UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica etsinf

Ejercicios Tema 3

Programación (PRG)

Departamento de Sistemas Informáticos y Computación

CC BY NC SA

Ejercicio Herencia: Yogures

- Diseña la clase **Yogur** sabiendo que un yogur siempre tiene 120.5 calorías.
  - Se requiere implementar un método consultor para obtener sus calorías.
- A continuación, diseña la clase **YogurDesnatado** sabiendo que siempre tiene la mitad de calorías que un Yogur normal.
- Finalmente, construye una clase **TestYogures** con un método principal que cree un objeto Yogur y un YogurDesnatado y muestre sus calorías.

Ejercicio Herencia: Felino

- Dada la clase **Felino**, diseñar la clase **Gato** de manera que cuando hable, emita un maullido.

```
public class Felino {
   protected String loQueHablo;
   public Felino(){
            loQueHablo = “Soy un Felino”;
   }
   public String habla(){ return loQueHablo;}
}
```

- La solución deberá incorporar algún mecanismo de herencia para reutilizar el código ya desarrollado (clase **Felino**).

Ejercicio: Clase Cilindro (I)

- Dadas las siguientes clases Java:

```
public class Circulo {
        private String tipo;
                double radio;

        public Circulo(double r) {
                this.radio = r;
                this.tipo = ''Circulo'';
        }
        public Circulo(double r, String t) {this.radio = r; this.tipo = t;}

        public double area(){ return Math.PI * radio * radio;}
        public double perimetro(){ return 2 * Math.PI * radio;}
        public String toString(){ return ''Circulo de radio ''+radio+''\n'';}
}
```

Ejercicio: Clase Cilindro (II)

```
public class Cilindro extends Circulo {
    private double altura;

    public Cilindro(double radioBase, double altura){...}

     public double area(){
        return 2*Math.PI*radio*radio + 2*Math.PI*radio*altura;
     }
     public double volumen(){
         return Math.PI*radio*radio*altura;
     }
}
```

Ejercicio: Clase Cilindro (III)

1. En la clase Circulo, ¿Qué modificador de visibilidad se le debe de asignar al atributo radio para que sea accesible desde la clase Cilindro y favorezca el principio de Ocultación de Información?
2. De las siguientes implementaciones del Constructor de Cilindro, una de ellas es incorrecta, ¿Cuál?, ¿Por qué?
   a) super(radioBase, “Cilindro”); this.altura=altura;
   b) super.radio = radioBase; super.tipo = “Cilindro”; this.altura=altura;
3. En la clase Cilindro, modificar las implementaciones de los métodos área y volumen para favorecer el principio de reutilización de software.
4. Redefínase la clase Cilindro para que en lugar de SER UN Circulo, tenga un Circulo como base. No se permite modificar la especificación del constructor de Cilindro.

Actores y Películas: Errores

- Corrige el código para que *mostrarReparto* funcione:

```
public class Persona {
   private String nombre;
   public Persona(String nombre){
     this.nombre = nombre;
}}
```

```
public class Actor extends Persona {
private String pelicula;
public Actor(String nombre, String pelicula) {
   this.nombre = nombre;
   this.pelicula = pelicula;
}}
```

```
public class Peliculas {
  public static void mostrarReparto(Actor[] lista, String pelicula) {
   for (int i = 0; i <= lista.size; i++)
      if (lista[i].pelicula == pelicula)
            System.out.println(lista[i].toString());
   }
}
```

BonoMetro con Excepciones

1. Diseña la clase BonoMetro que contiene un único atributo llamado *saldo* y un método *fichar* que reduce el saldo disponible en una unidad siempre que hubiera saldo. En caso contrario se debe lanzar la excepción SaldoAgotadoException.
    1. *El saldo por defecto de un BonoMetro es de 10 viajes.*
2. Implementa la excepción SaldoAgotadoException.
3. Implementa una clase TestBonoMetro que verifique la funcionalidad previamente implementada.

Transferencia de Ficheros

- La siguiente definición de clase permite realizar la transferencia de un archivo a otra máquina mediante FTP:

```
public class CopyViaFTP{
   public static void copyTo(String hostName, String localFilePath) throws
UnableToTransferException {
    …
   }
}
```

- Escribir un programa que realice la transferencia del fichero /tmp/data a la máquina fileserver.upv.es.
- En caso de fallo, la operación se deberá reintentar un máximo de 3 veces e indicar al usuario el nº de intento.

Módulo de Autorización (I)

- La siguiente clase implementa un módulo de autorización basado en usuarios y contraseñas registrados:
- Clase **AuthModule**
  - public static void **check**(String username, String password) throws **InvalidUserException**, **InvalidPasswordException**, **ExpirationDeadlineException**
- Excepciones lanzadas:
  - InvalidUserException: Si el *username* no existe.
  - InvalidPasswordException: El *username* existe, pero la contraseña no coincide con la registrada en el sistema.
  - ExpirationDeadlineException (extends RuntimeException): La contraseña caducará en breve.

Módulo de Autorización (II)

- Se pide:
- Utilizar la clase AuthModule para implementar el siguiente método :
  - public static void **grantAccess**(String username, String password) throws **AccessDeniedException**;
- El método *grantAccess:*
  - Lanza la excepción en caso de que el nombre de usuario no exista o la contraseña no coincida con aquella registrada.
  - Debe mostrar por la salida estándar mensajes de información al usuario sobre el proceso de autorización.
- Asumir que todas las clases mencionadas ya están implementadas.

Ejercicio: Carnet Por Puntos (1/3)

- Dadas las siguientes clases:

```
public class CarnetDeConducir {
  protected String nombre;    protected int puntos;
  public CarnetDeConducir(String nombre ) {
     this.nombre=nombre; puntos = 12;
   }
  public String toString( )  {
      return nombre + "  (" + puntos + " puntos" + ")";
  }
  public final String getNombre( ) {
    return nombre;
  }
  public final void quitarPuntos ( int penalizacion ) {
    puntos -= penalizacion;
  }}
```

Ejercicio: Carnet Por Puntos (2/3)

```
public class DGT {
   public static void multar(CarnetDeConducir c, Scanner teclado) {
      System.out.println("Introduce la penalización:");
      int penalizacion = teclado.nextInt();
      c.quitarPuntos(penalizacion);
   }
}
```

- Un Carnet de Conducir tiene un crédito inicial de 12 puntos que se va perdiendo a medida que se cometen infracciones. Un saldo de puntos cero o negativo implica una Retirada Inmediata del Carnet de Conducir

Ejercicio: Carnet Por Puntos (3/3)

- Se pide:

1. Definir la excepción comprobada de usuario *RetiradaInmediataCarnet.*
2. Modificar el diseño actual del método *quitarPuntos* de la clase *CarnetDeConducir* para que, cuando el saldo de puntos de un Carnet de Conducir sea negativo o cero tras la penalización, lance la excepción *RetiradaInmediataCarnet*.
3. Modificar el método *multar* de la clase DGT para que muestre un mensaje de error por pantalla si la penalización de puntos comporta la retirada inmediata del carnet.