**CORE JAVA**

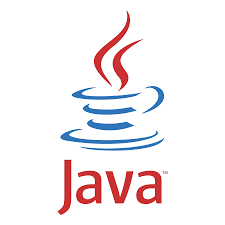
What is Java?

Java is a popular programming language, created in 1995.

It is owned by Oracle, and more than 3 billion devices run Java.

It is used for:

* Mobile applications (especially Android apps)
* Desktop applications
* Web applications
* Web servers and application servers
* Games
* Database connection
* And much, much more!

****

**Why Use Java?**

1. Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
2. It is one of the most popular programming languages in the world
3. It has a large demand in the current job market
4. It is easy to learn and simple to use
5. It is open-source and free
6. It is secure, fast and powerful
7. It has huge community support (tens of millions of developers)
8. Java is an object-oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
9. As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

|  |  |
| --- | --- |
| **Born** | James Arthur Gosling  May 19, 1955 (age 69)  [Calgary](https://en.wikipedia.org/wiki/Calgary), [Alberta](https://en.wikipedia.org/wiki/Alberta), Canada |
| **Alma mater** | * + [University of Calgary](https://en.wikipedia.org/wiki/University_of_Calgary) ([BSc](https://en.wikipedia.org/wiki/Bachelor_of_Science), 1977)   + [Carnegie Mellon University](https://en.wikipedia.org/wiki/Carnegie_Mellon_University) ([MA](https://en.wikipedia.org/wiki/Master_of_Arts); [PhD](https://en.wikipedia.org/wiki/Doctor_of_Philosophy), 1983) |
| **Known for** | [Java (programming language)](https://en.wikipedia.org/wiki/Java_(programming_language)) |
| **Children** | 2 |
| **Awards** | Officer of the [Order of Canada](https://en.wikipedia.org/wiki/Order_of_Canada)  IEEE John von Neumann Medal The Economist Innovation Award NAE Foreign Member  Fellow [Computer History Museum](https://en.wikipedia.org/wiki/Computer_History_Museum) |
| **Scientific career** | |
| **Institutions** | * [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) * [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation) * [Google](https://en.wikipedia.org/wiki/Google) * [Liquid Robotics](https://en.wikipedia.org/wiki/Liquid_Robotics)[[1]](https://en.wikipedia.org/wiki/James_Gosling#cite_note-nighthacks1-1) * [Amazon Web Services](https://en.wikipedia.org/wiki/Amazon_Web_Services) |
| [**Thesis**](https://en.wikipedia.org/wiki/Thesis) | [*Algebraic Constraints*](http://reports-archive.adm.cs.cmu.edu/anon/scan/CMU-CS-83-132.pdf) (1983) |
| [**Doctoral advisor**](https://en.wikipedia.org/wiki/Doctoral_advisor) | [Bob Sproull](https://en.wikipedia.org/wiki/Bob_Sproull) and [Raj Reddy](https://en.wikipedia.org/wiki/Raj_Reddy)[[2]](https://en.wikipedia.org/wiki/James_Gosling#cite_note-mathgene-2) |



[**James Gosling**](https://en.wikipedia.org/wiki/James_Gosling)

**James Gosling** [OC](https://en.wikipedia.org/wiki/Order_of_Canada) (born 19 May 1955) is a Canadian [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist), best known as the founder and lead designer behind the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)).[[3]](https://en.wikipedia.org/wiki/James_Gosling#cite_note-3)

Gosling was elected a member of the [National Academy of Engineering](https://en.wikipedia.org/wiki/National_Academy_of_Engineering) in 2004 for the conception and development of the architecture for the Java programming language and for contributions to [window systems](https://en.wikipedia.org/wiki/Windowing_system).

He is known as the father of the [Java programming language](https://en.wikipedia.org/wiki/Java_programming_language).[[](https://en.wikipedia.org/wiki/James_Gosling#cite_note-LeavesSun-9)

Gosling was with Sun Microsystems between 1984 and 2010 (26 years). At Sun he invented an early Unix windowing system called [NeWS](https://en.wikipedia.org/wiki/NeWS" \o "NeWS), which became a lesser-used alternative to the still used [X Window System](https://en.wikipedia.org/wiki/X_Window_System), because Sun did not give it an open source license. He got the idea for the [Java VM](https://en.wikipedia.org/wiki/Java_virtual_machine) while writing a program to port software from a [PERQ](https://en.wikipedia.org/wiki/PERQ) by translating Perq Q-Code to VAX assembler and emulating the hardware. He is generally credited with having invented the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language in 1994.

**3 Tier Architecture**

**Database**

**Middleware**

**Web-Technologies**

**Sql, MySql, MangoDB**

-Data Of the Application.

**Core Java, Advanced Java**

-Logic Of the Application.

**Html, Css, JavaScript ReactJS**

-Look & Feel

**FRONT-END**

**BACK-END**

**DATABASE**

**Language**

-It is Predefined set of rules and regulations that must be followed in order to established the communication between two different entities.

-Language are categorized into two types.

**1]Verbal Language**

**2]Programming Language.**

**1]Verbal Language**

-The type of language which we can read, write and speak that can be categorized as verbal language.

**e.g. :- English,Hindi,Marathi,Spannish.**

**2]Programming Language**

-Programming language it is set of instruction to perform a particular task by the machine which is given by the programmer is called as Programming language.

**OR**

-The type of language we can read, write but cannot speak that can be categorized as Programming language.

-It is generally used to communicate the machine.

-Based on its application programming language are further categorized into two types: -

-Based on its application programming language are further categorized into two types: -

**1]General Purpose Language**

**2]Domain Specific Language.**

**1]General Purpose Language**

-The Programming Language which can be used for all the purposes is called as General-Purpose Programming language.

-With the help of General-Purpose Language, we can build entire application.

**e.g. :- Java, Python, C, C++, Ruby, Dot Net.**

-General purpose Programming Language are further more classified into three types.

**i]High Level Language**

**ii]Low Level Language**

**iii]Middle Level Language.**

**i]High Level Language**

-The Programming Language which is writeable, Readable, Understandable & Executable by the programmer such language are called as High Level Programming Language.

**e.g.: - Java, Python, C, C++, Ruby, Dot Net.**

**ii]Low Level Language**

-The Language which is Understandable & Executable by the machine is called as Low-Level Programming Language.

**e.g.: - Binary language, Assembly Language.**

**iii]Middle Level Language**

-The Programming Language with having features of both Low Level & High Level Programming Language such languages are called as Middle Level programming Language.

**e.g.: - Java, Python, C, C++, Ruby, Dot Net.**

**2]Domain Specific Language**

-The Programming language which help to build only specific part of application that language can be called as Domain Specific Programming Language.

**e.g.: -Sql, MySql**

**WHAT IS JAVA?**

-Java is general purpose, Class-based, Object-Oriented Programming Language.

-Java is widely-used programming language for coding Web application. It has been a popular choice among the developers for over two decades.

-It is faster, Secure, Reliable Programming Language for coding everything from Mobile Apps and Enterprise Software to Big Data Applications and Server-Side Technologies.

-Java is a Multithreading, Object-Oriented and Network Centric Language that can be used as platform in itself.

 -Currently, Java is owned by Oracle and more than 3 billion devices run Java.

-Java is a general-purpose programming language intended to let programmers **Write Once, Run Anywhere (WORA)**. This means that compiled Java code can run on all platforms that support Java without the need to recompile*.*

-It is used for:

* Mobile applications (especially Android apps)
* Desktop applications
* Web applications
* Web servers and application servers
* Games
* Database connection

And much, much more.

What is Java programming language used for?

**What is Java programming language used for?**

Because Java is a free-to-use and a versatile language, it builds localized and distributed software. Some common uses of Java include:

**1.     Game Development**

Many popular mobile, computer, and video games are built in Java. Even modern games that integrate advanced technology like [machine learning](https://aws.amazon.com/what-is/machine-learning/) or virtual reality are built with Java technology.

**2.     Cloud computing**

Java is often referred to as WORA – Write Once and Run Anywhere, making it perfect for decentralized cloud-based applications. Cloud providers choose Java language to run programs on a wide range of underlying platforms.

**3.     Big Data**

Java is used for data processing engines that can work with complex data sets and massive amounts of real-time data.

**4.     Artificial Intelligence**

Java is a powerhouse of machine learning libraries. Its stability and speed make it perfect for [artificial intelligence](https://aws.amazon.com/what-is/artificial-intelligence/) application development like natural language processing and deep learning.

**5.     Internet of Things**

Java has been used to program sensors and hardware in edge devices that can connect independently to the internet.

**HISTORY OF JAVA?**

-The history of java was very interesting Java was Originally designed for Interactive Television but it was too advanced technology for that the digital cable television industry at that time.

-The History of Java Starts with the **Green Team Members** also known as **Green Team** initiated the project to develop a language are digital device such as Set-Up Boxes, Television etc.

-The Principles of Creating Java Programming were “Simple, Robust, Portable, Platform Independent, Secured, High Performance, Multithreaded, Architecture, Neutral, Object-Oriented, Interpreted, Dynamic, Platform Independent.

-Java Programming Language was originally developed by **Sun-Microsystem** which has initiated by **James Gosling** and released in **1995** as Core Component of Sun Microsystem.

## Java Name History

### **GreenTalk**

James Gosling was leading a team named as 'Green' team. Target of this team was to create a new project which can. Initially [C++](https://www.tutorialspoint.com/cplusplus/index.htm) was the original choice to develop the project. James Gosling wanted to enhance C++ to achieve the target but due to high memory usage, that idea was rejected and team started with a new language initially named as GreenTalk. The file extension used as .gt. Later this language was termed as Oak and finally to Java.

### **Oak**

James Gosling renamed language to Oak. There was an Oak tree in front of his office. James Gosling used this name as Oak represents solidarity and Oak tree is the national tree of multiple countries like USA, France, Romania etc. But Oak technologies already had Oak as a trademark and James team had to brainstorm another title for the language.

### **Finally Java**

Team put multiple names like DNA, Silk, Ruby and Java. Java was finalized by the team. James Gosling tabled Java title based on type of espresso coffee bean. Java is an island in Indonesia where new coffee was discovered termed as Java coffee. As per James Gosling, Java was among the top choice along with Silk. Finally Java was selected as it was quite unique and represented the essence of being dynamic, revolutionary and fun to say.

Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

The latest release of the Java Standard Edition is Java SE 21. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

Learn Java in-depth with real-world projects through our [Java certification course](https://www.tutorialspoint.com/certification/java-spring-boot-advanced-certification/index.asp?utm_source=tutorialspoint&utm_medium=java_tutorial_3p&utm_campaign=internal). Enroll and become a certified expert to boost your career.

## Java Versions History

-Over the period of nearly 30 years, Java has seen many minor and major versions. Following is a brief explaination of versions of java till date.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | Version | Date | Description |
| 1 | JDK Beta | 1995 | Initial Draft version |
| 2 | JDK 1.0 | 23 Jan 1996 | A stable variant JDK 1.0.2 was termed as JDK 1 |
| 3 | JDK 1.1 | 19 Feb 1997 | Major features like [**JavaBeans**](https://www.tutorialspoint.com/jsp/jsp_java_beans.htm), [**RMI**](https://www.tutorialspoint.com/java_rmi/index.htm), [**JDBC**](https://www.tutorialspoint.com/jdbc/index.htm), [**inner classes**](https://www.tutorialspoint.com/java/java_innerclasses.htm) were added in this release. |
| 4 | JDK 1.2 | 8 Dec 1998 | [**Swing**](https://www.tutorialspoint.com/swing/index.htm), JIT Compiler, Java Modules, [**Collections**](https://www.tutorialspoint.com/java/java_collections.htm) were introduced to JAVA and this release was a great success. |
| 5 | JDK 1.3 | 8 May 2000 | HotSpot [**JVM**](https://www.tutorialspoint.com/java/java-jvm.htm), JNDI, JPDA, JavaSound and support for Synthetic proxy classes were added. |
| 6 | JDK 1.4 | 6 Feb 2002 | Image I/O API to create/read JPEG/PNG image were added. Integrated [**XML parser**](https://www.tutorialspoint.com/java_xml/java_xml_parsers.htm) and XSLT processor (JAXP) and Preferences API were other important updates. |
| 7 | JDK 1.5 or J2SE 5 | 30 Sep 2004 | Various new features were added to the language like foreach, var-args, generics etc. |
| 8 | JAVA SE 6 | 11 Dec 2006 | 1. notation was dropped to SE and upgrades done to JAXB 2.0, JSR 269 support and JDBC 4.0 support added. |
| 9 | JAVA SE 7 | 7 Jul 2011 | Support for dynamic languages added to JVM. Another enhancements included string in switch case, compressed 64 bit pointers etc. |
| 10 | JAVA SE 8 | 18 Mar 2014 | Support for functional programming added. Lambda expressions,streams, default methods, new date-time APIs introduced. |
| 11 | JAVA SE 9 | 21 Sep 2017 | Module system introduced which can be applied to JVM platform. |
| 12 | JAVA SE 10 | 20 Mar 2018 | Unicode language-tag extensions added. Root certificates, threadlocal handshakes, support for heap allocation on alternate memory devices etc were introduced. |
| 13 | JAVA SE 11 | 5 Sep 2018 | Dynamic class-file constants,Epsilon a no-op garbage collector, local-variable support in lambda parameters, Low-overhead heap profiling support added. |
| 14 | JAVA SE 12 | 19 Mar 2019 | Experimental Garbage Collector,Shenandoah: A Low-Pause-Time Garbage Collector, Microbenchmark Suite, JVM Constants API added. |
| 15 | JAVA SE 13 | 17 Sep 2019 | Feature added - Text Blocks (Multiline strings), Enhanced Thread-local handshakes. |
| 16 | JAVA SE 14 | 17 Mar 2020 | Feature added - Records, a new class type for modelling, Pattern Matching for instanceof, Intuitive NullPointerException handling. |
| 17 | JAVA SE 15 | 15 Sep 2020 | Feature added - Sealed Classes, Hidden Classes, Foreign Function and Memory API (Incubator). |
| 18 | JAVA SE 16 | 16 Mar 2021 | Feature added as preview - Records, Pattern Matching for switch, Unix Domain Socket Channel (Incubator) etc. |
| 19 | JAVA SE 17 | 14 Sep 2021 | Feature added as finalized - Sealed Classes, Pattern Matching for instanceof, Strong encapsulation of JDK internals by default. New macOS rendering pipeline etc. |
| 20 | JAVA SE 18 | 22 Mar 2022 | Feature added - UTF-8 by Default, Code Snippets in Java API Documentation, Vector API (Third incubator), Foreign Function, Memory API (Second Incubator) etc. |
| 21 | JAVA SE 19 | 20 Sep 2022 | Feature added - Record pattern, Vector API (Fourth incubator), Structured Concurrency (Incubator) etc. |
| 22 | JAVA SE 20 | 21 Mar 2023 | Feature added - Scoped Values (Incubator), Record Patterns (Second Preview), Pattern Matching for switch (Fourth Preview),Foreign Function & Memory API (Second Preview) etc. |
| 22 | JAVA SE 21 | 19 Sep 2023 | Feature added - String Templates (Preview), Sequenced Collections, Generational ZGC, Record Patterns, Pattern Matching for switch etc. |

-Following are the give significant point that describe the History of Java.

