PROGRAMACIÓN **1º TRIMESTRE**
MARTES 15 Decembro 2015
NOME:

Unha empresa de paquetería desexa informatizar o seu sistema de envío de paquetes decidíndose por facer emprego da linguaxe Java. As características da mercancía a enviar a ter en conta son:

Según o peso, os paquetes poden ser:

Categoría A (hasta 5Kg)

Categoría B (hasta 20Kg)

Categoría C (+20 Kg)

*Según o volume, os paquetes serán (o volume se calculará largo*ancho*alto):*

Categoría 1 (hasta 15625cc) (25x25x25 cm)

Categoría 2 (hasta 125000cc) (50x50x50 cm)

Categoría 3 (hasta 1000000cc) (100x100x100 cm)

Categoría 4 (+1000000cc)

Os destiños estarán clasificados en Zona 1, Zona 2 ou Zona 3

O precio de envío se calculará de modo automático. Partindo dun precio base común de **50€**, se lle vai ir acumulando un coste por volume, peso e destiño. Os sobrecostes serán:

Peso	**Volume**	**Destiño**
Categoría A – 0 €	Categoría 1 – 0 €	Zona 1 - 0€
Categoría B – 10 €	Categoría 2 – 5 €	Zona 2 -5€
Cateegoría C – 30 €	Categoría 3 – 15 €	Zona 3 - 8€
	Categoría 4 – *0.08 cts por cada cc que exceda de 1000000cc*	

Si o envío é urxente, o sobrecoste adicional será de 10€.

Se pide:

1.- Elaborar a Clase Dimension, que servirá para xestionar o volume dos paquetes. Esta clase terá os seguintes atributos:

```
float ancho;        // Anchura do paquete (en cm)
float alto;         // Altura do paquete (en cm)
float profundo;     // Profundidade do paquete (en cm)
double volume;      // Volume en ccc do paquete
int categoria;      // Categoría do paquete segundo o seu volume
```

E os seguintes métodos públicos:

// Constructor. Almacenará os valores correspondentes nos atributos facendo os cálculos necesarios
Dimension(float ancho,float alto,float profundo);
// Devolverá a categoría que corresponde a esta dimensión
int getCategoria();
// Devolverá o sobrecusto que implica esta dimensión de paquete
double getSobrecusto();

2.- Elaborar a Clase **Paquete** que servirá para representar os obxectos paquete que se van a enviar. Esta clase debe ter os seguintes atributos:

```
Dimension d;        // Dimensións do paquete en cm (ancho, largo, e profundo)
float peso;         // Peso do paquete en kg
long codigo;        // Código do paquete[1].
```

E os seguintes métodos públicos:

// Constructor. Almacena os valores nos atributos xenerando o obxecto Dimension e calculando o código[1]
Paquete(float ancho,float alto,float profundo, float peso);
// Devolverá a categoría que lle corresponde ao paquete segundo o seu peso
char getCategoria();
// Devolverá o sobrecusto do paquete debido ao seu peso e volume
double getSobrecusto();
// Devolverá o obxecto Dimension coa información referente ao volume do paquete
Dimension getVolume();
// Devolverá o código de 10 díxitos do paquete
long getCodigo();

[1]O código do paquete será calculado automáticamente, e estará formado polos díxitos do ano, seguido de dous díxitos indicando o mes, dous indicando o día e 4 díxitos indicando o número de paquete. ***Para ir levando conta do número de paquete, se debe utilizar un atributo estático privado.***
Para o resto do código se fará uso da clase ***java.util.Calendar*** *documentada nas seguintes páxinas.*

3.- Elaborar a clase Envio, que se utilizará para xestionar os envíos dos paquetes. Esta clase terá os seguintes atributos :

```
Paquete paquete;        // Paquete a enviar
Localidade localidade;  // Localidade de destiño
String dirección;       // Dirección de destiño (rúa)
String telefono;        // Telefono de contacto
boolean urxente;        // Indica si o envío e urxente ou normal
String codigo;          // Codigo de envío do paquete.
```

E os seguintes métodos públicos:

// Constructor. Crea o obxecto almacenando a información nos atributos[2].
Envio(Paquete paquete,int codpostal,String direccion,String telefono,boolean urxente);
// Devolve o importe total a pagar polo envío do paquete.
double getCusto();
// Devolve o código do envío[2]
String getCodigo();

[2]*O codigo de envío terá o seguinte formato (é unha cadea de texto sen espacios): (U)Rxente ou (N)ormal, Categoria de Volume, Categoria de Peso, Zona de Envío,Codigo Postal e Código de Producto.*

Un exemplo sería o seguinte:

U 1 A 2 36940 201507050001

(Urxente,hasta* 15625cc, hasta 5Kg, envío a Zona 2, codigo postal 36940 e código de producto *201507050001).

