Hoja 4: Solución

Ejercicio 1

```
package figuras;
public class Cuadrado extends Rectángulo {


	public Cuadrado(int x, int y, int ancho) {
		super(x,y,ancho,ancho);
	}

	public String toString() {
		 return "("+getX()+", "+getY()+")"+" Lado: "+getLx();
	}
}
```

```
// error: un rectángulo no es un cuadrado; tipos incompatibles
// Cuadrado c1 = new Rectángulo(10,10,6,6);

// bien: un cuadrado es un rectángulo
Rectángulo r1 = new Cuadrado(20,20,5);

// declaraciones correctas
Cuadrado c2 = new Cuadrado(20,20,5);
Rectángulo r2 = new Rectángulo(10,10,6,6);

// error: un rectángulo no es un cuadrado; tipos incompatibles
// c2 = r2;

// bien: un cuadrado es un rectángulo
r2 = c2;

System.out.println(c2); // (20, 20) Lado: 5
System.out.println(r2); // (20, 20) Lado: 5
```

Ejercicio 2

```
// la clase es abstracta porque no podemos escribir el método coste
// NO se puede utilizar un interface, ya que algunas de los métodos sí se pueden
// escribir y un interface no permite escribir ninguno
public abstract class Factura {
	protected double importe; // total a pagar
	protected int númLlamadas; // número total de llamadas
	protected String compañía; // nombre de la compañía


	public Factura(String compañía) {  this.compañía = compañía;}

	public String getCompañía() {return this.compañía;}
	public int getNúmLlamadas() {return this.númLlamadas;}
	public double getImporte() {return this.importe;}


	public abstract double coste(int hora, int segundos);

	public void llamada(int hora, int segundos) {
		importe += coste(hora,segundos);
		númLlamadas++;
	}
```

```
	public String toString(){
		return "Compañía: "+getCompañía()+" - "+getNúmLlamadas()+
			" llamadas "+ "- Importe: "+getImporte()+" euros";
	}
}
```

Ejercicio 3

```
// sólo hay que implimentar el metodo abstracto coste
public class TeLeLe extends Factura {

	public TeLeLe(){
		super("TeLeLe");
	}

	public double coste(int hora, int duración) {
		double factor;

		factor =  hora >=0 && hora < 8   ? 0.01
			: hora >= 8 && hora < 14       ? 0.05
			: 0.02;

		return factor*duración + 0.25;
	}
}

import java.util.Scanner;

public class Principal {
	public static void main(String[] args) {
		// declaramos la factura
		TeLeLe factura = new TeLeLe();
		// para leer los datos
		Scanner sc = new Scanner(System.in);

		String seguir; // para preguntar que si quiere seguir
		do {
			// pedimos los datos de la llamada:
			System.out.print("Hora de inicio (de 0 a 24): ");
			int h = sc.nextInt();
			System.out.print("Duración (segundos): ");
			int s = sc.nextInt();
			factura.llamada(h,s);

			System.out.print("Más llamadas? (s/n) ");
			seguir = sc.nextLine();
		} while (seguir.equals("s") || seguir.equals("S"));


		System.out.println("Factura: "+factura);

	}
}
```

Ejercicio 4

Como nos piden que la clase no contenga medios incLocal o incVisitante no podemos usar herencia (en realidad sí podríamos, heredando y redefiniendo los métodos que queremos "ocultar" como privados, pero esta solución parece menos natural).

