JAVA Tutorial

PM DIGI CLOUD

For Beginners

#1

ZERO to HERO

What is Java?
How Platform Independent?
Installation and setup?
JVM vs JRE vs JDK

What is Java and Why you should learn?

PM DIGI CLOUD

Java is a **programming language** and a **platform**.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*.

Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

- Desktop Applications such as acrobat reader, media player, antivirus, etc.
- Web Applications such as irctc.co.in, javatpoint.com, etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc

PM DIGI CLOUD

Number 1 programming language
by developers community

TIOBE SOFTWARE
THE SOFTWARE QUALITY COMPANY

9 million
Java developers

5 billion
Java cards in use

3 billion
mobile phones run Java

eXo
The Enterprise Social Platform

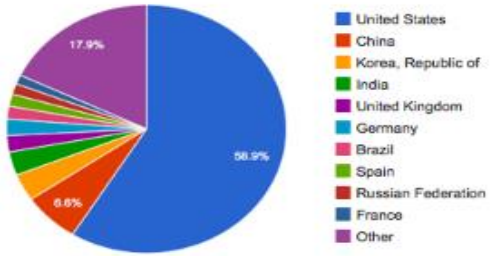
Today, Java is everywhere

Number 1
business applications langage

97%
of Enterprise Desktops Run Java

Language of choice
in all industries:

JAVA
Countries Distribution



History of Java

PM DIGI CLOUD

1992 – Launch OAK - > Java

1995 – Sun World Conference (23 march 1995) – Officially Launched as open source

2010 – Oracle Java

Java Version	Features / History
1995 – JDK Beta	First Beta Version , Developed by James Gosling
23 Jan 1996 – Java 1	First Public Release, Stable Version
19 Feb 1997 – Java 1.1	Inner Classes, Java Beans, JDBC, RMI
8 Dec 1998 – Java 1.2	Swing, JIT Compiler, Collections
8 May 2000 – Java 1.3	Hotspot JVM, JNDI, JPDA

History of Java (To be Contd…)

Java Version	Features / History
6 Feb 2002 – Java 1.4	Assertions, Image IO API, XML parsers, XSLT processors
30 Sep 2004 – Java 5	Generics, VarArgs, Autoboxing, Concurrency, Static Imports, Enum,Annotations ForEach loop etc
11 Dec 2006 – Java 6	JAXB 2, JDBC 4.0 support, Pluggable annotations
7 Jul 2011- Java 7	String in Switch, Try with Resources, java NIO package, Catching Multiple exceptions in single catch block
18 Mar 2014 – Java 8 **Oracle Commercial from Jan 2019**	forEach(), default and static method in interfaces, Lambda Expressions, Stream API, New Date Time API
21 Sep 2017 – Java 9	Jshell, Module System, Reactive stream, HTTP 2 client
20 Mar 2018 – Java 10	Local Variable Type Inference
25 Sept 2018 – Java 11	New String Class methods, Var for Lambda, Run java using single command
19 Mar 2019 – Java 12	Shenandoah Garbage Collector, Switch expression, New String methods

History of Java (To be Contd…)

PM DIGI CLOUD

Java Version	Features / History
17 Sept 2019 – Java 13	Socket API reimplementation, Text Blocks, DOM and SAX factories, Unicode 12.1 support

Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK March 2023 for Azul[11]	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	September 2029 for Azul At least September 2027 for Microsoft[14] At least September 2027 for Eclipse Adoptium	September 2029 or later September 2029 for Azul
Java SE 18	March 2022	September 2022 for OpenJDK and Adoptium	N/A
Java SE 19	September 2022	March 2023 for OpenJDK	N/A
Java SE 20	March 2023	September 2023 for OpenJDK	N/A
Java SE 21 (LTS)	September 2023	September 2028	September 2031[13]

Legend: Old version | Older version, still maintained | **Latest version** | Future release

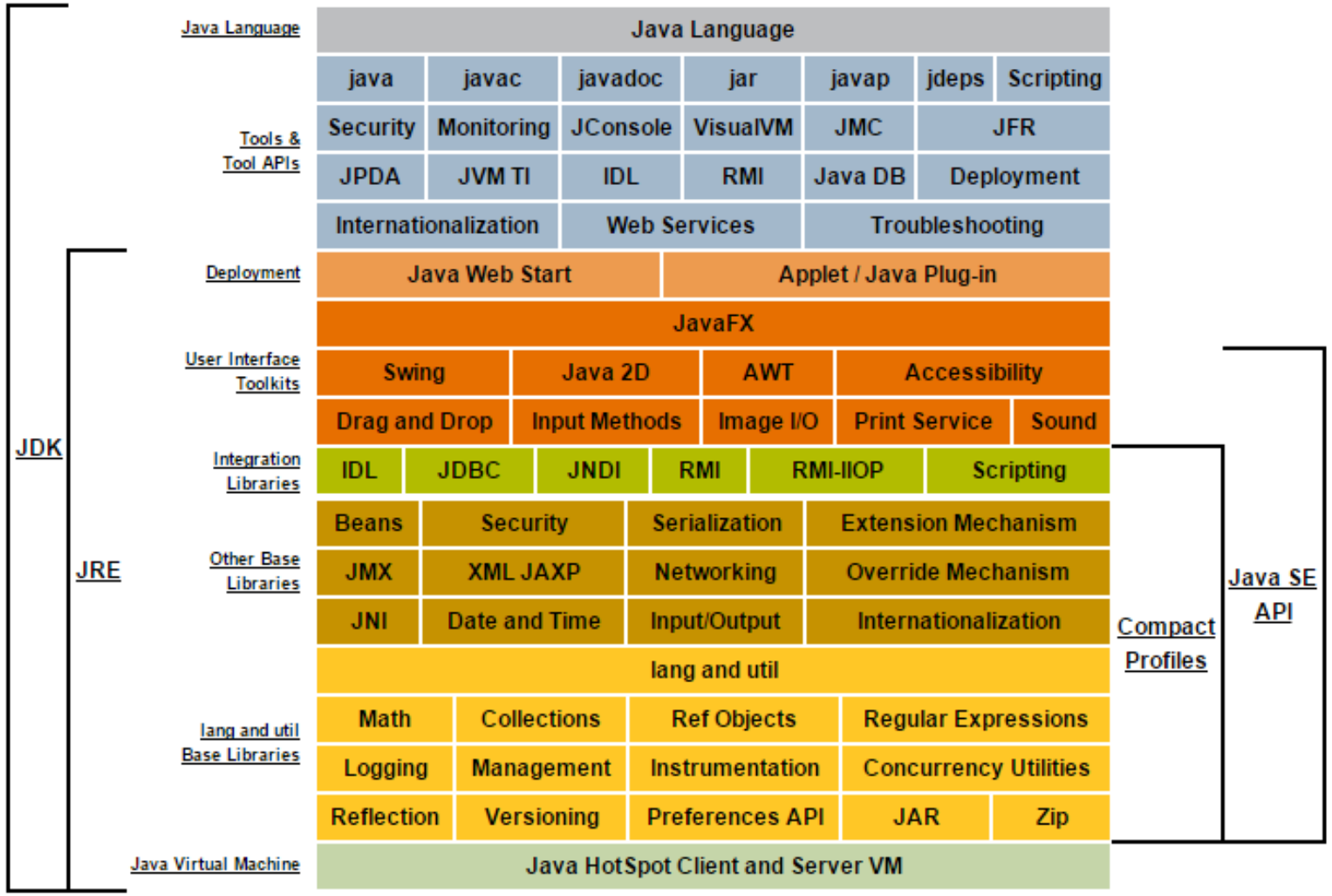
Features Of Java and Java vs C++

- Platform independent (Write once Run Anywhere)
- Extensive Libraries
- Garbage Collection
- Multi threaded
- OOPS
- Distributed
- Robust
- Secure
- Easy, etc

JVM vs JRE vs JDK

PM DIGI CLOUD

Description of Java Conceptual Diagram



Classes, Objects

- Classes
- Objects
- Fields (Data Types), Variables (Local, Instance, Static)
- Methods
- Operators
- Access Modifiers
- Constructor
- Blocks
- Nested Class
- Control Statements
- OOPS

JAVA Tutorial

PM DIGI CLOUD

For Beginners

#2

ZERO to HERO

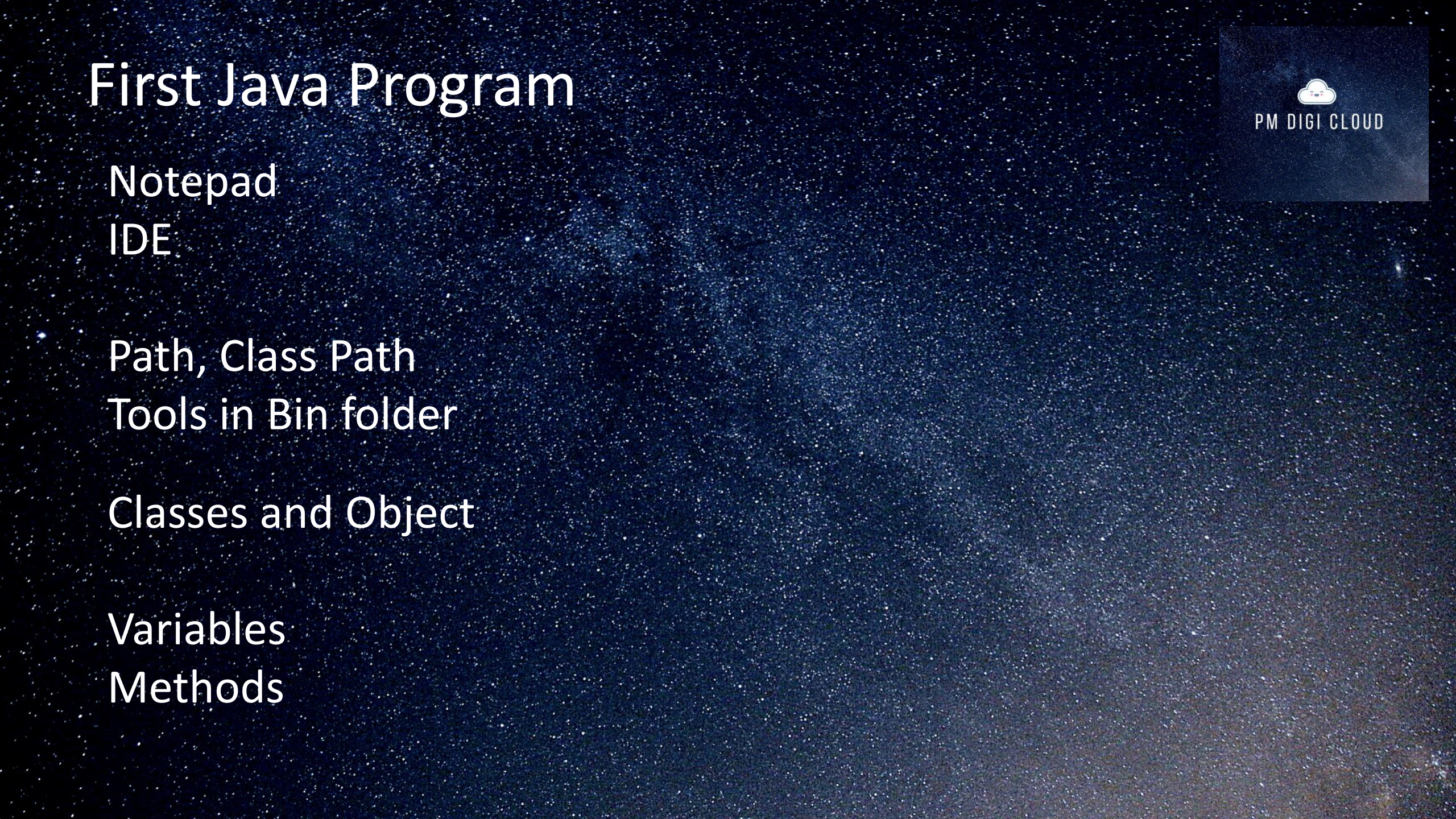
Creating First Java code?

Classes and Object?

Class Loaders?

Variables and Methods?

First Java Program



Notepad
IDE

Path, Class Path
Tools in Bin folder

Classes and Object

Variables
Methods

Class Loaders

1. **Bootstrap Class Loader**
2. **Extension Class Loader**
3. **System Class Loader**

Class loaders are responsible for **loading Java classes dynamically to the JVM (Java Virtual Machine) during runtime.**

Java classes aren't loaded into memory all at once, but rather when they're required by an application.

JAVA Tutorial

PM DIGI CLOUD

For Beginners

#3

ZERO to HERO

Constructor, Static Initializer
Access Specifier - Public Private
Instance, Local, Static Variables
Instance, Static Methods

Constructor, Initializer, Static Initializer

Constructor has the same name as class name whereas instance initialization block just have a body without any name or visibility type.

- No-Arg Constructor.
- Parameterized Constructor.
- Default Constructor.

- Instance Initializer
- Static Initializer

Packages, Encapsulation

PM DIGI CLOUD

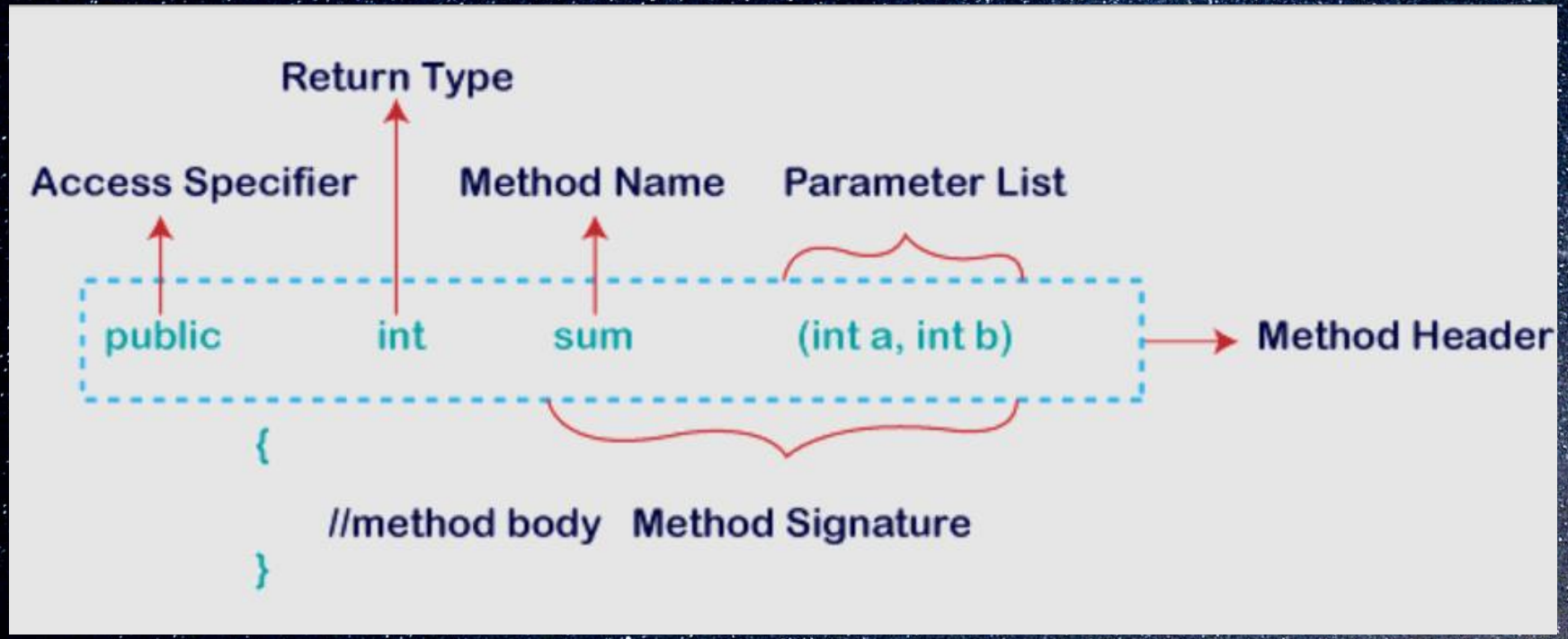
Access Specifiers in Java

		public	private	protected	default
Same Package	Class	YES	YES	YES	YES
	Sub class	YES	NO	YES	YES
	Non sub class	YES	NO	YES	YES
Different Package	Sub class	YES	NO	YES	NO
	Non sub class	YES	NO	NO	NO

