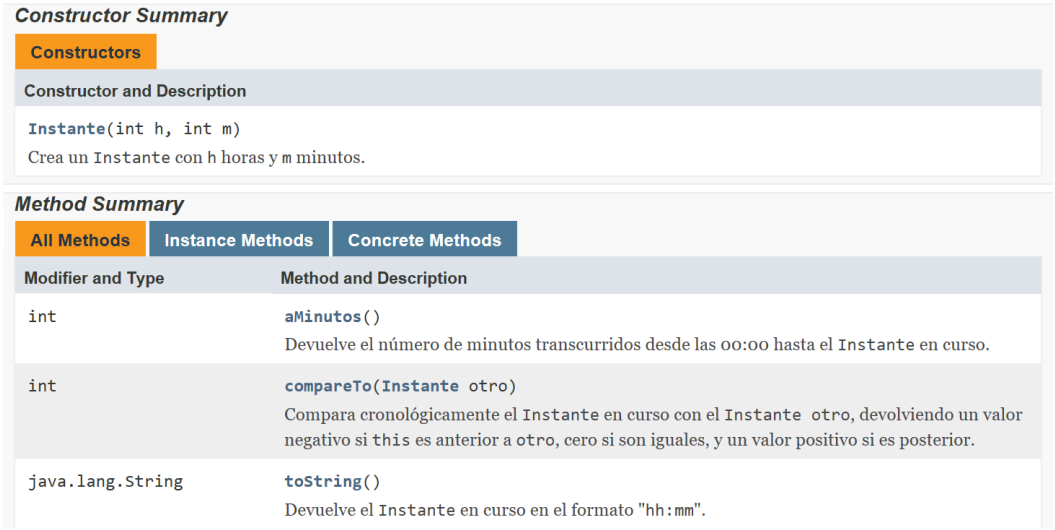
Primer Parcial de IIP (ETSInf)
6 de Noviembre de 2017. Duración: 1 hora y 30 minutos

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de IIP es de **2,4 puntos**

NOMBRE:	GRUPO:

1. 6 puntos Se quiere diseñar una clase Tipo de datos denominada `ProgramaRadio` para representar un programa de radio. Cada programa de radio tiene asociado un título, una hora de inicio y una hora de fin (siendo ambas del mismo día y la hora de inicio anterior a la de fin) y un tipo de programa que puede ser magazine, música o noticias. Para representar las horas de inicio y de fin se dispone de la clase `Instante` con la funcionalidad que se muestra en parte, a continuación, en su documentación:



Constructor Summary

Constructors

Constructor and Description
`Instante(int h, int m)` Crea un `Instante` con `h` horas y `m` minutos.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`int`	`aMinutos()` Devuelve el número de minutos transcurridos desde las 00:00 hasta el `Instante` en curso.
`int`	`compareTo(Instante otro)` Compara cronológicamente el `Instante` en curso con el `Instante otro`, devolviendo un valor negativo si `this` es anterior a `otro`, cero si son iguales, y un valor positivo si es posterior.
`java.lang.String`	`toString()` Devuelve el `Instante` en curso en el formato "`hh:mm`".

Se pide: implementar la clase `ProgramaRadio`, considerando que está en el mismo directorio que la clase `Instante`, con los atributos y métodos que se indican a continuación:

a) (0.25 puntos) Atributos de clase públicos y constantes de tipo entero:
   - `MAGAZINE`, con valor 0 que representa el tipo de programa magazine.
   - `MUSICA`, con valor 1 que representa el tipo de programa musical.
   - `NOTICIAS`, con valor 2 que representa el tipo de programa noticias.

   Estas constantes deberán ser utilizadas siempre que se requiera (tanto en la clase `ProgramaRadio` como en la clase `GestorRadio`).

b) (0.5 puntos) Atributos de instancia privados: `tipo` (`int`), `titulo` (`String`), `horaInicio` (`Instante`) y `horaFin` (`Instante`).

c) (1.25 puntos) Un constructor general tal que, dados el tipo de programa, su título, las horas y los minutos del inicio, y la duración en minutos, inicialice todos los atributos de instancia. Suponed que los datos son correctos.

d) (1 punto) Un método `equals` (que sobrescribe el de `Object`) para comprobar si dos programas de radio son iguales, esto es, si son del mismo tipo y tienen el mismo título.

e) (1 punto) Un método `toString` (que sobrescribe el de `Object`) para que, usando obligatoriamente una instrucción `switch` para convertir en texto el tipo de programa, devuelva el resultado con un formato como el mostrado en los siguientes ejemplos:

```
06:30 - Anda ya (Música)
14:30 - Todo noticias (Noticias)
16:00 - Julia en la Onda (Magazine)
```

f) (2 puntos) Dos programas de radio (que se emiten el mismo día) se consideran ordenados en la parrilla de programación de acuerdo con los siguientes criterios:
   - Va primero el programa que empieza antes.
   - A igual hora de inicio, va primero el que acaba antes.
   - A igual horario de inicio y fin, van primero los programas de noticias, luego los de música y, por último, los magazines.
   - En caso de que coincidan en horario y tipo, es indiferente el orden en que aparecen en la parrilla.

   Implementar un método `compareTo` que, dado un `ProgramaRadio` pasado como parámetro `p` que se emite el mismo día que `this`, devuelva un `int` negativo si `this` es anterior a `p` en la parrilla, positivo si `p` es anterior y `0` si es indiferente.

