Problema 5. Cuestiones sobre Excepciones

Objetivos

- Afianzar los conceptos relativos a las excepciones.
- Practicar el patrón de tratamiento de excepciones recuperables.

Desarrollo

Responder a las siguientes cuestiones.

Cuestión 1

(Apareció en el examen de septiembre de 2008)

Se dispone de la clase `Compuerta` ya realizada que tiene los métodos mostrados a continuación (la clase tiene más métodos, pero no son relevantes para el problema planteado):

abre

```
public void abre(int gradoApertura) throws Averiada
```

Abre la compuerta el grado de apertura indicado

Parameters:

gradoApertura grado de apertura deseado

Throws:

`Averiada` si se detecta que el mecanismo de la compuerta está averiado

gradoApertura

```
public int gradoApertura()
```

Retorna el grado de apertura actual de la compuerta

Returns:

grado de apertura

Se ha detectado un mal funcionamiento intermitente del mecanismo de apertura de las compuertas, de forma que puede ocurrir que el método `abre` lance la excepción `Averiada` en una llamada y abra la compuerta (sin lanzar la excepción) en la siguiente invocación. También se ha observado que en ocasiones la apertura lograda por alguna compuerta (leída con `gradoApertura`) no es la solicitada en el método `abre`.

Escribir un nuevo método de la clase `Compuerta` que permita corregir en lo posible esta situación. El método recibirá como parámetros el grado de apertura y el número de intentos de apertura (llamadas al método `abre`) que se desea realizar y finalizará al detectar que se ha alcanzado el grado de apertura solicitado. Deberá lanzar la excepción `Averiada` si en todos los intentos de apertura se ha lanzado la excepción `Averiada`. Si al menos uno de los intentos no lanzó la excepción `Averiada`, pero la apertura lograda después de todos los intentos no es la solicitada, deberá lanzar la excepción `AperturaIncorrecta`.

(Se proporciona la clase `Compuerta` en la página web de la asignatura)

Solución Cuestión 1

Alternativa 1 (con `while`)

```
public void abre(int gradoApertura, int intentos)
                            throws Averiada, AperturaIncorrecta {
  boolean siempreAveriada = true;
  int i = 0;

  // trata de abrir el número de intentos indicados o hasta que la
  // apertura sea correcta
  while ((i < intentos) && (gradoApertura() != gradoApertura)) {
    try {
      abre(gradoApertura);
      siempreAveriada = false;
    } catch (Averiada e) {
      // no hace nada, simplemente reintenta
    }
    i++;
  }

  // comprueba la razón por la que ha salido del lazo
  if (siempreAveriada)
    throw new Averiada();
  else if (gradoApertura() == gradoApertura)
      return; // la compuerta se ha abierto correctamente
  else
    throw new AperturaIncorrecta();
}
```

Alternativa 2 (con `for`). Más lío que la alternativa 1.

```
public void abre(int gradoApertura, int intentos)
                          throws Averiada, AperturaIncorrecta {
  boolean siempreAveriada = true;

  // intentamos en número de veces indicada
  for (int i = 0; i < intentos; i++) {
    try {
      abre(gradoApertura);
      siempreAveriada = false;
      if (gradoApertura() == gradoApertura)
        break; // la compuerta se ha abierto correctamente
    } catch (Averiada e) {
      // no hace nada, simplemente reintenta
    }
  }

  // comprueba la razón por la que ha finalizado el lazo
  if (siempreAveriada)
    throw new Averiada();
  if (gradoApertura() != gradoApertura)
    throw new AperturaIncorrecta();
  // si no entra en ninguno de los dos "ifs" es porque se ha
  //abierto correctamente así que el método finaliza sin más
}
```

Cuestión 2

Indica la salida por consola que se produciría si se ejecuta el programa siguiente con "CASO=0", "CASO=1" y "CASO=2".

```
public class PropagaExcepciones {
  private static final int CASO=...;

  public static class MiExcepción extends Exception {}
  public static class MiRuntimeExcepción
                                  extends RuntimeException {}

  private static void método2(int caso) throws MiExcepción {
    switch (caso) {
    case 0:
      throw new MiExcepción();
    case 1:
      throw new MiRuntimeExcepción();
    }
  }

  private static int método1() throws MiExcepción {
    int ret = 1;
    try {
      System.out.println("1:antes");
      método2(CASO);
      System.out.println("1:después");
      return 2;
    } catch (RuntimeException e) {
      System.out.println("1:catch");
    } finally {
      System.out.println("1:finally");
    }
    System.out.println("1:final");
    return ret;
  }

  public static void main(String[] args) {
    System.out.println("main:antes");
    int ret=0;
    try {
      ret = método1();
      System.out.println("main:después");
    } catch (MiExcepción e) {
      System.out.println("main:catch");
    }
    System.out.println("main:final. ret:"+ret);
  }
}
```

(puede valer 0, 1 o 2)

Solución cuestión 2

- CASO=0

```
main:antes
1:antes
1:finally
main:catch
main:final. ret:0
```

- CASO=1

```
main:antes
1:antes
1:catch
1:finally
1:final
main:después
main:final. ret:1
```

- CASO=2

```
main:antes
1:antes
1:después
1:finally
main:después
main:final. ret:2
```