**TABUCON, MARY GRACE P. BSIT 2-3**

**V. TEACHING-LEARNING ACTIVITIES**

**A. ENGAGE**: Misconception Commit your answer/document on your remote repository that was shared to your instructor github account.

**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.

1. **java 14 features**

Pattern Matching for instance of (Preview), Packaging Tool (Incubator), NUMA-Aware Memory Allocation for G1, JFR Event Streaming, Non-Volatile Mapped Byte Buffers, Helpful Null Pointer Exceptions.

**ADVANTAGES:**

You can use any other features.

**DISADVANTAGES:**

You can have little trouble if want to run your programs from old version of java. Sometimes when you execute your code it doesn’t execute the updated code but the code from last execution.

1. **Java 1.7 feature**

Introduces many language enhancements: Integral Types as Binary Literals.

**ADVANTAGES:**

Easy to run if you change the code of your program, it doesn’t need to build again and again to your code to take actions.

**DISADVANTAGES:**

Little less features and capabilities compare to JDK 14.

**B. EXPLORE: API Specifications list**

1. Enumerate at least 20 following API specifications of java, depending of the java version.

**API specifications of java standard edition 7**

1. [**java.awt.color**](https://docs.oracle.com/javase/7/docs/api/java/awt/color/package-summary.html) - Provides classes for color spaces.

**Class**: [ColorSpace](https://docs.oracle.com/javase/8/docs/api/java/awt/color/ColorSpace.html) -  
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.

**Exception**: [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.

1. [**java.awt.datatransfer**](https://docs.oracle.com/javase/7/docs/api/java/awt/datatransfer/package-summary.html) - Provides interfaces and classes for transferring data between and within applications.

**Interface:**

**Class:**

**Exception:**

1. [**java.awt.dnd**](https://docs.oracle.com/javase/7/docs/api/java/awt/dnd/package-summary.html) - 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.

**Interface:** [Autoscroll](https://docs.oracle.com/javase/7/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.

**Class:** [DnDConstants](https://docs.oracle.com/javase/7/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.

**Exception**: [InvalidDnDOperationException](https://docs.oracle.com/javase/7/docs/api/java/awt/dnd/InvalidDnDOperationException.html) - This exception is thrown by various methods in the java.awt.dnd package.

1. [**java.awt.geom**](https://docs.oracle.com/javase/7/docs/api/java/awt/geom/package-summary.html) - Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.

**Interface:** [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.

**Class:** [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.

**Exception**: [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.

1. [**java.awt.im.spi**](https://docs.oracle.com/javase/7/docs/api/java/awt/im/spi/package-summary.html)- Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.

**Interface:** [InputMethod](https://docs.oracle.com/javase/7/docs/api/java/awt/im/spi/InputMethod.html) – Defines the interface for an input method that supports complex text input.

**Class:**

**Exception:**

1. [**java.awt.image.renderable**](https://docs.oracle.com/javase/7/docs/api/java/awt/image/renderable/package-summary.html) - Provides classes and interfaces for producing rendering-independent images.

**Interface:** [ContextualRenderedImageFactory](https://javaalmanac.io/jdk/1.2/api/java/awt/image/renderable/ContextualRenderedImageFactory.html) - ContextualRenderedImageFactory provides an interface for the functionality that may differ between instances of RenderableImageOp.

**Class:** [ParameterBlock](https://javaalmanac.io/jdk/1.2/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.

1. [**java.awt.print**](https://docs.oracle.com/javase/7/docs/api/java/awt/print/package-summary.html) - Provides classes and interfaces for a general printing API.

**Interface**: [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.

**Class:** [Book](https://docs.oracle.com/javase/8/docs/api/java/awt/print/Book.html) - The Book class provides a representation of a document in which pages may have different page formats and page painters.

**Exception:** [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.

1. [**java.lang.management**](https://docs.oracle.com/javase/7/docs/api/java/lang/management/package-summary.html)- Provides the management interfaces for monitoring and management of the Java virtual machine and other components in the Java runtime.

**Interface**: [java.lang.management](https://docs.oracle.com/javase/7/docs/api/java/lang/management/package-summary.html) - The management interface for a buffer pool, for example a pool of [direct](https://docs.oracle.com/javase/7/docs/api/java/nio/ByteBuffer.html#allocateDirect(int)) or [mapped](https://docs.oracle.com/javase/7/docs/api/java/nio/MappedByteBuffer.html) buffers.

**Class:** [LockInfo](https://docs.oracle.com/javase/7/docs/api/java/lang/management/LockInfo.html) - Information about a lock.

1. [**java.lang.ref**](https://docs.oracle.com/javase/7/docs/api/java/lang/ref/package-summary.html) - Provides reference-object classes, which support a limited degree of interaction with the garbage collector.

**Class:** [PhantomReference](https://docs.oracle.com/javase/7/docs/api/java/lang/ref/PhantomReference.html)<T> - Phantom reference objects, which are enqueued after the collector determines that their referents may otherwise be reclaimed.

1. [**java.nio**](https://docs.oracle.com/javase/7/docs/api/java/nio/package-summary.html) - Defines buffers, which are containers for data, and provides an overview of the other NIO packages.

