simpleInterest = (principal * rate * time) / 100;

        System.out.println("Simple Interest = " + simpleInterest);
    }
}
```

3. **Write a Java program to declare and display values of different primitive data types such as int, float, double, char, and boolean.**

```
class PrimitiveDataTypes {
    public static void main(String[] args) {
        int number = 25;
        float percentage = 85.5f;
        double salary = 45000.75;
        char grade = 'A';
        boolean result = true;
        System.out.println("Integer value: " + number);
        System.out.println("Float value: " + percentage);
        System.out.println("Double value: " + salary);
        System.out.println("Character value: " + grade);
        System.out.println("Boolean value: " + result);   }   }
```

4. Write Java Program to Convert Celsius to Fahrenheit.

```
class CelsiusToFahrenheit {
    public static void main(String[] args) {
        float celsius = 30;
        float fahrenheit = (celsius * 9 / 5) + 32;

        System.out.println("Temperature in Fahrenheit = " + fahrenheit);
    }
}
```

5. Write a Java program to swap two numbers using a temporary variable and appropriate data types.

```
class SwapNumbers {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int temp;
        temp = a;
        a = b;
        b = temp;
        System.out.println("After Swapping:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
    } }
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Experiential 3

1. Write a Java program to demonstrate Autoboxing.
2. Write a Java program to demonstrate Unboxing.
3. Write a program where you add two Integer objects, but Java internally unboxes them to perform addition, and then autoboxes the result back into an Integer.
4. Write a Java program to convert a primitive data type to a wrapper object.
5. Write a Java program to convert a wrapper object to primitive data type.

Experiential 3

1. Write a Java program to demonstrate Autoboxing.

```
class AutoBoxingDemo {
    public static void main(String[] args) {
        int a = 10;
        Integer obj = a;   // Autoboxing

        System.out.println("Primitive value: " + a);
        System.out.println("Wrapper object: " + obj);
    }
}
```

Experiential 3

2. Write a Java program to demonstrate Unboxing.

```
class UnboxingDemo {
    public static void main(String[] args) {
        Integer obj = 25;
        int a = obj;   // Unboxing

        System.out.println("Wrapper object: " + obj);
        System.out.println("Primitive value: " + a);
    }
}
```

3. Write a program where you add two Integer objects, but Java internally unboxes them to perform addition, and then autoboxes the result back into an Integer.

```
class IntegerAddition {
    public static void main(String[] args) {
        Integer a = 15;
        Integer b = 20;

        Integer sum = a + b;  // Unboxing → addition → Autoboxing

        System.out.println("Sum = " + sum);
    }
}
```

4. Write a Java program to convert a primitive data type to a wrapper object.

```
class PrimitiveToWrapper {
    public static void main(String[] args) {
        int x = 50;
        Integer obj = Integer.valueOf(x);

        System.out.println("Primitive: " + x);
        System.out.println("Wrapper: " + obj);
    }
}
```

5. Write a Java program to convert a wrapper object to primitive data type.

```
class WrapperToPrimitive {
    public static void main(String[] args) {
        Integer obj = Integer.valueOf(100);
        int x = obj.intValue();

        System.out.println("Wrapper: " + obj);
        System.out.println("Primitive: " + x);
    }
}
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Wrapper Class

- Since the Primitive Data Types cannot be directly used as objects that's why Wrapper classes come into picture.
- Generic classes work with objects and don't support Primitives. As a result, Wrapper classes are needed as they convert primitive data types into objects
- Java wrapper classes wraps or represents the values of primitive data types as an object. When an object is created using a wrapper class, it contains a field which can store the primitive data types.
- Each primitive type (int, char, double, etc.) has a corresponding wrapper class in java.lang package

Wrapper Class

Primitive Data Type	Size	Wrapper Class	Example (Primitive)	Example (Wrapper)
byte	1 byte	Byte	byte b = 10;	Byte bObj = 10;
short	2 bytes	Short	short s = 100;	Short sObj = 100;
int	4 bytes	Integer	int i = 25;	Integer iObj = 25;
long	8 bytes	Long	long l = 5000L;	Long lObj = 5000L;
float	4 bytes	Float	float f = 5.5f;	Float fObj = 5.5f;
double	8 bytes	Double	double d = 45.75;	Double dObj = 45.75;
char	2 bytes	Character	char c = 'A';	Character cObj = 'A';
boolean	JVM dependent	Boolean	boolean flag = true;	Boolean flagObj = true;

Wrapper Class

Example 1: Wrapper Class Boxing and UnBoxing

