Ejercicios Tema 2

Ejercicios Adaptados de Apuntes y Exámenes de EDA
Germán Moltó
gmolto@dsic.upv.es
Estructuras de Datos y Algoritmos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

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.

Excepción en Base de Datos (I)

- Dada las siguientes definiciones de clases Java

```
public class ImposibleAbrirBD extends Exception {
}
```

```
public class BaseDeDatos {
   public BaseDeDatos(String nombre){...}
   private void abrir() throws ImposibleAbrirBD{...}
   public void inicializar() throws ImposibleAbrirBD{
      abrir();
      ...
   }
}
```


Excepción en Base de Datos (II)

```
public class TestBaseDeDatos {
   public static void main (String args[]){
      BaseDeDatos bd = new BaseDeDatos("MiBD");
      bd.inicializar();
   }}
```

- Se pide modificar el código de TestBaseDeDatos para que gestione la excepción de usuario ImposibleAbrirBD como sigue:
  - Notificar al usuario que no ha sido posible inicializar la Base de Datos mediante el mensaje "Imposible inicializar la Base de Datos" y, además, intentar volver a inicializar la Base de Datos en cuestión hasta un máximo de tres veces consecutivas.

Ejercicio Excepciones (I)

- Dado el siguiente código:

```
public class EjercicioExcepciones {
   public static String leer(){...}

public static void iniciaVector(int v[], int n, Random r) throws
   IllegalArgumentException{
         for (int i=0; i < v.length; i++) v[i] = r.nextInt(n);
}

   public static void actualizaInicia(int posicion)
    throws ArrayIndexOutOfBoundsException,IllegalArgumentException{
         int [] array = new int[3];
         array[posicion] = 25;
         Random r = new Random();
         iniciaVector (array, posicion, r);
   }
}
```


Ejercicio Excepciones (II)

```
public static void main(String [] args){
   String leido = leer();
   int denominador = Integer.parseInt(leido);
   int cociente = 42/denominador;
   System.out.println("El cociente de 42/"+denominador+" es:
   "+cociente);
   actualizaInicia(denominador);
}
```

- Se pide:
  - Modifica el método main para que se avise al usuario con un mensaje si la ejecución produce algún fallo.

Ejercicio: Carnet Por Puntos (1/3)

- Dadas las siguientes clases:

```
public interface CarnetPorPuntos {
    void quitarPuntos ( int penalizacion );
}
```

```
public class CarnetDeConducir implements CarnetPorPuntos  {
  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.