Se supon que a clase ***Localidade*** xa está programada, cos métodos e formato documentados nas seguintes páxinas.

4.- Escribir un programa Java que tras solicitarlle ao usuario:

alto, ancho e profundo do paquete.

Peso do paquete

Tipo de envío (normal/urxente)

Datos de envío (codigo postal, dirección (rúa), teléfono de contacto) .

Cree un obxecto Envio e visualice na pantalla o código de envío, datos de envío (Poboación, Dirección (rúa), código postal, Provincia, teléfono de contacto) e o importe total a pagar.

***NOTA:* Podes crear os métodos e atributos adicionais que consideredes oportunos. Únicamente se indican no enunciado os métodos públicos que deben estar presentes.**

Si pensas que é necesario engadir algún modificador a algún dos métodos, faino. NON necesitarás en ningún caso alterar os parámetros.

Supoñemos que xa están implementadas as clases ***Localidade*** e ***LocalidadeException.***

Criterios de avaliación. Total 10 puntos.

Parte 1 – Dimensión	***1.5 puntos.***	
- A clase está definida correctamente e cos modificadores de acceso axeitados		*0.25*
- A clase calcula correctamente o sobrecoste		*1*
- A clase calcula correctamente a categoría do volume		*0.25*
Parte 2 – Paquete	***2.5 puntos.***	
- A clase está definida correctamente e cos modificadores de acceso axeitados		*0.25*
- A clase calcula correctamente o sobrecoste debido ao peso		*0.25*
- A clase calcula correctamente o código de paquete		*1*
- O obxecto se constrúe correctamente		*1*
Parte 3 – Envío	***3 puntos.***	
- A clase está definida correctamente e cos modificadores de acceso axeitados		*0.25*
- A clase se constrúe correctamente		*0.25*
- Se xestionan correctamente as condicións de erro		*0.50*
- O Código de Envío se calcula correctamente		*1*
- Se calcula correctamente o custo de envío		*1*
Parte 4 – Programa	***2 puntos.***	
- O programa solicita correctamente os datos e visualiza un resultado correcto		*1*
- O programa realiza unha xestión de erros correcta		*1*
Claridade, Comentarios e Estructura do Código	***1 punto.***	

Documentación da clase Localidade

class Localidade {

```
/** Constructor – Construe o obxecto Localidade, si a empresa non traballa na localidade indicada polo código
postal, se lanza a excepción
*/
Localidade (int codpos) throws LocalidadeException {
}

/** Devolve o Código Postal correspondente a esta Localidade
*/
int getCodigoPostal() {
}

/** Devolve o número de zona correspondente á Localidade
*/
int getZone() {
}

/** Devolve o nome da poboación que corresponde con esta Localidade
*/
String getPoboacion() {
}

/** Devolve o nome da provincia correspondente con esta Localidade
*/
String getProvincia() {
}

/** Devolve o sobrecoste que acarrea esta localidade
*/
int getSobrecoste() {
}
```

}

A clase ***LocalidadeException*** hereda de *Exception*, e non proporciona ningún método adicional. Suponse que xa está programada.

Documentación necesaria da clase java.util.Calendar

`static Calendar getInstance()`

A clase calendar non utiliza o constructor para crear obxectos coa data actual, en lugar de esto fai uso dun método estático que se encarga de crear e devolver o Calendar. Para crear un obxecto Calendar coa data actual faríamos:

Calendar hoxe=Calendar.getInstance();

logo xa poderíamos utilizar os métodos do obxecto. Por exemplo a línea:

int ano=hoxe.get(Calendar.YEAR);

deixaría en ano o valor do ano actual.

`int get(int field)`

Devolve o valor do campo indicado en field. Algúns dos valores que se poden indicar en field son:

Calendar.MONTH (obten o número do mes, 0 é Xaneiro)
Calendar.DAY_OF_MONTH (obten o número do día)
Calendar.YEAR (obten o ano como número de 4 díxitos)