**Misconception Check**

In the past several years the technology evolved too fast and there’s a lot of updates in java versions that was developed and existing now. Select 2 versions of java and compare them in terms of features, advantages, and disadvantages.

**JAVA SE 9** - Java 9 would include better support for multi-gigabyte heaps, better native code integration, a different default garbage collector and a self-tuning JVM.

Advantages of Java SE 9 are to ease of testing and maintainability, it supports better performance. As Java SE 9 is going to divide JDK, JRE, JARs etc, into smaller modules, we can use whatever modules we want. So it is very easy to scale down the Java Application to Small devices and Its easy to support Less Coupling between components and Its easy to support Single Responsibilities.

Disadvantages of java SE 9 is that the problem of the interdependency of program products and libraries has not been solved yet, making installing new products and uninstalling old ones can be difficult and After Java 9 release, previous versions of Java no longer be supported, and clients can have to spend time and resources on migration.

Features are:

* Modularization of the JDK under Project Jigsaw.
* Provided Money and Currency API.
* Tight integration with JavaFX.
* Java implementation of reactive streams.
* More Concurrency Updates.
* Provided Java Linker.
* Automatic scaling and sizing.

**JAVA SE 8** - Java SE 8 is not supported in Windows XP but after JDK 8 update 25, we can install and run it under Windows XP.

Java 8 is set as a default version to download from java.com from October 2014.

Advantage of **Java 8** basically, default methods are added to an interface and do not have to be overridden in the interface implementation. These methods can be run directly from the interface. This was done for backward compatibility for your collections in your interfaces.

Disadvantages of Java 8 is that Parallel Streams can actually slow you down, default methods are distracting and issues that are still around.

Features are:

* Language-level support for Lambda expressions.
* Allowed developers to embed JavaScript code within applications.
* Annotation of Java Types.
* Provided Date and Time API.
* Repeating Annotations.
* Launching of JavaFX applications.
* Removal of permanent generation.

|  |  |
| --- | --- |
| **B. EXPLORE: API Specifications list**  1. Enumerate at least 20 following API specifications of java, depending of the java version.  Example: follow the format  JAVA version: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Packages Description**  Provides the classes necessary to create an applet and the classes an applet uses to  **java.applet** communicate with its applet context. | |
|  | **Interfaces**  **AppletContext** - This interface corresponds to an applet's environment: the document containing the applet and the other applets in the same document. **AppletStub -** When an applet is first created, an applet stub is attached to it using the applet's setStub method.  **AudioClip -** The AudioClip interface is a simple abstraction for playing a sound clip. |
|  |
| **Classes**  **Applet** - An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application. |
|  |
| **Exceptions** |
|  |
|  |  |