**API specifications of java STANDARD EDITION 8**

1. **Package org.omg.Dynamic** – this package contain the dynamic module specified in the OMG Portable interceptor specification, http://cgi.omg.org/cgi-bin /doc?ptc/2000-08-06,section 21.9.

**Interface**:

**Class:** Parameter – org/omg/Dynamic/Parameter.java .

**Exception:**

1. **Package org.ietf.jgss** – This package presents a framework that allows application developers to make use of security services like authentication, data integrity and data confidentially from a variety of underlying security mechanisms like Kerberos, using a unified API.

**Interface: GSSContext**- This interface encapsulates the GSS-API security context and provides the security services that are available over the context.

**GSSCredential**- This interface encapsulates the GSS-API credentials for an empty.

**GSSName**- this interface encapsulates a single GSS-API principal entity.

**Class**: **ChannelBinding**- This class encapsulates the concept of call-provided channel binding information.

**GSSManager**- this class serve as a factory for other important GSS-API classes and also provides information about the mechanisms that are supported.

**MessageProp** – this is a utility class used within the per-message GSSContext methods to convey per-message properties.

**Oid** – this class represents Universal Object Identifies (Oids) and their associated operations.

**Exception**: **GSSException** – this exception is thrown whenever a GSS-API error occurs, including any mechanism specific error.

1. **Package javax.rml.ssl** – provides implementations of RMIClientSocketFactory and RMIServerSocketFactoy over the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols.

**Interface:**

**Class: SsLMIClientSocketFactory** – An SsLRMICLientSocketFactory instance is used by the RMI runtime in order to obtain client sockets for RMI calls via SSL.

**SsLRMIServerSocketFactory** – an SsLRMIServerSocketFactory instance is used by the RMI runtime in order to obtain sever sockets for RMI calls via SSL.

**Exception:**

1. **Package java.rml.dgc**- provides classes and interface for RMI distributed garbage-collection (DGC).

**Interface: Dgc** - the dgc abstraction is used for the server side of the distributed garbage collection algorithm.

**Class: Lease**- a lease contains a unique VM identifier and a lease duration.

**VMID** – A VMID is a identifier that is unique across all java virtual machines.

**Exception**:

1. **Package javax.rmi** – contains user APIs for RMI-IIOP.

**Interface:**

**Class: PortableRemoteObject** – Server implementation objects may either inherit from javax.rmi.PortableRemoteObject or they may implement a remote interface and then use the exportObject method to register themselves as a server object.

**Exception:**

1. **Package javax.xml.datatype** – XML/Java Type Mappings.

**Interface:**

**Class: DataattypeConstants** – utility class to contain basic Datatype values as constants.

**DatatypeConstants.Field**- Type-safe enum class that represents six fields of the Duration class.

**DatatypeFactory** – Factory that creates new javax.xml.datatype Objects that map XML to/from java Objects.

**Duration –** Immutable representation of a time span as defined in the W3C XML Schema 1.0 specification.

**XMLGregorianCalendar** – Representation for W3C Schema 1.0 date/time datatypes.

**Exception:**

**DatatypeConfigurationException** – Indicates a serious configuration error.

1. **Package javax.imageio.spi** – a package of the java Image I/O API containing the plug-in interfaces for reader, writers, transcoders, and streams, and a runtime registry.

**Interface: RegisterableService** – An optional interface that may be provided by service provider objects that will be registered with a ServiceRigestry.

**Classes**: **IIORegistry**- A registry for service provider instances.

**IIOServiceProvider** – a super interface for functionality common to all image I/O service provider interfaces (SPIs).

**ImageInputStreamSpi** – The service provider interface (SPI) for ImageInputStreams.

**ImageOutputStreamSpi** – The service provider interface (SPI) for ImageOutputStreams.

**ImageReaderSpi** – The service provider interface (SPI) for ImageReaders.

**ImageReaderWriterSpi** – A super class containing instance variables and methods common to ImageReaderSpi and ImageWriterSpi.

**ImageTranscodeSpi** – The service provider interface (SPI) for ImageTranscoders.

**ImagewriterSpi** - The service provider interface (SPI) for ImageWriters.

**ServiceRegistry** – A registry for service provider instances.

**Exception**:

1. **tpPackage javax.xml.ws.ht** – This package defines APIs specific to the HTTP binding.

**Interface: HTTPBinding** – The HTTPBinding interface is an abstraction for the XML/HTTP binding.

**Class:**

**Exception: HTTPException** – the HTTPException exception represents a XML/HTTP fault.

1. **Package javax.xml.bind.util** – Useful client utility classes.

**Interface:**

**Class: JAXBResult** – JAXP Result implementation that unmarshals a JAXB object.

**JAXBSource** – JAXP Source implementation that unmarshals a JAXB-generated object.

**ValidationEventCollecto**r – ValidationEventHandler implementation that collects all events.

**Exception:**

1. **Package javax.ne**t – Provides classes for networking applications.

**Interface:**

**Class: ServerSocketFactory –** This class creates server sockets.

**SocketFactory –** This class creates sockets.

**Exception:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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** | | | |
| **Questions** | **YES** | **NO** | **MAYBE** |
| **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?** |  |  | **✔** |