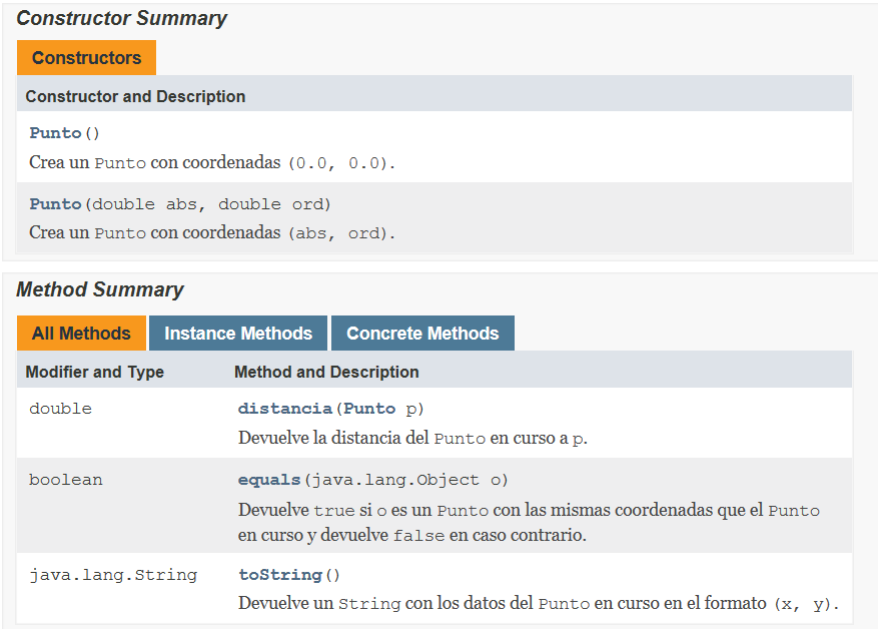
IIP Primer Parcial - ETSInf

9 de Noviembre de 2016. 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

1. 1 punto Se dispone de la clase `Punto` que define un punto en un espacio bidimensional real mediante los atributos `x` e `y` (representando abscisa y ordenada, respectivamente), con la funcionalidad que, en parte, se muestra, a continuación, en su documentación:



Constructor Summary

Constructors

Constructor and Description
`Punto()` Crea un `Punto` con coordenadas `(0.0, 0.0)`.
`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
`double`	`distancia(Punto p)` Devuelve la distancia del `Punto` en curso a `p`.
`boolean`	`equals(java.lang.Object o)` Devuelve `true` si `o` es un `Punto` con las mismas coordenadas que el `Punto` en curso y devuelve `false` en caso contrario.
`java.lang.String`	`toString()` Devuelve un `String` con los datos del `Punto` en curso en el formato `(x, y)`.

Se pide: Implementar el método `distancia` tal que devuelva la distancia del `Punto` en curso a otro `Punto p` dado, **redondeando** el resultado a **cuatro** decimales. Se recuerda que la distancia entre dos puntos (a_1, b_1) y (a_2, b_2) se calcula como $\sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}$.

Solución:

```
public double distancia(Punto p) {
    double abs = this.x - p.x;
    double ord = this.y - p.y;
    double dist = Math.sqrt(abs * abs + ord * ord);
    return Math.round(dist * 10000) / 10000.0;
    // o también: return Math.round(dist * Math.pow(10, 4)) / Math.pow(10, 4);
}
```

2. 6.5 puntos Utilizando la clase `Punto` descrita anteriormente, se desean representar los edificios del campus de Vera de la UPV. Para ello, se quiere diseñar una clase Tipo de datos denominada `Edificio` que contendrá información física acerca de la construcción (coordenadas GPS, código de identificación en un plano), así como del uso asignado al edificio (tipo de uso y nombre de la entidad que está haciendo uso del mismo).

Se pide: implementar la clase `Edificio` con los atributos y métodos que se indican a continuación:

a) (0.5 puntos) Atributos de clase públicos y constantes de tipo entero:

- `DEPARTAMENTO`, con valor 0 que representa el tipo de edificio departamental.
- `ESCUELA`, con valor 1 que representa el tipo de edificio dedicado a docencia como aularios, escuelas o laboratorios.

- `SERVICIOS`, con valor 2 que representa el tipo de edificios para otras actividades como cafeterías, oficinas, etc.

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

b) (0.5 puntos) Atributos de instancia privados `codigo` (`String`), `entidad` (`String`), `tipo` (`int`) y `coordenadas` (`Punto`).

c) (1.5 puntos) Dos constructores:
   - Un constructor general con los parámetros apropiados (uno de ellos de tipo `Punto`) para inicializar todos los atributos de instancia. Suponed que los datos son correctos.
   - Un constructor por defecto que cree un edificio departamental usado por la entidad `DSIC`, con código `1F` y en las coordenadas `(39.4625, -0.3472)`.

d) (0.5 puntos) Un método consultor y un método modificador del atributo `coordenadas`. Suponed que el valor del parámetro del modificador es correcto.