Variables & Methods in Detail

PM DIGI CLOUD

Instance, Static, Local Variable – Scope
Instance, Static Method Scoped



Note – Final variable and method would be overed in next video #4

JAVA Tutorial

PM DIGI CLOUD

For Beginners

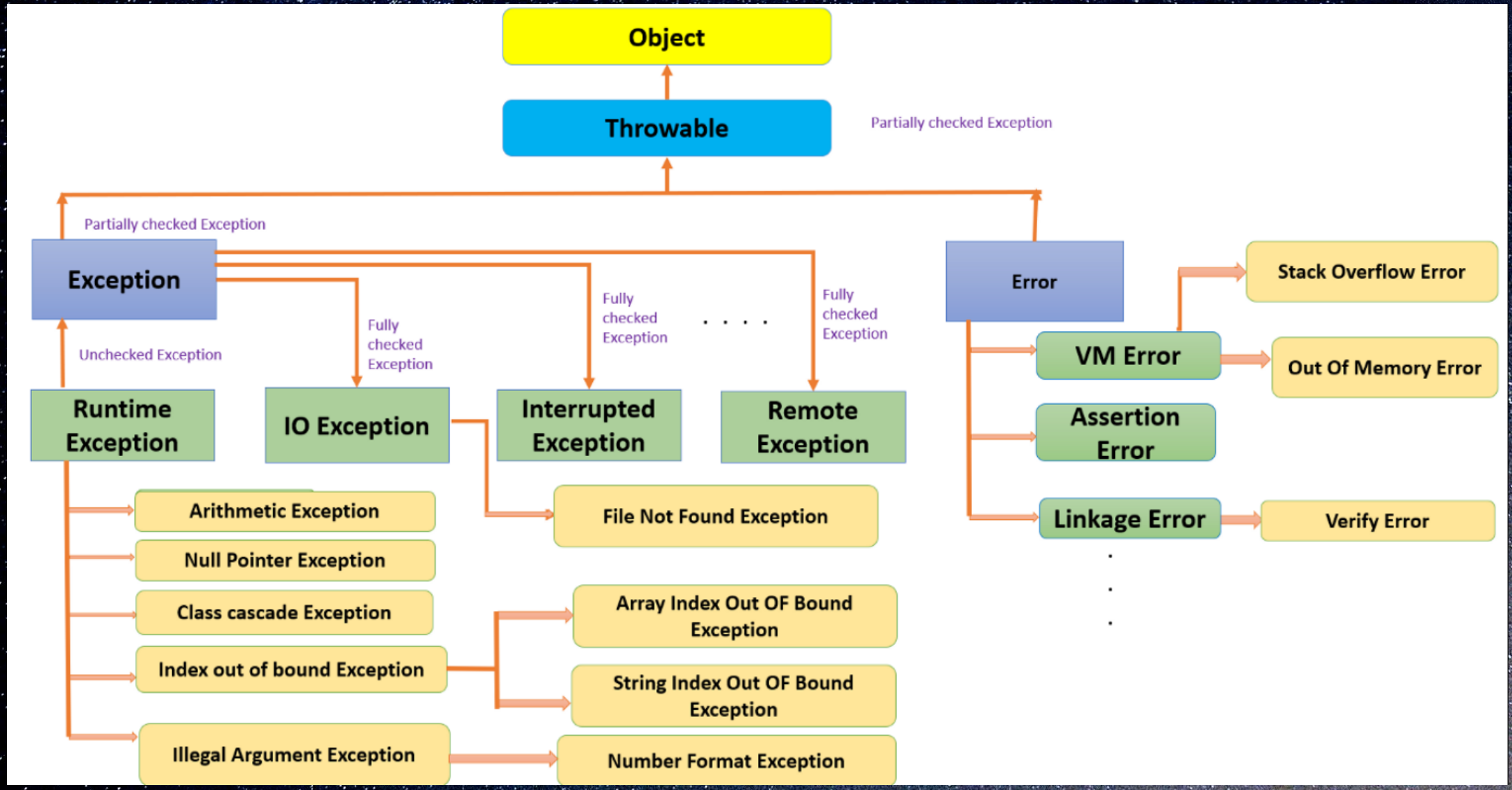
#4

ZERO to HERO

Exception Handling
(Try, Catch, Finally)
Checked vs Unchecked
Final vs Finally difference

Exception vs Error

PM DIGI CLOUD



Handling exceptions

PM DIGI CLOUD

```
try {
  //code
}
catch (ExceptionType1 e1) {
  // catch block
}
finally {
  // finally block always
executes
}
```

Throw and throws

try-with-resources

(introduced in Java 7)

JAVA Tutorial

PM DIGI CLOUD

#5

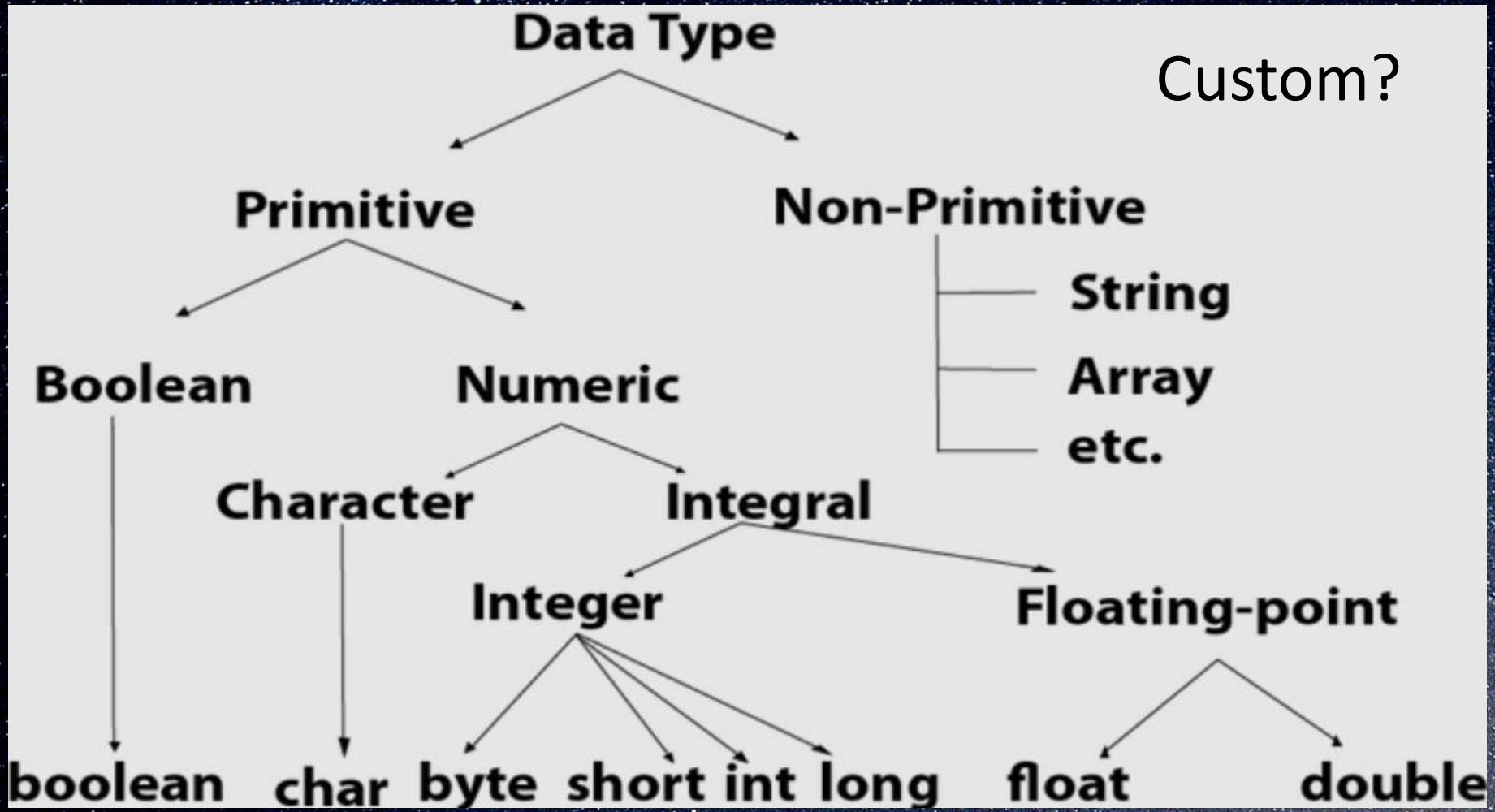
For Beginners

ZERO to HERO

Data Types & Casting
(Primitive vs Non-Primitive)
Stack vs Heap
Grabage Collection
(Young vs Old vs Perm gen)

#Data Types

PM DIGI CLOUD



Custom?