```
class s {
public static void main(String args[])  {
int a = 10;
Integer obj = Integer.valueOf(a); // Boxing
System.out.println(obj);

int b = obj.intValue();     // Unboxing
System.out.println(b);
}
}
```

Example 2: Wrapper Class Auto Boxing and UnBoxing

```
class s {
public static void main(String args[])  {
int num=2;
Integer num1=num; // Auto boxing
System.out.println(num1);
int num2=num1; // Auto Unboxing
System.out.println(num2);
String str="11";
int num3=Integer.parseInt(str);
System.out.println(num3);
}  }
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Type Casting in Java

Type casting in Java is the process of converting a variable from one data type to another.
Java is a strongly typed language, so type conversion must follow strict rules.

Types of Type Casting in Java

Java supports two main types of type casting:

1. Implicit Type Casting (Widening)
2. Explicit Type Casting (Narrowing)

Implicit Type Casting (Widening)

Implicit type casting occurs automatically when a smaller data type is converted into a larger data type.

- No data loss
- Done by the compiler
- Safe conversion

Order of Widening

byte → short → int → long → float → double

Implicit Type Casting (Widening)

Example:

```
class WideningExample {
    public static void main(String[] args) {
        int a = 10;
        double b = a;   // implicit casting
        System.out.println(b);
    }
}
```

Explicit Type Casting (Narrowing Conversion)

Explicit type casting is done **manually** when converting a **larger data type** into a **smaller data type**.

- Possible data loss
- Programmer must specify the cast
- Syntax uses (type)

Syntax :

targetType variable = (targetType) value;

Explicit Type Casting (Narrowing Conversion)

Example

```
class NarrowingExample {
    public static void main(String[] args) {
        double a = 10.75;
        int b = (int) a;   // explicit casting
        System.out.println(b);
    }
}
```

Type Casting with Characters

Characters are internally stored as **Unicode values**.

Example

```
class CharCasting {
    public static void main(String[] args) {
        char ch = 'A';
        int num = ch;
        System.out.println(num);
    }
}
```

Type Casting in Expressions

Java automatically **promotes smaller data types to int** while evaluating expressions.

Example

```
class PromotionExample {
    public static void main(String[] args) {
        byte a = 10;
        byte b = 20;
        int result = a + b;  // promoted to int
        System.out.println(result);
    }
}
```

Casting Between Objects (Reference Type Casting)

It is the process of converting one class type reference into another class type reference within the same inheritance hierarchy.

- Works only in inheritance / interface hierarchy
- Applies only to objects, not primitives.
- JVM checks casting at runtime

Types of Reference Type Casting

1. Upcasting //Upcasting is **implicit**
2. Downcasting //Downcasting is **explicit**

1. Which type casting is applied for the following code?

```
int a = 10;
double b = a;
```

A. Narrowing
B. Explicit casting
C. Implicit casting
D. Illegal casting

C. Implicit casting

What is the result of the following code?

```
double d = 12.99;
int i = (int) d;
System.out.println(i);
```

A. 12
B. 13
C. 12.99
D. Compilation error

Answer: A

What is the output of the following code?

```
byte a = 10;
byte b = 20;
byte c = (byte)(a + b);
System.out.println(c);
```

A. Compilation error
B. 30
C. 10
D. 20

Answer: B

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Conditional and Looping Statements

- Conditional statements: if, if-else, switch
- Iterative statements : for, while, do-while
- Transfer statements : break, continue, go to, return, try

