EVALUACION POR EXAMEN FINAL
NO-CONTINUA

Análisis y Diseño de Software (2012/2013)

Responde a cada apartado en hojas separadas

Apartado 1. (2.5 puntos)

Completa el programa de abajo, para calcular los precios de alquiler de vehículos (motos o turismos) según las siguientes condiciones. Todos los vehículos se crean con un *precio base de alquiler por día*, y la clase Vehículo les asigna automáticamente un *identificador* (ver salida esperada). Ninguno de estos 2 atributos podrá ser modificado una vez que toman valor. Las motos almacenan su cilindrada y existen dos tipos de turismo: de lujo y normales. Para simplificar, consideraremos cuatro tipos de carnet de conducir: sin puntos, de motos, de turismos y de camiones. Un conductor con carnet sin puntos no puede conducir ningún vehículo. Un conductor de camiones puede conducir también turismos y motos, y un conductor de turismos puede conducir también motos.

Para que un vehículo sea "alquilable" a un determinado conductor, éste debe cumplir los siguientes requisitos:

- para alquilar una moto es suficiente con tener carnet de moto o superior,
- para alquilar un turismo se debe tener carnet de turismo o superior pero además si el turismo es de lujo la edad del conductor debe ser > 25 años.

Con carácter general, *para todos los vehículos, su precio de alquiler para un determinado conductor* (que cumpla los requisitos anteriores) se calcula de la misma forma: *sumando al precio base por día un ajuste (positivo o negativo) por conductor*. Dicho *ajuste por conductor* será:

- **`-3`** para el alquiler de motos de cilindrada > 125, alquiladas por conductores de edad > 25 años o con carnet de turismo o superior;
- **`-2`** para motos de cilindrada no superior a 125, alquiladas por conductores de edad > 25;
- **`+1`** para turismos normales alquilados a conductores de edad no superior a 25.
- **`+2`** para turismos de lujo alquilados a conductores de edad no superior a 30.

Excepcionalmente, cuando el conductor no cumpla los requisitos de alquiler, indicados arriba, el precio de alquiler se considerará infinito (**`Double.POSITIVE_INFINITY`**).

Se pide: completar la declaración e inicialización de la variable **`precios`** del método **`main()`** y **completar el programa** para obtener la salida mostrada abajo (respetando el orden alfabético).

```
public class MainAlquilerVehiculos {
   public static void main(String[] args) {
      _____________________ precios = new _____________________________________;  // completar
      Conductor pedro = new Conductor("Pedro", 25, Carnet.TURISMO);
      Conductor marta = new Conductor("Marta", 26, Carnet.MOTO);
      Vehiculo[] vehiculos = { new TurismoLujo(120),     // Vehículo 1: turismo de lujo, 120 EUR/dia
                               new Moto(60, 250),        // Vehículo 2: moto, 60 EUR/dia, 250cc
                               new Turismo(80),          // Vehículo 3: turismo normal, 80 EUR/dia
                               new Moto(50, 125) };      // Vehículo 4: moto, 50 EUR/dia, 125cc
      for (Conductor c : Arrays.asList(pedro, marta))
        for (Vehiculo v : vehiculos)
          precios.put("Si " + c.getNombre() + " alquila " + v, v.precioAlquiler(c));

      for (String k : precios.keySet()) System.out.println(k +" su precio es: " + precios.get(k));

      Vehiculo lujoso = new TurismoLujo(190);  // turismo de lujo, 190 EUR/día
      System.out.println("¿lujoso es alquilable por Pedro? " + lujoso.alquilablePor(pedro));
      System.out.println(lujoso.precioAlquiler(new Conductor ("Ana", 27, Carnet.CAMION)));

      List<Vehiculo> vehiculosOrdenados  = Arrays.asList(vehiculos);
      Collections.sort(vehiculosOrdenados);
      System.out.println( Vehiculo.numVehiculos()+" vehiculos creados: "+ vehiculosOrdenados );
   }
}
```