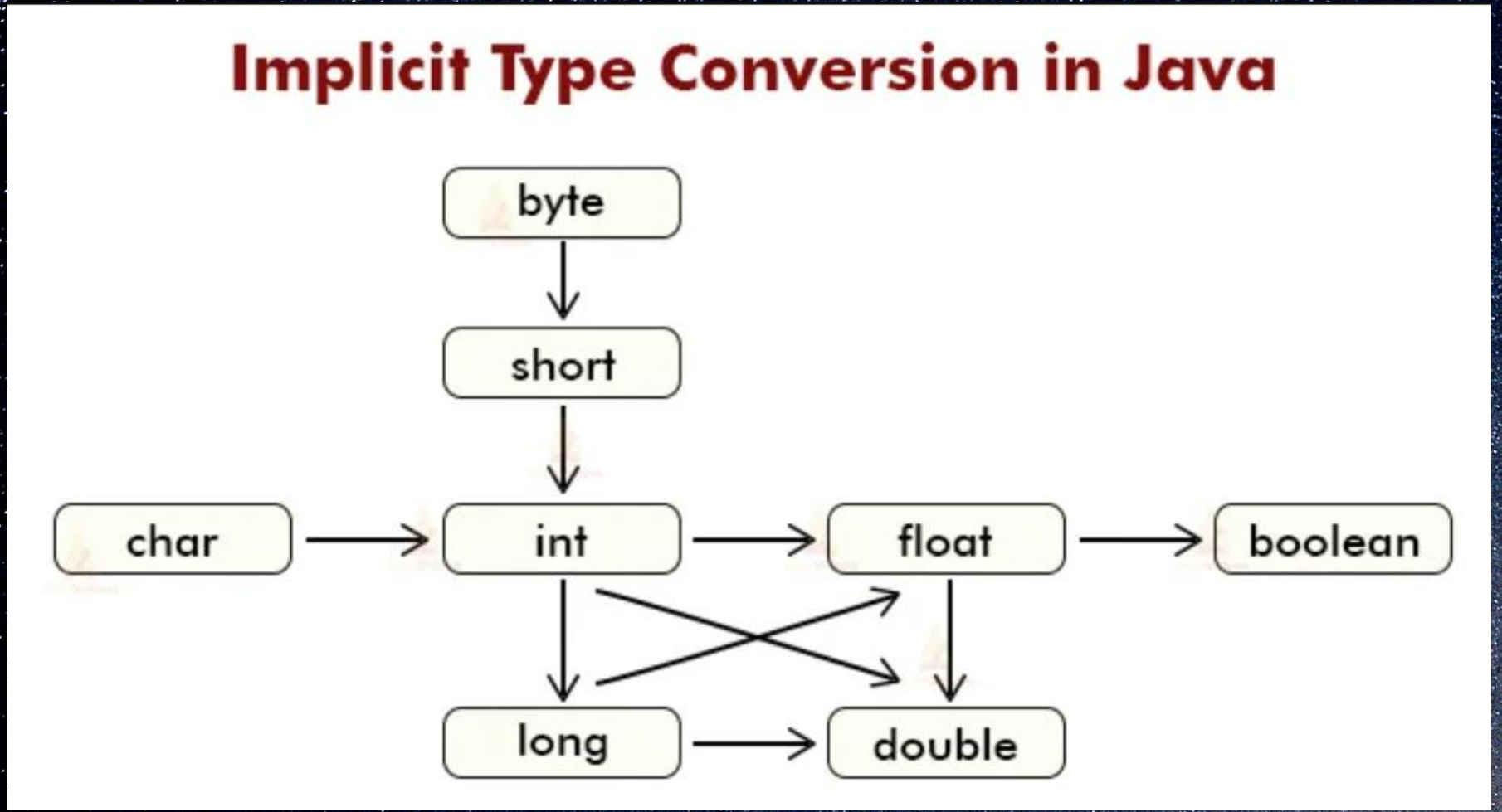
#Data Type Size

PM DIGI CLOUD

Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535
boolean	1	true or false

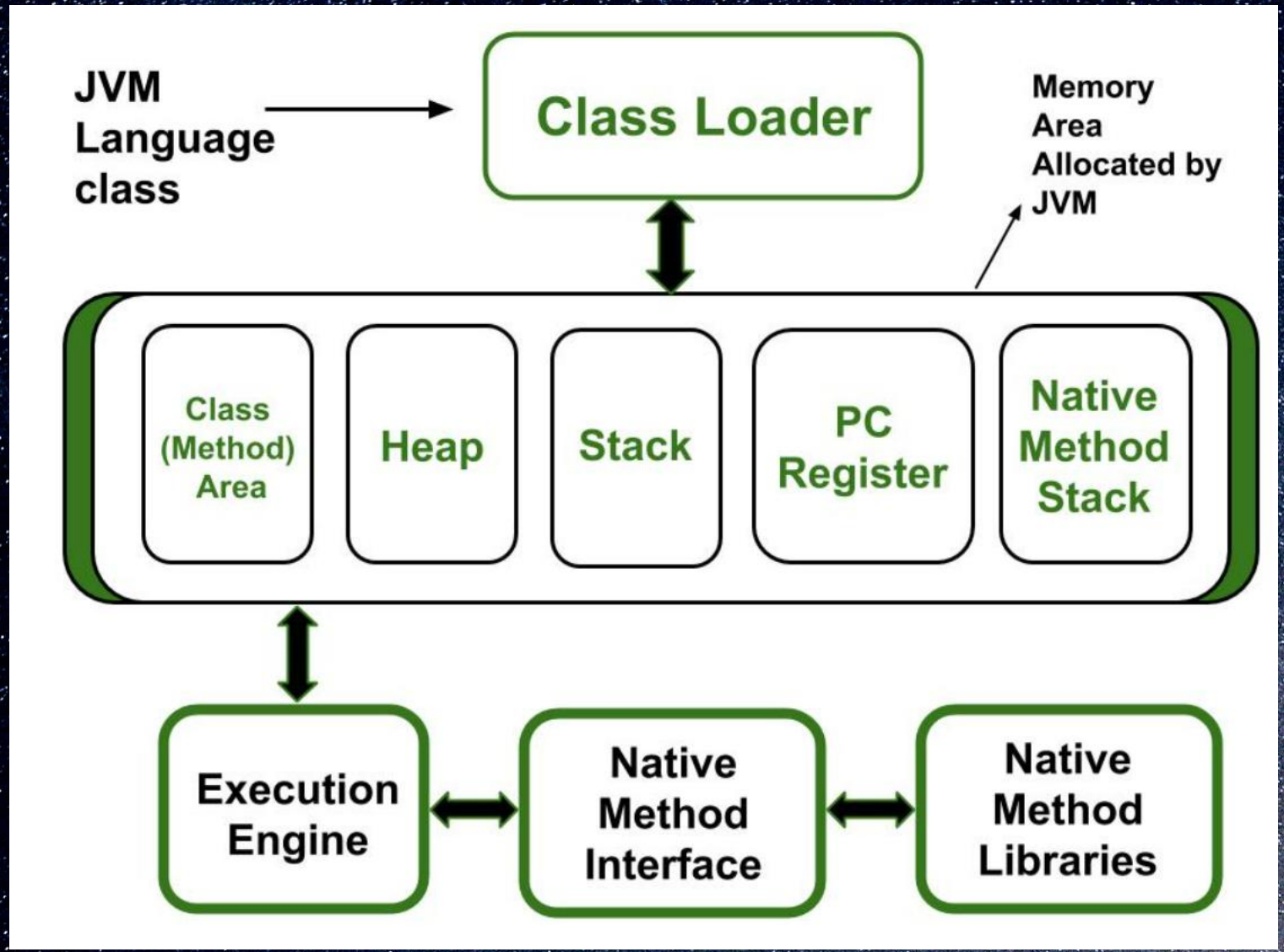
#Implicit Type Casting

PM DIGI CLOUD



#Memory mgmt. in JVM

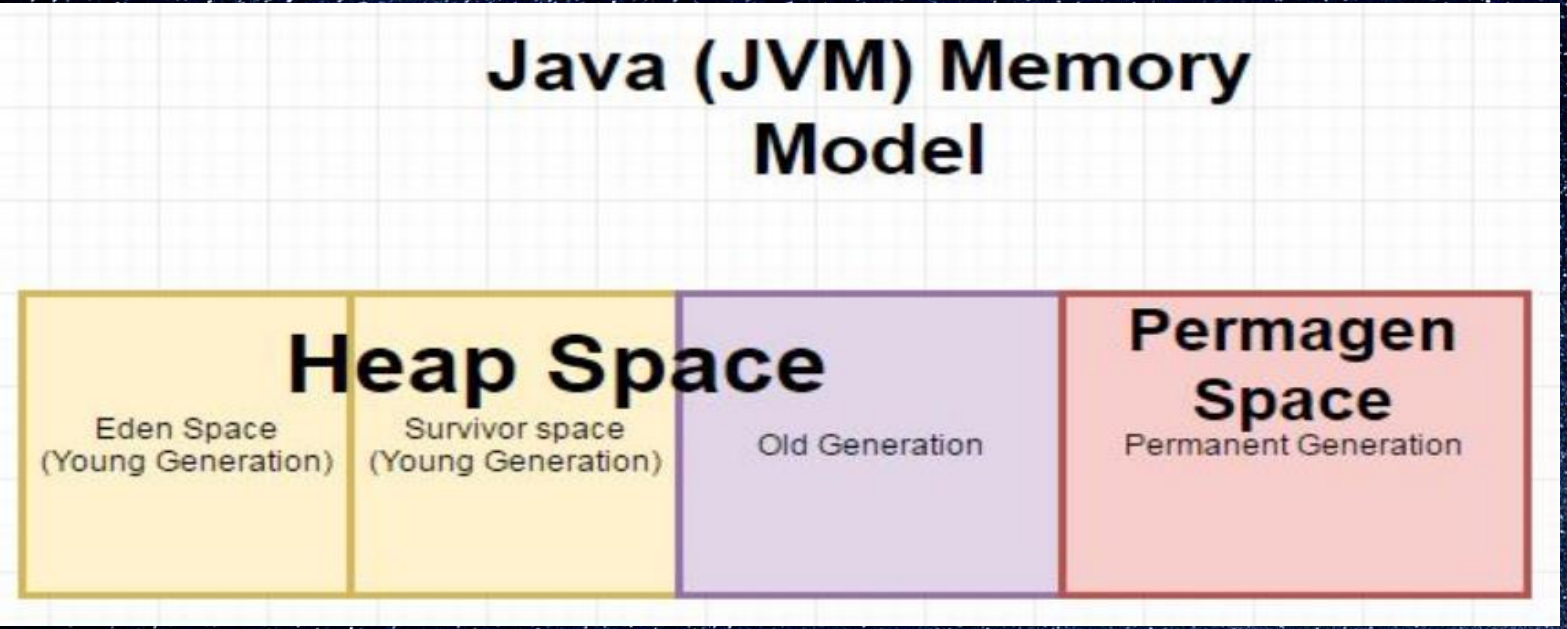
PM DIGI CLOUD



1-Each thread has a private JVM stack.
2-Method Area and Heap Area is shared with all Threads.

#Memory mgmt. in JVM

PM DIGI CLOUD



Heap memory is managed by Operating System.

For Java 11 and above

The Xmx value is 25% of the available memory with a maximum of 25 GB. However, where there is 2 GB or less of physical memory, the value set is 50% of available memory with a minimum value of 16 MB and a maximum value of 512 MB.

For Java 8

The Xmx value is half the available memory with a minimum of 16 MB and a maximum of 512 MB.

#Error

Stack Overflow Error

Out Of Memory Error

If the computation in a thread requires a larger Java Virtual Machine stack than is permitted, the Java Virtual Machine throws a *StackOverflowError*.

If a computation requires more heap than can be made available by the automatic storage management system, the Java Virtual Machine throws an *OutOfMemoryError*.

#Grabage Collection

PM DIGI CLOUD

It destroys un-referenced objects.

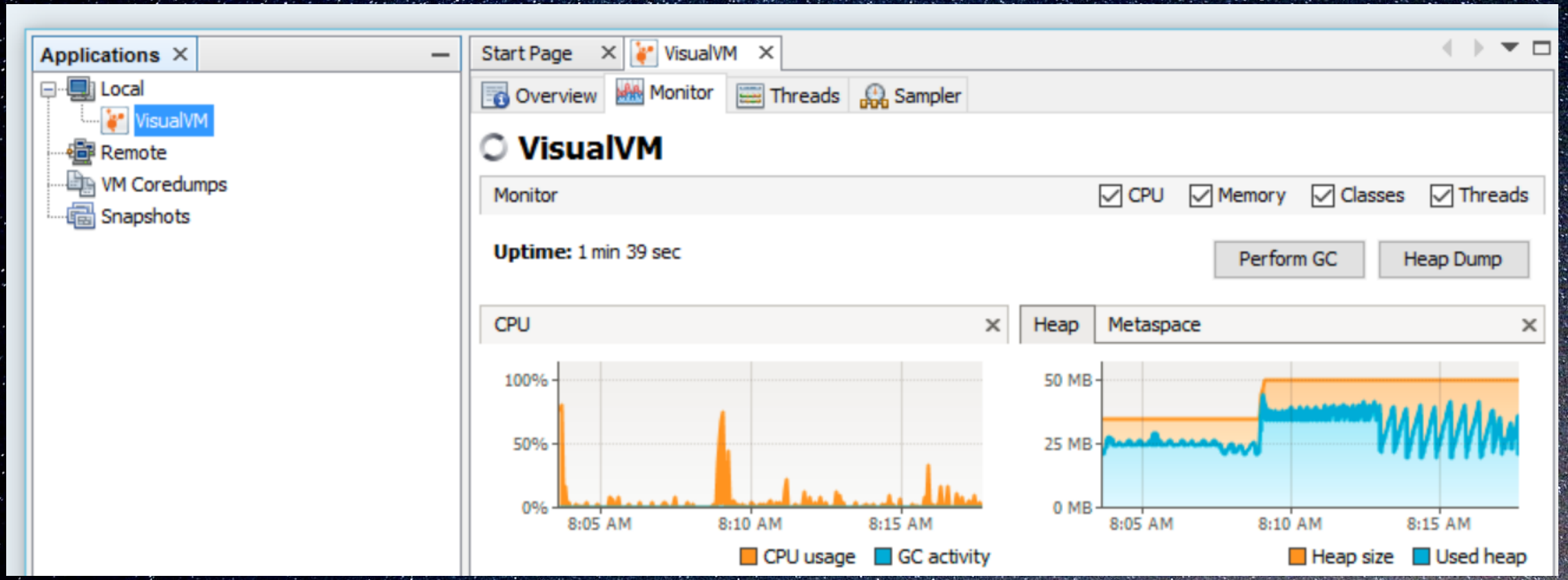
- **Serial Garbage Collector**
- **Parallel Garbage Collector**
- **CMS Garbage collector**
- **G1 Garbage Collector (Default from Java 9)**

```
Void gc();
Long freeMemory();
Long totalMemory();
```

#Grabage Collection

PM DIGI CLOUD

It destroys un-referenced objects.



JAVA Tutorial

PM DIGI CLOUD

#6

For Beginners

ZERO to HERO

Conditional , Looping and Jump

ifElse , Switch

While,For, ForEach

Break, Continue, Return

Why, How and When to use?

#Control Statements

PM DIGI CLOUD

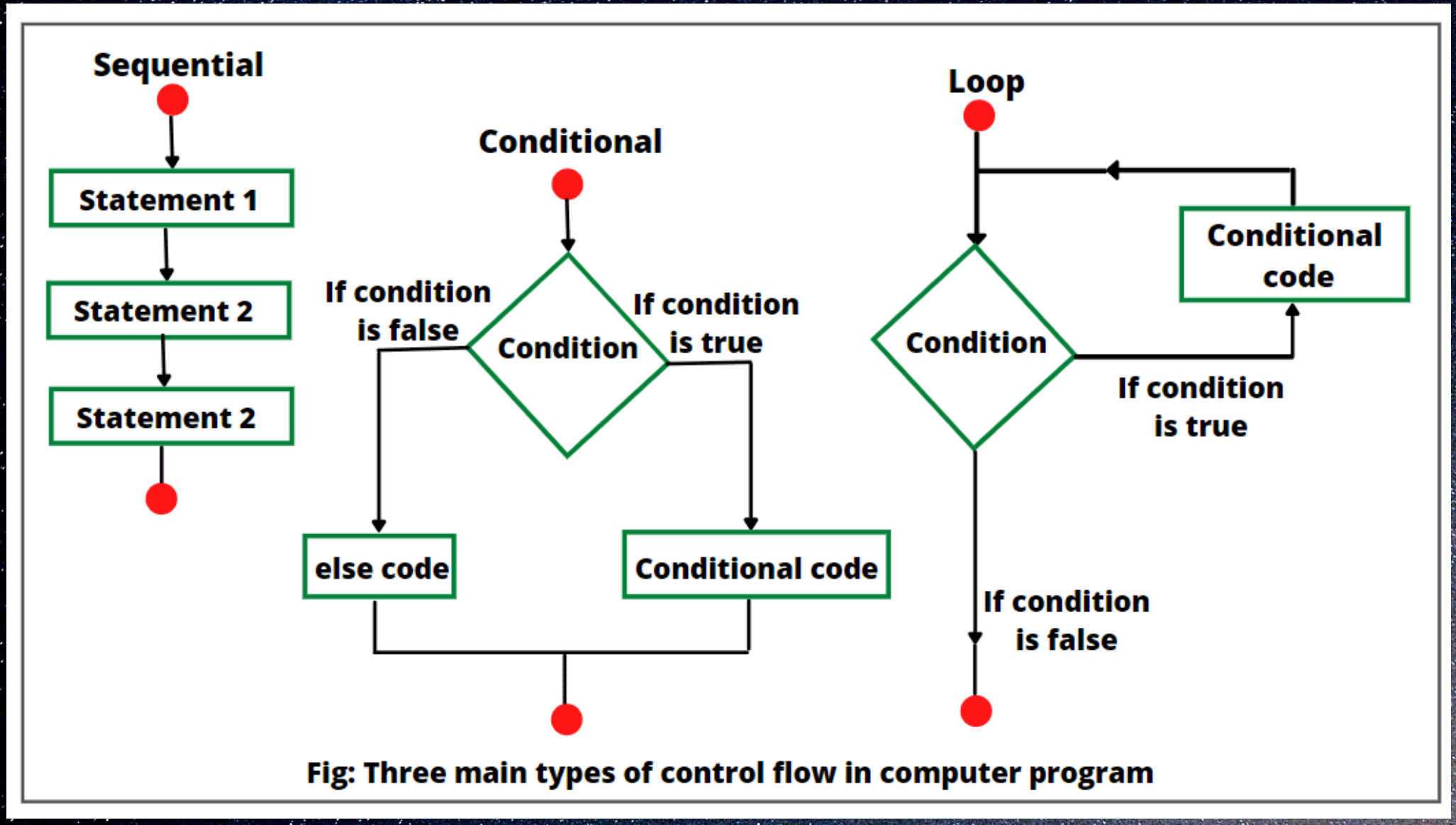
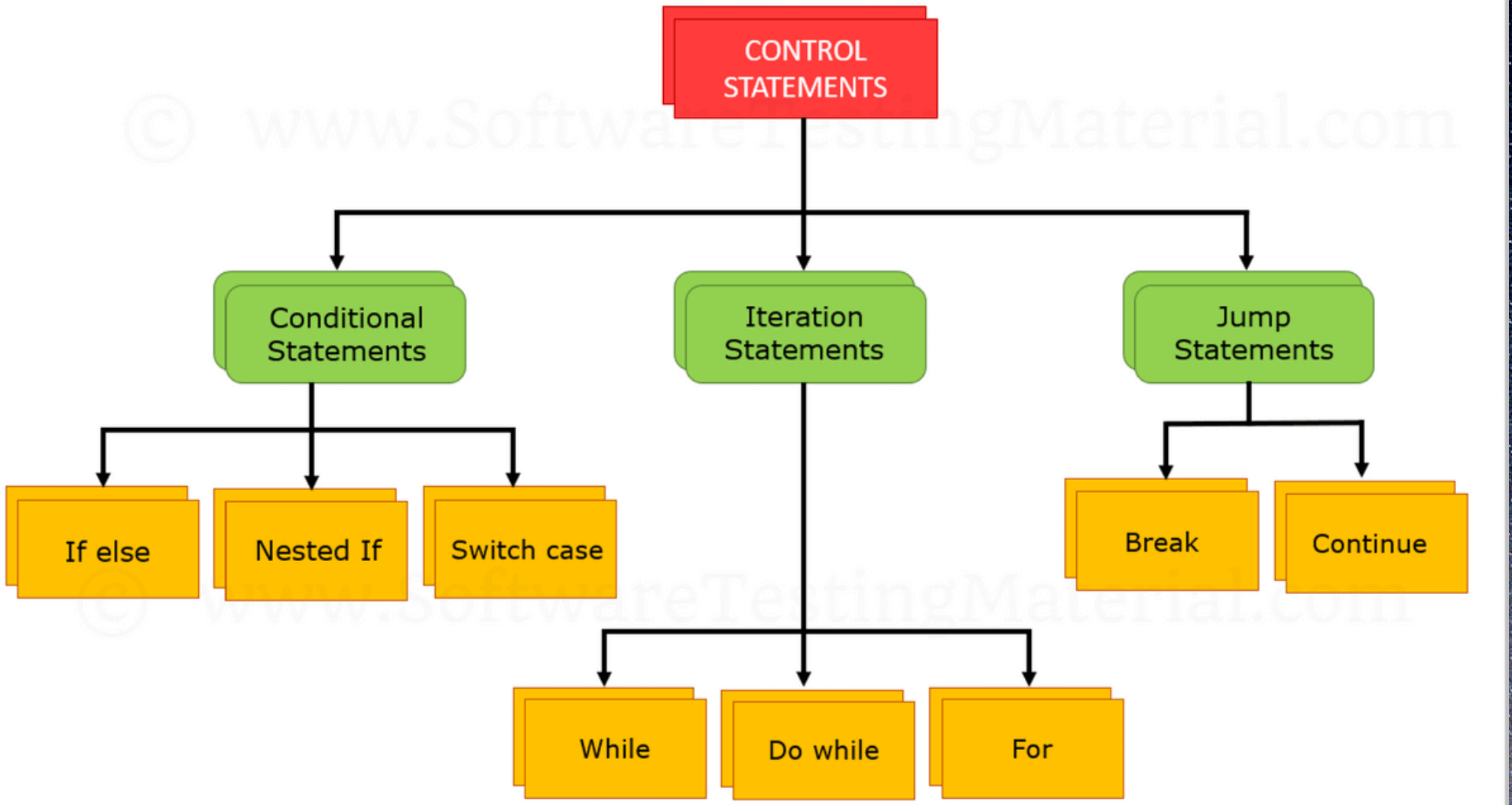