Conditional Statements

```
if ( condition ) {

    statements…

}
```

```
if ( condition ) {
    statements…
}
else
{
    statements
}
```

JAVA TECHNOLOGIES

Practice programs on if Condition

1. Program to check whether a number is positive
2. Program to check whether a number is even or odd
3. Program to check voting eligibility
4. Program to find the largest of two numbers
5. Program to check pass or fail
6. Program to find the largest of three numbers
7. Program to verify login credentials using if condition

1. Program to check whether a number is positive

```
class PositiveCheck {
    public static void main(String[] args) {
        int n = 10;

        if (n > 0) {
            System.out.println("Number is Positive");
        }
    }
}
```

2. Program to Check Whether a Number is Even or Odd

```
class EvenOdd {
    public static void main(String[] args) {
        int n = 8;

        if (n % 2 == 0) {
            System.out.println("Even Number");
        } else {
            System.out.println("Odd Number");
        }
    }
}
```

3. Program to Check Voting Eligibility

```
class VotingEligibility {
    public static void main(String[] args) {
        int age = 20;

        if (age >= 18) {
            System.out.println("Eligible to Vote");
        } else {
            System.out.println("Not Eligible to Vote");
        }
    }
}
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Syntax of looping Statements (do-while, while and for loop)

```
do {

   // statements to be executed

} while (condition);
```

```
for (initialization; condition; update) {

// statements to execute

}
```

```
while (condition) {

      //statements;
}
```

Program 1: Print Numbers from 1 to 10

```
class PrintNumbers {
    public static void main(String[] args) {
        int i = 1;

        while (i <= 10) {
            System.out.println(i);
            i++;
        }
    }
}
```

Program 2: Find Sum of First 10 Natural Numbers

```
class SumNatural {
    public static void main(String[] args) {
        int i = 1, sum = 0;

        while (i <= 10) {
            sum = sum + i;
            i++;
        }

        System.out.println("Sum = " + sum);
    }
}
```

Program 3: Find Factorial of a Number

```
class Factorial {
    public static void main(String[] args) {
        int n = 5;
        int fact = 1;

        while (n > 0) {
            fact = fact * n;
            n--;
        }
        System.out.println("Factorial = " + fact);
    }
}
```

JAVA TECHNOLOGIES

Practice programs on while loop

1. Program to print the multiplication table of a number
2. Program to count digits in a number
3. Program to check whether a number is a palindrome
4. Program to print alphabets from A to Z
5. Program to find the sum of digits of a number

Note : Write the java programs for same using do while statement

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Practice programs on for loop

1. Program to print numbers from 1 to 10
2. Program to print even numbers from 1 to 20
3. Program to find the sum of first 10 natural numbers
4. Program to find the factorial of a number
5. Program to print the multiplication table of a number

Program 1: Print Numbers from 1 to 10

```
class PrintNumbers {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
        }
    }
}
```

Program 2: Find Sum of First 10 Natural Numbers

```
class SumNatural {
    public static void main(String[] args) {
        int sum = 0;

        for (int i = 1; i <= 10; i++) {
            sum = sum + i;
        }

        System.out.println("Sum = " + sum);
    }
}
```

Switch Statement

```
switch (constant) {

        case const_1 : {   statements; break; }

         case const_2 : {   statements; break; }

         case const_n : {   statements; break; }

         default:   {   statements; break; }

}
```

Java Program: Menu-Driven Calculator Using switch–case

```
class SwitchCalculator {
    public static void main(String[] args) {
        int a = 10, b = 5;
        int choice = 2;
        switch (choice) {
            case 1:
                System.out.println("Addition = " + (a + b));
                break;
            case 2:
                System.out.println("Subtraction = " + (a - b));
                break;
            case 3:
                System.out.println("Multiplication = " + (a * b));
                break;
```

Java Program: Menu-Driven Calculator Using switch–case

//Cont…