1]**James Gosling**, **Mike Sheridan** & **Patrick Naughton** initiated Project in June 1991 the small team of Sun Engineers called as Green Team.

2]Initially it was designed for small, embedded system in electronic appliance like set-up boxes.

3]Firstly it was called “**GreenTalk**” by James Gosling & the file extension was **.g**.

4]After that it was call “**oak**” and was developed as a part of the Green Project.

5]Why Java was named as “**oak**”?

-oak is a Symbol of Strength and chosen as a national tree of many countries like the USA, France, Germany, Romania etc.

6]In 1995 oak was renamed as “**Java**” because it was already a trademark by “**oak Technologies**”.

**Why Java Programming Named “Java”?**

-Java is island in Indonesia where the first coffee produced called Java coffe.

-it is kind of espresso bean Java name was chosen by James Gosling while having a cup of Coffee nearby his office.

-Initially developed by James Gosling at Sun Microsystem which is Now Subsidiary of oracle Corporation and released in 1995.

**What is Object in Java**

-An Object is Real world entity.

-An object is a Runtime entity.

-The object is an entity which has State and Behaviour.

-The object is an instance of a class.

**Characteristics of Object**

-

**Types of Java**

**There are four types of Java language based on their editions:**

1. Java Standard Edition (Java SE)
2. Java Enterprise Edition (Java EE)
3. Java Micro Edition (Java ME).
4. JavaFX

**Editions of Java**

Let’s discuss all the Java editions in detail:

**1. Java Standard Edition (Java SE)**

-So, what is Java Standard Edition?

-The Java Standard Edition (Java SE) comprises APIs (Application Programming Interfaces) that offer fundamental programming features.

-This edition of Java specifies each aspect of Java programming, starting from its fundamental types and objects to its high-level classes for security, database manipulation, Graphical User Interface development, networking, and more.

-Java SE is considered the core foundation of the Java programming language and provides a set of libraries, APIs, tools, and runtime environments for building and deploying various types of applications, ranging from desktop to server applications and even mobile applications.

**Java SE includes the following key components:**

* [**Java Development Kit (JDK)**](https://www.tutorialsfreak.com/java-tutorial/java-development-kit)**:** The JDK is a software package that includes tools such as the Java compiler (javac), the Java Virtual Machine (JVM), and various utilities for developing and running Java applications. It also includes libraries and documentation.
* [**Java Runtime Environment (JRE)**](https://www.tutorialsfreak.com/java-tutorial/java-runtime-environment)**:** The JRE is a subset of the JDK and contains only the runtime components necessary to run Java applications. It includes the JVM and core class libraries.
* **Core Libraries:** Java Standard Edition provides a rich set of core libraries that offer functionalities for data structures, input/output operations, networking, graphical user interfaces (GUIs), and more. These libraries help developers build robust and efficient applications.
* **Java APIs:** Java SE includes a vast collection of APIs that developers can use to access various functionalities. These APIs cover areas like networking, file handling, concurrency, security, and more.
* **Language Features:** Java SE introduces the Java programming language itself, with features such as object-oriented programming, [**platform independence**](https://www.tutorialsfreak.com/java-tutorial/java-platform-independent), and strong type checking.
* **Compatibility:** Java SE emphasizes backward compatibility, allowing older Java applications to run on newer versions of the platform without major modifications.

**2. Java Enterprise Edition (Java EE)**

-Now, coming to what is Java EE.

-Java EE stands for Enterprise Edition. It was developed with the aim of extending Java SE by adding a collection of standards/specifications that define frequently-used features by commercial applications.

-The latest version of this Java edition includes over 40 specifications that help programmers build applications that use web services to transform object-relational data into entity-relationship models, exchange data, manage transactional interactions, and more.

**Commonly-used features of Enterprise Edition Java are as follows:**

* **Java Persistence API (JPA):** It allows Java developers to access an object/relational mapping mechanism so they can manage relational data in Java applications.
* **Java Server Pages (JSP):**JSP is a server-side programming language used to create device-independent and dynamic ways to build web applications.
* **Enterprise Java Beans (EJB):** It is among the Java APIs that are used for the development of corporate applications. EJB is a server-side software component that describes the business logic of an application.
* **Java Server Faces (JSF):** JSF API provides elements, such as commandButtons, inputText, and more, and assists in managing their states. It also offers server-side validation, data translation, etc.

**3. Java Micro Edition (Java ME)**

The full form of Java ME is Micro Edition. It is also one of the platforms of Java that was developed to facilitate mobile and embedded device applications.

Although it is not as popular as other editions of Java like Java EE and Java SE, the [**IoT (Internet of Things)**](https://www.tutorialsfreak.com/internet-of-things-iot-tutorial) gadgets recently gave this technology new hope. Java ME provides an API and small-footprint virtual machine to help run Java ME applications on compact devices.

**Common examples of Java ME applications are as follows:**

* Calculator for mobile that demonstrates the functionality of multi-threading.
* Location APIs.
* GPIO (General Purpose Input Output) applications.

**4. JavaFX**

JavaFX platform provides a modern, hardware-accelerated graphics and media engine for developing rich desktop applications. We can use JavaFX Script, a simple yet powerful scripting language, to enable the development of rich online applications, desktop applications, and GUI applications. Java created it to replace Swing as the default GUI library.

**Below, we have mentioned the key applications of JavaFX.**

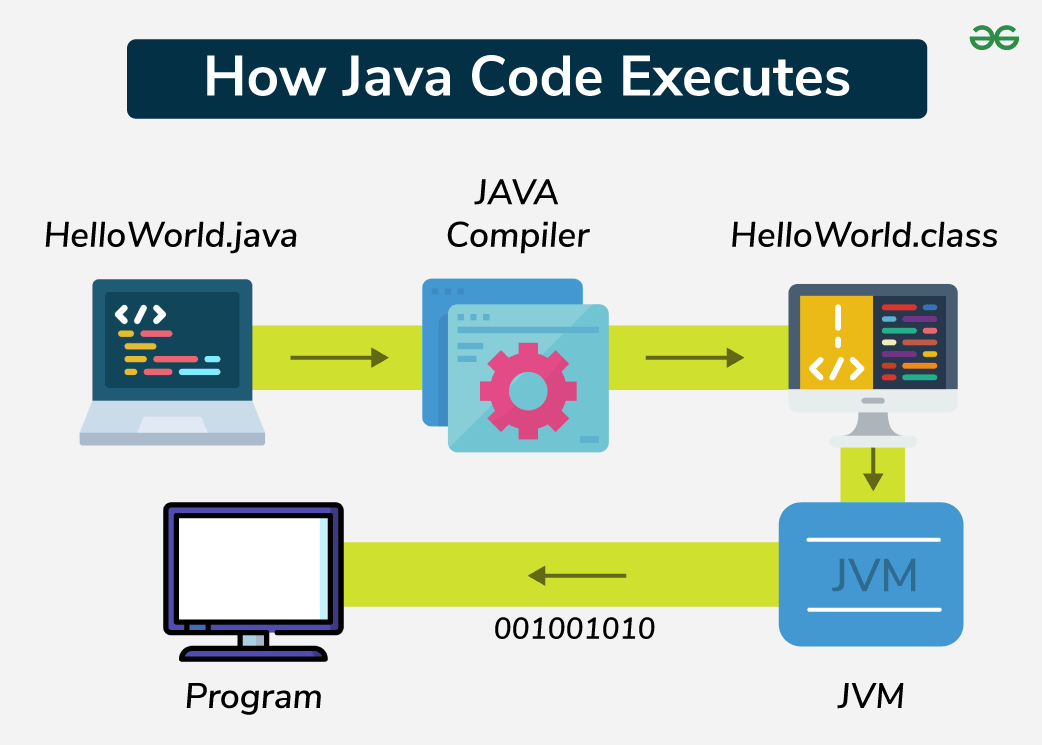
* **GEONS** Ground System Software (GGSS) Nasa is used in space technology.
* **NEOS** (New Eurovision Operations System) is used in television media.
* **FORUM** Carl Zeiss Meditec AG is used in the field of medicine.
* Quote Monitor application is used in the finance sector.

**Features Of java**



**How Java Code Executes**

The **execution** of a **Java application code** involves three main steps:



*How Java Code Executes*

**1. Creating the Program**

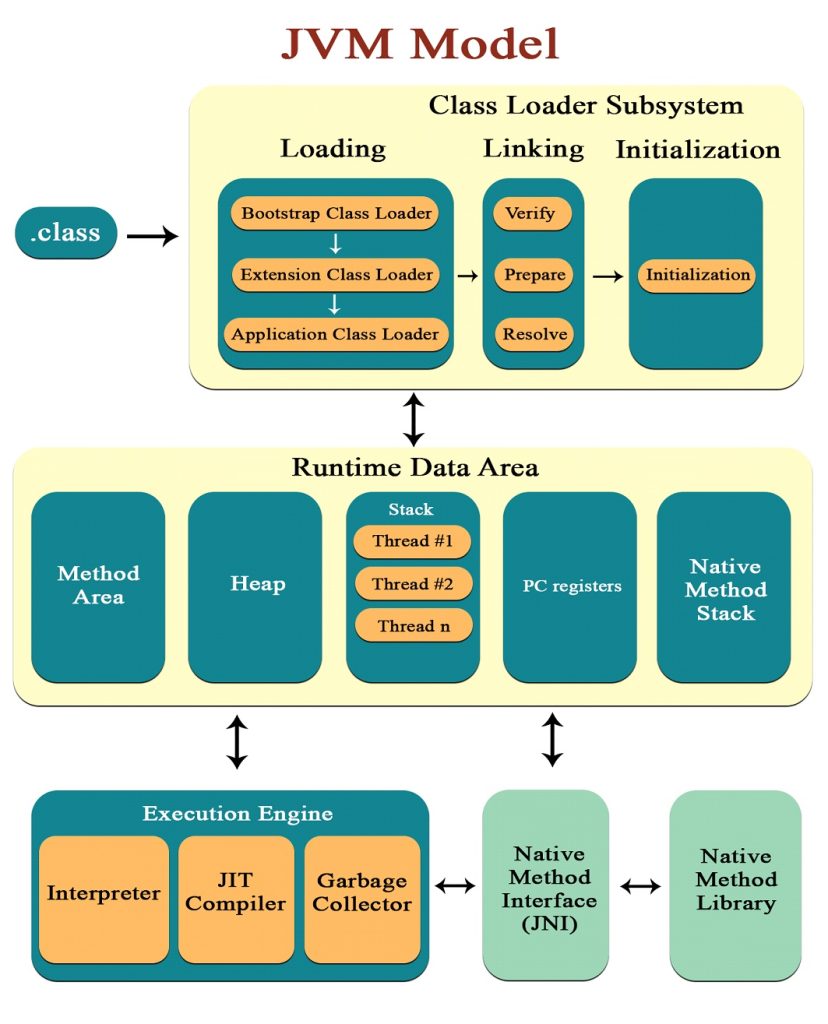
Java programs are written using a text editor or an [**Integrated Development Environment (IDE)**](https://www.geeksforgeeks.org/best-java-ide-for-developers/) like [**IntelliJ IDEA**](https://www.geeksforgeeks.org/step-by-step-guide-to-install-intellij-idea/), [**Eclipse**](https://www.geeksforgeeks.org/how-to-download-and-install-eclipse-on-windows/), or [**NetBeans**](https://www.geeksforgeeks.org/how-to-install-netbeans-java-ide-on-windows/). The source code is saved with a **.java extension**.

**2. Compiling the Program**

The [**Java compiler (javac)**](https://www.geeksforgeeks.org/what-are-the-roles-of-java-compiler-and-interpreter/) converts the source code into bytecode, which is stored in a .class file. This bytecode is platform-independent and can be executed on any machine with a **JVM**.

**3. Running the Program**

The **JVM executes the compiled bytecode**, translating it into machine code specific to the operating system and hardware.

****

**TOKENS**

-Tokens are the smallest elements of Java.

