CIS 36B – 15th Class / Lab Assignment – **10 Points-**

Student Name ______________________ **Student ID** ______________________ **Point Total**

https://docs.oracle.com/javase/tutorial/uiswing/components/frame.html

Section 1: Define / Answer

API- (**Application Programming Interface**) An **API** is an Interface which is used for accessing an application or a service from a program. An **API**makes it possible to use programs from within programs, therefore it is the foundation for modular systems with clearly **defined** Interfaces between separate components.

GUI- graphical (rather than purely textual) user interface to a computer. As you read this, you are looking at the GUI or graphical user interface of your particular Web browser. The term came into existence because the first interactive user interfaces to computers were not graphical; they were text-and-keyboard oriented and usually consisted of commands you had to remember and computer responses that were infamously brief. The command interface of the DOS operating system (which you can still get to from your Windows operating system) is an example of the typical user-computer interface before GUIs arrived. An intermediate step in user interfaces between the command line interface and the GUI was the non-graphical *menu-based interface*, which let you interact by using a mouse rather than by having to type in keyboard commands.

Component Classes-is the abstract superclass of the nonmenu-related Abstract Window Toolkit components.

Class `Component` can also be extended directly to create a lightweight component. A lightweight component is a component that is not associated with a native window. On the contrary, a heavyweight component is associated with a native window. The `isLightweight()` method may be used to distinguish between the two kinds of the components.

Container Classes-A generic Abstract Window Toolkit(AWT) container object is a component that can contain other AWT components.

Components added to a container are tracked in a list. The order of the list will define the components' front-to-back stacking order within the container. If no index is specified when adding a component to a container, it will be added to the end of the list (and hence to the bottom of the stacking order).

Helper Classes-

is used to assist in providing some functionality, which isn't the main goal of the application or class in which it is used. An instance of a helper class is called a **helper object** (for example, in the delegation pattern).

Helper classes are often created in introductory programming lessons, after the novice programmer has moved beyond creating one or two classes.

CIS 36B – 15th Class / Lab Assignment – **10 Points-**
Student Name______________________________ **Student ID**______________________________ **Point Total**

A utility class is a special case of a helper class in which the methods are all static. In general, helper classes do not have to have all static methods, and may have instance variables and multiple instances of the helper class may exist.

Using helper classes can be done in multiple ways:

- Instantiating them directly (as above)
- via dependency injection
- by making their methods `static` and accessing them in a static way, like `IOUtils.closeQuietly(inputStream)` closes an `InputStream` wihtout throwing exceptions.
- at least my convention is to name classes with only static methods and not dependencies `XUtils`, and classees that in turn have dependencies / need to be managed by a DI container `XHelper`

java.awt.Container

A generic Abstract Window Toolkit(AWT) container object is a component that can contain other AWT components.

Components added to a container are tracked in a list. The order of the list will define the components' front-to-back stacking order within the container. If no index is specified when adding a component to a container, it will be added to the end of the list (and hence to the bottom of the stacking order).

javax.swing.JFrame

An extended version of `java.awt.Frame` that adds support for the JFC/Swing component architecture. You can find task-oriented documentation about using `JFrame` in *The Java Tutorial*, in the section How to Make Frames.

The `JFrame` class is slightly incompatible with `Frame`. Like all other JFC/Swing top-level containers, a `JFrame` contains a `JRootPane` as its only child. The **content pane** provided by the root pane should, as a rule, contain all the non-menu components displayed by the `JFrame`. This is different from the AWT `Frame` case. As a conveniance `add` and its variants, `remove` and `setLayout` have been overridden to forward to the `contentPane` as necessary.

javax.swing.JPanel

`JPanel` is a generic lightweight container. For examples and task-oriented documentation for JPanel, see How to Use Panels, a section in *The Java Tutorial*.

javax.swing.JApplet

CIS 36B – 15th Class / Lab Assignment – **10 Points-**
Student Name____________________ **Student ID**____________________ **Point Total**

javax.swing.JDialog

An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application.

The `Applet` class must be the superclass of any applet that is to be embedded in a Web page or viewed by the Java Applet Viewer. The `Applet` class provides a standard interface between applets and their environment.

java.awt.Graphics

The `Graphics` class is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images.

A `Graphics` object encapsulates state information needed for the basic rendering operations that Java supports. This state information includes the following properties:

java.awt.Color

The `Color` class is used to encapsulate colors in the default sRGB color space or colors in arbitrary color spaces identified by a `ColorSpace`. Every color has an implicit alpha value of 1.0 or an explicit one provided in the constructor. The alpha value defines the transparency of a color and can be represented by a float value in the range 0.0 - 1.0 or 0 - 255. An alpha value of 1.0 or 255 means that the color is completely opaque and an alpha value of 0 or 0.0 means that the color is completely transparent. When constructing a `Color` with an explicit alpha or getting the color/alpha components of a `Color`, the color components are never premultiplied by the alpha component.

java.awt.Font

The `Font` class represents fonts, which are used to render text in a visible way. A font provides the information needed to map sequences of *characters* to sequences of *glyphs* and to render sequences of glyphs on `Graphics` and `Component` objects.

java.awt.FontMetrics

The `FontMetrics` class defines a font metrics object, which encapsulates information about the rendering of a particular font on a particular screen.

java.awt.Dimension

The `Dimension` class encapsulates the width and height of a component (in integer precision) in a single object. The class is associated with certain properties of components. Several methods defined by the `Component` class and the `LayoutManager` interface return a `Dimension` object.

CIS 36B – 15th Class / Lab Assignment – **10 Points-**
Student Name ______________________________ **Student ID** ______________________________ **Point Total**

Normally the values of `width` and `height` are non-negative integers. The constructors that allow you to create a dimension do not prevent you from setting a negative value for these properties. If the value of `width` or `height` is negative, the behavior of some methods defined by other objects is undefined.

java.awt.LayoutManager

Defines the interface for classes that know how to lay out `Container`s.

Swing's painting architecture assumes the children of a `JComponent` do not overlap. If a `JComponent`'s `LayoutManager` allows children to overlap, the `JComponent` must override `isOptimizedDrawingEnabled` to return false.

Layout Managers-

A layout manager is an object that controls the size and the position of components in a container. Every `Container` object has a `LayoutManager` object that controls its layout.

CIS 36B – 15th Class / Lab Assignment – **10 Points-**

Student Name______________________________ **Student ID**____________________________ **Point Total**

Task 1:

NO PROGRAMMING TASK.

COME UP WITH PROJECT IDEA. COMPLETE ALL PREVIOUS ASSIGNMENTS AND PROGRAMMING TASKS.