```
case 4:
            System.out.println("Division = " + (a / b));
            break;
        default:
            System.out.println("Invalid Choice");
    }
  }
}
```

Java Program: Menu-Driven Calculator Using switch with Loop

```
class SwitchLoopCalculator {
    public static void main(String[] args) {
        int a = 10, b = 5;
        int choice = 1;
        while (choice <= 4) {
            switch (choice) {
                case 1:
                    System.out.println("Addition = " + (a + b));
                    break;
                case 2:
                    System.out.println("Subtraction = " + (a - b));
                    break;
```

Java Program: Menu-Driven Calculator Using switch with Loop

Cont…

```
case 3:
                System.out.println("Multiplication = " + (a * b));
                break;
            case 4:
                System.out.println("Division = " + (a / b));
                break;
            default:
                System.out.println("Invalid Choice");
        }
        choice++;   // loop control
    }   } }
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Practice programs on for loop

1. Program to print numbers from 1 to 10
2. Program to print even numbers from 1 to 20
3. Program to find the sum of first 10 natural numbers
4. Program to find the factorial of a number
5. Program to print the multiplication table of a number

Program 1: Print Numbers from 1 to 10

```
class PrintNumbers {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
        }
    }
}
```

Program 2: Find Sum of First 10 Natural Numbers

```
class SumNatural {
    public static void main(String[] args) {
        int sum = 0;

        for (int i = 1; i <= 10; i++) {
            sum = sum + i;
        }

        System.out.println("Sum = " + sum);
    }
}
```

Switch Statement

```
switch (constant) {

        case const_1 : {   statements; break; }

         case const_2 : {   statements; break; }

         case const_n : {   statements; break; }

         default:   {   statements; break; }

}
```

Java Program: Menu-Driven Calculator Using switch–case

```
class SwitchCalculator {
    public static void main(String[] args) {
        int a = 10, b = 5;
        int choice = 2;
        switch (choice) {
            case 1:
                System.out.println("Addition = " + (a + b));
                break;
            case 2:
                System.out.println("Subtraction = " + (a - b));
                break;
            case 3:
                System.out.println("Multiplication = " + (a * b));
                break;
```

Java Program: Menu-Driven Calculator Using switch–case

//Cont…

```
case 4:
            System.out.println("Division = " + (a / b));
            break;
        default:
            System.out.println("Invalid Choice");
    }
  }
}
```

Java Program: Menu-Driven Calculator Using switch with Loop

```
class SwitchLoopCalculator {
    public static void main(String[] args) {
        int a = 10, b = 5;
        int choice = 1;
        while (choice <= 4) {
            switch (choice) {
                case 1:
                    System.out.println("Addition = " + (a + b));
                    break;
                case 2:
                    System.out.println("Subtraction = " + (a - b));
                    break;
```

Java Program: Menu-Driven Calculator Using switch with Loop

Cont…

```
case 3:
                System.out.println("Multiplication = " + (a * b));
                break;
            case 4:
                System.out.println("Division = " + (a / b));
                break;
            default:
                System.out.println("Invalid Choice");
        }
        choice++;   // loop control
    }   } }
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

String Class

String Class is part of **java.lang** package and represents a sequence of characters. It offers a wide range of methods for creating, manipulating, and processing text in Java applications.

- **Immutable:** Once created, the value of a string cannot be changed. Any modification creates a new string object, which helps ensure data integrity and security.
- **String Class:** Strings are instances of the String class in Java, which provides numerous methods for string manipulation, such as length(), charAt(), substring(), replace(), and more.
- **Concatenation:** You can concatenate strings using the + operator

String Class

Immutable: Once created, the value of a string cannot be changed. Any modification creates a new string object, which helps ensure data integrity and security.