Fig: Three main types of control flow in computer program

#Control Statements- Java

PM DIGI CLOUD



© www.SoftwareTestingMaterial.com

Control Statements

Conditional

- *calling**IfElse**Control();*
- *calling**Switch**Control();*

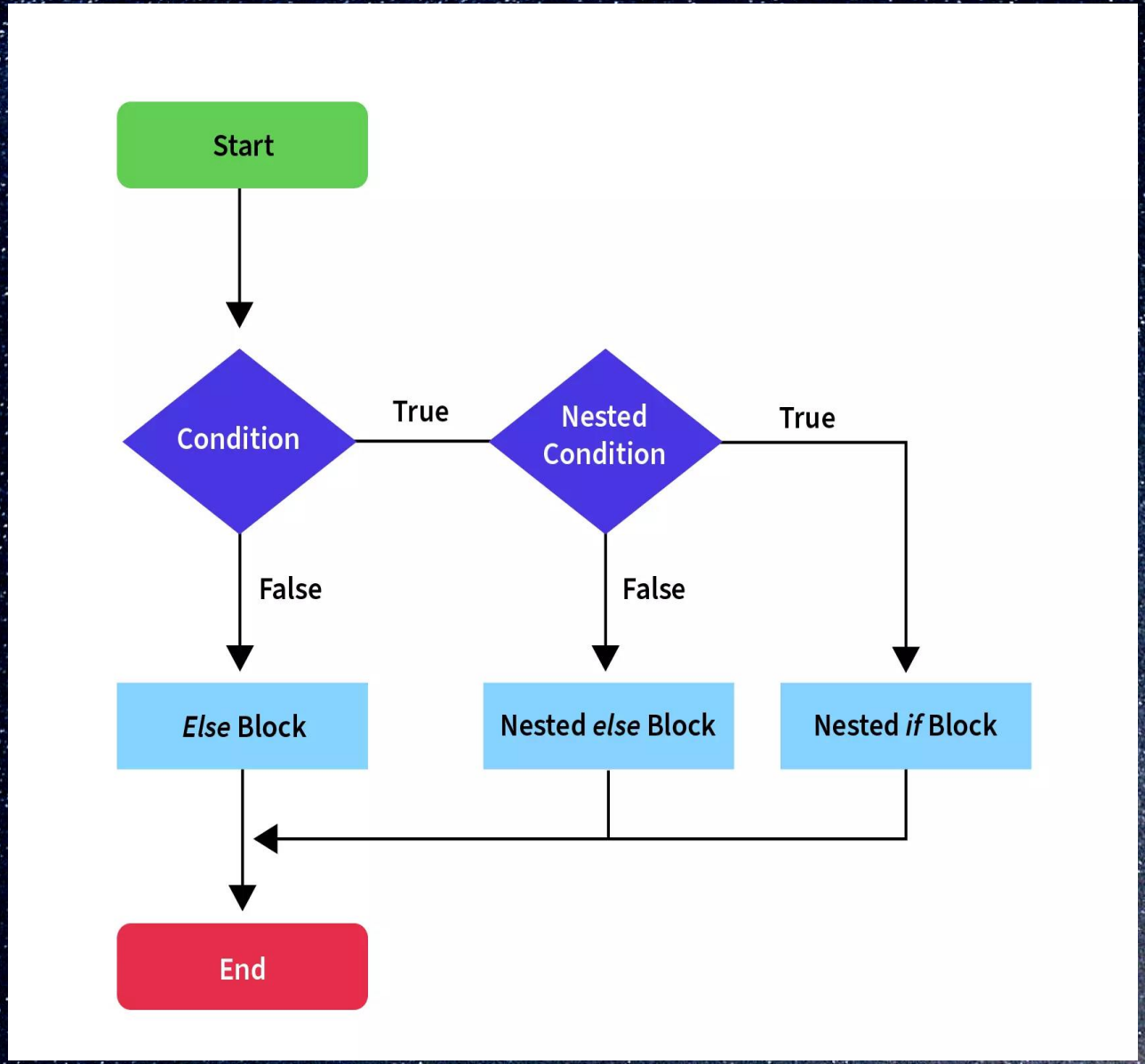
Looping

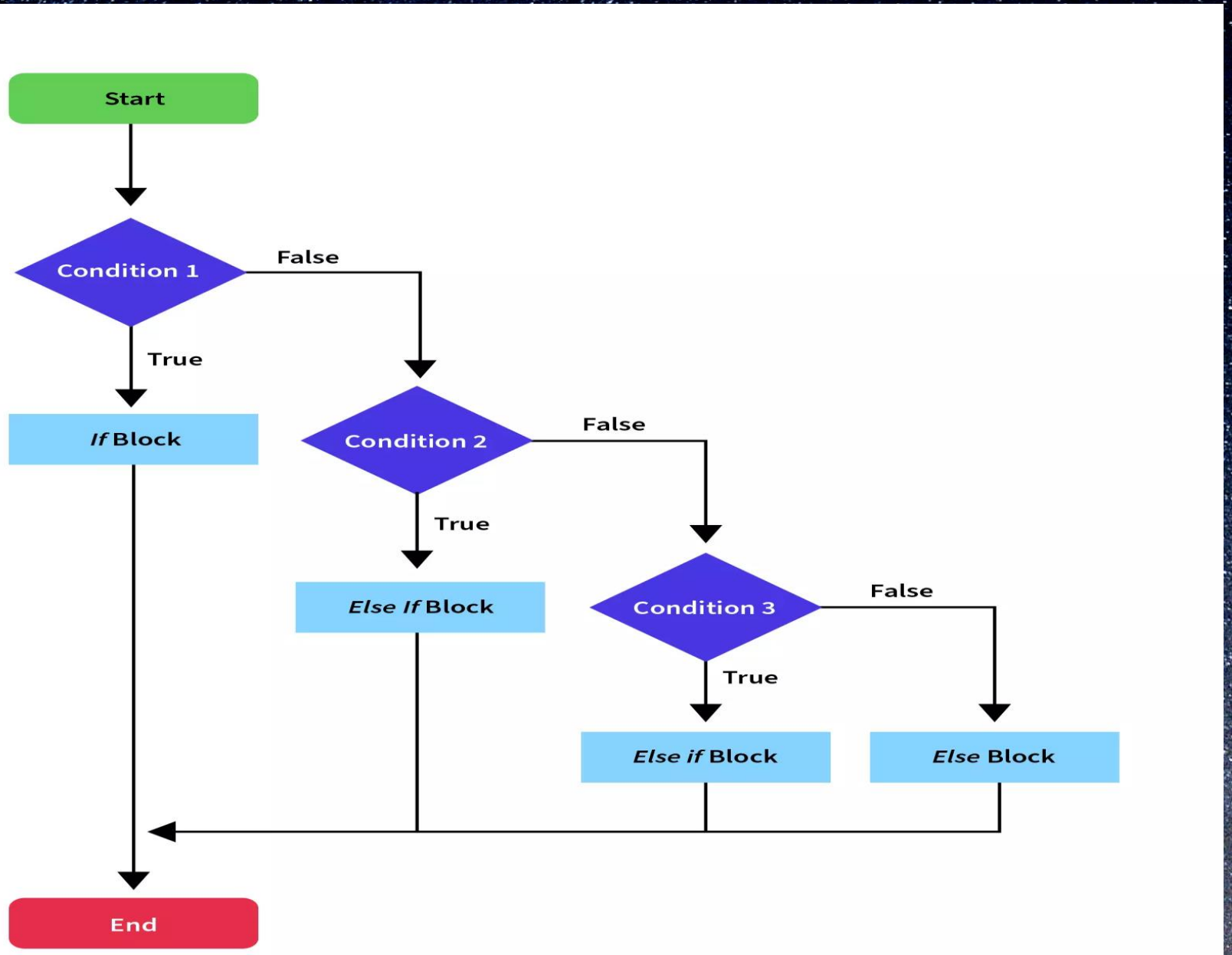
- *calling**While**LoopControl();*
- *calling**For**LoopControl();*
- *calling**ForEach**LoopControl();*

Jump/Branching Statements

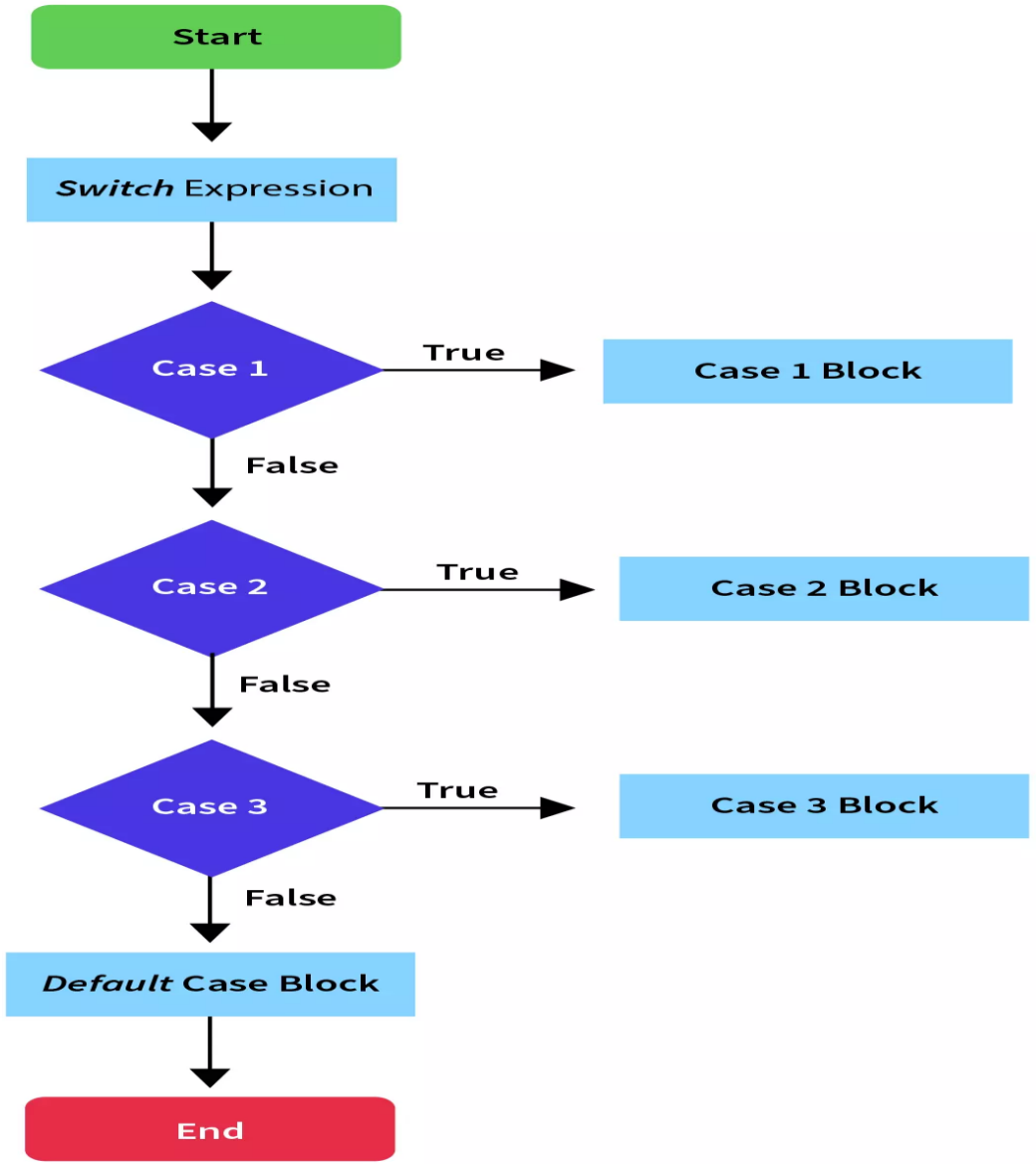
- *using**Break**InControl();*
- *using**Continue**InControl();*
- *using**Return**InControl();*

