Prueba 2 de Evaluación Continua

Análisis y Diseño de Software (2013/2014)

Ejercicio 1: Programación Orientada a Objetos (3,5 puntos)

Se va a desarrollar una aplicación orientada a objetos para una empresa que organiza actividades de ocio. Cada actividad se caracteriza por una descripción y un precio base. El precio total de cada actividad se calcula sumando su precio base y un coste extra que se calcula de manera diferente para cada tipo de actividad. Hay dos tipos de actividades: práctica deportiva y visita a museo. Cada práctica deportiva lleva asociado un nivel de riesgo, que se utiliza para calcular su coste extra (multiplicado riesgo por precio base). Todas las visitas a museos tienen un precio base constante de 5.00€, y su coste extra se calcula según sea una visita guiada (4.00€), bilingüe (8.00€), escolar (descuento de 1.00€), o individual (sin coste extra). Además, se debe llevar automáticamente una contabilidad de la suma de los precios totales de todas las actividades deportivas que se hayan creado.

Se pide escribir el código Java necesario para modelar los objetos descritos arriba de forma que al ejecutar la siguiente clase de prueba se produzca la salida mostrada abajo. No escribas métodos innecesarios para esta ejecución, pero presta especial atención al diseño de las clases necesarias y las relaciones entre ellas y entres sus métodos.

```
public class Apartado1 {
        public static void main(String[] args) {
           Actividad[] actividades = { new VisitaMuseo("Reina Sofia", TipoVisitaMuseo.BILINGUE),
                                       new VisitaMuseo("Arqueológico", TipoVisitaMuseo.ESCOLAR),
                                       new PracticaDeportiva("Cañada Real Bicicleta", 20.0, 0.1),
                                       new PracticaDeportiva("Descenso Rapido Sella", 50.0, 0.25)
                                     };
          System.out.println("Precio base fijo para museos: " + VisitaMuseo.COSTE_FIJO);
          for (Actividad a : actividades)
             System.out.println(a + ": precio " + a.precio() + " incluye " + a.extras() +" de complementos");
          System.out.println("Suma de precios de actividades deportivas " + PracticaDeportiva.total());
        }
}
```

Salida esperada:

```
Precio base fijo para museos: 5.0
Reina Sofia: precio 13.0 incluye 8.0 de complementos
Arqueológico: precio 4.0 incluye -1.0 de complementos
Cañada Real Bicicleta: precio 22.0 incluye 2.0 de complementos
Descenso Rapido Sella: precio 62.5 incluye 12.5 de complementos
Suma de precios de actividades deportivas 84.5
```

Solución:

```
public abstract class Actividad {
	private String descripcion;
	protected double precioBase;
	public Actividad(String d, double p) { descripcion=d; precioBase = p;}
	public double precio() { return precioBase + extras(); }
	public abstract double extras();
	public String toString() { return descripcion; }
}

public enum TipoVisitaMuseo { ESCOLAR, INDIVIDUAL, GUIADA, BILINGUE; }

public class VisitaMuseo extends Actividad {
	public final static Double COSTE_FIJO = 5.0;
	public TipoVisitaMuseo tipo;
	public VisitaMuseo (String d, TipoVisitaMuseo t) { super(d,COSTE_FIJO); tipo = t;  }
	public double extras() {
		if (tipo == TipoVisitaMuseo.GUIADA) return 4.0;
		else if (tipo == TipoVisitaMuseo.BILINGUE) return 8.0;
		else if (tipo == TipoVisitaMuseo.ESCOLAR)return -1.0;
		else return 0.0;
	}
	/* Otra alternativa es almacenar el coste de la visita en la clase TipoVisitaMuseo --- ver al final */
}

public class PracticaDeportiva extends Actividad {
	private static double total = 0;
	private double riesgo;
	public PracticaDeportiva(String d, double p, double r) {
		super(d,p); riesgo = r;
		PracticaDeportiva.total += precio();
	}
	public double extras() { return precioBase * riesgo; }
	public static double total() { return PracticaDeportiva.total; }
}
```

/* la alternativa con el coste en el enumerado `TipoVisitaMuseo` simplifica el método `extras()` de la clase `VisitaMuseo`, y por lo tanto hace más sencilla la ampliación para otros tipos de visitas a museo */

```
enum TipoVisitaMuseo {
	ESCOLAR(-1.0), INDIVIDUAL(0.0), GUIADA(4.0), BILINGUE(8.0);

	private double costeExtra;
	private TipoVisitaMuseo(double coste) { this.costeExtra = coste; }

	public double costeExtra() { return costeExtra; }
}

class VisitaMuseo extends Actividad {
	public final static Double COSTE_FIJO = 5.0;
	public TipoVisitaMuseo tipo;
	public VisitaMuseo (String d, TipoVisitaMuseo t) { super(d,COSTE_FIJO); tipo = t;	}
	public double extras() {
		return tipo.costeExtra();
	}
}
```