```
String s = "Hello";
s.concat(" World");
System.out.println(s);
```

JAVA TECHNOLOGIES

Java provides two String class options in string to modify after its creation:

- **StringBuffer**
- **StringBuilder**.

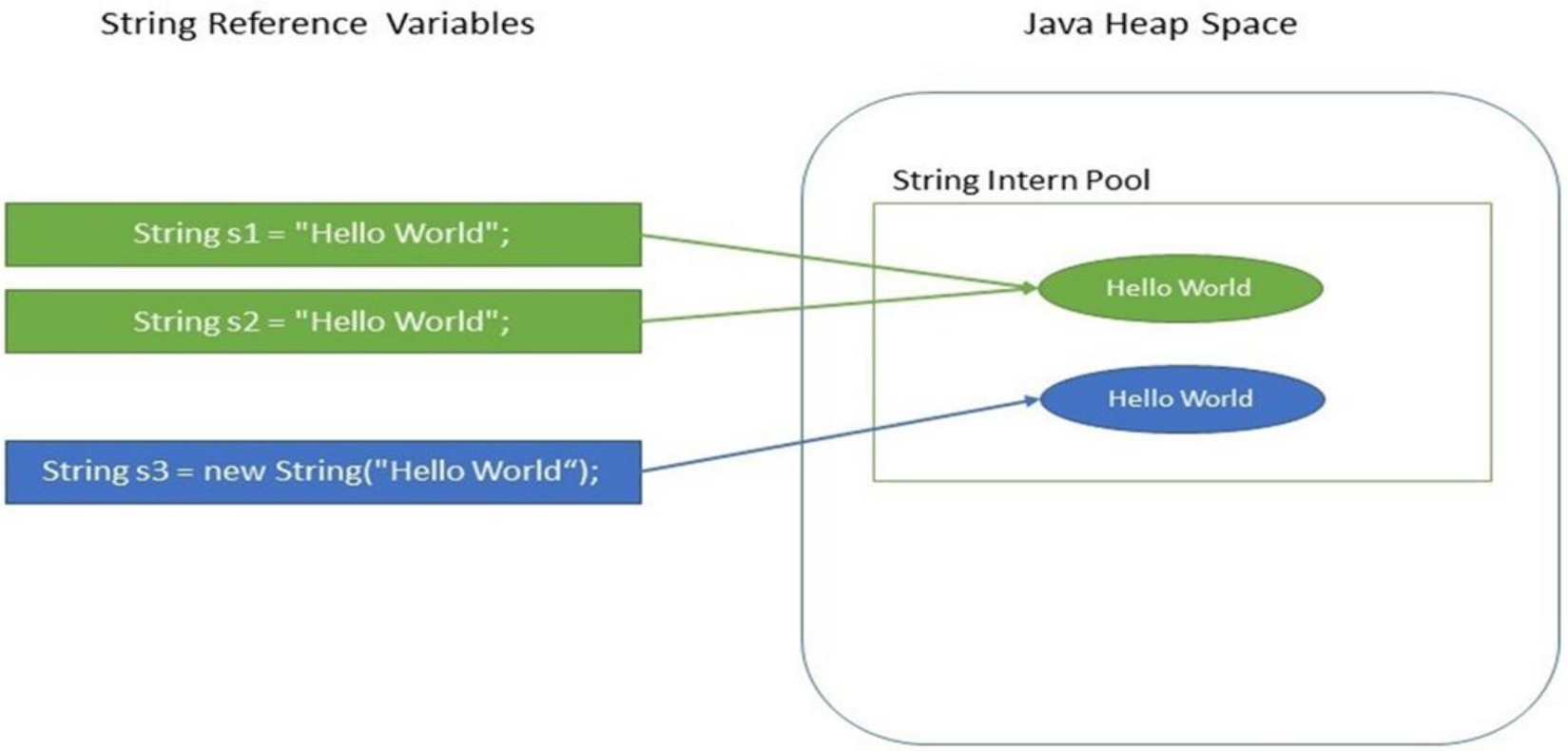
Both hold strings that can be modified after they are created

```
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World");
System.out.println(sb);
```

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(" World");
System.out.println(sb);
```

There are 2 ways of creating Strings

1.String literal
2.new keyword



String is basically an object that represents sequence of char values. An array of characters works same as java string.

For example:

1. char[] ch={‘P’,’E’,’S’};
2. String s=new String(ch);.