If Else

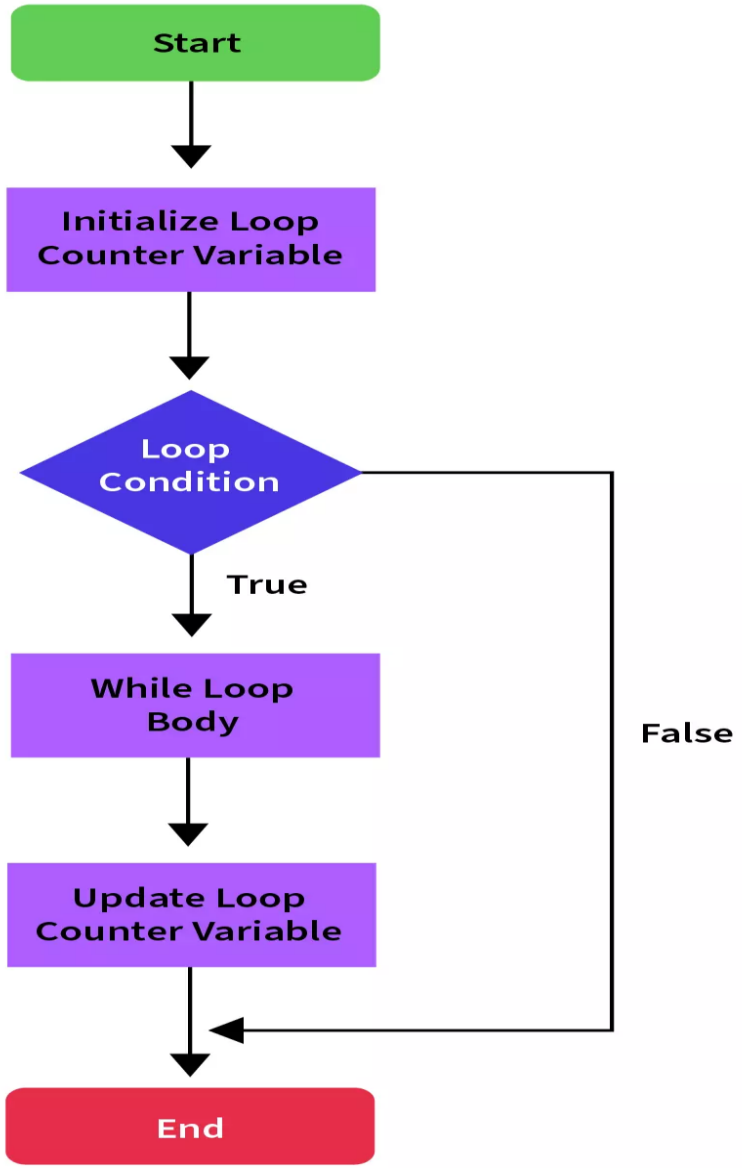

Start
Condition
True
Nested Condition
True
False
False
Else Block
Nested else Block
Nested if Block
End



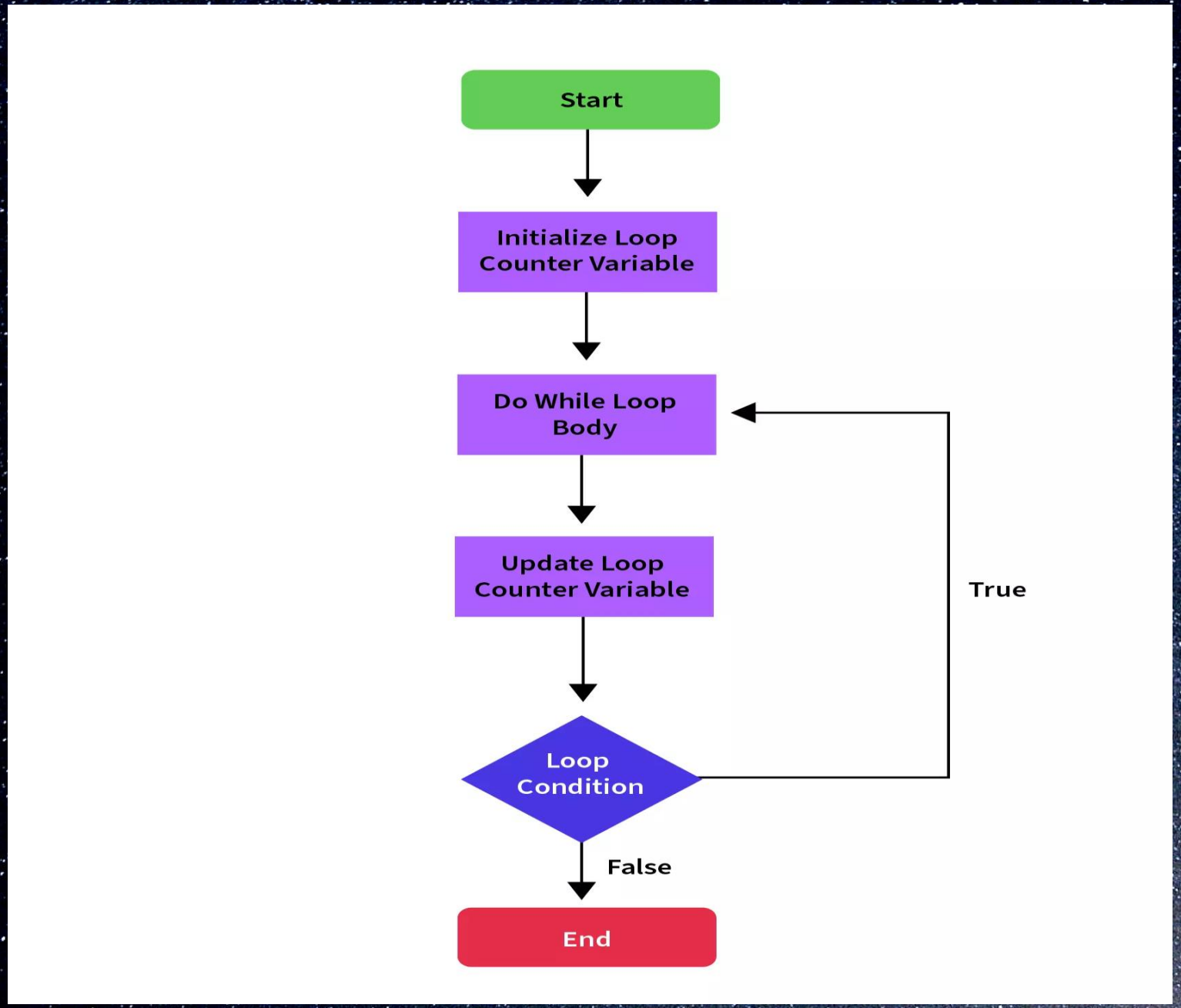
PM DIGI CLOUD

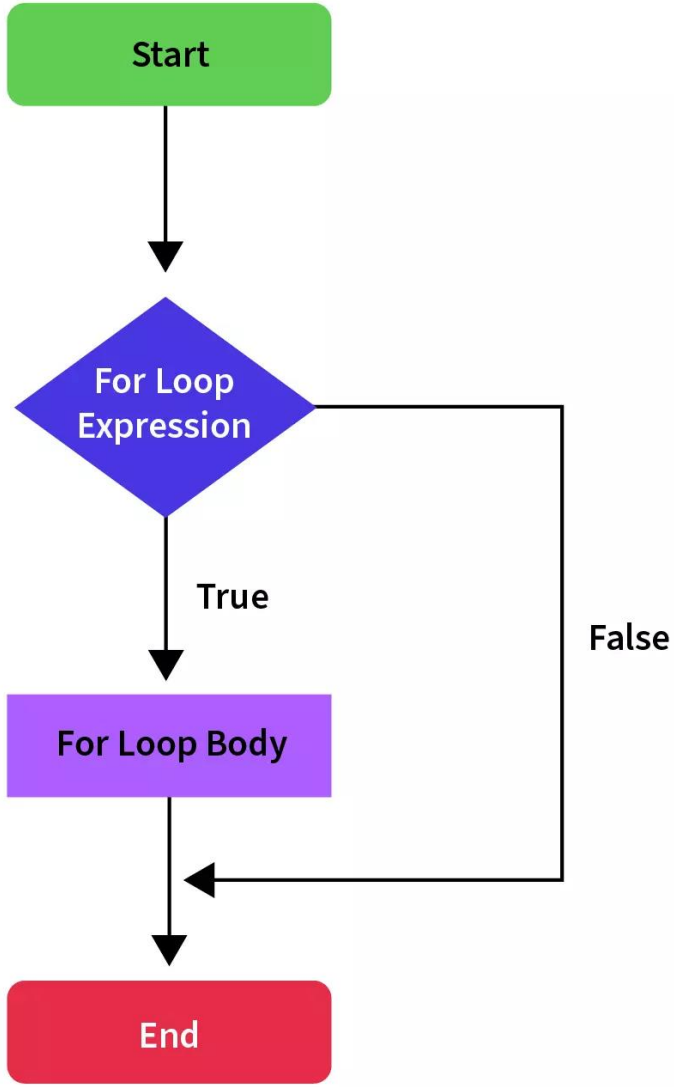


PM DIGI CLOUD

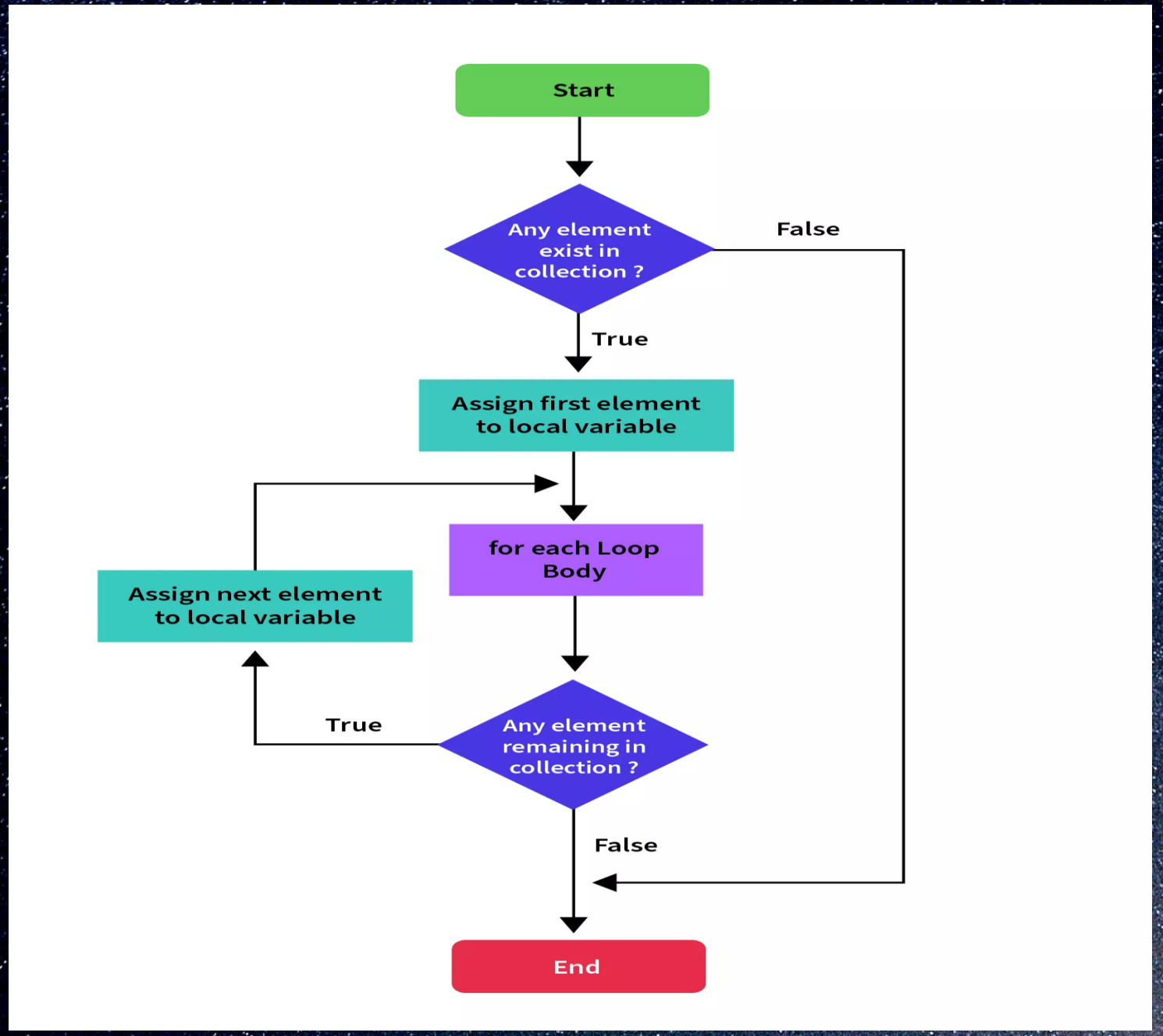


PM DIGI CLOUD

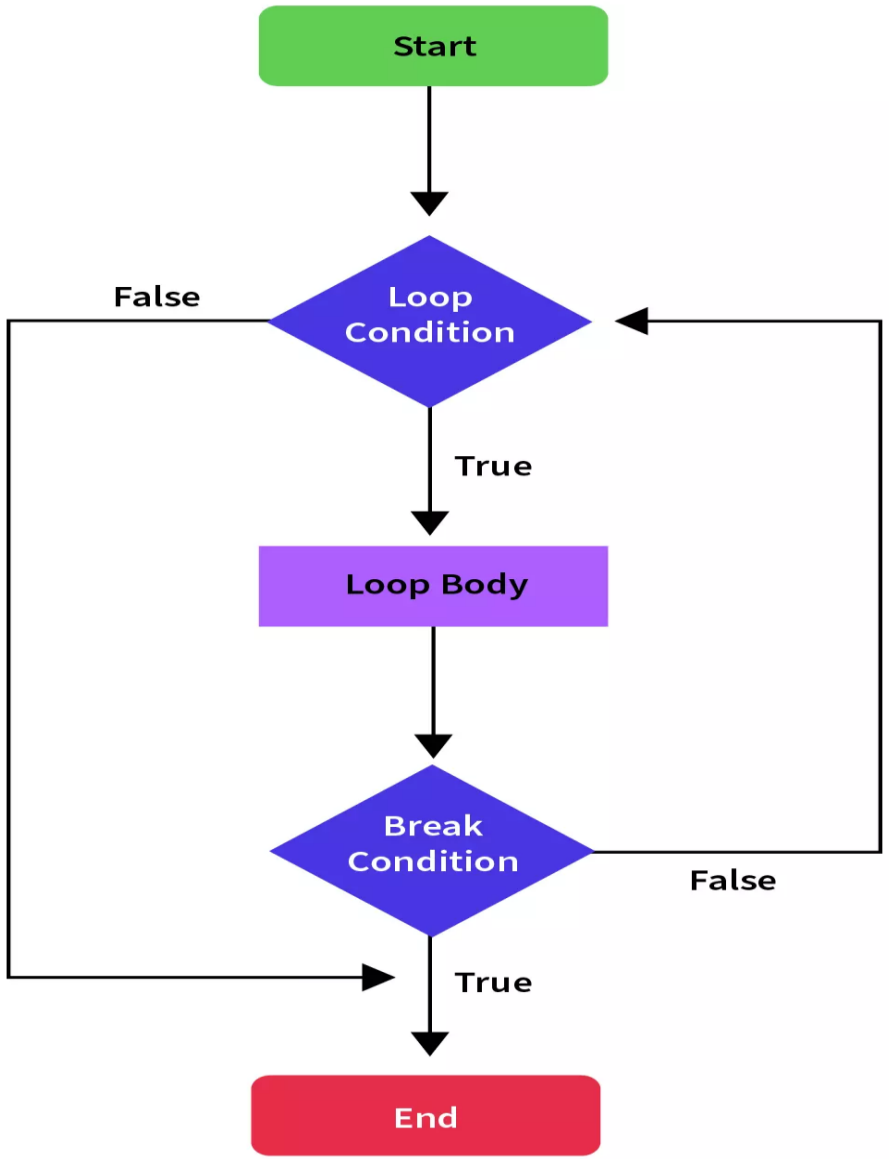

Start
Initialize Loop Counter Variable
Do While Loop Body
Update Loop Counter Variable
Loop Condition
True
False
End