Salida esperada:

```
Si Marta alquila Veh. num 1 su precio es: Infinity
Si Marta alquila Veh. num 2 su precio es: 57.0
Si Marta alquila Veh. num 3 su precio es: Infinity
Si Marta alquila Veh. num 4 su precio es: 48.0
Si Pedro alquila Veh. num 1 su precio es: Infinity
Si Pedro alquila Veh. num 2 su precio es: 57.0
Si Pedro alquila Veh. num 3 su precio es: 81.0
Si Pedro alquila Veh. num 4 su precio es: 50.0
¿lujoso es alquilable por Pedro? false
192.0
5 vehiculos creados: [Veh. num 4, Veh. num 2, Veh. num 3, Veh. num 1]
```

Apartado 2 (3 puntos)

Se quieren construir unas clases Java para la creación de facturas. Una factura puede contener *líneas de venta* y *descuentos*, y agrupa elementos relacionados en *grupos de venta*. Los grupos de venta tienen un nombre y a su vez pueden contener líneas de venta, descuentos y grupos de venta. Una línea de venta viene descrita por un nombre de elemento, un precio unitario, número de unidades que se compran e IVA soportado. Un descuento se describe mediante un concepto y la cantidad que se descuenta. Los grupos de venta no admiten duplicados, considerándose iguales dos elementos con el mismo nombre.

Las facturas deben poder imprimirse con el formato que se muestra más abajo, así como validarse mediante el método **`validar()`**. Dicho método devuelve **`true`**, si el contenido de la factura es correcto, o lanza una excepción, que indica el tipo de error dependiendo del tipo de excepción. Por simplificar considera sólo dos tipos de errores: *Facturas negativas*, cuando los descuentos exceden los gastos, o *grupo de venta vacío*, cuando un grupo no tiene elementos.

Se pide:

(a) Utilizando patrones de diseño, completa el siguiente programa, para que la salida sea la de más abajo. (**2,75 puntos**)

(b) ¿Qué patrón o patrones has usado?, indícalo en el código y explica brevemente qué decisiones has tomado a la hora de implementarlos. (**0,25 puntos**)

```
class Factura {
	private String nombre;
	private GrupoVenta gv;

	public Factura(String n, GrupoVenta g) {
		this.nombre = n;
		this.gv = g;
	}
	// Completar…
}
// Completar…

public class Main {
  public static void main(String[] args) {
	GrupoVenta hardware = new GrupoVenta("Hardware");
	LineaVenta impresora = new LineaVenta("Impresora", 730, 2, 28);
	hardware.add(impresora);
	hardware.add(new LineaVenta("Altavoces", 140, 4, 28));
	GrupoVenta ordenadores = new GrupoVenta("Ordenadores");
	ordenadores.add(new LineaVenta("HP Proliant", 1730, 1, 28));
	ordenadores.add(new LineaVenta("Toshiba - Satellite 15.6'", 529.99, 2, 28));
	ordenadores.add(new Descuento("Cliente UAM", 100));
	ordenadores.add(new Descuento("Cliente UAM", 10));  // Repetido, no se admite
	hardware.add(ordenadores);
	Factura f = new Factura("Pepe Pérez", hardware);
	hardware.delete(impresora);
	System.out.println(f);
	hardware.add(new GrupoVenta("Pen Drives"));
	try {
		f.validar();
	} catch (FacturaExcepcion fe) {
		System.out.println(fe);
	}
   }
}
```

Salida:

```
Factura a nombre de: Pepe Pérez
 * Hardware. Total=4187.9744€ {
 + Altavoces (4 unidades): 716.8 €
 * Ordenadores. Total=3471.1744€ {
 + HP Proliant (1 unidades): 2214.4 €
 + Toshiba - Satellite 15.6' (2 unidades): 1356.7744 €
 - Cliente UAM (descuento): 100.0 €
 }
 }
El grupo de venta Pen Drives está vacio
```

Apartado 3 (2.5 puntos)

