**CASUYON, VLADIMIR S. BSIT 2-2**

**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:**

The programmer/user can use new or other available features.

**DISADVANTAGES:**

The user/programmer will have a trouble in running if they want to run the program from the old version.

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.

**JAVA version: Java™ Platform, Standard Edition 7**

**Packages and Description**

1. [**java.awt**](https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html) - Contains all of the classes for creating user interfaces and for painting graphics and images.

**Interfaces** - [ActiveEvent](https://docs.oracle.com/javase/7/docs/api/java/awt/ActiveEvent.html) - An interface for events that know how to dispatch themselves.

**Classes** - [AlphaComposite](https://docs.oracle.com/javase/7/docs/api/java/awt/AlphaComposite.html) - The AlphaComposite class implements basic alpha compositing rules for combining source and destination colors to achieve blending and transparency effects with graphics and images.

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.
2. [**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.event**](https://docs.oracle.com/javase/7/docs/api/java/awt/event/package-summary.html) - Provides interfaces and classes for dealing with different types of events fired by AWT components.

**Packages:** [java.applet](https://docs.oracle.com/javase/7/docs/api/java/applet/package-summary.html) - Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.

1. [**java.awt.font**](https://docs.oracle.com/javase/7/docs/api/java/awt/font/package-summary.html) - Provides classes and interface relating to fonts.

**Modifier and Type**: static int - [BOLD](https://docs.oracle.com/javase/7/docs/api/java/awt/Font.html#BOLD) The bold style constant.

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.

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.beans**](https://docs.oracle.com/javase/7/docs/api/java/beans/package-summary.html) - Contains classes related to developing beans -- components based on the JavaBeans™ architecture.
2. [j**ava.beans.beancontext**](https://docs.oracle.com/javase/7/docs/api/java/beans/beancontext/package-summary.html) - Provides classes and interfaces relating to bean context.
3. [**java.io**](https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html) - Provides for system input and output through data streams, serialization and the file system.

**Interface:** [Closeable](https://docs.oracle.com/javase/9/docs/api/java/io/Closeable.html) - A Closeable is a source or destination of data that can be closed.

**Class:** [BufferedInputStream](https://docs.oracle.com/javase/9/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.

**Exception:** [CharConversionException](https://docs.oracle.com/javase/9/docs/api/java/io/CharConversionException.html) - Base class for character conversion exceptions.

1. [**java.lang**](https://docs.oracle.com/javase/7/docs/api/java/lang/package-summary.html) - Provides classes that are fundamental t the design of the Java programming language.
2. [**java.lang.annotation**](https://docs.oracle.com/javase/7/docs/api/java/lang/annotation/package-summary.html)- Provides library support for the Java programming language annotation facility.
3. [**java.lang.invoke**](https://docs.oracle.com/javase/7/docs/api/java/lang/invoke/package-summary.html) - The java.lang.invoke package contains dynamic language support provided directly by the Java core class libraries and virtual machine.

**Class:** [CallSite](https://docs.oracle.com/javase/7/docs/api/java/lang/invoke/CallSite.html) - A CallSite is a holder for a variable [MethodHandle](https://docs.oracle.com/javase/7/docs/api/java/lang/invoke/MethodHandle.html" \o "class in java.lang.invoke), which is called its target.

**Exception:** [WrongMethodTypeException](https://docs.oracle.com/javase/7/docs/api/java/lang/invoke/WrongMethodTypeException.html) - Thrown to indicate that code has attempted to call a method handle via the wrong method type.

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.
2. [**java.nio.channels.spi**](https://docs.oracle.com/javase/7/docs/api/java/nio/channels/spi/package-summary.html) **-** Service-provider classes for the [java.nio.channels](https://docs.oracle.com/javase/7/docs/api/java/nio/channels/package-summary.html) package.

|  |  |  |  |
| --- | --- | --- | --- |
| 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?** |  |  | **✔** |