|  |
| --- |
| The **JRE is** the environment within which the virtual machine runs. **JRE is** the container, **JVM is** the content. Java Runtime Environment contains **JVM**, class libraries, and other supporting files. It does not contain any development tools such as compiler, debugger, etc |
| Java Program Execution Process in Detail |
| CLASS LOADING: |
|  |
| * ClassLoader always follows the Delegation Hierarchy Principle. * Whenever JVM comes across a class, it checks whether that class is already loaded or not. * If the Class is already loaded in the method area then the JVM proceeds with execution. * If the class is not present in the method area then the JVM asks the Java ClassLoader Sub-System to load that particular class, then ClassLoader sub-system hands over the control to Application ClassLoader. * Application ClassLoader then delegates the request to Extension ClassLoader and the Extension ClassLoader in turn delegates the request to Bootstrap ClassLoader. * Bootstrap ClassLoader will search in the Bootstrap classpath(JDK/JRE/LIB). If the class is available then it is loaded, if not the request is delegated to Extension ClassLoader. * Extension ClassLoader searches for the class in the Extension Classpath(JDK/JRE/LIB/EXT). If the class is available then it is loaded, if not the request is delegated to the Application ClassLoader. * Application ClassLoader searches for the class in the Application Classpath. If the class is available then it is loaded, if not then a ClassNotFoundException exception is generated. |
| **What if I write static public void instead of public static void?**  The program compiles and runs correctly because the order of specifiers doesn't matter in Java. |
| **What are the various access specifiers in Java?**  **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.  **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.  **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.  **Private** The private class, methods, or variables defined as private can be accessed within the class only. |
| 30Javatpoint  Javatpoint1020  New line  In the first case, 10 and 20 are treated as numbers and added to be 30. Now, their sum 30 is treated as the string and concatenated with the string Javatpoint. Therefore, the output will be 30Javatpoint.  In the second case, the string Javatpoint is concatenated with 10 to be the string Javatpoint10 which will then be concatenated with 20 to be Javatpoint1020. |
| 200Javatpoint  Javatpoint200 |
| Wrapper Classes    **Wrapper classes in Java**  The wrapper class in Java provides the mechanism to convert **primitive into object and object into primitive**.  Since J2SE 5.0, autoboxing and unboxing feature convert primitives into objects and objects into primitives automatically.  The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.   * **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value. * **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes. * **Synchronization:** Java synchronization works with objects in Multithreading. * **java.util package:** The java.util package provides the utility classes to deal with objects. * **Collection Framework:** Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only. |
| Jar vs war file  A .war file is a Web Application Archive which runs inside an application server while a  .jar is Java Application Archive that runs a desktop application on a user's machine.  .jar files: The .jar files contain libraries, resources and accessories files like property files.  .war files: The war file contains the web application that can be deployed on any servlet/jsp container. The .war file contains jsp, html, javascript and other files necessary for the development of web applications. |
| Pass by Value: **Java is Strictly Pass by Value!**  **public** **class** myClass {  **public** **static** **void** main(String[] args) {  **int** x = 5;  *change*(x);  System.***out***.println(x);  }  **public** **static** **void** change(**int** x) {  x = 10;  }  }  o/p: 5 (Pass by Value)      o/p:  [1, 2, 3, 4, 5, 6, 7]  [1, 2, 3, 4, 5] |