e) (1 punto) Un método `equals` (que sobrescribe el de `Object`) para comprobar si dos edificios son iguales teniendo en cuenta los datos de la construcción e independientemente del uso, esto es, si tienen el mismo código y las mismas coordenadas. Nótese que un edificio puede usarse a la vez por dos entidades, por ejemplo, el edificio `1G` por la `ETSINF` y el `DISCA`.

f) (1 punto) Un método `toString` (que sobrescribe el de `Object`) para que, usando obligatoriamente una instrucción `switch`, devuelva el resultado con un formato como el mostrado en los siguientes ejemplos (para las coordenadas GPS se dispone del método `toString` de la clase `Punto`):

```
Edificio departamental 1F (DSIC), GPS: (39.4625, -0.3472)
Edificio departamental 1G (DISCA), GPS: (39.4826, -0.3470)
Edificio docente 1G (ETSINF), GPS: (39.4826, -0.3470)
Edificio de servicios 3N (Cafeteria BBAA), GPS: (39.4841, -0.3443)
```

g) (1.5 puntos) Un método `masCercaRectorado` que, dado un `Edificio` pasado como parámetro `e`:
   - Si la distancia del edificio `this` a las coordenadas de rectorado `(39.4823, -0.3457)` es menor que la del edificio `e`, devuelve `-1`.
   - Si, por el contrario, la distancia del edificio `this` a rectorado es mayor que la del edificio `e`, devuelve `1`.
   - Si ambas distancias son iguales se considera que:
     - Si los tipos de edificio son diferentes se considera más cerca los de tipo servicio, luego docentes y, por último, departamentales, devolviendo `-1` o `1` según el caso. Por ejemplo, si el edificio `this` es el `DISCA` y el edificio `e` es la `ETSINF` (edificios departamental y docente, respectivamente, a la misma distancia de rectorado), el método devolverá `1` indicando que la `ETSINF` está más cerca de rectorado que el `DISCA`. Pero si `this` es la `ETSINF` y `e` es el `DISCA`, devolverá `-1`.
     - Si los tipos son iguales, devuelve `0`.

Solución:

```
public class Edificio {
    public static final int DEPARTAMENTO = 0;
    public static final int ESCUELA = 1;
    public static final int SERVICIOS = 2;
```

```
    private String codigo, entidad;
    private int tipo;
    private Punto coordenadas;

    public Edificio(String c, String e, int t, Punto p) {
        codigo = c;
        entidad = e;
        tipo = t;
        coordenadas = p;
    }

    public Edificio() {
        this("1F", "DSIC", DEPARTAMENTO, new Punto(39.4625, -0.3472));
    }

    public Punto getCoordenadas() { return coordenadas; }

    public void setCoordenadas(Punto p) { coordenadas = p; }

    public boolean equals(Object o) {
        return o instanceof Edificio
            && codigo.equals(((Edificio) o).codigo)
            && coordenadas.equals(((Edificio) o).coordenadas);
    }

    public String toString() {
        String res = "Edificio ";
        switch (tipo) {
            case DEPARTAMENTO:
                res += "departamental "; break;
            case ESCUELA:
                res += "docente "; break;
            case SERVICIOS:
                res += "de servicios "; break;
        }
        res += codigo + " (" + entidad + "), GPS: " + coordenadas;
        return res;
    }

    public int masCercaRectorado(Edificio e) {
        Punto rectorado = new Punto(39.4823, -0.3457);
        double distThis = coordenadas.distancia(rectorado);
        double distE = e.coordenadas.distancia(rectorado);
        int resultado = 0;
        if (distThis < distE) { resultado = -1; }
        else if (distThis > distE) { resultado = 1; }
        else if (tipo < e.tipo) { resultado = 1; }
        else if (tipo > e.tipo) { resultado = -1; }
        return resultado;
    }
}
```

3. 2.5 puntos **Se pide**: implementar la clase Programa `GestorEdificios` con un método `main` que realice las siguientes acciones:

a) (1 punto) Crear un objeto `e1` de tipo `Edificio` usando el constructor general, para representar a un edificio docente usado por la entidad `ETSINF`, con código `1G` y coordenadas `(39.4826, -0.3470)`.

b) (0.5 puntos) Crear un objeto `e2` de tipo `Edificio` usando el constructor por defecto.

c) (1 punto) Invocar al método `masCercaRectorado` para comparar `e1` con `e2` y, a continuación, mostrar por pantalla el edificio más cercano a rectorado tras el literal `"El edificio más cercano a rectorado es "` o, si están igual de cerca, el mensaje `"Ambos edificios están igual de cerca de rectorado"`.

Solución:

```
public class GestorEdificios {
    public static void main(String[] args) {
        Edificio e1 = new Edificio("1G", "ETSINF", Edificio.ESCUELA,
            new Punto(39.4826, -0.3470));
        Edificio e2 = new Edificio();

        int comp = e1.masCercaRectorado(e2);
        String res = "El edificio más cercano a rectorado es ";
        if (comp == -1) { res += e1; }
        else if (comp == 1) { res += e2; }
        else { res = "Ambos edificios están igual de cerca de rectorado"; }
        System.out.println(res);
    }
}
```