-Token are classified into: -

**1]KEYWORDS**

**2]IDENTIFIERS**

**3]SEPERATORS**

**4]LITERALS**

**1]KEYWODRS**

-Keywords are the words whose **meaning** actually known by the **Compiler**.

-There are **53** keywords in Java.

-The Keyword are the Predefined Words.

-Keywords are the Reserved words in Java.

-We can identify Keywords of Java in: -

1]They are Highlighted in **Blue** Colour.

2]They are written in terms of Lowercase.

**e.g.:- class, public static, void etc**

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

-Beside these keywords you cannot use **true** **false** & **null** as identifiers because it is **Literals**.

**2]IDENTIFIERS**

-Identifiers its name given for elements of Java for Identification purpose by Programmer.

-Elements of Identifiers in Java are: -

1]ClassName

2]VariableName

3]MethodName

4]InterfaceName

5]PackageName

**Class Naming Conventions**

**RULES: -**

1]Program Name should be start with letter **(A-Z) (a-z)**

Either its **Uppercase** or **Lowercase** depends on Company Standards.

**e.g.: -Program.java**

**Not Mandatory**

**program.java**

2]We cannot give space between Program Name.

3]The Underscore **(\_)** and Doller **($)** symbol are used between Program name.

4]The Program Name cannot start with Numeric Values.

**OPERATORS**

**1]Unary Operator**

**2]Binary Operator**

**3]Ternary Operator**

**1]UNARY OPERATOR**

-The operator which takes One Value At a time Is called as unary Operator.

1. **Increment Operator**
2. **Decrement Operator**
3. **Cast Operator**

**I.INCREMENT OPERATOR**

**1]Post-Increment Operator**

**2]Pre-Increment Operator**

**1]POST-INCREMENT OPERATOR**

**Steps to Perform Post-Increment Operator: -**

1]Use the value present inside the variable.

2]Later Increment the value which is present inside the variable by 1 & updated.

**Example No 1: -**

Class Program1

{

Public static void main (String args[])

{

int a=1;

int b=a++;

int c=b++;

System.out.println(a);

System.out.println(b);

System.out.println(c);

}

}

O/P: -

🡺2

🡺2

🡺1

**Example No 2: -**

Class Program2

{

Public static void main (String args[])

{

int a=1;

System.out.println(a);

System.out.println(a++);

System.out.println(a);

}

}

**O/P: -**

🡺1

🡺2

🡺2

**2]PRE-INCREMENT OPERATOR**

**Steps to Perform Per-Increment Operator: -**

1]Increment the value which is present inside the variable by 1 & updated.

2]Use the updated the value which is present inside the variable.

**Example No 1: -**

Class Program1

{

Public static void main (String args[])

{

int a=1;

int b=++a;

int c=++b;

System.out.println(a);

System.out.println(b);

System.out.println(c);

}

}

**O/P: -**

🡺2

🡺3

🡺3

**Example No 2: -**

Class Program2

{

Public static void main (String args[])

{

int a=1;

int b=2;

int c=++a + ++b + ++a + ++b;

System.out.println(a);

System.out.println(b);

System.out.println(c);

}

}

**O/P: -**

🡺3

🡺4

🡺12

**Example No 3: -**

Class Program3

{

Public static void main (String args[])

{

int a=2;

int b=2;

int c=a++ + ++a + ++b + b++ + ++a;

System.out.println(a);

System.out.println(b);

System.out.println(c);

}

}

**O/P: -**

🡺5

🡺4

🡺17

**Write the Difference Between Post-Increment & Pre-Increment Operator?**

|  |  |  |
| --- | --- | --- |
| Sr No | Pre-Increment Operator | Post-Increment Operator |
| 1 | The Increment Operator Which is Prefixed to its Variable name is called as Pre-Increment Operator | The Increment Operator Which is Suffixed to its Variable name is called as Post-Increment Operator |
| 2 | **STEP-1: -**Increment the value present inside the variable by **1** & **updated.** | **STEP-1: -**Use the updated value which is present inside the variable. |
| 3 | **STEP-2: -**Use the updated value which is present inside the variable. | **STEP-2: -**Increment the value present inside the variable by **1** & **updated.** |
| 4 | **Ex: -++a;** | **Ex: -a++;** |

**Decrement Operator**

The Decrement operator is a Unary Operator

It is used to decrement the value present inside the variable by 1 & updated.

Decrement Operator is denoted by ­­--.

Decrement Operator IS Classified into 2 types: -

**1]Post-Decrement Operator**

**2]Pre-Decrement Operator**

**Write a Difference Between Post-Decrement & Pre-Decrement Operator?**

**Ans: -**

|  |  |  |
| --- | --- | --- |
| Sr No | Pre-Decrement Operator | Post-Decrement Operator |
| 1 | The Decrement Operator Which is Prefixed to its Variable name is called as Pre-Decrement Operator | The Decrement Operator Which is Suffixed to its Variable name is called as Pre-Decrement Operator |
| 2 | **STEP-1: -** Decrement the value present inside the variable by **1** & **updated.** | **STEP-1: -**Use the updated value which is present inside the variable. |
| 3 | **STEP-2: -**Use the updated value which is present inside the variable. | **STEP-2: -** Decrement the value present inside the variable by **1** & **updated.** |
| 4 | **Ex: - --a;** | **Ex: -a--;** |

**Que: - Can we perform the increment or decrement for the char, int, double, string, Boolean datatype?**

**Ans: -** We can perform Increment & Decrement only char, long, double datatype.

We can’t perform the Increment & Decrement for the Boolean & String Datatype.

**III.CAST OPERATOR**

QNO1: -Convert Double to Float

class DoubleToFloat

{

public static void main (String [] args)

{

double a=203;

float b=(float)a;

System.out.println(b);

}

}

**O/P: -**

🡺 203.0

QNO2: -Convert Double to Char

class DoubleToChar

{

public static void main (String [] args)

{

double a=97;

char b=(char)a;

System.out.println(b);

}

}

**O/P: -**

🡺 a

QNO3: -Convert Float to Int

class FloatToInt

{

public static void main (String[] args)

{

float a=20.6f;

int b=(int)a;

System.out.println(b);

}

}

**O/P: -**

🡺 20

QNo4: -Convert Float to Char

class FloatToChar

{

public static void main (String [] args)

{

float a=97.35f;

char b=(char)a;

System.out.println(b);

}

}

**O/P: -**

**🡺** a

QNO5: -Convert long to char

class LongToChar

{

public static void main (String [] args)

{

long a=111;

char b=(char)a;

System.out.println(b);

}

}

**O/P: -**

🡺 o

QNO6: -Convert Float to Long

class FloatToLong

{

public static void main (String [] args)

{

float a=20.5f;

long b=(long)a;

System.out.println(b);

}

}

**O/P: -**

🡺 20

QNO7: -Convert Double to Long

class DoubleToLong

{

public static void main(String[] args)

{

double a=202435.55;

long b=(long)a;

System.out.println(b);

}

}

**O/P: -**

🡺202435

**2]BINARY OPERATOR**

1. **Relational Operator**
2. **Arithmetic Operator**
3. **Logical Operator**

**I. Relational Operator**

-Relational operator it is an Binary Operator it takes two values at a time.

-The output type of relational operator is a Boolean Condition. [True/False]

- Relational operator is classified into:-

Class RealtionalOperators

{

Public static void main (String args [])

{

int a=3;

int b=4;

System.out.println(a>b);

System.out.println(a<b);

System.out.println(a>=b);

System.out.println(a<=b);

System.out.println(a==b);

System.out.println(a!=b);

}

}

**O/P: -**

🡺 false

🡺 true

🡺 false

🡺 true

🡺 false

🡺 true

**II. Arithmetic Operators**

-Arithmetic operator it is a binary operator it takes two values at a time.

-Arithmetic operator is used to performs the Arithmetic operations

-Arithmetic operator is classified into: -

Class ArithematicOperator

{

Public static void main (String args[])

{

Int a=4;

Int b=2;

System.out.println(a+b);

System.out.println(a-b);

System.out.println(a\*b);

System.out.println(a/b);

System.out.println(a%b);

}

}

**O/P: -**

🡺 6

🡺 2

🡺 8

🡺 2

🡺 0

**III.Logical Operators**

-Logical operator it is an Binary Operator Which Takes two value at a time

-Logical Operator is classified into: -

1. **AND &&**
2. **OR ||**
3. **NOT !**

**I.AND OPERATOR: -**

Both Condition Are Satisfied Then It will Print **True**.

**Truth Table: -**

|  |  |  |
| --- | --- | --- |
| Input -1 | Input-2 | Output |
| True | True | **True** |
| True | False | **False** |
| False | True | **False** |
| False | False | **False** |

**II.OR OPERATOR: -**

Anyone Condition Are Satisfied then it will print **True.**

**Truth Table: -**

|  |  |  |
| --- | --- | --- |
| Input -1 | Input-2 | Output |
| True | True | **True** |
| True | False | **True** |
| False | True | **True** |
| False | False | **False** |

**III.NOT OPERATOR: -**

Not Operator It is a Unary Operator Which takes only One Input at a time. Not Operator belongs to Logical Operator

**Truth Table: -**

|  |  |
| --- | --- |
| Input | Output |
| True | **False** |
| False | **True** |

Class LogicalOperator

{

Public static void main (String args [])

{ **//AND &&**

System.out.println(true && true);

System.out.println(true && false);

System.out.println(false && true);

System.out.println(False && False);

**//OR ||**

System.out.println(true || true);

System.out.println(true || false);

System.out.println(false || true);

System.out.println(false || false);

**//NOT !**

System.out.println(true || true);

}

}

**2]TERNARY OPERATOR**

1. **CONDITIONAL OPERATOR**

Conditional Operator it is a Ternary Operator it takes three values at a time.

Syntax: -

**CONDITION ? TASK 1 : TASK 2;**

**TRUE**

**False**

1]If the condition is true the control will transfer to **TASK1**.

2]If the condition is false the control will transfer to **TASK2**.

QNO1: -Write a Program to find greater between two numbers**?**

Class Program1

{

Public static void main (String args [])

{

int a=4;

int b=2;

Int res = a>b ? a : b;//Storing the result

System.out.println(res);

}

}

**O/P: -**

**🡺**4

QNO2: -Write a Program to check the given number is even or odd using Conditional Operator**?**

Class Program2

{

Public static void (String args [])

{

int a=4;

String res = a%2 ==0?” Even”: “Odd”;

System.out.println(res);

}

}

**O/P: -**

**🡺**4

QNO3: -Write a Program to find Minimum of Two Numbers Using Conditional Operators?

class FindMinimunNum

{

public static void main(String args[])

{

int a=15;

int b=7;

int res = b<a ? b : a;

System.out.println(res);

}

}

**O/P: -**

**🡺**7

QNO4: -Write a Program to check the given number is Positive or Negative?

class PosNegNum

{

public static void main(String[] args)

{

int a=5;

System.out.println(a>0?"POSITIVE":"NEGATIVE");

}

}

**O/P:-**

**🡺POSITIVE**

QNO5: -Write a Program to find the Maximum of Three Numbers Using Conditional Operator?

class FindMaximumThreeNum

{

public static void main(String[] args)

{

int a=3;

int b=5;

int c=11;

String res=a>b && a>c?"A Is Greater":b>a && b>c?"B Is Greater":"C Is Greater";

System.out.println(res);

}

}

**O/P:-**

**🡺C Is Greater**

QNO6: -Write a Program to find the Minimum of Four Numbers Using Conditional Operator?

class MinimumFourNum

{

public static void main(String[] args)

{

int a=24;

int b=11;

int c=5;

int d=3;

String res = a<b && a<c && a<d ? "A Is Smaller ": b<a && b<c && b<d ? "B Is Smaller": c<a && c<b && c<d ? "C Is Smaller" : "D Is Smaller";

System.out.println(res);

}

}

**O/P:-**

**🡺 D Is Smaller**

**EXERCISE**

QNO1.Write a Program to perform addition of three numbers?

Class AddThreeNum

{

Public static void main(String args[])

{

int a=1;

int b=2;

int c=3;

int res = a + b + c;

System.out.println(res);

}

}

**O/P: -**

**🡺 6**

QNO2: -Write a Program to find the Square of a given number?

class FindSquare

{

public static void main (String[] args)

{

int num=2;

int res=num\*num;

System.out.println("The Square Of:"+num+":-Is:"+res);

}

}

**O/P:-**

**🡺 The Square Of:2: -Is:4**

QNO3: -Write a Program to find the Cube of a given number?

class FindCube

{

public static void main (String[] args)

{

int num=3;

int res=num\*num\*num;

System.out.println("The Cube Of:"+num+”: -Is:"+res);

}

}

**O/P: -**

**🡺 The Cube Of:4: -Is:64**

QNO4.Write a Program to calculate the Simple Interest?

class SimpleInterest

{

public static void main(String[] args)

{

float P=10000, R=5, T=5;

float SI=(P\*R\*T)/100;

System.out.println("Simple Interest = " + SI);

}

}

**O/P:-**

**🡺 2500.0**

QNO5.Write a Program to Calculate area of Circle?

class AreaCircle

{

public static void main (String[] args)

{

int radius=5;

double pi=3.142, area;

System.out.println("Radius:- " +radius);//Concat The String + Integer Value

area = pi \* radius \* radius;//Formula For Calculating The Area Of The Circle

System.out.println("Area Of Circle Is:- " + area);

}

}

**O/P: -**

**🡺78.55**

QNO6.Write a Program to Calculate area of Square

class AreaSquare

{

public static void main(String[] args)

{

double side=5;

double area=side\*side;

System.out.println("Area Of Square Is:- " + area);

}

}

**O/P: -**

**🡺The Area of Square Is: -25.0**

QNO7.Write a Program to Calculate area of Rectangle?

class AreaRectangle

{

public static void main(String[] args)

{

int width=5;

int height=10;

int area=width\*height;//Formula to Calculate the Area of Rectangle

System.out.println("Area Of Rectangle :- " + area);

}

}

**O/P:-**

**🡺50**

**DECISION MAKING STATEMENTS**

🡺Decision Making Statement it is used to check the condition before performing a particular task.

