Metodologías de Diseño y Programación

Tarea 1

Profesor: Alexandre Bergel
Auxiliar: Juan Pablo Sandoval A.
Ayudantes: Alejandro Infante , Daniel Notario

September 9, 2013

1 iClock

iClock planea ser una "librería" que permite mostrar un reloj digital. *iClock* tiene tres tipos de relojes:

- **Classic** – El reloj "clásico" muestra la hora, minuto y segundo, ademas de una etiqueta que puede ser A.M. o P.M. , donde el valor de los minutos y segundos pueden variar entre 0 y 59. Sin embargo, el valor de la hora varia entre 0 y 12. Por ejemplo, "02:12:37 - A.M.", "11:59:59 - A.M."
- **Modern** – El reloj "moderno" muestra la hora, minuto y segundo. A diferencia del reloj "clásico" este no cuenta con las etiquetas "A.M" o "P.M" , pero el valor de la hora puede variar entre 0 y 23. Por ejemplo, "21:09:08".
- **Chronometer** – El cronometro solo muestra los minutos y segundos. El cronometro es un reloj especial el valor del segundo varia entre 0 y 59, pero el valor de los minutos no tiene limite. Por ejemplo, "121:31" (121 minutos y 31 segundos).

Note que en los ejemplos si el valor de la hora, minuto o segundo es menor a 10 se muestra con un cero adelante.

2 Tarea 1

El objetivo de la tarea 1 desarrollar algunas funcionalidades de iClock. Cada reloj debe proveer una forma de: (i) incrementar un segundo a un tiempo actual, (ii) ver si dos relojes están sincronizados (tienen la misma hora,min, seg), (iii) sincronizar dos relojes (iv) imprimir la hora correspondiente a un reloj en consola usando su propio formato.

No es necesario que cree una UI (user interface), es suficiente que se pueda imprimir en consola. Tampoco es necesario que los relojes actualicen su hora automáticamente (no usen threads).

A continuación se listan las funcionalidades que deben desarrollar para el proyecto *iClock* junto con algunos ejemplos:

- Crear las clases, atributos y constructores para poder crear objetos de los diferentes tipos de relojes Classic, Mordern y Chronometer.

- Sobre-escribir el metodo toString() para los diferentes tipos de relojes, para que devuelva el tiempo según el correspondiente formato de cada reloj. Por ejemplo:

```
IClock clockA= new ClassicClock(3,5,2,Classic.PM); // 03:05:02 PM
IClock clockB= new ModernClock(15,5,2); // 15:05:02
IClock clockC= new Chronometer(900, 2); // 900 minutes y 2 segundos

System.out.println(clockA); // imprime 03:05:02 PM
System.out.println(clockB); // imprime 15:05:02
System.out.println(clockC); // imprime 900:02
```

- Agregar un método que permita saber si dos relojes están sincronizados (*i.e.,* clockA.isSyncronized(clockB), devuelve true si los relojes están sincronizados y false si no lo estan). Dos relojes están sincronizados si tienen la misma hora del día (dependiendo el caso). Por ejemplo:

```
IClock clockA= new ClassicClock(3,5,2,Classic.PM); // 03:05:02 PM
IClock clockB= new ModernClock(15,5,2); // 15:05:02
IClock clockC= new Chronometer(900, 2); // 900 minutes y 2 segundos, que son 15
      horas 5 minutos y 2 segundos
IClock clockD= new ModernClock(15,5,2); // 15:05:02

IClock clockE= new ModernClock(15,5,3); // 15:05:03
IClock clockF= new ClassicClock(3,5,2,Classic.AM); // 03:05:02 AM
```

  A, B, C, D están sincronizados. A y E no están sincronizados porque E tiene un segundo mas. A y F no están sincronizados porque uno es AM y el otro PM. Note que para ver si un reloj esta sincronizado con algún cronometro, primero se debe calcular el numero de horas, minutos y segundos cronometrados. Para luego compararlos con las horas, minutos y segundos de los demás relojes.

- Agregar un metodo que permita sincronizar dos relojes (*i.e.,* clockA.syncronize(clockB)). Por ejemplo:

```
IClock clockB= new ModernClock(15,5,2); // 15:05:02
IClock clockD= new ModernClock(15,5,5); // 15:05:05

System.out.println(clockB.isSyncronized(clockD)); // imprime false

clockB.syncronize(clockD);  //  el reloj B debe actualizar al reloj D con su hora.
System.out.println(clockB); // imprime 15:05:02
```

```
System.out.println(clockD); // imprime 15:05:02

System.out.println(clockB.isSyncronized(clockD)); // imprime true
```

- Agregar un método que permita incrementar un segundo a un reloj. Por ejemplo:

```
IClock clockA= new ClassicClock(3,5,2,Classic.PM); // 03:05:02 PM
System.out.println(clockA); // imprime 03:05:02 PM
clockA.increment();
System.out.println(clockA); // imprime 03:05:03 PM

IClock clockB= new ModernClock(15,59,59); // 15:59:59
System.out.println(clockB); // imprime 15:59:59
clockB.increment();
System.out.println(clockB); // imprime 16:00:00
```

- Agrege al menos tres tests para probar la correctitud de los metodos: `increment`,`toString`, `isSyncronized` y `syncronize`.

- Documentar las clases del proyecto y generar JAVADOCS. Nota: debe adjuntar los JAVADOCS generados. Para ver la forma de documentar código puede ver el siguiente link: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html Un ejemplo sencillo de como se documenta una clase es la clase Point2D http://javasourcecode.org/html/open-source/jdk/jdk-6u23/java/awt/geom/Point2D.java.html

- (**opcional**, +1 punto extra) Crear una interfaz grafica que muestre la hora en forma de un reloj digital, el reloj debe actualizarse cada segundo (usando uso del metodo increment). Para ayudarlos en este ultimo punto, se subió a u-cursos un ejemplo, buscar en material docente-auxiliar, proyecto iCounter. Este ejemplo tiene un interfaz grafica que muestra un contador que se actualiza cada segundo :) .

Cada una de las funcionalidad anteriores deben estar testadas y documentadas. Se debe aplicar todos los conceptos y buenas practicas vistas en clases.

Entregables Se debe subir a u-cursos la carpeta correspondiente a su proyecto eclipse (código fuente + javadocs).