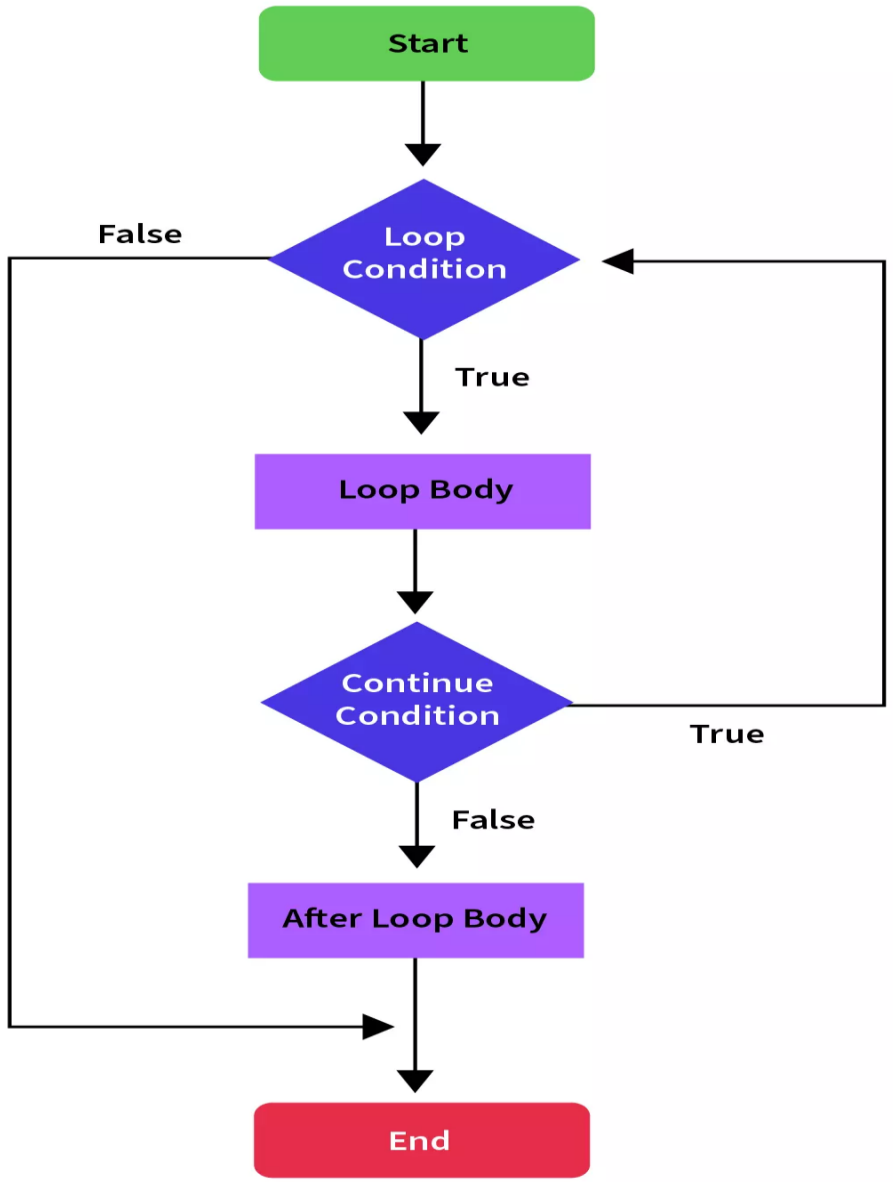
PM DIGI CLOUD



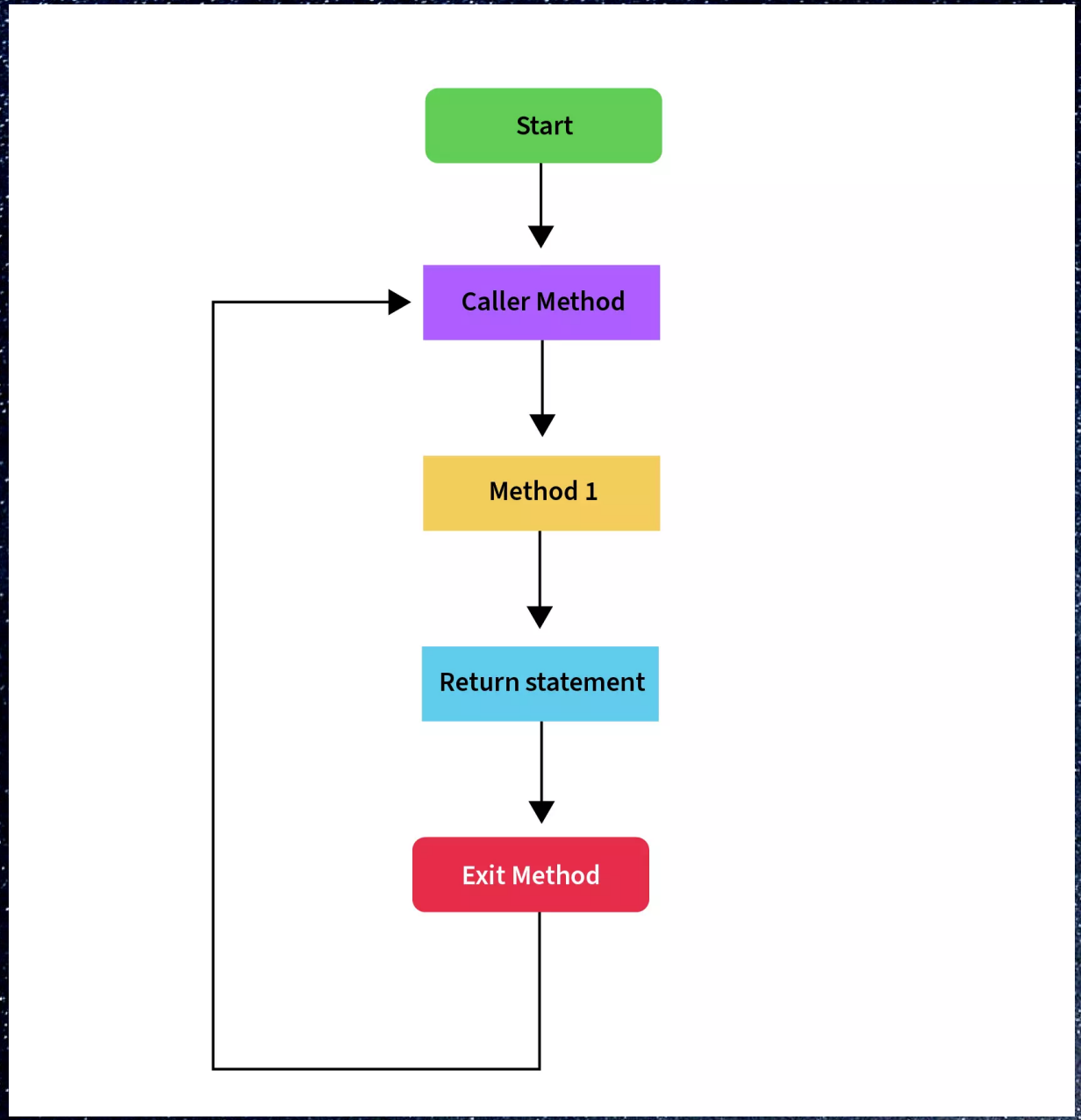
PM DIGI CLOUD



PM DIGI CLOUD



PM DIGI CLOUD



PM DIGI CLOUD

JAVA Tutorial

For Beginners

PM DIGI CLOUD

#7 ZERO to HERO

Arrays in detail #HandsOn

How Array store in memory

Multi Dimensional Arrays

Cloning in Arrays

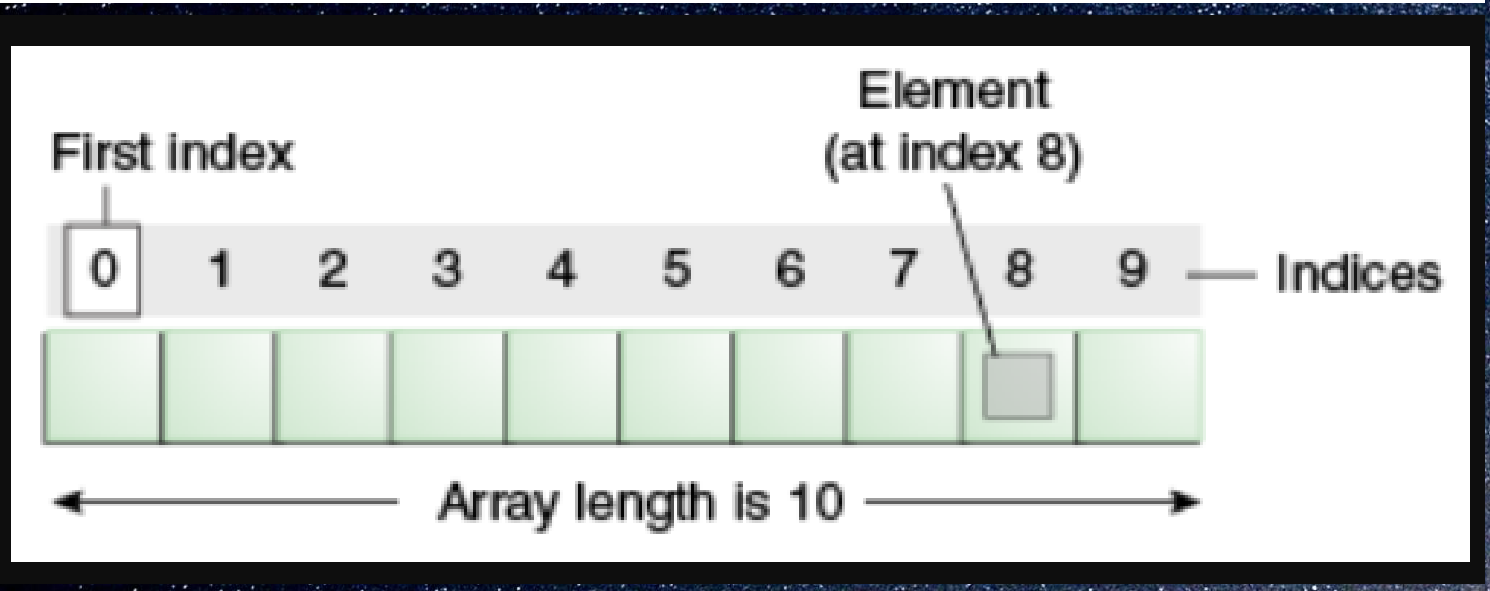
Compare, fill, binary search , copy of, equals methods in Array

#Arrays

PM DIGI CLOUD

Array is an object in java which contains similar types of data.

```java
int[] arr; //declare an integer array
arr = new int[3]; // create an array of 3 elements

//initialize array elements
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
```



#Arrays 2D

PM DIGI CLOUD

```
int[][] a = new int[3][4];
```

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

```
int[][] a = {
        {1, 2, 3},
        {4, 5, 6, 9},
        {7},
};
```

	Column 1	Column 2	Column 3	Column 4
Row 1	1 a[0][0]	2 a[0][1]	3 a[0][2]	
Row 2	4 a[1][0]	5 a[1][1]	6 a[1][2]	9 a[1][3]
Row 3	7 a[2][0]			

Array vs Array List

ArrayList	Array
Dynamic size(will grow and shrink automatically)	Fixed size
Can work with objects only	Can work with objects and primitives
All the manipulations are done by using methods(add, get, set)	The manipulations are done by using [] and indexes
size() is used to get number of elements	length is used to get number of elements

#Arrays

Passing an Array to a method or function

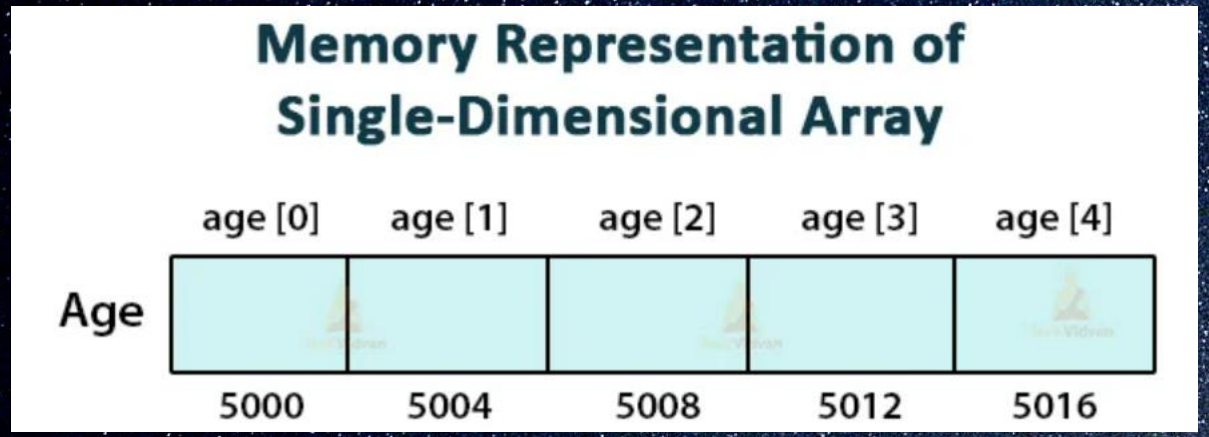
```
int[] num = new int[10];   //array declaration

methodName (num);       //calling the method

methodName (int [] x)     //passing an array to method
{
  //method body
}
```

#Arrays in detail - 1

PM DIGI CLOUD

Arrays are the object of a class, and the **class 'Object'** is the direct superclass of arrays. The members of an array are:

1. The field **public final length** which stores the number of elements of the array. length can be either zero or positive, but never be negative.
2. Calling Clone to single array is deep cloning and multi dimensional araay is shallow copy of cloning.
3. ArrayIndexOutOFBoundExceptions
4. Array [M][N] contains M*N elements



Arrays in detail -2

int compare(array1, array2)

void fill(originalArray, value)

boolean equals(array1, array2)

int binarySearch(array [], value)

copyOf(originalArray, newLength)

JAVA Tutorial

PM DIGI CLOUD

For Beginners

#8 ZERO to HERO

Abstract Classs vs Interface

Why, How and When to use?

Abstract class – Parent Class

PM DIGI CLOUD

An abstract class is defined as a class that's declared with the "abstract" keyword.
It should include at least one abstract method.
An abstract class cannot be instantiated, meaning you cannot create an object with it.

```
3  public abstract class VehicleParent {
4
5          abstract void start();
6          abstract void stop();
7          abstract void drive();
8          abstract void changeGear();
9          abstract void reverse();
0  }
1
```

```
abstract class Animal {
    Animal() {
        ....
    }
}

class Dog extends Animal {
    Dog() {
        super();
        ...
    }
}
```

Interface – Contract

An interface is a blueprint used to implement a class. It is a collection of abstract methods and contains no concrete methods, unlike abstract class. Interface can extend another interface as well.

