Programación, Curso 2017-2018

Problema 1: Órdenes de la *shell*; Datos de una clase; Tipos primitivos; Sangrado

Datos personales	
Apellidos:	
Nombre:	

1 Datos de una clase

Objetivos

Distinguir entre los diferentes datos que pueden encontrarse en una clase

Descripción

Indicar qué datos se encuentran en el código Java de la siguiente clase, indicando para cada uno si es un atributo, argumento, variable local o constante literal, así como el tipo de dato.

Observar que los datos variables siempre se definen en Java con el formato:

```
tipo nombreVariable
```

donde en ocasiones se pone el modificador "private" delante. El tipo puede ser un tipo predefinido (como int, double, …) o una clase (como String).

Por otro lado, las constantes literales se expresan directamente con su valor.

```
/**
 * Clase que gestiona los jugadores de los distintos equipos de baloncesto
 */
public class Jugador
{
   private String nombre; //nombre con el que se conoce al jugador
   private int edad; //la edad en años del jugador
   private double altura; //la altura del jugador, en metros
   private int posicion; // posicion que ocupa el jugador de entre las posibles posiciones

   //Constantes de calse publicas que limitan las posibles posiciones que ocupa un jugador.
   public static final int BASE = 0;
   public static final int ESCOLTA = 1;
   public static final int ALERO = 2;
   public static final int PIVOT = 3;


   /**
    * Constructor de la clase Jugador
    * @param nombre    nombre con el que se conoce al jugador
    * @param edad      la edad en años del jugador
```

```
     * @param altura    la altura del jugador, en metros
     * @param posicion  0=BASE, 1=ESCOLTA, 2=ALERO, 3=PIVOT
     */
    public Jugador(String nombre, int edad, double altura, int posicion)
    {
       this.nombre = nombre;
       this.edad = edad;
       this.altura = altura;
       this.posicion = posicion;
    }

    /**
     * Metodo observador que devuelve el nombre del jugador
     * @return     nombre con el que se conoce al jugador
     */
    public String getNombre()
    {
       return nombre;
    }

     /**
     * Metodo observador que devuelve la edad del jugador
     * @return     la edad del jugador en años
     */
    public int getEdad()
    {
       return edad;
    }

     /**
     * Metodo observador que devuelve la altura del jugador
     * @return     altura del jugador en metros
     */
    public double getAltura()
    {
       return altura;
    }

     /**
     * Metodo observador que devuelve la posicion en la que juega el jugador
     * @return     posicion en la que juega el jugador
     */
    public int getPosicion()
    {
       return posicion;
    }

    public String toString()
    {
       String cadena = nombre + ". " + edad + "años. " + altura + "m. ";
       switch(posicion)
       {
          case BASE:
             cadena+="Base.";
             break;
          case ESCOLTA:
```

```
            cadena+="Escolta.";
            break;
         case ALERO:
            cadena+="Alero.";
            break;
         case PIVOT:
            cadena+="Pivot.";
            break;
         default:
            break;

      }
      return cadena;
   }

}
```

Respuesta

Crear una lista de los datos que hay en la clase, agrupándolos según:

- atributos
- argumentos
- variables locales
- constantes literales

Para cada dato, indicar de qué tipo es (entero, real, carácter, booleano, String, ...)

2 Literales (cambiar, por poco didáctico)

Objetivos

Familiarizarse con los literales de los tipos primitivos

Descripción

Reescribir la siguiente clase Java para que los literales utilizados se correspondan con el tipo de la variable a la que se asigna o de la expresión en que aparece (aunque no sea necesario para que la clase compile correctamente).

Respuesta

Adaptar los literales de la siguiente clase para lograr una coincidencia de los tipos de sus expresiones:

```
/**
* Clase que sirve para practicar con los literales de los números enteros y reales
*/
public class Literales
{
   // atributos
   private double medida=3.0;
   private float x;
```

```
    private int i;
    private short s=12;
    private long num=73;

    /**
     * Metodo que modifica atributos y retorna un calculo.
     */
    public double funcion(float y, byte b) {
        x=y/2.0;
        int j=i+b*3;
        i=s*b+8;
        num=num+8*j;
        return x*j+y+2*medida;
    }

}
```

3 Sangrado

Objetivos

Familiarizarse con el concepto de sangrado.

Descripción

Contestar a una pregunta sobre la importancia del sangrado y hacer un ejercicio para sangrar correctamente las instrucciones de una clase Java.

Respuesta

a) ¿Para qué es importante el sangrado del código fuente?

<poner aquí la respuesta, máximo dos líneas>

b) Adaptar la siguiente clase utilizando el sangrado que te parezca más adecuado:

```
/**
 * Contiene los datos de un ciclista y métodos para estimar
 * su potencia y cambiar las condiciones
 *
 * @author (Michael)
 * @version (23-oct-2013)
 */
public class Ciclista
{
// atributos, en unidades del sistema internacional; ángulos en grados
private double m; // masa del conjunto ciclista+bicicleta, Kg
private double ang; // angulo de la pendiente, grados
private double vc; // velocidad del ciclista, m/s
private double vViento; // velocidad del viento, m/s
private double k=0.226; // coeficiente aerodinámico, Kg/m
private double cr=0.038; // coeficiente de rodadura, m/s2
private double g=9.8; // gravedad, m/s2
```

```
/**
 * Pone los valores de los atributos obteniéndolos de los argumentos del
 * mismo nombre
 */
public Ciclista (double m, double ang, double vc, double vViento)
{
this.m=m;
this.ang=ang;
this.vc=vc;
this.vViento=vViento;
}

/**
* retorna la potencia total del ciclista
*/
public double potenciaTotal()
{
double vAire=vc+vViento;
double pAero= k*vc*vAire*vAire;
double pRod=cr*m*vc;
double pAsc=m*g*vc*Math.sin(Math.toRadians(ang));
return pAero+pRod+pAsc;
}


/**
* modifica los atributos ang, vc y vViento haciéndolos iguales a nuevoAng,
* nuevaVelCicl y nuevaVelViento
*/
public void cambiaCondiciones (double nuevoAng, double nuevaVelCicl,
double nuevaVelViento)
{
this.ang=nuevoAng;
this.vc=nuevaVelCicl;
this.vViento=nuevaVelViento;
}

}
```

4 Órdenes Unix

Objetivos

Aprender el funcionamiento de algunas órdenes de la *shell* de Unix

Descripción

Consultar el funcionamiento de las siguientes órdenes de la *shell* de Unix, resumir el funcionamiento en menos de tres líneas por orden y escribir un breve ejemplo que las utilice haciendo lo que se pide:

- `if`: mostrar un mensaje en pantalla (con la orden `echo`) si un directorio determinado existe y otro mensaje distinto si no existe
- `cat`: unir tres ficheros, obteniendo un fichero nuevo cuyo contenido es el contenido de un

fichero existente, seguido del contenido de otro, y del de un tercero.

- grep: mostrar las líneas de los ficheros de un directorio que contengan la secuencia de caracteres formada por dos barras de dividir: "//"

Para hacer pruebas de los ejemplos se puede crear un directorio y en él crear los ficheros que se necesiten para probar.

Respuesta:

<poner aquí el resumen de lo que hace cada orden seguido del ejemplo de uso que se pide>