**Decision Making Statement is Classified into 5 Types: -**

**1]If Statement**

**2]If-Else Statement**

**3]Else-If Ladder**

**4]Nested-If**

**5]Switch**

**1]If Statement**

Syntax: -

**If (Condition)**

**TRUE**

**Statement-1;**

**False**

**Statement-2;**

Class IfStatement

{

Public static void main (String args[])

{

String Shop==” Open”

if (Shop==” Close”)

System.out.println(“Purchase the Sugar Packet 1 Kg”);

System.out.println(“Go Back to The Home”);

}

}

**O/P: -**

**🡺**Go Back to The Home

Class p2

{

Public static void main(String args[])

{

String==”Open”;

If (Shop==”Open”)

{ //if block

System.out.println(“Purchase”);

System.out.println(“1 Kg Sugar”);

System.out.println(“1 Kg Salt”);

System.out.println(“Pond Powder”);

}

System.out.println(“Go Back to The Home”);

}

}

**O/P: -**

**🡺 Purchase**

**🡺 1 Kg Sugar**

**🡺 1 Kg Salt**

**🡺 Pond Powder**

**🡺Go Back to The Home**

**2]If -Else Statement**

Syntax: -

**if (Condition)**

**TRUE**

**{**//if block

Statement-1;

**False**

Statement-2;

**}**

**else**

**{** //else block

Statements;

**}**

🡺If Block get executed when the condition is True.

🡺For false condition we can perform Task with the help of Else Block.

🡺Else block is not mandatory.

Class p3

{

Public static void main (String args [])

{

String==” Open”;

If (Shop==” Close”)

{ //if block

System.out.println(“Purchase”);

System.out.println(“1 Kg Sugar”);

System.out.println(“1 Kg Salt”);

System.out.println(“Pond Powder”);

}

else

{

System.out.println(“Borrow from The Neighbor”);

}

System.out.println(“Go Back to The Home”);

}

}

**O/P: -**

**🡺 Borrow from The Neighbor**

**🡺Go Back to The Home**

**EXERCISE**

QNO1: -Write a program to check the given character is Uppercase Using If Else Statement?

class CheckUppercase

{

public static void main (String [] args)

{

char ch='0';

if (ch>='A' && ch<='Z')

{

System.out.println(ch+”: Is A Uppercase Character");

}

else

{

System.out.println(ch+”: Is Not Uppercase Character");

}

}

}

**O/P: -**

**🡺 A: Is an Uppercase Character**

QNO2:-Write A Program to Check the given number Positive or Negative Using If Else Statement?

class CheckPosNeg

{

public static void main (String [] args)

{

int num=3;

if (num>0)

{

System.out.println(num+”: The Given Number Is Positive");

}

else

{

System.out.println(num+”: The Given Number Is Negative");

}

}

}

**O/P: -**

**🡺 3: The Given Number Is Positive**

QNO3: - Write A Program to check whether the given number is Even or Odd using If Else Statement?

class EvenOdd

{

public static void main (String[] args)

{

int num=4;

if (num%2==0)

{

System.out.println(num+" : Given Number Is Even");

}

else

{

System.out.println(num+" : Given Number Is Odd");

}

}

}

**O/P: -**

**🡺 4: Given Number Is Even**

QNO4:-Write A Program to find the maximum of Two Numbers using If Else Statement?

class MaxTwoNum

{

public static void main(String[] args)

{

int a=111;

int b=535;

if (a>b)

{

System.out.println(a+":A Is Greater");

}

else

{

System.out.println(b+" : B Is Greater");

}

}

}

**O/P: -**

**🡺 535: B Is Greater**

QNO5: -Write A Program to check the given character is Vowels or Not?

class CheckVowels

{

public static void main(String[] args)

{

char ch='a';

if (ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')

{

System.out.println(ch+”: Given Charcater Is Vowel");

}

else

{

System.out.println(ch+”: Given Character Is Not Vowel");

}

}

}

**Ans: -**

**O/P: -**

**🡺 a: Given Character Is Vowel**

Syntax: -

**if (Condition)**

**{** //if block

Statement-1;

Statement-2;

**}**

**else if (Condition)**

**{** //else-if block

Statements;

**}**

**else if (Condition)**

**{** //else-if block

Statements;

**}**

**else**

**{** //else block

Statements;

**}**

**3]Else-If Ladder**

🡺Verifying more than one condition Is Called as Else-If Ladder.

QNO1: -Write a Program to check whether the given number is

**Conditions: -**

1. If the Number Is Divisible by 3 and 7 print “Hi”.
2. If the Number Is Divisible by 2 or 7 print “Bye”.
3. If the Number Is Even and not an Odd Number print “Goodbye”.
4. If the none of the condition are satisfied print “Block”.

class p1

{

Public static void main (String args[])

{

int n=4;

if(n%3==0 && n%7==0)

{

System.out.println(“Hi”);

}

else if(n%2==0 && n%7==0)

{

System.out.println(“Bye”);

}

else if(n%2==0)

{

System.out.println(“Goodbye”);

}

else

{

System.out.println(“Block”);

}

}

}

**ASSIGNMENT QUESTIONS**

QNO1: -Write a program the maximum of Four numbers using else-if ladder?

class MaxFourNum

{

public static void main(String[] args)

{

int a=444;

int b=291;

int c=111;

int d=15;

if (a>b && a>c && a>d)

{

System.out.println(a+" : A is Greater");

}

else if (b>c && b>d)

{

System.out.println(b+" : B is Greater");

}

else if (c>d)

{

System.out.println(c+" : C Is Greater");

}

else

{

System.out.println(d+" : D is Greater");

}

}

}

QNO2: -Write a program to check whether the given character is Uppercase, Lowercase or Special Character?

class CheckCharacter

{

public static void main(String[] args)

{

char ch='#';

if (ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')

{

System.out.println(ch+" : It is an Vowel");

}

else if (ch>='A' && ch<='Z')

{

System.out.println(ch + ": It Is An Uppercase Letter");

}

else if (ch>='a' && ch<='z')

{

System.out.println(ch + " : It Is an Lowercase Letter");

}

else if (ch>='0' && ch<='9')

{

System.out.println(ch + " : It Is an Didit");

}

else

{

System.out.println(ch + ": It Is An Special Character");

}

}

QNo4: -Write a program to calculate marks of Four Subjects and Display the Results

**Condition: -**

1. If any subject marks are less than 35% print “**Fail**”.
2. If the Percentage is Greater than 80% print “**Distinction”.**
3. If the Percentage is Greater than 60% print **“First Class”.**

class CalMarks

{

public static void main(String[] args)

{

int Sub1=74, Sub2=77, Sub3=83, Sub4=77;

int Total=Sub1+Sub2+Sub3+Sub4;

double Per=((Total\*100)/400);

if (Per<35)

{

System.out.println(Per+" : Fail");

}

else if (Per>75)

{

System.out.println(Per+" : Distinction");

}

else if (Per>60)

{

System.out.println(Per+" : First Class");

}

else if (Per>50)

{

System.out.println(Per+" : Second Class");

}

else if (Per>35)

{

System.out.println(Per+" : Pass");

}

}

}

1. If the Percentage is Greater than 50% print **“Second Class”.**
2. If the Percentage is Greater than 35% print **“Pass”.**

**4]Nested IF**

-And If condition inside another If is called as Nested-If Condition

**Syntax: -**

**if (Condition)**

**TRUE**

**{** //Outer-if block

**if (Condition)**

**{** //Inner-if block

Statement-1;

**False**

**False**

Statement-2;

**}**

**else**

**{** //Inner-else block

Statements;

**}**

**}**

**else**

**{** //else block

Statements;

**}**

**ASSIGNMENT QUESTIONS**

QNO1: - Write a program to check the given number is Even or Odd if the number is Greater than 35.

class NumEvenOddNumG35

{

public static void main(String[] args)

{

int num=24;

if (num>35)

{

if (num%2==0)

{

System.out.println(num+ " : This Is Even Number");

}

else

{

System.out.println(num+ " : This Is Odd Number");

}

}

else

{

System.out.println(num+ " : Number Is Less Than 35");

}

}

}

QNO2: -Write a program to check the given character is Vowel or Not if the character is Uppercase?

class CharVowel

{

public static void main(String[] args)

{

char ch='a';

if (ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')

{

if (ch>='A' && ch<='Z')

{

System.out.println(ch +" : The Given Character Is Uppercase");

}

else

{

System.out.println(ch + " : The Given Character Is Lowercase");

}

}

else

{

System.out.println(ch + " : The Character Is Special Symbol");

}

}

}

**5]Switch**

-It is a Special Decision-Making Statement.

-Inside a switch we cannot passes condition.

-Inside Switch we can Pass **Expression** or **Literals**.

-In switch we can compare **Case Value** with **Switch Value.**

Syntax 1: -

**Switch(expression/literals)**

**{**

Case value: action;

Case value: action;

Case value: action;

default: action;

**}**

**Default Keyword: -**

-Default is a keyword in java.

-The Nature of default keyword is working like else keyword.

-Suppose All cases are not satisfied then the default case is executed.

**Break Keyword: -**

-Break is a keyword in java.

- For stop execution of switch we need to use Break Keyword neither it will be execute the all the cases.

Syntax 2: -

**Switch(expression/literals)**

**{**

**Case value : {**

**action;**

**}**

**Case value : {**

**action;**

**}**

**Case value : {**

**action;**

**}**

**default : {**

**action;**

**}**

**}**

**LOOPING STATEMENT**

It is used to verify the condition multiple times.

-Looping Statement it is classified into 3 types: -

**1]While Loop**

**2]Do-While Loop**

**3]For Loop**

**1]While Loop**

Syntax: -

**While(condition)**

**{**

**False**

**True**

Statements;

Update;

**}**

QNO1: -Write a program to print the first five number?

Class Print5Num

{

Public static void main(String args[])

{

int n=0;

while(n<5)

{

System.out.println(n);

n++;

}

}

}

QNO2: -Write a program to print first five characters?

class Print5Char

{

public static void main(String[] args)

{

char ch='A';

while (ch<='E')

{

System.out.println(ch);

ch++;

}

}

}

QNO3: -Write a program to print All the given numbers between 0 to 10?

class Print10Num

{

public static void main(String[] args)

{

int num=1;

while (num<=10)

{

System.out.println(num);

num++;

}

}

}

QNO4: -Write a program to print First 10 numbers in Reverse Order?

class Print10NumRev

{

public static void main(String[] args)

{

int num=10;

while (num<=1)

{

System.out.println(num);

num++;

}

}

}

QNO5: -Write a program to print All the numbers which are divisible by 3 & 7 between o to 100?

class PrintNumDiv3A7Bet100

{

public static void main(String[] args)

{

int num=1;

while (num<=100)

{

if (num%3==0 && num%7==0)

{

System.out.println(num);

}

num++;

}

}

}

QNO6: - Write a program to print first 10 Odd Numbers in Reverse Orders?

class Print10OddNumRev

{

public static void main(String[] args)

{

int num=20;

while (num<=20)

{

if (num%2==1)

{

System.out.println(num);

}

num--;

}

}

}

QNO7: - Write a program to perform Addition of first 10 Numbers using looping Statements?

class Addition10Num

{

public static void main(String[] args)

{

int num=0;

int Sum=0;

while (num<=10)

{

Sum=Sum+num;

num++;

}

System.out.println("Addition Of 10 Num:-"+Sum);

}

}

**2]Do-While Loop**

Syntax: -

**do**

**{**

Statements;

Update;

**} While(condition);**

Class Program1

{

Public static void main(String args[]

{

int num=1;

do

{

Sop(“$”);

a++;

}

While(a<=5);

}

}

**NOTE: - We can perform task for false condition only once using Do-While Loop.**

Class Program2

{

Public static void main(String args[])

{

Int num=0;

Do

{

Sop(“$”);

++a;

}

}

}

**3]For Loop**

Syntax: -

**for (Initialization ; Condition ; Update)**

**{**

**True**

Task;

**}**

**Nested-For Loop**

Syntax: -

**for (Initialization ; Condition ; Update)**

**{**

**for (Initialization ; Condition ; Update)**

**{**

Task;

**}**

**}**

-A For Loop inside another for loop is called as Nested for Loop.

**PATTERN PROGRAM LOGIC**

**COL**

C=3

C=2

C=1

C=0

**ROW**

C=4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00  \* | 01  \* | 02  \* | 03  \* | 04  \* |
| 10  \* | 11  **\*** | 12  **\*** | 13  **\*** | 14  **\*** |
| 20  \* | 21  **\*** | 22  **\*** | 23  **\*** | 25  **\*** |
| 30  \* | 31  **\*** | 32  **\*** | 33  **\*** | 34  **\*** |
| 40  \* | 41  **\*** | 42  **\*** | 43  **\*** | 44  **\*** |

R=0

R=1

R=2

R=3

R=3

Class Pattern1

{

Public static void main(String args[])

{

for(int r=0;r<5;r++)

{

for(int c=0;c<5;c++)

{

System.out.print(“\*”);

}

System.out.println();

}

}

**O/P:-**

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

1.

1 2 3 4 5

6 7 8 9 1

2 3 4 5 6

7 8 9 1 2

3 4 5 6 7

2.