Se desea diseñar un sistema genérico para crear flujos de trabajo en Java. Un flujo de trabajo (*workflow*) está definido por varias tareas (tipo *Task*) conectadas de forma que al introducir información de entrada en la primera tarea se obtiene una salida que es a su vez la entrada para la segunda tarea, y así sucesivamente.
Cada tarea tendrá un método que recibe una entrada y la transforma en una salida. Para conectar tareas se crearán conectores que son tipos especiales de tareas que implementarán la lógica de diversas formas de conexión.
Tu programa debe considerar dos tipos de conexiones:

a) *SerialConnector*, que conecta dos tareas, de forma que la salida de la primera es la entrada de la segunda. Esta clase usa tres parámetros genéricos, el tipo de la entrada, el tipo intermedio y el tipo de salida.
b) *ListConnector*, que aplica una misma tarea a una lista de entradas, produciendo una lista de salidas que tendrá el mismo tamaño que la lista de entradas. Los parámetros genéricos son el tipo de los elementos de la lista de entrada y el tipo de elementos de la lista de salida.

Dadas las siguientes clases de tareas de utilidad `Max` y `String2Double` (cada una definida en su fichero .java correspondiente), **se pide**: completar el siguiente programa, para que la salida sea la de más abajo, implementando las clases o interfaces necesarias y rellenando los espacios indicados

```
public class Max<T extends Comparable<T>>  implements Task<List<T>, T> {
	@Override
	public T execute(List<T> input) {
		T result = null;
		for (T i:input){
			if (result == null || result.compareTo(i) < 0){
				result = i;
			}
		}
		return result;
	}

}
```

```
public class String2Double implements Task<String, Double> {
	@Override
	public Double execute(String input) { return Double.parseDouble(input); }
}
```

```
public class Main {                                    // completar espacios subrayados
	public static void main(String[] args) {
		Max<Double> maxDouble= new Max<Double>();
		Max<String> maxString= new Max<String>();
		Task<_________________________> toDouble = new String2Double();
		Task<_________________________> workflow =
			new SerialConnector<List<String>, ________________ , Double>(
					new ListConnector<String, Double>(toDouble),
					maxDouble);
		String[] test = {"3.2", "1", "19", "0"};
		System.out.println(workflow.execute(Arrays.asList(test)));
		System.out.println(maxString.execute(Arrays.asList(test)));
	}
}
```

Salida:

```
19.0
3.2
```

Apartado 4. (2 puntos)

Una *Universidad* quiere construir una aplicación informática para la confección de los horarios de las distintas asignaturas de grado de sus titulaciones. Un *grado* tiene una duración de 4 años, cada año con dos *cuatrimestres*. Una *asignatura* está descrita por un *nombre*, *abreviatura*, *curso*, *cuatrimestre* en el que se imparte y créditos ECTS. Las asignaturas pueden ser de *teoría* solamente, solamente de *prácticas* o teórico-prácticas. Una asignatura de teoría tiene uno o más grupos de teoría que vienen descrito por un código de 3 cifras. Las asignaturas con prácticas tienen uno o más grupos de prácticas, que tienen un código de 4 cifras. En ambos casos, cada grupo tiene asignado un *idioma* de impartición, así como un *horario* (con los días de la semana y hora de inicio y fin en que se imparte), un lugar de impartición y un profesor. Los profesores vienen descritos por un nombre, e-mail y teléfono de contacto, así como un número de despacho. Un profesor puede impartir de uno a ocho grupos de teoría o prácticas.

Se pide:

(a) Un diagrama de clases que represente el enunciado anterior. **(1.4 puntos)**

(b) Añade los siguientes métodos a la(s) clase(s) adecuada(s) de tu diagrama de clases, indicando sus parámetros de entrada y valor de retorno: **(0.6 puntos)**

   a. Un método que compruebe si hay solapamiento de horario entre las asignaturas de un determinado curso y cuatrimestre.
   b. Un método que calcule la media de horas lectivas semanales de prácticas de cierto curso de una determinada titulación.
   c. Un método que añada un grupo de prácticas a una asignatura.

Nota: No hace falta incluir código Java, ni constructores ni métodos *getters* y *setters* en el diagrama de clases.