OR

String s=“PES";

The String class supports several constructors.

To create an empty String, call the **default constructor.**

For example,
String s = new String();

will create an instance of String with no characters in it.

Length() Method

Code:

```
char chars[] = { 'a', 'b', 'c' };
String s = new String(chars);     //defensive copy of array char[ ]
System.out.println(s.length());
```

Character Extraction

✓ **charAt( ):To extract a single character from a String**

```
char ch;
ch = "abc".charAt(1);
```

✓ **getChars( ): To extract more than one character at a time,**

```
void getChars(int sourceStart, int sourceEnd, char target[ ], int targetStart)
```

```
String s = "This is a demo of the getChars method.";
int start = 10;
int end = 14;
char buf[] = new char[end - start];
s.getChars(start, end, buf, 0);
System.out.println(buf);
```

String Concatenation

Example:

```
int age = 9;
String s = "He is " + age + " years old.";
System.out.println(s);
```

output : He is 9 years old

```
String s = "four: " + 3 + 3;
System.out.println(s);
```

Output four: 33

```
String s = "four: " + (3 +3);
```

Now s contains the string "four: 6".

String Conversion

When Java converts data into its **string representation** during concatenation, it does so by calling one of the overloaded versions of the string conversion method valueOf( ) defined by String.

valueOf( ) is overloaded for all the primitive types and for type Object.

- ✓ For the primitive types, **valueOf( )** returns a string that contains the human-readable equivalent of the value with which it is called.
- ✓ For objects, **valueOf( ) calls the toString( )** method on the object.

String Conversion

- ✓ When valueOf() is used with an object, it checks if the object is null.
- ✓ If the object is not null, valueOf() calls the object's **toString()** method to get its string representation. If the object is null, it returns the string "null"

```
class conversion
{
public static void main(String args[]) {
        int intnum=30;
        String s1=String.valueOf(intnum);
        System.out.println(s1);
        String s2=new String("STRING DATA");
        String s4=new String("JAVA");
          String s3=String.valueOf(s2);
        System.out.println(s3);
        System.out.println(s3+s4);
        conversion Obj = null;
        String strNull = String.valueOf(Obj);
        System.out.println(strNull);              }  }
```

- ✓ **The equals( ) method** compares the characters inside a String object.
- ✓ **The == operator** compares two object references to see whether they refer to the same instance

```
class EqualsNotEqualTo {
public static void main(String args[])
 {
String s1 = "Hello";
String s2 = new String(s1);
//String s3=new String(“JAVA”);
//String s4=new String(“JAVA”);

System.out.println(s1 + " equals " + s2 + " -> " + s1.equals(s2));// Content equality
System.out.println(s1 + " == " + s2 + " -> " + (s1 == s2));   //reference equality
} }
```

Modify String

- ✓ substring( )
- ✓ concat( )
- ✓ replace( )
- ✓ trim( )

Changing case of string

- ✓ String toLowerCase( )
- ✓ String toUpperCase( )

Strings are Immutable.

- In Java, once a String object is created, its contents cannot be changed.
- Any method like replace(), concat(), substring(), etc.it does not modify the original string, instead it creates a new String object.

```
class Test {
    public static void main(String[] args) {
        String s = "banana";
        String result = s.replace('a', 'o');

        System.out.println("Original: " + s);   // banana
        System.out.println("Replaced: " + result); // bonono
    }
}
```

JAVA TECHNOLOGIES

Write a Java code to build a simple program to validate user email addresses.

- Trim extra spaces
- Check if it contains @Extract domain name
- Convert domain to lowercase
- Remove spaces
- Verify ending (.com, .org, .in etc.)