0 1 0 1 0

0 1 0 1 0

0 1 0 1 0

0 1 0 1 0

0 1 0 1 0

3.

| \* | \* |

\* | \* | \*

| \* | \* |

\* | \* | \*  
| \* | \* |

4.

A A A A A

B B B B B

C C C C C

D D D D D

E E E E E

5.

A B C D E

A B C D E

A B C D E

A B C D E

A B C D E

6.

1 2 2 2 2

2 1 2 2 2

2 2 1 2 2

2 2 2 1 2

2 2 2 2 1

7.

\* 1 \* 2 \*

3 \* 4 \* 5

\* 6 \* 7 \*

8 \* 9 \* 1

\* 1 \* 2 \*

**CONTROL TRANSFER STATEMENT**

Control transfer statement it is used to stop the execution and transfer the control out of the Block.

**Control transfer statement it is classified into two types: -**

**1]Break**

**2]Continue**

**3]Return**

**1]Break**

-Break Is a Keyword in Java.

-Break it is a Control Transfer Statement.

-Break keyword it is used to **Stop** the **Execution** & transfer the control to **End** of the **Block**.

-Break keyword it is always declared at the end of the Block.

**Syntax:-**

{

break;

}

-We cannot use break keyword in between the Block.

-We get An Compile time Error called **Unreachable Statement**.

**EX:-**

**{**

**break**

**n++;**

**}**

**#Program**

Class BreakDemo

{

Public static void main(String args[])

{

System.out.println(“Main Begins”);

Int n=0;

While(n<5)

{

System.out.println(n);

If(n==2)

{

System.out.println(“if Block);

N++;

break;

}

**Transfer The Control to The End of The Loop**

System.out.println(n);

N++;

}

System.out.println(“Main End”);

}

}

**2]Continue**

-Continue is a Keyword in Java.

-It is a Control Transfer statement.

-Continue Keyword it is used to Transfer the control to Beginning of the Loop Condition.

-Continue always has to be declared in the End of the Block.

Syntax:-

{

N++;

continue;

}

-We cannot use the Continue Keyword in Between The Blocks. We Get compile Time Error Called **Unreachable Statement.**

**Ex:-**

{

Continue;

N++;

}

**#Program**

Class ContinueDemo

{

Public static void main(String args[])

{

System.out.println(“Main Begins”);

Int n=0;

While(n<5)

{

System.out.println(n);

**Transfer The Control to The Beginning.**

If(n==2)

{

System.out.println(“if Block);

N++;

continue;

}

System.out.println(n);

N++;

}

System.out.println(“Main End”);

}

}

QNO1.Can We use Break & Continue together inside the single Block.

**Ans:-** No, We get Compile Time Error Unreachable Statement

Class BreakContinueDemo

{

If(n==2)

{

System.out.println(“if Block”);

N++;

Continue;//CTE

Break;

}

}

QNO2.Can We declare Break & Continue keyword together inside a Single Program?

**Ans: -**Yes, we declare together inside single program but both break & continue keyword should be in different blocks.

class BreakContinueDifferentBlock

{

public static void main(String[] args)

{

System.out.println("Main Start");//1

int a=0;

while (a<5)//\*{0}

{

System.out.println("Hi");//2

if(a==2)/\*{false}\*/

{

System.out.println("GoodBye");

a++;

continue;

}

System.out.println("Bye");//3

a++;

break;

}

System.out.println("Main End");//4

}

}

**METHODS**

- Methods it is a set of instructions to performs a particular task.

-We can Create methods inside Class Block.

-To create a method, we need to take the help of **Method Header.**

**-**A Method Consist of two elements Method Header and Method Body.

**Method Naming Conventions**

**Rules: -**

**1.If a method consists a single word the entire word has to be written in terms of Lowercase.**

**EX: -**

demo ();

test ();

add (int a, int b, int c);

**2.If a method name consists of more than one word the first Letter of second word has to be written in terms of Uppercase[Camel Case].**

**EX: -**

demo ();

test ();

add (int a, int b, int c);

**METHOD HEADER**

-A method header gives the additional information about the method.

**Syntax: -**

**AccessModifier ReturnType MethodName (formal Arguments)**

**{** //method body

**}**

**EX: -**

Public static void Amol (int a, int b)

-A method header consist of following things.

**1.Access Modifier**

**2.ReturnType**

**3.MethodName**

**4.Formal Arguments**

**Formal Arguments**

-The arguments which holds the actual value is called as Formal arguments.

**Actual Arguments**

-It is the actual value Pass by the Programmer.

**METHOD SIGNATURE**

**Syntax: -** //Method Signature

**AccessModifier ReturnType MethodName (formal Arguments)**

**{** //method body

**}**

-Method Signature Consist of: -

**1.Method Name**

**2.Formal Arguments**

**METHOD CALL STATEMENT**

-It is a process of transferring the Control from Calling Method to Called Method.

-For Execution of Method Calling is mandatory.

-We can call method using Method Call Statement.

**CALLING METHOD**

-A method which tries to call another method is called as Calling Method.

**CALLED METHOD**

-The Method Which Is under Execution Is Called as Called Method.

-We can identify a method with the help of Method Signature.

Class MethodDemo

{

Public static void demo()

{

System.out.println(“Demo P6”);

}

Public static void main (String args[])

{

demo();//MCS

}

}

**Called Method**

**demo()**

**{**

**SOP(“demo()”);**

**}**

**Main()**

**{**

**demo();//MCS**

**}**

**Calling Method**

**METHODS TYPES**

-We can create two types methods

**1.No Argument Method**

-The No Argument mean it does not accept any argument that why is called as No Argument Method.

Class NoArgsMethod

{

Public static void demo()

{

System.out.println(“Demo P6”);

}

Public static void main (String args[])

{

demo();

}

}

**2.Parameterized Method**

-The method which accept any argument is called as Parameterized Method.

Class ParameterizedDemo

{

Public static void demo(String name)

{

System.out.println(“Hi This is ”+name);

}

Public static void main (String args[])

{

Demo(“Amol”);

}

}

**ASSIGNMENTS QUESTIONS**

QN01.Design a method to check the given number is Even or odd?

QN02.Design a method to check whether given number is Positive or Negative

QN03.Design a method to check the given character is Vowel or not

QNO4.Design a method to print the all the Even Number Between 0 to 20.

QNO5.Design a method to perform Addition of three Numbers

QNO6.Design a method to find Maximum of Three Numbers

QNO7.Design a Method to Print First 10 Characters?

QNO8.Design a Method to find Summation of first 10 Numbers?

QNO9.Design a method to find the Summation of all the Numbers which are Divisible by 3 & 5 Between 0 to 100?

QNO10.Design a method to count all the numbers which are divisible by 3 & 7

Between 0 to 200?

QNO11.Design a method to find the factorial of 7.

QNO12.Design a method to find the Cube of given Number.

QNO13.Design a method to find Simple Interest

QNO14.Design a method to print first 10 Odd numbers in reverse order.

QNO15.Design a to find the Area of the Circle.

QNO16.design A method to find Area of the Square.

QNO17.Design A method to find the area of the rectangle.

QNOO18.Design A Method to find the Minimum of four numbers.

QNO19.design A method to check whether the given character is uppercase or Lowercase.

**RETURNTYPE**

-It is a process of transferring the **Control** from **Called** **Method** back to its **Calling** **Method** is called As **ReturnType.**

Class ReturnTypeDemo

{

Public static int omkar(int money)

{

System.out.println(money);

Int profit=5\*money;

System.out.println(profit);

return 200;

}

Public static void main(String args[])

{

Int var=omkar(100);

System.out.println(Var);

}

}

**Called Method**

**omkar(int money)**

**{**

**SOP(money);**

**Int profit=5\*money;**

**Sop(profit);**

**Return 200;**

**}**

**}**

**Control**

**+**

**Values**

**Control**

**+**

**Values**

**Calling Method**

**Main()**

**ReturnType**

**{**

**Int var=omakr(100);//MCS**

**}**

-We can return two types of values

**1]Primitive Values**

**-Byte, Short, Char, int, Long, Float, Double, Boolean**

**2]Non-Primitive Values: -**

**-String, Class, Array.**

**VOID**

-Void it is Keyword in java.

-If the method ReturnType is void it does not return any value back to Programmer. (Calling Method).

-Void Mean Nothing.

**NOTE 1: -**

1]We can call a method from a variable such that we can Store Return value and use the Return Value More than once.

2]In this case method get executed at only once at a time.

class MethodCallingVar

{

public static String fruit(String name)

{

String res=name;

String var=res +" juice";

return var;

}

public static void main(String args[])

{

String res=fruit("Apple");

System.out.println(res);

System.out.println(res);

System.out.println(res);

}

}

**NOTE 2: -**

-We can call a method from a Printing statement such that we can use the Return Value at Only Once.

class MethodCallingPrinting

{

public static String fruit(String name)

{

String res=name;

String var=res +" juice";

return var;

}

public static void main(String args[])

{

System.out.println(fruit("Apple"));

}

}

**fruit(String name)**

**{**

**String res=name;**

**Staring var=res+”Juice”;**

**return var;**

**}**

**}**

**Control**

**+**

**Values**

**Control**

**+**

**Values**

**Calling Method**

**Main()**

**{**

**Sop(fruit(“Apple”);//MCS**

**}**

**NOTE:-**

-If the Method return type is **void**  then you cannot using that using variable or from printing statement.

-For Calling method from variable or Printing statement we that method should be returning something.

class MethodCallingDemo

{

public static void fruit(String name)

{

String res = name;

String var = res +" juice";

System.out.println(var);

}

public static void main(String args[])

{

fruit("Apple");

String res = fruit("Apple");//CTE

System.out.println(fruit("Apple"));//CTE

}

}

class MethodCalling

{

public static String fruit(String name)

{

String res = name;

String var = res+" Juice ";

return var;

}

public static void main (String args[])

{

//(1st)Not Using the Return Value

fruit("Apple");

//(2nd) Store the return value & use the variable

String res = fruit("Apple");

System.out.println(res);

//(3r)Printing the Return value Directly From Printing Statement

System.out.println(fruit("Apple"));

}

}

**NOTE3:**

**-**For the Void Keyword (ReturnType) Return Statement is not Mandatory. [But still it will just transfer the control from Called Method Back to its Calling Method.

class MethodReturnType

{

public static void test()

{

System.out.println("test()-method");

return ;//not mandatory

}

public static void main (String args[])

{

test();

}

}

**METHOD OVERLOADING**

-A class consist of More than one Method with Same Name and Different Arguments is called is **Method** **Overloading.**

-Inside the class block we cannot declare more than one Method with Same Name & Same Arguments.

class MethodOverloading

{

public static void test ()

{

System.out.println("test-()");

}

public static void test (int a)

{

System.out.println("test(int)");

}

public static void main(String args[])

{

test(10);

}

}

**NOTE:-**

-If the we pass character data and there is multiple methods that accept the different datatype(int,long,float,double).

-Then the character type methods gets executed because it doing method Typecasting [Wideing].

-It checks nearest datatype.

============== WIDENING PROCESS ===============

byte

Char 🡺 int 🡺 long 🡺 float 🡺double.

short

class MethodOverloading

{

public static void test ()

{

System.out.println("test-()");

}

public static void test (int a)

{

System.out.println("test(int)");

}

public static void main(String args[])

{

test(10);

}

}

**test (double b)**

**{**

**test (double a)**

**}**

**{**

**test(int a)**

**}**

**test(int a)**

**main()**

**{**

**Test(‘a’);**

**}**

**RECURSIVE METHOD**

-The method calling itself is called as Recursive Method.

class RecursiveMethodDemo

{

public static void test()

{

System.out.println("test()");

test();

}

public static void main(String[] args)

{

test();

}

}

-In the above program the test method calls itself such that it provides an Infinite Loops.

-In order to stop the Infinite Loop, we need to Base Condition.

-Base Condition Is the condition which is used to stop the Infinite Loop.

-Base condition is condition is used to stop the method to executing from infinite loops.

**BASE-CONDITION**

class RecursiveMethod

{

public static void test(int n)

{

System.out.println(n);

if (n==5)

{

return;

}

n++;

test(n);

}

public static void main(String[] args)

{

test(0);

}

}

-In the above program we have used the base condition such that method cannot executed infintly.

-Base condition it is used to Stop the Execution and Transfer the Control back to its Calling Method.

QNo1.Write a program to print first five numbers using Recursive Method?

OR

QNo1.Write a program to print the first five numbers without using the Looping Statement?

class Print5Num

{

public static void printNum(int num)

{

System.out.println(num);

if (num==5)

{

return;

}

num++;

printNum(num);

}

public static void main(String[] args)

{

System.out.println("Start");

printNum(0);

System.out.println("End");

}

}

**METHOD EXECUTION FLOW**

class MethodExecutionFlow

{

public static void Suraj(String fruit)

{

System.out.println(Aniket("Apple Juice"));

}

public static boolean Aniket(String Juice)

{

System.out.println(Akash("Rose"));

return true;

}

public static String Akash(String Flower)

{

System.out.println("Proposed A Girl");

return "She Said I like Suraj";

}

public static void main(String args[])

{

Suraj("Apple");

}

}

🡺Proposed A Girl

🡺She Said I like Suraj

🡺true

Akash(String Fruit)

Aniket(String Fruit)

{

Sop(“Proposed A Girl”);

return “She Said I like Suraj”;

}

{

Sop(Akash(“Rose”));

return true;

}

Suraj(String Fruit)

{

Sop(Aniket(“Apple Juice”);

return;

}

main

{

Suraj(“Apple”);

}

**ARRAY**

-Array it is a continuous block of memory where we can store the Same type of values.