Like a class, an interface can contain methods and variables, though the declared methods default to abstract.

```java
package practice;

import java.io.File;

public interface ISender {
    void send(File fileToBeSent);
}

```

```java
interface A {
  // members of A
}

interface B {
  // members of B
}

class C implements A, B {
  // abstract members of A
  // abstract members of B
}
```

```java
interface A {
   ...
}
interface B {
   ...
}

interface C extends A, B {
   ...
}
```

Abstract vs Interface

PM DIGI CLOUD

Abstract Class	Interface
An abstract class can contain both abstract and non-abstract methods.	Interface contains only abstract methods.
An abstract class can have all four; static, non-static and final, non-final variables.	Only final and static variables are used.
To declare abstract class abstract keywords are used.	The interface can be declared with the interface keyword.
It supports multiple inheritance.	It does not support multiple inheritance.
The keyword ‘extend’ is used to extend an abstract class	The keyword implement is used to implement the interface.
It has class members like private and protected, etc.	It has class members public by default.

Imp Points

- You can only extend to one class.
- You can implement interface and extend class as well.
- You can implement multiple interface.
- You can extend interface with interface.
- You can extend Abstract with Abstract.
- It is must to implement methods in child classes extending abstract class until they again are being defined as abstract
- Best practice is define constant only in interface if need of variable.

JAVA Tutorial

For Beginners

PM DIGI CLOUD

#9 ZERO to HERO

Strings in detail #HandsOn

How String store in memory

String pool vs String Heap

String Builder vs String Buffer

Imp methods in Strings

String

PM DIGI CLOUD

In Java, a string is a sequence of characters
We use **double quotes** to represent a string in Java.

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

    // create strings
    String first = "Java";
    String second = "Python";
    String third = "JavaScript";

    // print strings
    System.out.println(first);   // print Java
    System.out.println(second);  // print Python
    System.out.println(third);   // print JavaScript
  }
}
```

Note: Strings in Java are not primitive types (like int, char, etc). Instead, all strings are objects of a predefined class named String.

And, all string variables are instances of the String class.

String Operations

contains()	checks whether the string contains a substring
substring()	returns the substring of the string
join()	join the given strings using the delimiter
replace()	replaces the specified old character with the specified new character
replaceAll()	replaces all substrings matching the regex pattern
replaceFirst()	replace the first matching substring
charAt()	returns the character present in the specified location
getBytes()	converts the string to an array of bytes
indexOf()	returns the position of the specified character in the string

String Operations

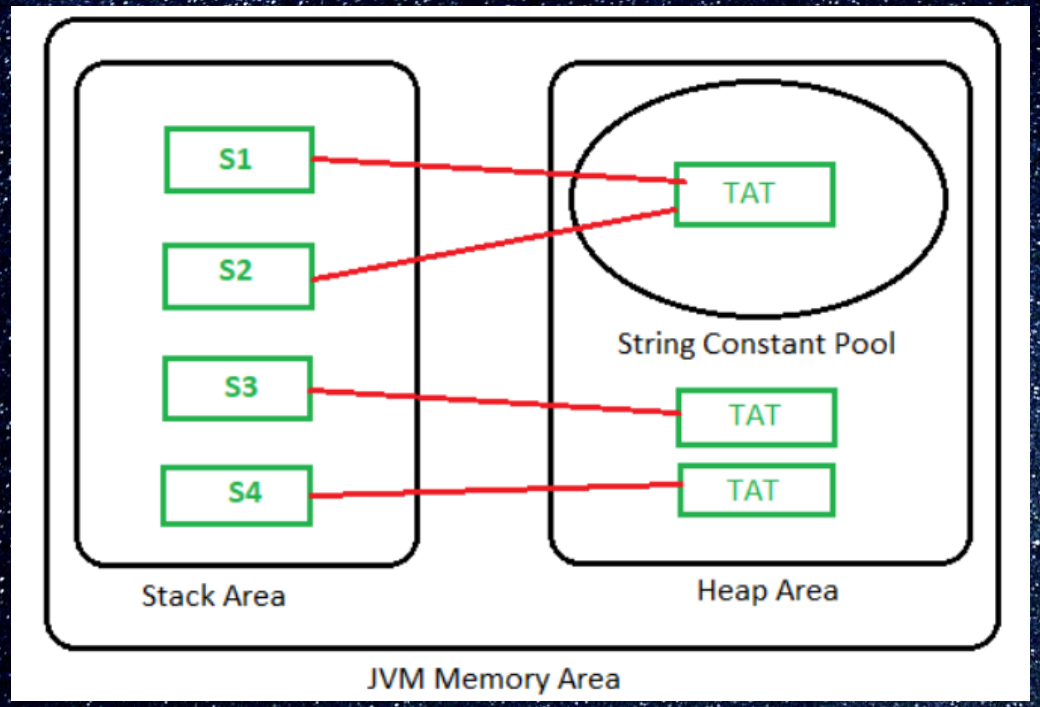
compareTo()	compares two strings in the dictionary order
compareToIgnoreCase()	compares two strings ignoring case differences
trim()	removes any leading and trailing whitespaces
format()	returns a formatted string
split()	breaks the string into an array of strings
toLowerCase()	converts the string to lowercase
toUpperCase()	converts the string to uppercase
valueOf()	returns the string representation of the specified argument
toCharArray()	converts the string to a char array

String Operations

matches()	checks whether the string matches the given regex
startsWith()	checks if the string begins with the given string
endsWith()	checks if the string ends with the given string
isEmpty()	checks whether a string is empty of not
intern()	returns the canonical representation of the string
contentEquals()	checks whether the string is equal to charSequence
hashCode()	returns a hash code for the string
subSequence()	returns a subsequence from the string

String and memory

PM DIGI CLOUD



When we create a *String* object using the *new()* operator, it always creates a new object in heap memory.

On the other hand, if we create an object using *String* literal syntax e.g. “Baeldung”, it may return an existing object from the String pool, if it already exists. Otherwise, it will create a new String object and put in the string pool for future re-use.

PM DIGI CLOUD

#shorts Quick Info

Pass by Value

vs

Pass by Reference

in JAVA

Java

Java is Pass by Value only

PM DIGI CLOUD

Pass by Value: The method parameter values are copied to another variable and then the copied object is passed, that's why it's called pass by value.

Pass by Reference: An alias or reference to the actual parameter is passed to the method, that's why it's called pass by reference.

Java is always Pass by Value and not pass by reference, we can prove it with a simple example.
we pass a copy of the reference and hence it's pass by value.

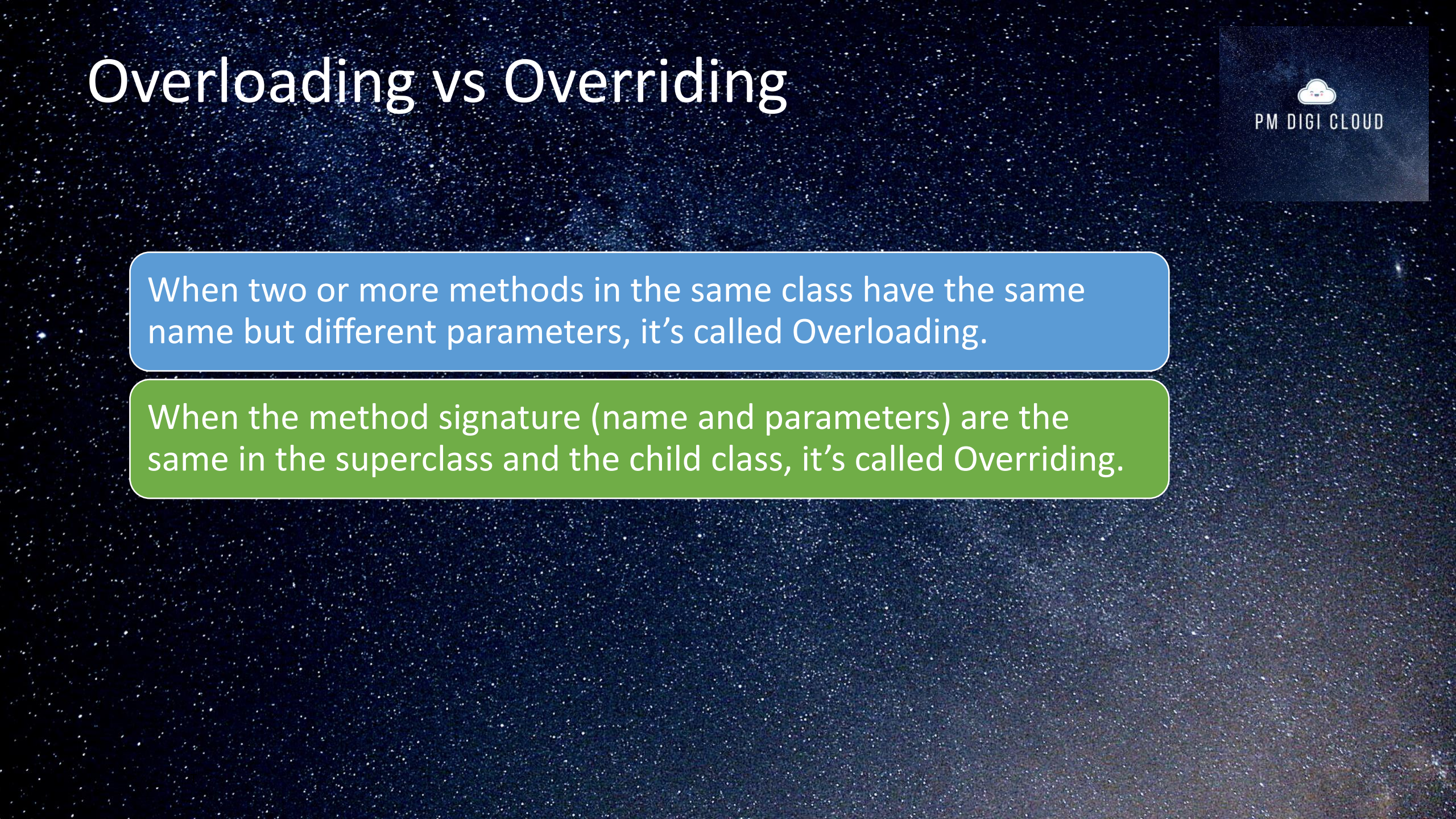
PM DIGI CLOUD

#shorts Quick Info

Overloading vs Overriding in JAVA

Java

Overloading vs Overriding



When two or more methods in the same class have the same name but different parameters, it’s called Overloading.

When the method signature (name and parameters) are the same in the superclass and the child class, it’s called Overriding.

Overloading vs Overriding

- Overriding implements Runtime Polymorphism whereas Overloading implements Compile time polymorphism.
- The method Overriding occurs between superclass and subclass. Overloading occurs between the methods in the same class.
- Overriding methods have the same signature i.e. same name and method arguments. Overloaded method names are the same but the parameters are different.
- With Overloading, the method to call is determined at the compile-time. With overriding, the method call is determined at the runtime based on the object type.
- If overriding breaks, it can cause serious issues in our program because the effect will be visible at runtime. Whereas if overloading breaks, the compile-time error will come and it’s easy to fix.

PM DIGI CLOUD

#shorts Quick Info

Final Variable
Final Method
Finally block
in JAVA

Java

Final vs Finally

- Finally is used in try catch finally block
- Final variable is constant which can not be changed
- Final methods can not be overridden

JAVA Tutorial

PM DIGI CLOUD

#10

For Beginners

ZERO to HERO

Know the difference

== and equals

Why, How and When to use?

== and equals

== and equals by default compare reference of objects

Equals can only compare value of object if you have overridden and mentioned logic to do so.

== and equals compare values by default in String

Equals and HasCode contract must know if using HashMap or has set

JAVA Tutorial

PM DIGI CLOUD

For Beginners

#11 ZERO to HERO

How to sort

Primitive variables

Objects

Comparable vs Comparator

Sorting of Objects

Java provides two interfaces to sort objects using data members of the class:

Comparable
Comparator

Collections.sort()

A comparable object is capable of comparing itself with another object.

Unlike Comparable, Comparator is external to the element type we are comparing. It’s a separate class. We create multiple separate classes (that implement Comparator) to compare by different members.

JAVA Tutorial

PM DIGI CLOUD

#12

For Beginners

ZERO to HERO

Serialization

File Reading

File Writing

#Serialization

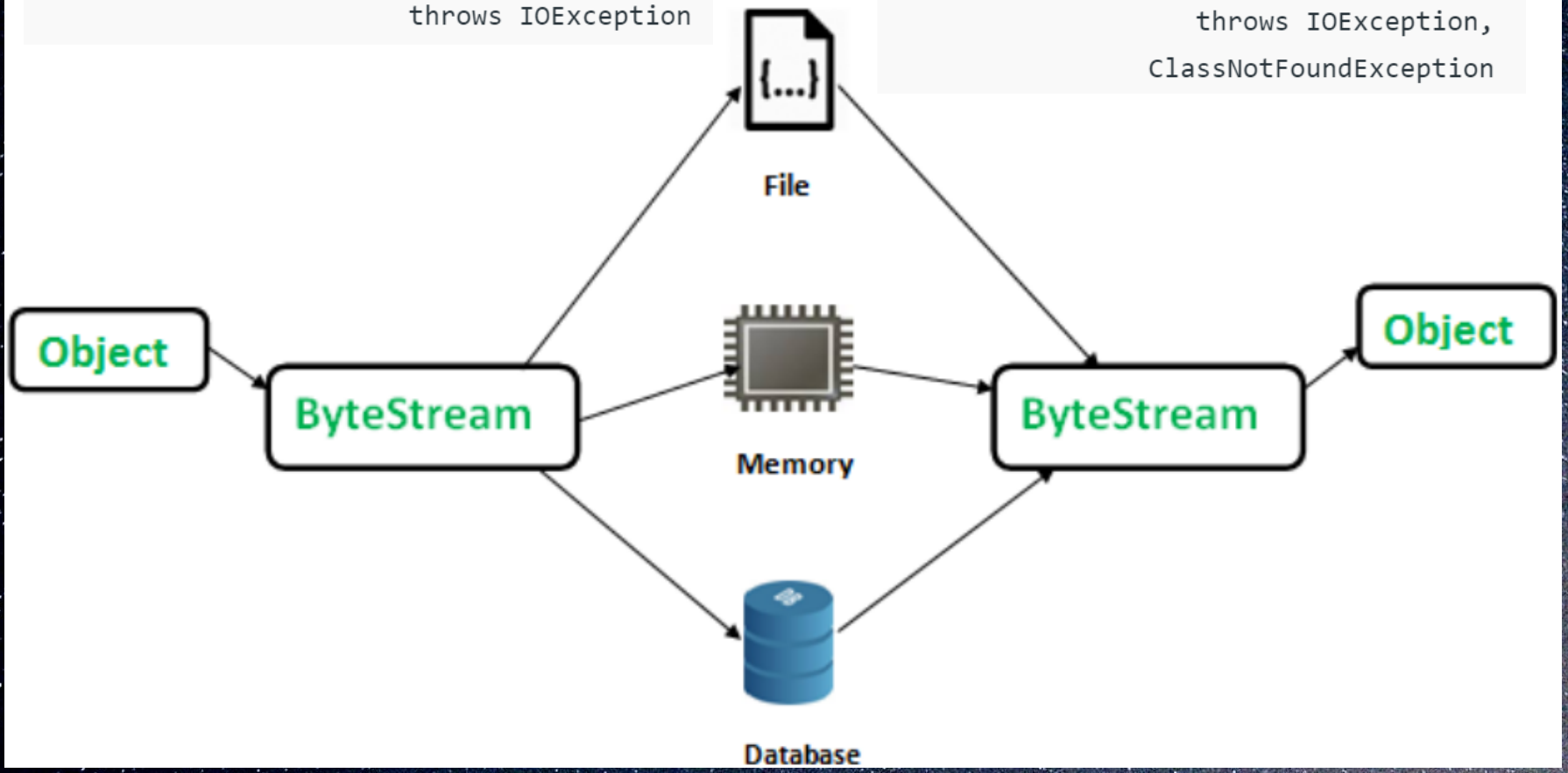
PM DIGI CLOUD

Serialization