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

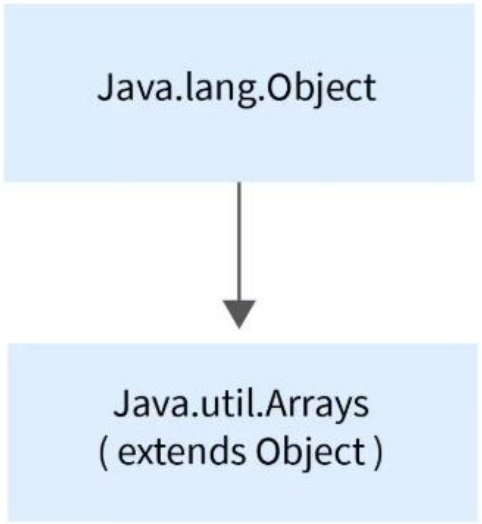
Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Arrays in Java (Helper Class)

- The arrays class is part of the Java collection framework in the java.util package.
- It only consists of static methods and methods of the object` class.
- Using Arrays, we can create, compare, sort, search, stream, and transform arrays.
- We can easily access the methods of this class by their class name.



Arrays in Java (Helper Class)

Methods in Java Array Class

- The static methods of the Java Array class could be used to perform operations like:
- Filling the elements
- Sorting the elements
- Searching for the elements
- Converting the array elements to String
- Many more...

Arrays in Java (Helper Class)

```
import java.util.Arrays;

class ArraysClass {
        public static void main(String[] args)
        {
          int Arr[] = { 10, 20, 11, 21, 31 };
          Arrays.sort(Arr);
          int Key = 31;
          System.out.println(Key + " found at index = "+ Arrays.binarySearch(Arr, Key));
        }
}
```

Arrays in Java (Helper Class)

```
import java.util.Arrays;

class ArraysClass {
    public static void main(String[] args)
    {
        int Arr[] = { 10, 20, 11, 21, 31 };

        System.out.println("Integer Array is: " + Arrays.toString(Arr));
    }
}
```

Arrays in Java (Helper Class)

```
import java.util.Arrays;

public class Scaler {
    public static void main(String[] args)
    {

        int Arr[] = { 10, 20, 11, 21, 31 };
        int Arr1[] = { 10, 11, 21 };

        System.out.println("Integer Arrays on comparison are : " + Arrays.equals(Arr, Arr1));
    }
}
```

Arrays in Java (Helper Class)

```
import java.util.Arrays;

public class Scaler {
    public static void main(String[] args)
    {
        int intArr[] = { 10, 20, 11, 21, 31 };

        System.out.println("Integer Array is: " + Arrays.hashCode(intArr));
    }
}
```

Benefits of Arrays Class in Java

- Arrays class makes it easier to perform common operations on an array.
- No need to use complicated loops to work with an array.
- No need to reinvent the wheel for binary search and sorting operations.

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Practice programs

1. Write a Java program to create a class Sum with two integer data members and a member function to calculate and display their sum.

2. Create a class Employee with data members empId, empName, and salary. Write a member function to display employee information.

JAVA TECHNOLOGIES

Write a Java program to create a class Sum with two integer data members and a member function to calculate and display their sum.

```
class Sum {
    int a = 10;
    int b = 20;
    void calculateSum() {
        int sum = a + b;
        System.out.println("Sum = " + sum);
    }
 public static void main(String[] args) {
      Sum s = new Sum();
      s.calculateSum();
    }
}
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

Practice programs

1. Write a Java program to create a class Sum with two integer data members and a member function to calculate and display their sum.

2. Create a class Employee with data members empId, empName, and salary. Write a member function to display employee information.

JAVA TECHNOLOGIES

Write a Java program to create a class Sum with two integer data members and a member function to calculate and display their sum.