Además de la clase tendremos que escribir dos clases para las excepciones:

```
package deportes;

public class NombreEquipoNoVálido extends Exception {
}
```

```
package deportes;

public class PuntuaciónNoVálida extends Exception {
}
```

```
package deportes;


public class MarcadorBaloncesto {

    private Marcador m;

       public MarcadorBaloncesto(String nombreLocal, String nombreVisitante) {
          m = new Marcador(nombreLocal,nombreVisitante);
       }

       // tiro libre anotado
       // eq es el nombre del equipo
       public void tiroLibre(String eq) throws NombreEquipoNoVálido {
              if (m.getNombreLocal().equals(eq))
                 m.incLocal();
              else if (m.getNombreVisitante().equals(eq))
                 m.incVisitante();
              else
                 throw new NombreEquipoNoVálido();

       }

       public void canasta(String eq, int punt)
                 throws NombreEquipoNoVálido,PuntuaciónNoVálida {

              if (punt!=2 && punt !=3)
                throw new PuntuaciónNoVálida();


              if (m.getNombreLocal().equals(eq))
                 m.incLocal(punt);
              else if (m.getNombreVisitante().equals(eq))
                 m.incVisitante(punt);
              else
                 throw new NombreEquipoNoVálido();


       }
       public String toString(){ return m.toString(); }
}
```

Ejercicio 5

```
// 1)
package lógica;
public interface Fórmula {
	public boolean valoración(ValorVar t[]);
}

// 2)
package lógica;
// Fórmula formada por una sola variable
public class Variable implements Fórmula{
	private String nombre;

	public Variable(String n){this.nombre=n;}
		public boolean valoración(ValorVar t[]){
		boolean encontrado=false;
		boolean resultado=false;
		int i=0;
		// recorremos el array hasta encontrar la
		// posición que ocupa la variable
		while(i<t.length && !encontrado)
		if (t[i].nombreVar().equals(this.nombre))
			encontrado=true;
		else i++;
			// por el enunciado, suponemos que siempre se encuentra
			resultado = t[i].valor();
		return resultado;
	}
}

// 3)
package lógica;
public class Negación implements Fórmula {
	private Fórmula f;
	public Negación(Fórmula f) {this.f = f; }
	public boolean valoración(ValorVar t[]){return !f.valoración(t);}
}

// 4)
package lógica;
public class And implements Fórmula{
	private Fórmula f1,f2;
	// constructora
	// parámetros: dos fórmulas f1 y f2.
	// Esta fórmula representará la conjunción de f1 y f2
	public And(Fórmula f1, Fórmula f2) { this.f1 = f1; this.f2=f2; }
	public boolean valoración(ValorVar t[]){
		return f1.valoración(t) && f2.valoración(t);
	}
}

// 5)
package lógica;
public class Or implements Fórmula {
	private Fórmula f1,f2;
	// constructora
	// Esta fórmula representará la disyunción de f1 y f2
	public Or(Fórmula f1, Fórmula f2) { this.f1 = f1; this.f2=f2; }
	public boolean valoración(ValorVar t[]){
		return f1.valoración(t) || f2.valoración(t);
	}
}
```

Ejercicio 6

```
import lógica.*;

public class Principal {

    public static void main(String[] args) {
       // Nota: También está bien si se declaran como de tipo Fórmula
       Variable X = new Variable("X");
       Variable Y = new Variable("Y");
       Variable Z = new Variable("Z");

       And f1 = new And(X,Z);
       And f2 = new And(Y,Z);
       Negación f3 = new Negación(f2);
       Or f = new Or(f1,f3); // la fórmula pedida

       // la valoración
       ValorVar t[] = {
              new ValorVar("X",true),
              new ValorVar("Y",true),
              new ValorVar("Z",false)
       };

       // escribimos la valoración
       System.out.println(f.valoración(t));
    }
}
```

Ejercicio 7

```
public class TercerGrado extends Newton {
       private double a,b,c,d;

       public TercerGrado(double a, double b, double c, double d) {
              this.a = a; this.b = b; this.c = c; this.d = d;
       }

       public double f(double x) {
              return a*x*x*x + b*x*x + c*x + d;
       }

       public double df(double x) {
              return 3*a*x*x + 2*b*x + c;
       }

       public double raíz() {
              return raíz(0);
       }
}
```

Ejercicio 8

```
public class Principal {
       public static void main(String[] args) {
          TercerGrado p = new TercerGrado(4,-2,3,1);
          System.out.println(p.raíz());
       }
}
```