**OR**

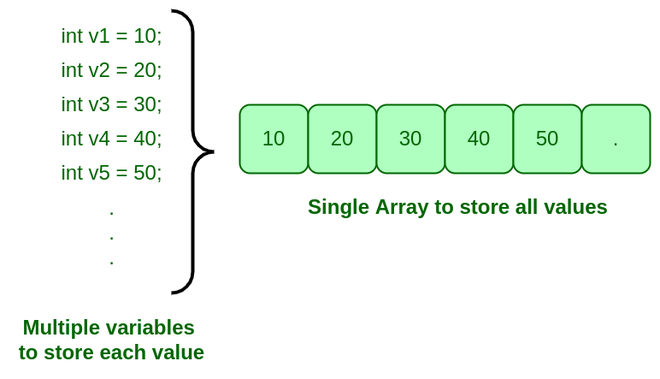
-Array it is a continuous block of memory where we can store the **Homogenous type** of values.

-Homogenous means same type of values.

**Need Of Array**

-Before array we can store value using normal variable suppose if there is multiple values so to store that multiple value we need to create the multiple variables.

-So using array we can store the multiple elements.



-In order to store same type of values we need to create an **Object** [Block of Memory]

-Before creating an object, we need to create a block of memory to store the Address of the object.

-To store Homogenous type of values using Array we need create:-

**1]Array Reference Variable**

**2]Array Object**

**1]Array Reference Variable**

-Array reference variable it is used to store the Address of the Array object.

-We can create an Array for both Primitive & NonPrimitive Datatype.

**Syntax:-**

**1] datatype[] VariableName;**

**2] datatype VariableName [];**

class ArrayDemo

{

public static void main(String[] args)

{

int[] a;

byte[] b;

long[] c;

short[] d;

float[] e;

double[] f;

char[] g;

boolean[] h;

String[] i;

}

}

class ArrayDemo1

{

public static void main(String[] args)

{

int a [];

byte b [];

long c [];

short d [];

float e [];

double f [];

char g [];

boolean h [];

String i [];

}

}

**2]Array Object**

-Array object is consist of Address.

-We can create an Array Reference variable for both Primitive or NonPrimitive datatype.

-We can create an Array object in two ways:-

**1]Declaration & Initialization**

**2]Using New Keyword**

**1]DECLARATION & INITIALIZATION**

**Syntax**:

**Datatype[] VariableName={value1,value2,value3…..value n};**

//QNO1:-Creating Array using Array Declaration And Initialization

class ArrayDeclarationAInitialization

{

public static void main(String[] args)

{

int [] arr = {1,2,3,4,5};//Declaration And Initialization Of Array

System.out.println(arr);

System.out.println(arr.length)

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--------->>>>>OUTPUT>>>>------------\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-->>[I@3fee733d \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int [] arr = {1,2,3,4};

**@123**

**@123**

|  |
| --- |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

Array Object

-If we print the **arr** then it print the address of the array.

-For accessing the array element you have use the index value.

**NOTE:-**

-Index value always start from 0.

-Length is always start from 1.

**Length Property**

-It is used to count the number the units present inside the array object.

-Array always consist of **Fixed-Length.**

-We can access the length of array with the help of Length variable

**Syntax:-**

**ArrayReferenceVariableName.length;**

class ArrayLength

{

public static void main(String[] args)

{

int[] array = {1,2,3,4,5};

System.out.println(array.length);

}

}

**Access Array Element**

-We can access the units of array object with the help of Index Value.

-Index Value always start with **0**[**Zero**].

-To access index value we need to take the help of array Reference Variable.

**Syntax:-**

**ArrayReferenceVariable [Index Value];**

class ArrayDemo

{

public static void main(String[] args)

{

int[] array = {10,20,30};

System.out.println(array[0]);

System.out.println(array[1]);

System.out.println(array[2]);

System.out.println(array.length);

}

}

Int arr = {10,20,30};

**@123**

|  |
| --- |
| 10 |
| 20 |
| 30 |

0 1 1 2

**Count**

**Length**

**@123**

**Index Value**

3 2

**NOTE:-**

-Declaration an Initialization of array cannot be perform in multiple statement. [We get compile Time Error].

//QNO2:-Declaration And Initialization Array In Clone Multiple Statement

class DeclarationAInitializationMultipleStatement

{

public static void main(String[] args)

{

int [] n;

n = {1,2,3,4,5};//Declaration And Initialization Array Cannot Be Clone In Multiple Statement

System.out.println(n);//We get compile Time Error

}

}

DeclarationAInitializationMultipleStatement.java:8: error: illegal start of expression

n={1,2,3,4,5};//Declaration And Initialization Array Cannot Be Clone In Multiple Statement

^

DeclarationAInitializationMultipleStatement.java:8: error: not a statement

n={1,2,3,4,5};//Declaration And Initialization Array Cannot Be Clone In Multiple Statement

^

DeclarationAInitializationMultipleStatement.java:8: error: ';' expected

n={1,2,3,4,5};//Declaration And Initialization Array Cannot Be Clone In Multiple Statement

^

DeclarationAInitializationMultipleStatement.java:9: error: <identifier> expected

System.out.println(n);//WE get compile Time Error

**NOTE:-**

-Other way to printing the Array element using loops.

-Using for loop we can print easily the array element.

class ArrayAccess

{

public static void main(String[] args)

{

int[] arr = {11,22,33,44,55};

System.out.println(arr);

System.out.println(arr.length);

for(int index=0;index<arr.length;index++)

{

System.out.println(arr[index]);

}

}

}

**2]Using New Keyword**

**Syntax**:-

**Datatype[] VariableName=new datatype[Size];**

**New Keyword: -**

-New keyword it is used to create an **Object** [A block of Memory].

class ArrayNewDemo

{

public static void main(String[] args)

{

int[] arr = new int[3];

System.out.println(arr);

System.out.println(arr.length);

}

}

[I@6d06d69c

3

Press any key to continue . . .

class ArrayInsertDemo

{

public static void main(String[] args)

{

int[] arr = new int[3];

System.out.println(arr);

System.out.println(arr.length);

System.out.println(arr[0]);

System.out.println(arr[1]);

System.out.println(arr[2]);

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;

System.out.println(arr[0]);

System.out.println(arr[1]);

System.out.println(arr[2]);

}

}

[I@6d06d69c

3

0

0

0

10

20

30

Press any key to continue . . .

**Index Value**

**Address**

int[ ] arr = new[3];

**[I@6d06d69c**

|  |
| --- |
| 0 10 |
| 0 20 |
| 0 30 |

**[I@6d06d69c**

0

**Array Object**

1

2

arr[0]=10;

arr[1]=20;

arr[2]=30;

**Default**

**Value**

QNo1.Write a program to find the summation of an Array.

class SummationFiveNumStoreArray

{

public static void main(String[] args)

{

int num[]={1,2,3,4,5};

int sum=0;

for(int index=0;index<num.length;index++)

{

sum=sum+num[index];

}

System.out.println(sum);

}

}

15

Press any key to continue . . .

**ArrayIndexOutOfBoundsException**

-When we try to access Array, units based on its Index Value.

-If the Index Value is **Greater** or Equal’s to **Length** of Array Object, we get an

**Runtime** **Exception** called as **ArrayIndexOutOfBoundsException**.

-Whenever an Exception occurs the execution of the program is forcefully Stopped.

class ArrayIndexOutOfBoundsException

{

public static void main(String[] args)

{

System.out.println("Main Starts");

int[] arr = new int[3];

System.out.println(arr[0]);

System.out.println(arr[1]);

System.out.println(arr[2]);

System.out.println(arr[3]);//Exception

System.out.println("Main Ends");

}

}

Main Starts

0

0

0

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3

at ArrayIndexOutOfBoundsException.main(ArrayIndexOutOfBoundsException.java:10)

Press any key to continue . . .

-In above program we already define the length

-If we trying access the element if the **Index value** is greater or Equal to the length then we **ArrayIndexOutOfBoundsException.**

-If this exception occurs then the remaining code of the program is not excuted.

**ARRAY DISADVANTAGES / LIMITATION OF ARRAY**

1]Array is fixed in size.

2]Array can hold only Homogenous Type of elements.

3]Arrays concept is not implemented based on some standard data structure hence there no predefined or readymade method support is not available for array.

**Scanner Class**

Scanner is a In build class in java it is used to take obtain the input of primitive types like int,double,char,string.

-It is easiest way to read the input from the user.

-Scanner class present inside **java.util** package we need to import it.

-Using Scanner class we take input from user during Runtime.

Syntax: -import packageName Member To Imported

Syntax: - import java.util.Scanner

**Scanner Classes Method**

**Byte ======> nextByte()**

**Short** **======> nextShort()**

**Int ======> nextInt()**

**Long ======> nextLong()**

**Float ======> nextFloat()**

**Double ======> nextDouble()**

**Char ======> next()**

**charAt(0)**

**Boolean ======> nextBoolean()**

**String ======> next()**

**nextLine()**

**next():-next method is takes input as a Word**

**nextLine() :-nextLine method is takes input as a Line**

**STRING**

-String is a Non-Primitive Datatype.

-String it is an **Inbuild Class** Which is Present **java.lang** Package.

-The Fully Qualified Name Of String Class **java.lang.String**.

-String class is a **final**.

-String Class Extends Object Class and Implements Two Interfaces

**1]Comparable Interface**

**2]Serializable Interface**

**3]Char Sequence Interface**

Char Sequence Interface

Serializable Interface

Comparable Interface

OBJECT

CLASS

**EXTENDS** **IMPLEMENTS**

**STRING**

-**String** is a **Concrete Class** in java.

-We can create the **Object** for the String Class in two ways

**1]Using Assignment Operator[Literal Way]**

**2]Using New Keyword.**

**#Program**

Class P1

{

Public static void main(String args[])

{

String s1=”java”;

System.out.println(s1);

String s2=new String(“Java”);

System.out.println(s2);

}

}

Java.lang.String@123

“JAVA”

String s1 = ”java”;

Java.lang.String@123

-The Created Object using the Assignment operator for the String it will be Store Inside the String Constant Pool Area.

-We Cannot modify The Value Of the String Object Because it is final in Nature

-String Constant Pool Does Not Allow To Store Duplicate Object.

Java.lang.String@123

“JAVA”

String s1 = ”java”;

Java.lang.String@123

Java.lang.String@123

“JAVA”

String s2 = ”java”;

Java.lang.String@123

-If the Value of the String Object Is Same it does not created Object for the New String Object inside String Constant Pool Area.

Java.lang.String@123

“JAVA”

String s1 = ”java”;

Java.lang.String@123

Java.lang.String@123

“JAVA”

String s2 = new String(”java”;)

Java.lang.String@123

-If The We Creating Object for String Using Assignment then the

Object will be Created Inside the String Constant Pool Area.

-If the Object Created String using The New Keyword then the Object will be Created inside the Heap Area.

#Program

Class P2

{

Public static void main(String args[])

{

String s1=”java”;

String s2=”java”;

String s3=new String(“java”);

String s4=new String(“java”);

System.out.println(s1==s2);//true

System.out.println(s1==s3);//false

System.out.println(s2==s4);//false

System.out.println(s3==s4);//false

}

}

-Heap Area Allow To Create the Duplicate Object of the String.

Java.lang.String@123

“JAVA”

String s1 = ”java”;

Java.lang.String@123

Java.lang.String@123

“JAVA”

String s2 = ”java”;

Java.lang.String@123

Java.lang. String@123

“JAVA”

String s3 = new String(”java”;)

Java.lang.String@1234

“JAVA”

Java.lang.String@1234545

String s4 = new String(”java”;)

Java.lang.String@12345

-The object Class Method Overridden from String Class such that it won’t return the Address instead it will return the Value Present Object.

#Program

Class P3

{

String s1=”java”;

String s2=new String(“java”);

//Compare the Address Of the Object

System.out.println(s1==s2);

//Compare The Value Present Inside the Object

System.out.println(s1.equals(s2));

}

}

Java.lang.String@123

“JAVA”

String s1 = ”java”;

Java.lang.String@123

//Compare Address

S1==s2(Address)

Java.lang.String@1234

“JAVA”

String s2 = new String(”java”;)

Java.lang.String@1234

//Compare The Value Present Inside the Objects

s1.equals(s2);

#Program

Class P4

{

Public static void main(String args[]

{

String s1=”java”;

String s2=”java”;

System.out.println(s1==s2);

System.out.println(s1.equals(s2));

System.out.println(s1.equalsIgnoreCase(s2));

System.out.println(s2.charAt(0));

System.out.println(s2.length());

String s3=”mango”;

System.out.println(s3.toUppercase());

String s4=”MANGO”;

System.out.println(s4.toLowerCase());

String s5=”JUICE”;

System.out.println(s4.concat(s5));

}

}

**equalsIgnoreCase()**

-It is Used to Compare the Value the Two Different String Object by ignoring the Case Sensitive.

-The Return Type IS Boolean Condition

**charAt()**

-It is Used to Extract the Character a given String based on its **Index Value**.

Syntax:

**charAt(0);**

**length()**

-It is used to Calculate or Count the Number Of Elements which is Present inside the Given String.

-The ReturnType Integer Value.

**toUpperCase()**

-It is used to convert Lowercase String to Uppercase String.

**toLowerCase()**

-It is used to convert Uppercase String to Lowercase String.

**concat()**

-It is Used to Merge the Values of two different String Objects.

**compareTo()**

-It used to compare the Two Different String and Return the Output

In Integer Format.