```
public final void writeObject(Object obj)
                        throws IOException
```

De-Serialization

```
public final Object readObject()
                    throws IOException,
                ClassNotFoundException
```



#Points to Remember in serialization

- Serializable is a marker interface (has no data member and method).
- If a parent class has implemented Serializable interface then child class doesn’t need to implement it but vice-versa is not true.
- Only non-static data members are saved via Serialization process.
- Static data members and transient data members are not saved via Serialization process.So, if you don’t want to save value of a non-static data member then make it transient.
- Constructor of object is never called when an object is deserialized.
- Associated objects must be implementing Serializable interface.

#SerialVersionUID

The Serialization runtime associates a version number with each Serializable class called a SerialVersionUID, which is used during Deserialization to verify that sender and receiver of a serialized object have loaded classes for that object which are compatible with respect to serialization.

InvalidClassException is thrown of id id different

A Serializable class can declare its own UID explicitly by declaring a field name.
It must be static, final and of type long.A Serializable class can declare its own UID explicitly by declaring a field name.
It must be static, final and of type long.

#File Writing

FileWriter is useful to create a file writing characters into it.

This class inherits from the OutputStream class.
The constructors of this class assume that the default character encoding and the default byte-buffer size are acceptable. To specify these values yourself, construct an OutputStreamWriter on a FileOutputStream.

FileWriter is meant for writing streams of characters. For writing streams of raw bytes, consider using a FileOutputStream.
FileWriter creates the output file if it is not present already.

#File Reading

FileReader is useful to read data in the form of characters from a ‘text’ file.

This class inherited from the InputStreamReader Class.
The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate. To specify these values yourself, construct an InputStreamReader on a FileInputStream.

FileReader is meant for reading streams of characters. For reading streams of raw bytes, consider using a FileInputStream.

JAVA Tutorial

PM DIGI CLOUD

#12

For Beginners

ZERO to HERO

Know everything about

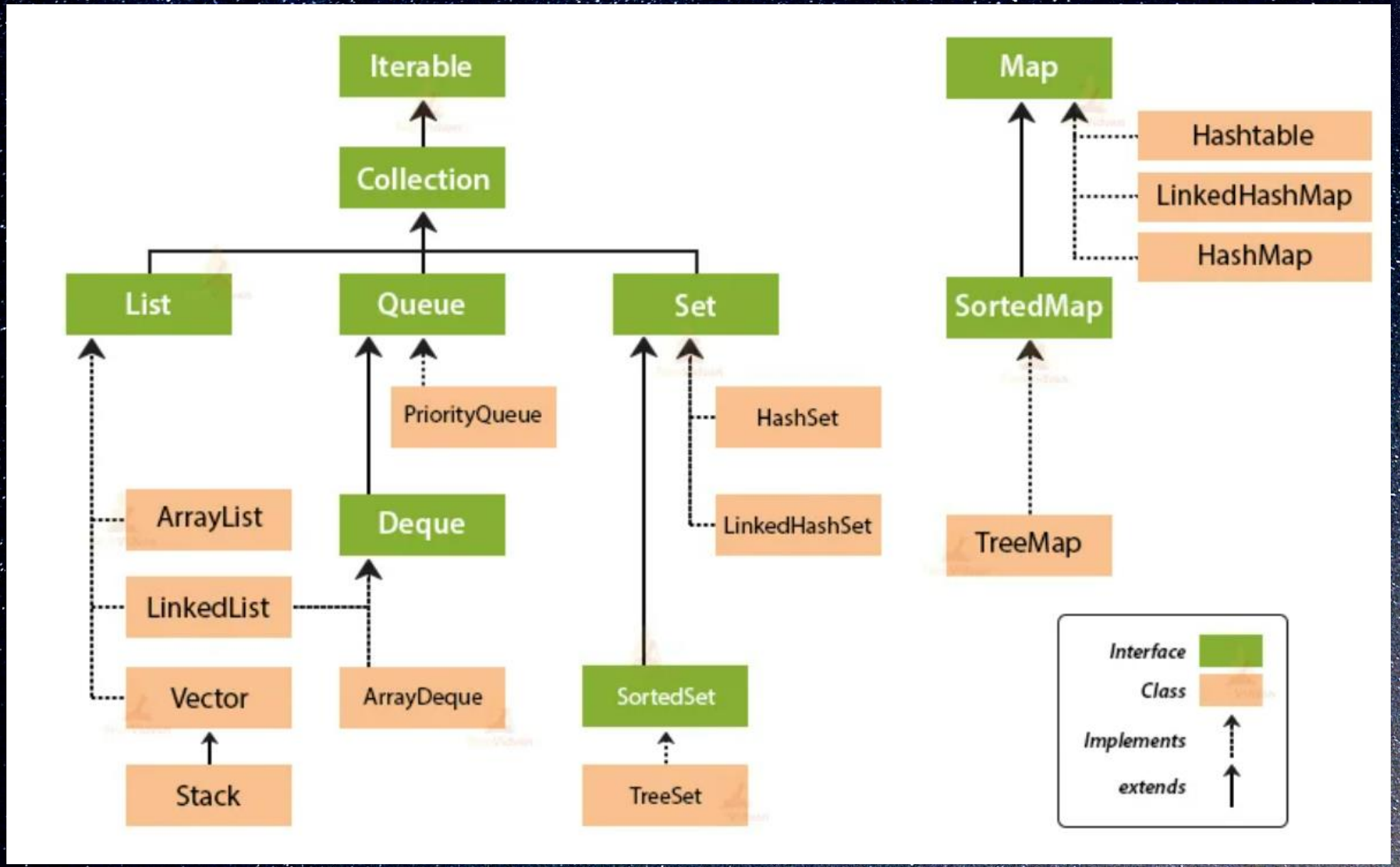
Collection Interface

List Set Map

Collections utility class

#Collection

PM DIGI CLOUD



#Collection

PM DIGI CLOUD

Interface	**Has Duplicates?**	Implementations					Historical
Set	no	HashSet	...	LinkedHashSet	...	TreeSet	...
List	yes	...	ArrayList	...	LinkedList	...	Vector, Stack
Map	no duplicate keys	HashMap	...	LinkedHashMap	...	TreeMap	Hashtable, Properties

"ordering" will here refer to the order of items returned by an Iterator.
"sorting" will here refer to sorting items according to Comparable or Comparator.

#Collection – Principal Features

- HashMap has slightly better performance than LinkedHashMap, but its iteration order is undefined
- HashSet has slightly better performance than LinkedHashSet, but its iteration order is undefined
- TreeSet is ordered and sorted, but slower
- TreeMap is ordered and sorted, but slower
- LinkedList has fast adding to the start of the list, and fast deletion from the interior via iteration
- For and , the re-insertion of an item does not affect insertion order.
- For , 'access order' is from the least recent access to the most recent access.
- keys of a Map, items in a Set recommended to be mutable
- To retain the order of a as specified in an ORDER BY clause, insert the records into a or a .

#Collection – Iteration Order

- HashSet - undefined
- HashMap - undefined
- LinkedHashSet - insertion order
- LinkedHashMap - insertion order of keys (by default), or 'access order'
- ArrayList - insertion order
- LinkedList - insertion order
- TreeSet - ascending order, according to Comparable / Comparator
- TreeMap - ascending order of keys, according to Comparable / Comparator

JAVA Tutorial

PM DIGI CLOUD

#13

For Beginners

ZERO to HERO

Multi - Threading

#Handson

Threading

Process and Thread are two basic units of execution. Concurrency programming is more concerned with java threads.

- Java Threads are lightweight compared to processes, it takes less time and resource to create a thread.
- Threads share their parent process data and code
- Context switching between threads is usually less expensive than between processes.
- Thread intercommunication is relatively easy than process communication.

Java provides two ways to create a thread programmatically.

1. Implementing the java.lang.Runnable interface.
2. Extending the java.lang.Thread class.

Thread Safety in Java

1. Synchronization is the easiest and most widely used tool for thread safety in java.
2. Use of Atomic Wrapper classes from java.util.concurrent.atomic package. For example AtomicInteger
3. Use of locks from java.util.concurrent.locks package.
4. Using thread safe collection classes, check this post for usage of ConcurrentHashMap for thread safety.
5. Using volatile keyword with variables to make every thread read the data from memory, not read from thread cache.

JAVA Tutorial

PM DIGI CLOUD

For Beginners

#14 ZERO to HERO

Generics

Wrapper Classes

Java Generics

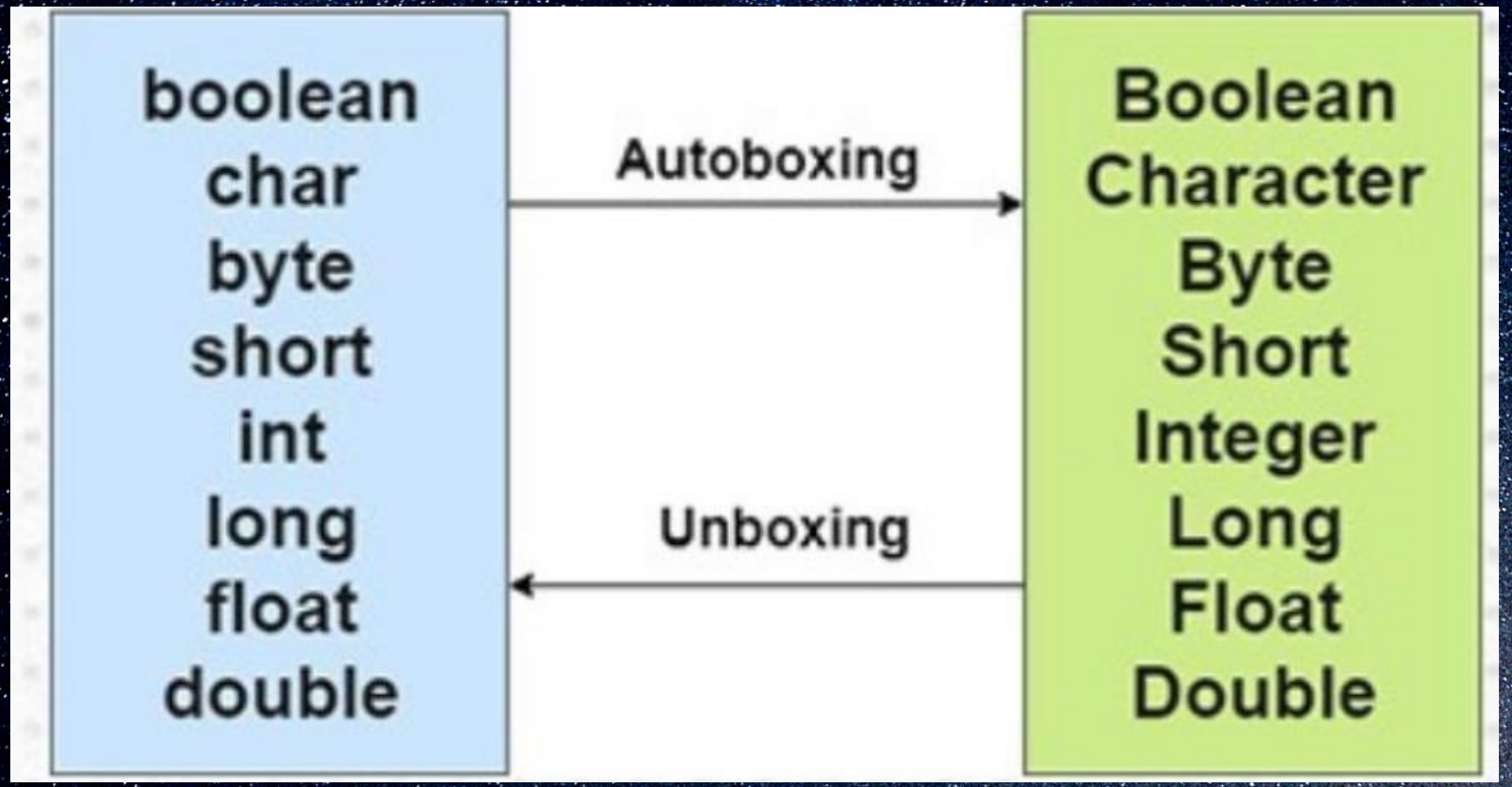
We can create a class that can be used with any type of data. Such a class is known as Generics Class.

```
class GenericsClass<T> {...}
```

```
public <T> void genericMethod(T data) {...}
```

Java Wrapper classes

PM DIGI CLOUD



JAVA Tutorial

PM DIGI CLOUD

#15

For Beginners

ZERO to HERO

Stream

Lambda

Inner Classes

Java Lambda Expressions

Lambda Expressions were added in Java 8.

A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

```
parameter -> expression
```

```
(parameter1, parameter2) -> expression
```

```
(parameter1, parameter2) -> { code block }
```

Streams

1. A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
2. Streams don’t change the original data structure, they only provide the result as per the pipelined methods.
3. Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

Different Operations On Streams

map: The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.

```
List number = Arrays.asList(2,3,4,5);
List square = number.stream().map(x->x*x).collect(Collectors.toList());
```

filter: The filter method is used to select elements as per the Predicate passed as argument.

```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().filter(s-
>s.startsWith("S")).collect(Collectors.toList());
```

sorted: The sorted method is used to sort the stream.

```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().sorted().collect(Collectors.toList());
```

Different Operations On Streams

collect: The collect method is used to return the result of the intermediate operations performed on the stream.

```
List number = Arrays.asList(2,3,4,5,3);
Set square = number.stream().map(x->x*x).collect(Collectors.toSet());
```

forEach: The forEach method is used to iterate through every element of the stream.

```
List number = Arrays.asList(2,3,4,5);
number.stream().map(x->x*x).forEach(y->System.out.println(y));
```

reduce: The reduce method is used to reduce the elements of a stream to a single value.

The reduce method takes a BinaryOperator as a parameter.

```
List number = Arrays.asList(2,3,4,5);
int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);
```

JAVA Tutorial

PM DIGI CLOUD

#16

For Beginners

ZERO to HERO

Creating Web Application Servlet JSP

JAVA Tutorial

PM DIGI CLOUD

#17

For Beginners

ZERO to HERO

Creating REST Web Services API