```
class Sum {
    int a = 10;
    int b = 20;
    void calculateSum() {
        int sum = a + b;
        System.out.println("Sum = " + sum);
    }
 public static void main(String[] args) {
      Sum s = new Sum();
      s.calculateSum();
    }
}
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Practice programs

1. Write JAVA program to demonstrate string methods
2. Write a Java program to perform the following operations:
   - ✓ Remove extra spaces at the beginning and end.
   - ✓ Check if the email contains '@' symbol.
   - ✓ Extract the username (before @).
   - ✓ Extract the domain name (after @).
   - ✓ Convert the domain name to lowercase.
   - ✓ Check whether the email ends with .com, .org, or .in.
   - ✓ Replace "Student" with "User" in the username.
   - ✓ Display the final formatted email.

```
public class StringMethodsDemo {
    public static void main(String[] args) {
        String str = "  Java Programming  ";
        String str2 = "JAVA";
        System.out.println("Original: " + str);
        System.out.println("Length: " + str.length());
        String trimmed = str.trim();
        System.out.println("Trimmed: " + trimmed);
        System.out.println("Uppercase: " + trimmed.toUpperCase());
        System.out.println("Lowercase: " + trimmed.toLowerCase());
        System.out.println("Character at 2: " + trimmed.charAt(2));
        System.out.println("Equals: " + trimmed.equals(str2));
        System.out.println("Equals Ignore Case: " + trimmed.equalsIgnoreCase(str2));
        System.out.println("Contains 'Prog': " + trimmed.contains("Prog"));
```

```
System.out.println("Index of 'a': " + trimmed.indexOf('a'));
System.out.println("Last Index of 'a': " + trimmed.lastIndexOf('a'));
System.out.println("Substring(5): " + trimmed.substring(5));
System.out.println("Substring(0,4): " + trimmed.substring(0,4));
System.out.println("Replace a with o: " + trimmed.replace('a','o'));
System.out.println("Starts with Java: " + trimmed.startsWith("Java"));
System.out.println("Ends with ming: " + trimmed.endsWith("ming"));
String[] words = trimmed.split(" ");
System.out.println("Words:");
for(String w : words) {
    System.out.println(w);       }
char[] chars = trimmed.toCharArray();
System.out.println("Characters:");
for(char c : chars) {
    System.out.print(c + " ");       }   }   }
```

JAVA TECHNOLOGIES

SUBJECT CODE : UQ24CA251B

Samyukta D Kumta

Computer Applications

JAVA TECHNOLOGIES

Practice Programs

Write Program to perform followings.

1. Find sum and average of numbers
2. Store and display student details
3. Store marks where each student has different subjects

JAVA TECHNOLOGIES

1. Find sum and average of numbers

```
public class SimpleArray {
    public static void main(String[] args) {
        int arr[] = {10, 20, 30, 40, 50};
        int sum = 0;
        for(int i = 0; i < arr.length; i++) {
            sum += arr[i];
        }
        double avg = (double)sum / arr.length;
        System.out.println("Sum = " + sum);
        System.out.println("Average = " + avg);
    }
}
```

2. Store and display student details

```
class Student {
    int roll;
    String name;
    double marks;
    void setData(int r, String n, double m) {
        roll = r;
        name = n;
        marks = m;
    }
    void display() {
        System.out.println("Roll: " + roll + " Name: " + name + " Marks: " +
marks);
    }  }
```

```
public class ClassArray {
    public static void main(String[] args) {
        Student s[] = new Student[3];
        for(int i = 0; i < s.length; i++) {
            s[i] = new Student();
        }
        s[0].setData(1, "Anu", 85);
        s[1].setData(2, "Ravi", 90);
        s[2].setData(3, "Meena", 88);
        for(int i = 0; i < s.length; i++) {
            s[i].display();
        }
    }
}
```

3. Store marks where each student has different subjects

```
public class JaggedArray {
    public static void main(String[] args) {
        int marks[][] = new int[3][];
        marks[0] = new int[]{85, 90};      // Student 1
        marks[1] = new int[]{78, 88, 91};  // Student 2
        marks[2] = new int[]{92};          // Student 3
        for(int i = 0; i < marks.length; i++) {
            System.out.println("Student " + (i+1) + " marks:");
            for(int j = 0; j < marks[i].length; j++) {
                System.out.print(marks[i][j] + " ");
            }
            System.out.println();
        }    }  }
```

THANK YOU

Samyukta D Kumta
Department of Computer Applications
samyuktad@pes.edu