#Program P2

Class P1

{

Public static void main(String args[])

{

String s1=”java”;

String s2=”java”;

System.out.println(s1.compareTo(s2));//0

System.out.println(“a”.compareTo(“A”));//32

System.out.println(“A”.compareTo(“a”));//-32

}

}

**Case 1: String 1 == String 2 Output: 0(Zero)**

**Case 2:String1 > String 2 Output :Positive**

**Case 3: String1 < String 2 Output :Negative**

**indexOf()**

­-It is used to return Index Value of a given Character or String.

-Index Value Always Will be in Integer Format.

-If the Given Character or String is Not Present then it will Return A Negative value (**-1**).

-If we Pass the String Inside the Index Of Method it will Return Index value of the First Character of the given String.

**Syntax:**

**1]indexOf**

**2]indexOf(character,index\_value)**

**3]indexOf(String)**

**4]indexOf(String,index\_value)**

**#Programs**

Class P2

{

Public static void main(String args[])

{

String s1=”Maharashtra”;

System.out.println(s1);

System.out.println(s1.charAt(1));

System.out.println(s1.indexOf(‘a’));

System.out.println(s1.indexOf(‘c’));

System.out.println(s1.indexOf(‘a’,2));

System.out.println(s1.indexOf(“tra”));

System.out.println(s1.indexOf(“ras”));

System.out.println(s1.indexOf(“ras”,5));//-1

}

}

**substring()**

-It is used to Extract a character or a String From a Given String.

**Syntax:**

**substring(index\_value)**

**substring(Start Index\_Value,End Index\_Value)**

#Program

Class P4

{

Public static void main(String args[])

{

String s1=”corejava”;

System.out.println(s1);

System.out.println(s1.substring(4));

//0<4

System.out.println(s1.substring(0,4));

//4<5

System.out.println(s1.substring(4,5));

System.out.println(substring(0));

System.out.println(substring(8));

}

}

-For the substring we Pass Greater Index Value we cannot get StringIndexOutBoundsException it prints the Empty Space.

#Program

Class P5

{

Public static void main(String args[])

{

String s1=”Maharashtra”;

For(int r=0;r<s1.length();r++)

{

char m=s1.charAt(r);

System.out.println();

}

}

}

**ASSIGNMENT QUESTIONS**

QNo1.Write A Program to Print the Uppercase Characters Which is Present inside the Given String.

Ans:-

class PrintUppercase

{

public static void main(String[] args)

{

String str="MaHaraStra";

System.out.println(str.length());

for (int i=0;i<str.length();i++)

{

if (str.charAt(i)>='A' && str.charAt(i)<='Z')

{

System.out.println(str.charAt(i));

}

}

}

}

QNo2.Write A Program to Count Number Of Lowercase Character Present in Given String.

Ans:-

class CountLowerCase

{

public static void main(String[] args)

{

String str="JavaClass";

System.out.println(str);

int count=0;

for (int r=0;r<str.length();r++)

{

if (str.charAt(r)>='a' && str.charAt(r)<='z')

{

System.out.println(str.charAt(r));

count++;

}

}

System.out.println("The Number Of Lowercase Letteres :"+count);

}

}

QNo3.Write the Program to Count Number of Vowels Present in a Given String.

Ans:

class CheckVowels

{

public static void main(String[] args)

{

String str="i love Maharastra";

System.out.println(str);

for (int r=0;r<str.length();r++)

{

if (str.charAt(r)=='a' || str.charAt(r)=='e' || str.charAt(r)=='i' ||str.charAt(r)=='o' || str.charAt(r)=='u')

{

System.out.println("The Vowels Are:"+str.charAt(r));

}

}

}

}

QNo4.Write A Program To Count the Number of Uppercase Vowels and Lowercase Vowels Present in a Given String.

Ans:-

class CountUppLowVowels

{

public static void main(String[] args)

{

String str="I Love Pune.It BeautifAEul Place";

System.out.println(str);

int count=0;

for (int r=0;r<str.length();r++)

{

if (str.charAt(r)=='a' || str.charAt(r)=='e' || str.charAt(r)=='o' || str.charAt(r)=='i' || str.charAt(r)=='u')

{

System.out.println("The Lowercase Vowels Count Is :"+str.charAt(r));

count++;

}

}

System.out.println("The Lowercase Vowels Are: "+count);

if (count!=0)

{

count=0;

for (int r=0;r<str.length();r++)

{

if (str.charAt(r)=='A' || str.charAt(r)=='E' || str.charAt(r)=='I' || str.charAt(r)=='O' || str.charAt(r)=='U')

{

System.out.println("The Uppercase Vowels Are :"+str.charAt(r));

count++;

}

}

}

System.out.println("The Upperacse Vowels Count Is : "+count);

}

}

QNo5.Write A Program to Count the Number Of Digit in Given String.

Ans:-

class CountDigit

{

public static void main(String[] args)

{

String str="WWC13";

System.out.println(str);

int Digit=0;

for (int r=0;r<str.length();r++)

{

if (str.charAt(r)>='0' && str.charAt(r)<='9')

{

System.out.println(str.charAt(r));

Digit++;

}

}

System.out.println("The Number Of Digit :"+Digit);

}

}

QNo6.Write a Program to Reverse a String.

class ReverseString

{

public static void main(String[] args)

{

String str="java";

String Rev= "";

for (int r=0;r<str.length();r++)

{

Rev=str.charAt(r) + Rev;

}

System.out.println("Reverse String Is :"+Rev);

}

}

Note : We cannot modify the value which is Present inside the String Object.Hence the Object of the String it is also called as

**Immutable [Modification Not Allowed].**

#Program

Class P1

{

Public static void main(String args[])

{

String s1=”sam”;

System.out.println(s1);

String s2=”sung”;

System.out.println(s2);

String s3=s1.concat(s2));

System.out.println((s1));

System.out.println((s2));

System.out.println(s3));

}

}

@100

“Sam”

String s1

@100

@200

“sung”

String s2

@200

@300

“Samsung”

String s3=s1.concat(s3)

@300

-With the help of concat() we can Merge value of two different object and create New Object.

**CONVERTING AN ARRAY TO STRING**

Program

Class P2

{

Public static void main(String args[])

{

char[] ch={‘c’,’a’,’t’};

int[] n={1,2,3};

String s1=new String(ch);

System.out.println(s1);

String s2=String .valueOf(ch);//boxing

System.out.println(s2);

System.out.println(n);

System.out.println(ch);

System.out.println(ch.toString());

}

}

**O/P**

**🡺cat**

**🡺cat**

**🡺[I@39ed]**

**🡺cat**

**🡺[C@123]**

**toCharArray()**

-It is used to convert String into Array of characters.

#Program

Class P3

{

Public static void main(String args[])

{

String s1=”JavaClass”;

System.out.println(s1);

//Converting String to Array

Char ch[]=s1.toCharArray();

for(int n=0;n<ch.length;n++)

{

System.out.println(ch[n]);

}

}

}

**O/P**

**🡺JavaClass**

**🡺J**

**🡺a**

**🡺v**

**🡺a**

**🡺C**

**🡺l**

**🡺a**

**🡺s**

**🡺s**

**trim()**

-It is used to delete the Space of Starting and end of the given String.

#program

Class P4

{

Public static void main(String args[])

{

String s1=” ABCDE “;

System.out.println(s1);

System.out.println(s1.length());

String s2=s1.trim();

System.out.println(s2);

System.out.println(s2.length());

}

}

**o/p**

**🡺\_\_\_ABCDE\_\_\_\_\_**

**🡺13**

**🡺ABCDE**

**🡺5**

**spilt()**

-It is used to divide the Multiple String.

#Program

Class P5

{

Public static void main(String args[])

{

String s1=”pug husky elephant”;

System.out.println(s1);

System.out.println(s1.contais(“husky”));

//Array

String[] s2=s1.spilt(“”);

for(int n=0;n<s2.length;n++)

{

System.out.println(s2[n]);

}

}

}

String s2

@A500

@100

@A500

@100

pug

@300

@200

@300

elephant

husky

@200

**STRING BUFFER**

-StringBuffer is a inbuild Class In Java which helps programmer to create **Mutable** Objects.

-StringBuffer don’t have concepts called **String Constant Pool**

-StringBuffer is **Thread Safe** class mean **Multithreading** is Not possible in StringBuffer.

-We can create object for StringBuffer in only one-way i.e

With the help of **New** keyword.

-There are some inbuild Non-Static Methods which are Present inside the StringBuffer class.

-Methods of StringBuffer class are as follows:

|  |  |  |
| --- | --- | --- |
|  | Methods | Use Of Methods |
| 1. | **appends(String)** | -This Method works similar to the concat method of String Class.  But instead of creating new object  **appends()** methods update the String Literals which is present inside the current object. |
| 2. | **replaceOf(int StartIndex,int EndIndex,String regex)** | -It will replace the substring between the given index range and add the newly passed String into the Old String. |
| 3. | **replace(Old char,New char)** | -This Method used to replace the character with newly passed character |
| 4. | **reverse()** | -This method is to Reverse the String which is present inside current object. |
| 5. | **capacity()** | -This method used to return the capacity of StringBuffer. |

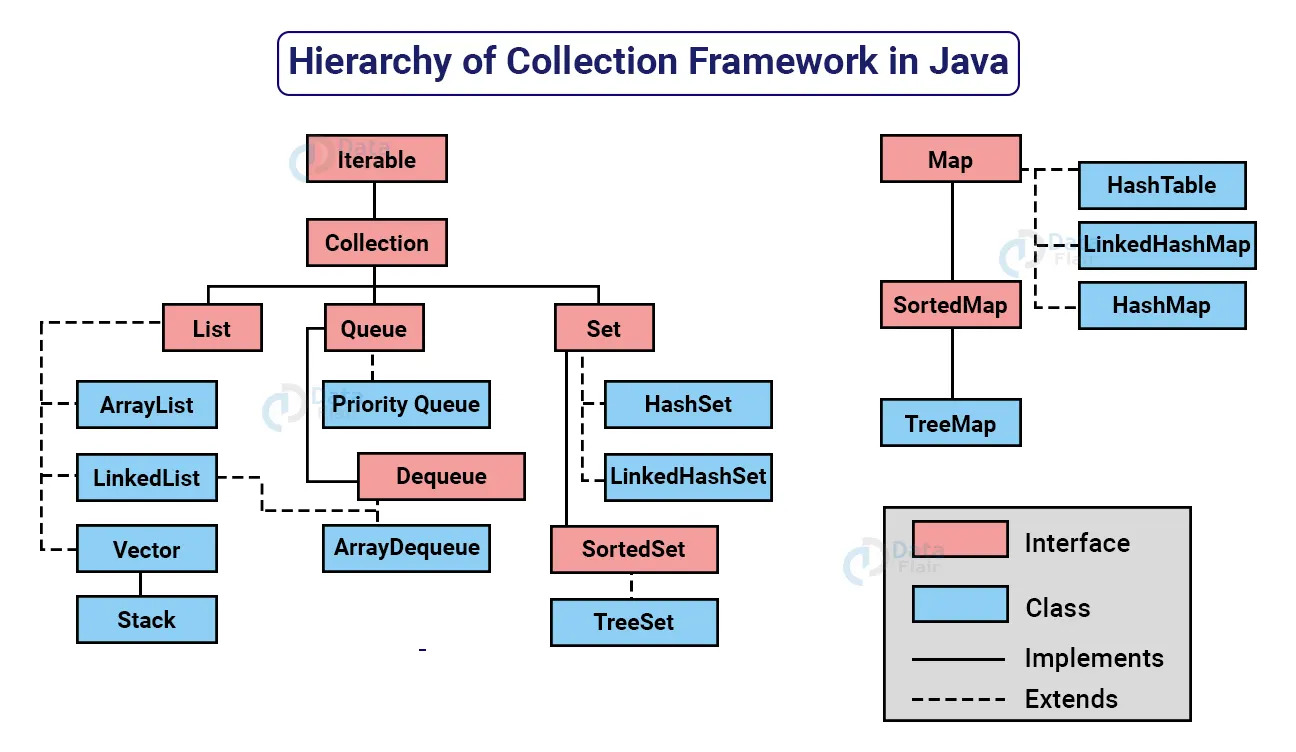
QNo1.Write down the Difference between String and StringBuffer.

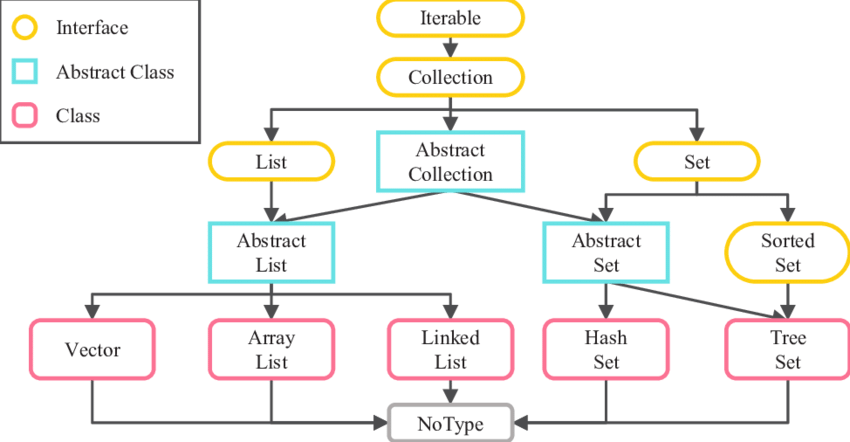
QNo2.Create class and declare a String variable and print first three character of the given String in Uppercase and rest String into Lowercase.

