**Práctica # 14 Anotaciones**

Nombre: Gian Molina Raigoza Matrícula: 1636155 Calificación:\_\_\_\_\_\_ Ponderación: 8% Día: 14/05/2019 Hora: \_\_\_\_\_\_:\_\_\_\_\_\_

**ObjetivosGenerales:**

* Implementar anotaciones ya definidas por java como @Deprecated.
* Utilizar las anotaciones que se proporcionan en los comentarios.

**Actividad #1 – Conceptos de Anotaciones en Java**

**Ponderación:** 20%

Responda a las siguientes preguntas:

1. Defina el concepto de anotación bajo el contexto de la programación en Java.

**Es una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución. También las anotaciones pueden añadirse a los elementos de programa tales como clases, métodos, metadatos, campos, parámetros, variables locales, y paquetes.**

1. ¿Cuales son las anotaciones que se añaden automaticamente a cada clase en su IDE netbeans?
   1. **@author-----Se añade el nombre del autor**
   2. **@param------Provee parámetros para los recursos incrustados con el elemento object**
2. ¿Qué es una anotación personalizada y para que podría usar una?

**Son metadatos que se pueden asociar a clases, miembros, métodos o parámetros para una mejor comprensión de un codigo.**

**Nos dan información sobre el elemento que tiene la anotación y permiten definir cómo queremos que sea tratado por el framework de ejecución.**

1. En sus palabras, ¿por qué cree que seria importante utilizar anotaciones en sus desarrollos de Java?

**Son importantes porque mediante ellas se puede comprender mejor un codigo que ha sido realizado por alguien y saber quién es el dicho autor y desde cuando fue hecho para futuras actualizaciones y comprender la lógica y el funcionamiento del mismo si algún día requiere mantenimiento o una mejora.**

**Actividad #2 – Anotaciones en Java**

**Ponderación:** 80%

Realice un programa que administre la ganancia de una máquina en una zona industrial. Utilice las anotaciones que proporciona Java.

**Procedimiento**

1. Cree las siguientes clases:

|  |
| --- |
| **Máquina** |
| * String nombre * double precioUnitario * int unidadesProducidad * double costoEmpleado * List<Empleado> empleados |
| + Maquina(String nombre, double precioUnitario, double costoEmpleado)  + int calcularCantidadEmpleados() @deprecated  + int calcularCantidadEmpleadosMaquina()  + double calcularGanancia()  + void agregarEmpleado(Empleado e)  + void imprimir()  + Getter y Setters |

|  |
| --- |
| **Empleado** |
| * String nombre * double sueldo |
| + Empleado(String nombre, double sueldo)  + Getter y Setters |

1. En la clase Maquina:
   * El método calcularCantidadEmpleados() estará @Deprecated y calculará el tamaño del ArrayList de empleados mediante un for y un contador. Añada una anotación en los comentarios del método: @deprecated.
   * El método calcularCantidadEmpleadosMaquina() calculará el tamaño del ArrayList de empleados utilizando solamente la función size().
   * El método calcularGanancia() calculará la ganancia total de la maquina que será la cantidad de unidades producidas por el precio unitario menos el costo de todos los trabajadores en ella.
   * El método agregarEmpleado(Empleado e) acepta como parámetro un empleado y agregara este a la lista de empleados del objeto.
   * El método imprimir mostrara todos los atributos de la maquina y los atributos de todos sus empleados.
2. Añada las anotaciones siguientes en comentarios según apliquen. Agréguelas en las clases y métodos que creó, así como en clases y métodos futuros.
   * @author
   * @param
   * @return
   * @since
   * @see
   * @version
3. Cree una clase que tenga un método main, y declare una Maquina.
4. Pedir al usuario los datos de la maquina.
5. Agregue un menú en el main que ejecutará las siguientes acciones:
6. Agregar Empleado a Maquina.
7. Imprimir Maquina.
8. Salir.