Solución:

```
public class ProgramaRadio {
    public static final int MAGAZINE = 0;
    public static final int MUSICA = 1;
    public static final int NOTICIAS = 2;

    private int tipo;
    private String titulo;
    private Instante horaInicio;
    private Instante horaFin;

    public ProgramaRadio(int tip, String tit, int hora, int min, int duracion) {
        tipo = tip;
        titulo = tit;
        horaInicio = new Instante(hora, min);
        int fin = horaInicio.aMinutos() + duracion;
        horaFin  = new Instante(fin / 60, fin % 60);
    }

    public boolean equals(Object o) {
        return o instanceof ProgramaRadio
            && tipo == ((ProgramaRadio) o).tipo
            && titulo.equals(((ProgramaRadio) o).titulo);
    }

    public String toString() {
        String res = horaInicio + " - " + titulo + " (";
        switch (tipo) {
            case MAGAZINE:
                res += "Magazine)";
                break;
            case MUSICA:
                res += "Música)";
                break;
            case NOTICIAS:
                res += "Noticias)";
        }
        return res;
    }

    public int compareTo(ProgramaRadio p) {
        int res = horaInicio.compareTo(p.horaInicio);
        if (res == 0) {
            res = horaFin.compareTo(p.horaFin);
            if (res == 0) { res = p.tipo - tipo; }
        }
        return res;
    }
}
```

2. 2 puntos **Se pide**: implementar la clase Programa `GestorRadio`, en el mismo directorio que `ProgramaRadio`, con un método `main` que realice las siguientes acciones:

   a) (0.25 puntos) Crear un objeto `p1` de tipo `ProgramaRadio`, para representar a un programa de radio magazine, de título `Julia en la Onda`, que comienza a las 16:00 y tiene una duración de 180 minutos.
   b) (0.25 puntos) Crear un objeto `p2` de tipo `ProgramaRadio`, para representar a un programa de radio musical, de título `Anda ya`, que comienza a las 6:30 y tiene una duración de 300 minutos.
   c) (1.5 puntos) Comparar `p1` con `p2` usando el método `compareTo` y, en función de su resultado, mostrarlos por pantalla según su orden en la parrilla.

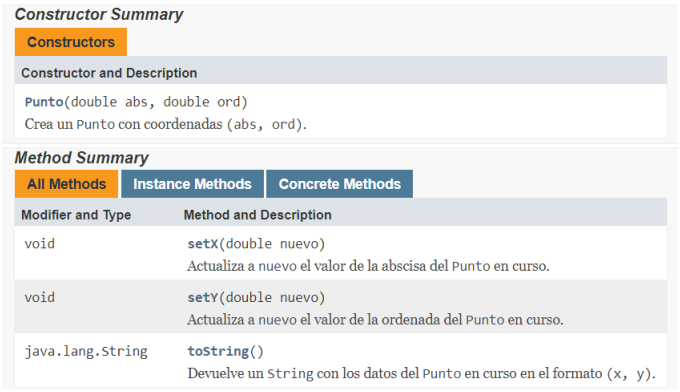
Solución:

```
public class GestorRadio {
    public static void main(String[] args) {
        ProgramaRadio p1 = new ProgramaRadio(ProgramaRadio.MAGAZINE, "Julia en la Onda",
                                             16, 0, 180);

        ProgramaRadio p2 = new ProgramaRadio(ProgramaRadio.MUSICA, "Anda Ya",
                                             6, 30, 300);

        if (p1.compareTo(p2) < 0) { System.out.println(p1 + "\n" + p2); }
        else { System.out.println(p2 + "\n" + p1); }
    }
}
```

3. 2 puntos Se dispone de la clase `Punto` que define un punto en un espacio bidimensional real (con dos atributos representando su abscisa y su ordenada), con la funcionalidad que se muestra en parte, a continuación, en su documentación:



Constructor Summary

Constructors

Constructor and Description
`Punto(double abs, double ord)` Crea un `Punto` con coordenadas `(abs, ord)`.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`void`	`setX(double nuevo)` Actualiza a nuevo el valor de la abscisa del `Punto` en curso.
`void`	`setY(double nuevo)` Actualiza a nuevo el valor de la ordenada del `Punto` en curso.
`java.lang.String`	`toString()` Devuelve un `String` con los datos del `Punto` en curso en el formato `(x, y)`.

Dada la siguiente clase programa:

```
public class Ejercicio3 {
    public static void main(String[] args) {
        Punto p = new Punto(1.0, -1.0);
        double x = 1.0, y = -1.0;
        System.out.println("Antes de cambiarCoord: x = " + x + ", y = " + y
                           + ", p = " + p);

        cambiarCoord(x, y, p);
        System.out.println("Tras cambiarCoord una vez: x = " + x + ", y = " + y
                           + ", p = " + p);

        cambiarCoord(x, y, p);
        System.out.println("Tras cambiarCoord 2 veces: x = " + x + ", y = " + y
                           + ", p = " + p);
    }

    public static void cambiarCoord(double x, double y, Punto p) {
        p.setX(y);
        p.setY(x);
    }
}
```

Se pide: Completar qué se muestra por pantalla tras su ejecución.

```
Antes de cambiarCoord: x =  1.0 , y =  -1.0 , p = ( 1.0 , -1.0 )

Tras cambiarCoord una vez: x =  1.0 , y =  -1.0 , p = ( -1.0 ,  1.0 )

Tras cambiarCoord 2 veces: x =  1.0 , y =  -1.0 , p = ( -1.0 ,  1.0 )
```