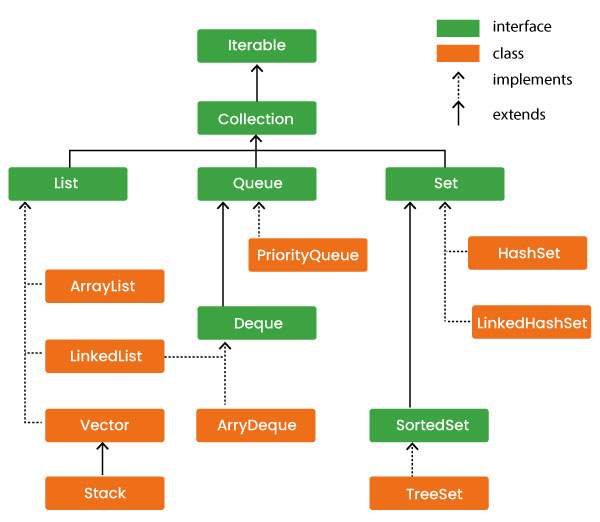
QNo3. Create class and declare a String variable and print Last three character of the given String in Uppercase and rest String into Lowercase.

QNo4.Create class and declare the String Variable and print the first letter in Uppercase for Every Word without using Inbuild Method.

**COLLECTION FRAMEWORK**







-With the help of Collection framework, we can store the Objects and maintain in organized way.

-The Collection framework it is Present inside the **java.util** Package**.-**

-The Fully Qualified Name For All the Collection class is **java.util.ClassName.**

**Java Collection Interfaces**

The collections framework defines several interfaces. This section provides an overview of each interface −

|  |  |
| --- | --- |
| Sr.No. | Interface & Description |
| 1 | [**The Collection Interface**](https://www.tutorialspoint.com/java/java_collection_interface.htm)  This enables you to work with groups of objects; it is at the top of the collections hierarchy. |
| 2 | [**The List Interface**](https://www.tutorialspoint.com/java/java_list_interface.htm)  This extends **Collection** and an instance of List stores an ordered collection of elements. |
| 3 | [**The Set**](https://www.tutorialspoint.com/java/java_set_interface.htm)  This extends Collection to handle sets, which must contain unique elements. |
| 4 | [**The SortedSet**](https://www.tutorialspoint.com/java/java_sortedset_interface.htm)  This extends Set to handle sorted sets. |
| 5 | [**The Map**](https://www.tutorialspoint.com/java/java_map_interface.htm)  This maps unique keys to values. |
| 6 | [**The Map.Entry**](https://www.tutorialspoint.com/java/java_mapentry_interface.htm)  This describes an element (a key/value pair) in a map. This is an inner class of Map. |
| 7 | [**The SortedMap**](https://www.tutorialspoint.com/java/java_sortedmap_interface.htm)  This extends Map so that the keys are maintained in an ascending order. |
| 8 | [**The Enumeration**](https://www.tutorialspoint.com/java/java_enumeration_interface.htm)  This is legacy interface defines the methods by which you can enumerate (obtain one at a time) the elements in a collection of objects. This legacy interface has been superceded by Iterator. |

**JAVA COLLECTION CLASSES**

-Java provides a set of standard collection classes that implement Collection interfaces. Some of the classes provide full implementations that can be used as-is and others are abstract class, providing skeletal implementations that are used as starting points for creating concrete collections.

-The standard collection classes are summarized in the following table −

|  |  |
| --- | --- |
| Sr.No. | Class & Description |
| 1 | **AbstractCollection**  Implements most of the Collection interface. |
| 2 | **AbstractList**  Extends AbstractCollection and implements most of the List interface. |
| 3 | **AbstractSequentialList**  Extends AbstractList for use by a collection that uses sequential rather than random access of its elements. |
| 4 | [**LinkedList**](https://www.tutorialspoint.com/java/util/java_util_linkedlist.htm)  Implements a linked list by extending AbstractSequentialList. |
| 5 | [**ArrayList**](https://www.tutorialspoint.com/java/util/java_util_arraylist.htm)  Implements a dynamic array by extending AbstractList. |
| 6 | **AbstractSet**  Extends AbstractCollection and implements most of the Set interface. |
| 7 | [**HashSet**](https://www.tutorialspoint.com/java/util/java_util_hashset.htm)  Extends AbstractSet for use with a hash table. |
| 8 | [**LinkedHashSet**](https://www.tutorialspoint.com/java/util/java_util_linkedhashset.htm)  Extends HashSet to allow insertion-order iterations. |
| 9 | [**TreeSet**](https://www.tutorialspoint.com/java/util/java_util_treeset.htm)  Implements a set stored in a tree. Extends AbstractSet. |
| 10 | **AbstractMap**  Implements most of the Map interface. |
| 11 | [**HashMap**](https://www.tutorialspoint.com/java/util/java_util_hashmap.htm)  Extends AbstractMap to use a hash table. |
| 12 | [**TreeMap**](https://www.tutorialspoint.com/java/util/java_util_treemap.htm)  Extends AbstractMap to use a tree. |
| 13 | [**WeakHashMap**](https://www.tutorialspoint.com/java/util/java_util_weakhashmap.htm)  Extends AbstractMap to use a hash table with weak keys. |
| 14 | [**LinkedHashMap**](https://www.tutorialspoint.com/java/util/java_util_linkedhashmap.htm)  Extends HashMap to allow insertion-order iterations. |
| 15 | [**IdentityHashMap**](https://www.tutorialspoint.com/java/util/java_util_identityhashmap.htm)  Extends AbstractMap and uses reference equality when comparing documents. |

-The AbstractCollection, AbstractSet, AbstractList, AbstractSequentialList and AbstractMap classes provide skeletal implementations of the core collection interfaces, to minimize the effort required to implement them.

-The following legacy classes defined by java.util have been discussed in the previous chapter −

|  |  |
| --- | --- |
| Sr.No. | Class & Description |
| 1 | [**Vector**](https://www.tutorialspoint.com/java/util/java_util_vector.htm)  This implements a dynamic array. It is similar to ArrayList, but with some differences. |
| 2 | [**Stack**](https://www.tutorialspoint.com/java/util/java_util_stack.htm)  Stack is a subclass of Vector that implements a standard last-in, first-out stack. |
| 3 | [**Dictionary**](https://www.tutorialspoint.com/java/util/java_util_dictionary.htm)  Dictionary is an abstract class that represents a key/value storage repository and operates much like Map. |
| 4 | [**Hashtable**](https://www.tutorialspoint.com/java/util/java_util_hashtable.htm)  Hashtable was part of the original java.util and is a concrete implementation of a Dictionary. |
| 5 | [**Properties**](https://www.tutorialspoint.com/java/util/java_util_properties.htm)  Properties is a subclass of Hashtable. It is used to maintain lists of values in which the key is a String and the value is also a String. |
| 6 | [**PriorityQueue**](https://www.tutorialspoint.com/java/util/java_util_priorityqueue.htm)  PriorityQueue class is an unbounded priority queue based on a priority heap. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects. |
| 7 | [**BitSet**](https://www.tutorialspoint.com/java/util/java_util_bitset.htm)  A BitSet class creates a special type of array that holds bit values. This array can increase in size as needed. |
| 8 | [**ArrayDeque**](https://www.tutorialspoint.com/java/util/java_util_arraydeque.htm)  ArrayDeque class provides resizable-array and implements the Deque interface. Array deques have no capacity restrictions so they grow as necessary to support usage. |
| 9 | [**EnumMap**](https://www.tutorialspoint.com/java/util/java_util_enummap.htm)  EnumMap class is a specialized Map implementation for use with enum keys. All of the keys in an enum map must come from a single enum type that is specified, explicitly or implicitly, when the map is created. |
| 10 | [**Queue**](https://www.tutorialspoint.com/java/java_util_queue.htm)  The queue interface is provided in java.util package and it implements the Collection interface. The queue implements FIFO i.e. First In First Out. This means that the elements entered first are the ones that are deleted first. |
| 11 | [**Deque**](https://www.tutorialspoint.com/java/util/java_util_deque.htm)  EnumMap class is a specialized Map implementation for use with enum keys. All of the keys in an enum map must come from a single enum type that is specified, explicitly or implicitly, when the map is created. |

**ITERABLE**

-In simple words, the iterable interface is a common interface that allows us to iterate over a collection of objects. It was first introduced with the release of JDK 1.5 and made available in **'java.lang**' package.

-The Java Collection Framework extends this interface, hence all the classes available in this collection framework by default implement the iterable interface.

-In other words, the classes of collection framework such as ArrayList, TreeSet, TreeMap and HashMap are iterable.

General Syntax of Iterable Interface

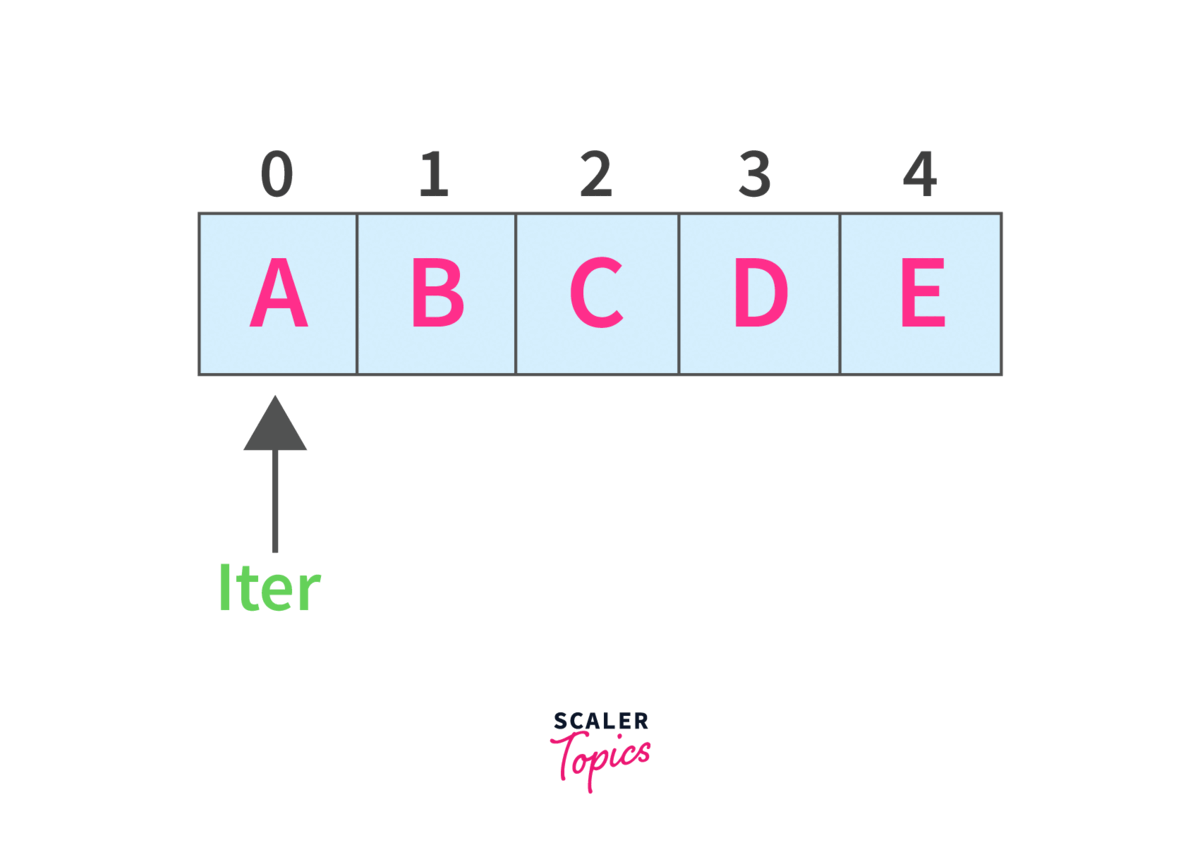
public interface Iterable<TypeOfElement> { }

General Syntax of Iterator()

Iterator<TypeOfCollection> instanceOfIterator = nameOfCollection.iterator();

The Iterator interface provides the following methods

* **hasNext()** − It returns true if there are more elements to iterate over.
* **next()** − It returns the next element in the iteration if hasNext() returns true.
* **remove()**− It removes the current element from the collection.



**COLLECTIONS**

-The Collection interface is the root interface of the collections framework hierarchy.

-If we want to represent a group of individual objects as a single entity then we should go for collection.

-A **Collection** is an interface allowing items to be grouped within a single container object. It is found in the java.util package and is implemented by List, [Set](https://www.codecademy.com/resources/docs/java/set), and [Queue](https://www.codecademy.com/resources/docs/java/queue).

-There are many varied Collection types in Java, and the special [Collections](https://www.codecademy.com/resources/docs/java/collections) class is provided to manipulate them.

**Syntax**

import java.util;

Collection<DataType> c = new CollectionClass<DataType>();

-Where DataType is the data type to be stored in the collection and CollectionClass is some class implementing the Collection interface.

**Methods**

The Collection interface utilizes the following methods:

|  |  |  |
| --- | --- | --- |
| Method | Syntax | Description |
| .add() | .add(item) | Adds item to the collection if it wasn’t a member already. |
| .addAll() | .addAll(collection) | Adds the elements in collection to the collection. |
| .clear() | .clear() | Removes all the items in the collection. |
| .contains() | .contains(item) | Returns true if item is a member of the collection. |
| .containsAll() | .contains(collection) | Returns true if all the items in collection are contained in the collection. |
| .isEmpty() | .isEmpty() | Returns true if the collection contains no elements. |
| .remove() | .remove(item) | Removes item from the collection. |
| .removeAll() | .removeAll(collection) | Removes all items in collection from the collection. |
| .size() | .size() | Returns the number of elements in the collection. |
| .toArray() | .toArray() | Returns an array containing all the elements in the collection. |
|  | .toArray(array) | Returns an array whose data type is the same as array. |