|  |  |
| --- | --- |
| **B. EXPLORE: API Specifications list**  1. Enumerate at least 20 following API specifications of java, depending of the java version.  JAVA version: Java Platform, Standard Edition 8 API Specification  **Packages Description**  **java.awt** contains all of the classes for creating user interfaces and for painting graphics and images. | |
| **Java.awt.datatransfer** | **Interfaces**  **Active Event** – an interface for events that know how to dispatch themselves.  **Composite**- The Composite interface, along with [Composite Context](https://docs.oracle.com/javase/8/docs/api/java/awt/CompositeContext.html), defines the methods to compose a draw primitive with the underlying graphics area. |
|  |
| Classes  **ColorSpace** - This abstract class is used to serve as a color space tag to identify the specific color space of a Color object or, via a ColorModel object, of an Image, a BufferedImage, or a GraphicsDevice. |
|  |
| **Exceptions**  [**CMMException**](https://docs.oracle.com/javase/8/docs/api/java/awt/color/CMMException.html)**-** This exception is thrown if the native CMM returns an error.  **Profile data Exception** – This exception is thrown when an error occurs in accessing or processing an ICC Profile object |
| Provides interfaces and classes for transferring data between and within applications   |  | | --- | | **Interfaces**  [**ClipboardOwner**](https://docs.oracle.com/javase/8/docs/api/java/awt/datatransfer/ClipboardOwner.html) - Defines the interface for classes that will provide data to a clipboard.  [**FlavorMap**](https://docs.oracle.com/javase/8/docs/api/java/awt/datatransfer/FlavorMap.html) - A two-way Map between "natives" (Strings), which correspond to platform- specific data formats, and "flavors" (DataFlavors), which correspond to platform-independent MIME types. | | **Classes**  [**Clipboard**](https://docs.oracle.com/javase/8/docs/api/java/awt/datatransfer/Clipboard.html)- A class that implements a mechanism to transfer data using cut/copy/paste operations.  **Data Flavor –** a data flavor provides metaa information about data. | | **Exceptions**  [**MimeTypeParseException**](https://docs.oracle.com/javase/8/docs/api/java/awt/datatransfer/MimeTypeParseException.html)**-** A class to encapsulate MimeType parsing related exceptions.  Unsupported Flavor Exception – Signals that the requested data is not supported in this flavor. | |
| **java.awt.dnd****java.awt.font****java.awt.geom** [**java.awt.im**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/package-summary.html)  [**java.awt.im.spi**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/spi/package-summary.html) **java.awt.image.renderable** **Java.awt.print**  **Java.beans**  **Java.beans.beancontecxt**  **Java.io**  **Java.lang** **java.lang.annotation** **Java.lang.instrument**  **Java.lang.invoke**  **Java.lang.reflect**  **Java.net**  **Java.nio.file**  **Java.rmi** | **Description**  Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.   |  | | --- | | **Interfaces**  [**Autoscroll**](https://docs.oracle.com/javase/8/docs/api/java/awt/dnd/Autoscroll.html) - During DnD operations it is possible that a user may wish to drop the subject of the operation on a region of a scrollable GUI control that is not currently visible to the user.  [**DropTargetListener**](https://docs.oracle.com/javase/8/docs/api/java/awt/dnd/DropTargetListener.html) - The DropTargetListener interface is the callback interface used by the DropTarget class to provide notification of DnD operations that involve the subject DropTarget. | | **Classes**  [**DnDConstants**](https://docs.oracle.com/javase/8/docs/api/java/awt/dnd/DnDConstants.html)- This class contains constant values representing the type of action(s) to be performed by a Drag and Drop operation.  Drag Gesture event – passed to Drag Gesture Listener’s drag Gesture Recognized() method when a particular Drag Gesture Recognizer detects that a platform dependent drag initiating gesture has occurred on the Component that it is tracking. | | **Exceptions**  [**InvalidDnDOperationException**](https://docs.oracle.com/javase/8/docs/api/java/awt/dnd/InvalidDnDOperationException.html) - This exception is thrown by various methods in the java.awt.dnd package. |   **Descriptions**    Provides classes and interface relating to fonts.   |  | | --- | | **Interfaces**  [**MultipleMaster**](https://docs.oracle.com/javase/8/docs/api/java/awt/font/MultipleMaster.html) - The MultipleMaster interface represents Type 1 Multiple Master fonts.  **OpenType** – Represents OpenType and trueTypee fonts. | | **Classes**  [**FontRenderContext**](https://docs.oracle.com/javase/8/docs/api/java/awt/font/FontRenderContext.html) - The FontRenderContext class is a container for the information needed to correctly measure text.  Glyph Justification info – represents information about the justification properties of a glyph.ss | | **Exceptions** |     **Description**  Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.   |  | | --- | | **Interfaces**  [**PathIterator**](https://docs.oracle.com/javase/8/docs/api/java/awt/geom/PathIterator.html) - The PathIterator interface provides the mechanism for objects that implement the [**Shape**](https://docs.oracle.com/javase/8/docs/api/java/awt/Shape.html) interface to return the geometry of their boundary by allowing a caller to retrieve the path of that boundary a segment at a time. | | **Classes**  [**AffineTransform**](https://docs.oracle.com/javase/8/docs/api/java/awt/geom/AffineTransform.html) - The AffineTransform class represents a 2D affine transform that performs a linear mapping from 2D coordinates to other 2D coordinates that preserves the "straightness" and "parallelness" of lines.  [**Arc2D.Double**](https://docs.oracle.com/javase/8/docs/api/java/awt/geom/Arc2D.Double.html) - This class defines an arc specified in double precision. | | **Exceptions**  [**IllegalPathStateException**](https://docs.oracle.com/javase/8/docs/api/java/awt/geom/IllegalPathStateException.html) - The IllegalPathStateException represents an exception that is thrown if an operation is performed on a path that is in an illegal state with respect to the particular operation being performed, such as appending a path segment to a **[GeneralPath](https://docs.oracle.com/javase/8/docs/api/java/awt/geom/GeneralPath.html" \o "class in java.awt.geom)** without an initial moveto. |   **Descriptions**  Provides classes and interfaces for the input method framework.   |  | | --- | | **Interfaces**  [**InputMethodRequests**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/InputMethodRequests.html) -InputMethodRequests defines the requests that a text editing component has to handle in order to work with input methods. | | **Classes**  [**InputContext**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/InputContext.html) - Provides methods to control text input facilities such as input methods and keyboard layouts.  [**InputMethodHighlight**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/InputMethodHighlight.html) - An InputMethodHighlight is used to describe the highlight attributes of text being composed. | | **Exceptions** |   **Descriptions**  Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.   |  | | --- | | **Interfaces**  [**InputMethod**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/spi/InputMethod.html) - Defines the interface for an input method that supports complex text input.  [**InputMethodDescriptor**](https://docs.oracle.com/javase/8/docs/api/java/awt/im/spi/InputMethodDescriptor.html) - Defines methods that provide sufficient information about an input method to enable selection and loading of that input method. | | **Classes** | | **Exceptions** |   **Descriptions**  Provides classes and interfaces for producing rendering-independent images.   |  | | --- | | **Interfaces**  [**ContextualRenderedImageFactory**](https://docs.oracle.com/javase/8/docs/api/java/awt/image/renderable/ContextualRenderedImageFactory.html)- ContextualRenderedImageFactory provides an interface for the functionality that may differ between instances of RenderableImageOp.  [**RenderedImageFactory**](https://docs.oracle.com/javase/8/docs/api/java/awt/image/renderable/RenderedImageFactory.html) - The RenderedImageFactory interface (often abbreviated RIF) is intended to be implemented by classes that wish to act as factories to produce different renderings, for example by executing a series of BufferedImageOps on a set of sources, depending on a specific set of parameters, properties, and rendering hints. | | **Classes**  [**ParameterBlock**](https://docs.oracle.com/javase/8/docs/api/java/awt/image/renderable/ParameterBlock.html) - A ParameterBlock encapsulates all the information about sources and parameters (Objects) required by a RenderableImageOp, or other classes that process images.  [**RenderableImageProducer**](https://docs.oracle.com/javase/8/docs/api/java/awt/image/renderable/RenderableImageProducer.html) - An adapter class that implements ImageProducer to allow the asynchronous production of a RenderableImage. | | **Exceptions** |   **Descriptions**  Provides classes and interfaces for a general printing API.   |  | | --- | | **Interfaces**  [**Pageable**](https://docs.oracle.com/javase/8/docs/api/java/awt/print/Pageable.html) - The Pageable implementation represents a set of pages to be printed.  [**PrinterGraphics**](https://docs.oracle.com/javase/8/docs/api/java/awt/print/PrinterGraphics.html) - The PrinterGraphics interface is implemented by [**Graphics**](https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html) objects that are passed to [**Printable**](https://docs.oracle.com/javase/8/docs/api/java/awt/print/Printable.html) objects to render a page. | | **Classes**  **Book -** The Book class provides a representation of a document in which pages may have different page formats and page painters.  Paper - The Paper class describes the physical characteristics of a piece of paper. | | **Exceptions**  [**PrinterAbortException**](https://docs.oracle.com/javase/8/docs/api/java/awt/print/PrinterAbortException.html) - The PrinterAbortException class is a subclass of **[PrinterException](https://docs.oracle.com/javase/8/docs/api/java/awt/print/PrinterException.html" \o "class in java.awt.print)** and is used to indicate that a user or application has terminated the print job while it was in the process of printing.  [**PrinterIOException**](https://docs.oracle.com/javase/8/docs/api/java/awt/print/PrinterIOException.html) - The PrinterIOException class is a subclass of **[PrinterException](https://docs.oracle.com/javase/8/docs/api/java/awt/print/PrinterException.html" \o "class in java.awt.print)** and is used to indicate that an IO error of some sort has occurred while printing. |   **Descriptions**  Contains classes related to developing beans -- components based on the JavaBeans™ architecture.   |  | | --- | | **Interfaces**  [**AppletInitializer**](https://docs.oracle.com/javase/8/docs/api/java/beans/AppletInitializer.html) -This interface is designed to work in collusion with java.beans.Beans.instantiate.  [**Customizer**](https://docs.oracle.com/javase/8/docs/api/java/beans/Customizer.html) - A customizer class provides a complete custom GUI for customizing a target Java Bean. | | **Classes**  [**BeanDescriptor**](https://docs.oracle.com/javase/8/docs/api/java/beans/BeanDescriptor.html) - A Bean Descriptor provides global information about a "bean", including its Java class, its displayName, etc.  [**DefaultPersistenceDelegate**](https://docs.oracle.com/javase/8/docs/api/java/beans/DefaultPersistenceDelegate.html) - The DefaultPersistenceDelegate is a concrete implementation of the abstract PersistenceDelegate class and is the delegate used by default for classes about which no information is available. | | **Exceptions**  [**IntrospectionException**](https://docs.oracle.com/javase/8/docs/api/java/beans/IntrospectionException.html) - Thrown when an exception happens during Introspection.  [**PropertyVetoException**](https://docs.oracle.com/javase/8/docs/api/java/beans/PropertyVetoException.html) - A PropertyVetoException is thrown when a proposed change to a property represents an unacceptable value. |     **Descriptions**  Provides classes and interfaces relating to bean context.   |  | | --- | | **Interfaces**  [**BeanContext**](https://docs.oracle.com/javase/8/docs/api/java/beans/beancontext/BeanContext.html) - The BeanContext acts a logical hierarchical container for JavaBeans.  [**BeanContextChildComponentProxy**](https://docs.oracle.com/javase/8/docs/api/java/beans/beancontext/BeanContextChildComponentProxy.html) - This interface is implemented by BeanContextChildren that have an AWT Component associated with them. | | **Classes**  [**BeanContextChildSupport**](https://docs.oracle.com/javase/8/docs/api/java/beans/beancontext/BeanContextChildSupport.html) - This is a general support class to provide support for implementing the BeanContextChild protocol.  [**BeanContextMembershipEvent**](https://docs.oracle.com/javase/8/docs/api/java/beans/beancontext/BeanContextMembershipEvent.html) - A BeanContextMembershipEvent encapsulates the list of children added to, or removed from, the membership of a particular BeanContext. | | **Exceptions** |   **Descriptions**  Provides for system input and output through data streams, serialization and the file system.   |  | | --- | | **Interfaces**  [**Closeable**](https://docs.oracle.com/javase/8/docs/api/java/io/Closeable.html) - A Closeable is a source or destination of data that can be closed.  [**DataOutput**](https://docs.oracle.com/javase/8/docs/api/java/io/DataOutput.html) - The DataOutput interface provides for converting data from any of the Java primitive types to a series of bytes and writing these bytes to a binary stream. | | **Classes**  [**BufferedInputStream**](https://docs.oracle.com/javase/8/docs/api/java/io/BufferedInputStream.html) - A BufferedInputStream adds functionality to another input stream-namely, the ability to buffer the input and to support the mark and reset methods.  [**BufferedReader**](https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html) - Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines. | | **Exceptions**  [**CharConversionException**](https://docs.oracle.com/javase/8/docs/api/java/io/CharConversionException.html)- Base class for character conversion exceptions.  [**FileNotFoundException**](https://docs.oracle.com/javase/8/docs/api/java/io/FileNotFoundException.html) - Signals that an attempt to open the file denoted by a specified pathname has failed. |   **Descriptions**  Provides classes that are fundamental to the design of the Java programming language.   |  | | --- | | **Interfaces**  [**Appendable**](https://docs.oracle.com/javase/8/docs/api/java/lang/Appendable.html) - An object to which char sequences and values can be appended.  [**CharSequence**](https://docs.oracle.com/javase/8/docs/api/java/lang/CharSequence.html) - A CharSequence is a readable sequence of char values. | | **Classes**  [**Boolean**](https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html) - The Boolean class wraps a value of the primitive type boolean in an object.  [**Character**](https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html) - The Character class wraps a value of the primitive type char in an object. | | **Exceptions**  [**ArithmeticException**](https://docs.oracle.com/javase/8/docs/api/java/lang/ArithmeticException.html) - Thrown when an exceptional arithmetic condition has occurred.  [**ArrayStoreException**](https://docs.oracle.com/javase/8/docs/api/java/lang/ArrayStoreException.html) - Thrown to indicate that an attempt has been made to store the wrong type of object into an array of objects. |   **Descriptions**  Provides library support for the Java programming language annotation facility.   |  | | --- | | **Interfaces**  [**Annotation**](https://docs.oracle.com/javase/8/docs/api/java/lang/annotation/Annotation.html) - The common interface extended by all annotation types. | | **Classes** | | **Exceptions**  [**AnnotationTypeMismatchException**](https://docs.oracle.com/javase/8/docs/api/java/lang/annotation/AnnotationTypeMismatchException.html) - Thrown to indicate that a program has attempted to access an element of an annotation whose type has changed after the annotation was compiled (or serialized).  [**IncompleteAnnotationException**](https://docs.oracle.com/javase/8/docs/api/java/lang/annotation/IncompleteAnnotationException.html) - Thrown to indicate that a program has attempted to access an element of an annotation type that was added to the annotation type definition after the annotation was compiled (or serialized). |   **Descriptions**  Provides services that allow Java programming language agents to instrument programs running on the JVM.   |  | | --- | | **Interfaces**  [**ClassFileTransformer**](https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/ClassFileTransformer.html) - An agent provides an implementation of this interface in order to transform class files.  **Instrumentation – this class provides services needed to instrument java programming language code.** | | **Classes**  [**ClassDefinition**](https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/ClassDefinition.html) - This class serves as a parameter block to the Instrumentation.redefineClasses method. | | **Exceptions**  [**IllegalClassFormatException**](https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/IllegalClassFormatException.html) - Thrown by an implementation of **[ClassFileTransformer.transform](https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/ClassFileTransformer.html" \l "transform-java.lang.ClassLoader-java.lang.String-java.lang.Class-java.security.ProtectionDomain-byte:A-)** when its input parameters are invalid.  unmodifiableClassException – thrown by an implementation of instrumentation.redefineClasses when one of the specified classes cannot be modified |     **Descriptions**  The java.lang.invoke package contains dynamic language support provided directly by the Java core class libraries and virtual machine.   |  | | --- | | **Interfaces**  [**Method Handle Info**](https://docs.oracle.com/javase/8/docs/api/java/lang/invoke/MethodHandleInfo.html) - A symbolic reference obtained by cracking a direct method handle into its constituents symbolic parts. | | **Classes**  [**Call Site**](https://docs.oracle.com/javase/8/docs/api/java/lang/invoke/CallSite.html) - A Call Site is a holder for a variable [**Method Handle**](https://docs.oracle.com/javase/8/docs/api/java/lang/invoke/MethodHandle.html), which is called its target.  [**Lambda Metafactory**](https://docs.oracle.com/javase/8/docs/api/java/lang/invoke/LambdaMetafactory.html) - Methods to facilitate the creation of simple "function objects" that implement one or more interfaces by delegation to a provided **[MethodHandle](https://docs.oracle.com/javase/8/docs/api/java/lang/invoke/MethodHandle.html" \o "class in java.lang.invoke)**, possibly after type adaptation and partial evaluation of arguments. | | **Exceptions**  [**Lambda Conversion Exception**](https://docs.oracle.com/javase/8/docs/api/java/lang/invoke/LambdaConversionException.html) – Lambda Conversion Exception  Wrong Method Type Exception – Thrown to indicate that code has attempted to call a method handle via wrong method type. |   **Descriptions**  Provides classes and interfaces for obtaining reflective information about classes and objects.   |  | | --- | | **Interfaces**  [**Annotated Array Type**](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/AnnotatedArrayType.html) – Annotated Array Type represents the potentially annotated use of an array type, whose component type may itself represent the annotated use of a type.  [**Annotated Parameterized Type**](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/AnnotatedParameterizedType.html) – Annotated Parameterized Type represents the potentially annotated use of a parameterized type, whose type arguments may themselves represent annotated uses of types. | | **Classes**  [**Accessible Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/AccessibleObject.html) - The Accessible Object class is the base class for Field, Method and Constructor objects.  Modifier -The Modifier class provides static methods and constants to decode class and member access modifiers. | | **Exceptions**  [**Invocation Target Exception**](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/InvocationTargetException.html) – Invocation Target Exception is a checked exception that wraps an exception thrown by an invoked method or constructor.  [**Malformed Parameters Exception**](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/MalformedParametersException.html) - Thrown when [**the java.lang.reflect package**](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Executable.html#getParameters--) attempts to read method parameters from a class file and determines that one or more parameters are malformed. |   **Descriptions**  Provides the classes for implementing networking applications.   |  | | --- | | **Interfaces**  [**Content Handler Factory**](https://docs.oracle.com/javase/8/docs/api/java/net/ContentHandlerFactory.html) - This interface defines a factory for content handlers.  Cookie store -A Cookie Store object represents a storage for cookie. | | **Classes**  **Authenticator -** The class Authenticator represents an object that knows how to obtain authentication for a network connection.  **Cache Response -** Represent channels for retrieving resources from the ResponseCache. | | **Exceptions**  **Bind Exception -** Signals that an error occurred while attempting to bind a socket to a local address and port.  [**Http Retry Exception**](https://docs.oracle.com/javase/8/docs/api/java/net/HttpRetryException.html) - Thrown to indicate that a HTTP request needs to be retried but cannot be retried automatically, due to streaming mode being enabled. |   **Descriptions**  Defines interfaces and classes for the Java virtual machine to access files, file attributes, and file systems.   |  | | --- | | **Interfaces**  **Copy Option -** An object that configures how to copy or move a file.  [**DirectoryStream.Filter**](https://docs.oracle.com/javase/8/docs/api/java/nio/file/DirectoryStream.Filter.html)<T> - An interface that is implemented by objects that decide if a directory entry should be accepted or filtered. | | **Classes**  **Files -** This class consists exclusively of static methods that operate on files, directories, or other types of files.  File System - Provides an interface to a file system and is the factory for objects to access files and other objects in the file system. | | **Exceptions**  [**Access Denied Exception**](https://docs.oracle.com/javase/8/docs/api/java/nio/file/AccessDeniedException.html) - Checked exception thrown when a file system operation is denied, typically due to a file permission or other access check.  [**Closed Watch Service Exception**](https://docs.oracle.com/javase/8/docs/api/java/nio/file/ClosedWatchServiceException.html) - Unchecked exception thrown when an attempt is made to invoke an operation on a watch service that is closed. |   **Descriptions**  Provides the RMI package.   |  | | --- | | **Interfaces**  **Remote -** The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. | | **Classes**  [**MarshalledObject**](https://docs.oracle.com/javase/8/docs/api/java/rmi/MarshalledObject.html)<T> - A MarshalledObject contains a byte stream with the serialized representation of an object given to its constructor.  [**RMISecurityManager**](https://docs.oracle.com/javase/8/docs/api/java/rmi/RMISecurityManager.html) – Deprecated. *Use*[***Security Manager***](https://docs.oracle.com/javase/8/docs/api/java/lang/SecurityManager.html)*instead.* | | **Exceptions**  **Access Exception -**An AccessException is thrown by certain methods of the java.rmi.Naming class (specifically bind, rebind, and unbind) and methods of the java.rmi.activation.ActivationSystem interface to indicate that the caller does not have permission to perform the action requested by the method call.  Connect exception - A ConnectException is thrown if a connection is refused to the remote host for a remote method call. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **C. EXPLAIN: Reading**  To understand the module activities, read and practice the reading materials on the internet, search it in google to explore more information about the usage and features of java API specifications in different versions. | | | | |
| **D. ELABORATE: Additional Details**  More in java API Specifications:  Give the Requirements for Writing a Java API Specifications.  This document describes the requirements for writing API specifications for the Java platform. The specification for each Java TMplatform API library is made up of its Javadoc comments and additional support documentation called out in the doc comments.  This document has five sections that correspond to the sections of an API specification.   * Top-level specification, Package specification, Class/interface specification, Field and Method specification. | | | | |
| **E.EVALUATE**  **Self-Assessment.**   * Kindly check (✔) the box of your answer for each question. In this way, we will be able to assess how much we have learned and what are the things that needs to be improved. | | | | |
| **Questions** | **YES** | **NO** | **MAYB E** |  |
| **1. Did I work hard on this module?** |  |  |  |
| **2. Did I understand what my teacher asked me to do?** |  |  |  |
| **3. Did I spend enough time to finish answering this module?** |  |  |  |
| **4. Did I make good use of available resources?** |  |  |  |
| **5. Did I check/ review my work for possible errors?** |  |  |  |
| **6. Did I learn something in this module?** |  |  |  |
| **7. Did I ask questions if I needed help?** |  |  |  |
| **8. Did I read the instructions carefully?** |  |  |  |
| **9. Did I set high standards for myself?** |  |  |  |
| **10. Did I meet the success criteria?** |  |  |  |