Unidad

3

DIPLOMATURA EN PROGRAMACION JAVA

Ejercicios



# Capítulo 6

# Excepciones y aserciones

# Capítulo 6

## *Ejercicio 1*

En este ejercicio se utilizará el bloque **try-catch** para manejar una simple rutina de excepción.

El programa VerificaExcepciones que se encuentra en Módulo 6\Ejercicio 1 imprime por pantalla los elementos con que se inicializa el vector vec, pero tiene un pequeño error, intenta acceder a los elementos más allá del último (el elemento 8, con subíndice 7) que es el límite del vector

```
public class VerificaExcepciones {
	public static void main(String[] args) {
		int vec[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
		for (int i = 0; true; i++) {
			System.out.println("vec[" + i + "] es '" + vec[i] + "'");
		}
	}
}
```

**Nota**: la mala implementación del código es intencional. Recordar que los vectores se recorren siempre verificando no exceder su largo máximo con la propiedad vec.length (en este caso). El propósito es experimentar con bloques que atrapen (**catch**) excepciones (exceptions).

1. Crear un espacio de trabajo en el directorio Capítulo 6 y llamarlo Ejercicio 1
2. Crear un proyecto Java y llamarlo Ejercicio 1 también
3. Compilar y ejecutar el programa.
4. Verificar que la salida es:

```
vec[0] es '1'
vec[1] es '2'
vec[2] es '3'
vec[3] es '4'
vec[4] es '5'
vec[5] es '6'
vec[6] es '7'
vec[7] es '8'
java.lang.ArrayIndexOutOfBoundsException: 8
	at VerificaExcepciones.main(VerificaExcepciones.java:5)
```

5. Modificar el programa para manejar la excepción ArrayIndexOutOfBoundsException colocando el ciclo **for** dentro de un bloque **try**
6. Escribir el bloque **catch** asociado con una referencia al objeto del tipo ArrayIndexOutOfBoundsException como argumento.

7. Dentro del bloque **catch** declarar System.out.println(e.getMessage()); para obtener e imprimir por pantalla el mensaje que devuelve la excepción e imprimir por pantalla un String que indique el nombre de la excepción ocurrida y que el programa va a terminar.
8. Forzar la terminación del programa con la sentencia System.exit(0);
9. Compilar y ejecutar el programa
10. Verificar que la salida sea

```
vec[0] es '1'
vec[1] es '2'
vec[2] es '3'
vec[3] es '4'
vec[4] es '5'
vec[5] es '6'
vec[6] es '7'
vec[7] es '8'
8
Se produjo la excepción ArrayIndexOutOfBoundsException
El programa finalizó por esta causa
```

11. Genere un nuevo proyecto en el espacio de trabajo en el que se encuentra. Llamarlo Excepciones2
12. Copie la clase VerificaExcepciones del proyecto Excepciones.
13. Elimine el manejo de la excepción para dejar la clase en su estado primario, como se muestra a continuación:

```
public class VerificaExcepciones {
	public static void main(String[] args) {
		int vec[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
		for (int i = 0; true; i++) {
			System.out.println("vec[" + i + "] es '" + vec[i] + "'");
		}
	}
}
```

14. Diseñe una aseveración que finalice el programa antes que se exceda el límite del vector. Es decir, el programa deberá finalizar por un error de **assert** y no por el lanzamiento de la excepción ArrayIndexOutOfBoundsException.
15. Para ejecutar el programa, seleccionar el menú Run -> Run Configurations. En la ventana que se despliega, seleccionar la pestaña (x)= Arguments y en el editor debajo del título VM Arguments escribir –ea para activar la verificación de las aseveraciones. Luego presionar el botón Run para ejecutar el programa.

**Nota**: Si se desea depurar el programa, se debe realizar el mismo procedimiento pero seleccionando Run -> Debug Configurations

16. Verificar que al ejecutar el programa la salida sea como la siguiente:

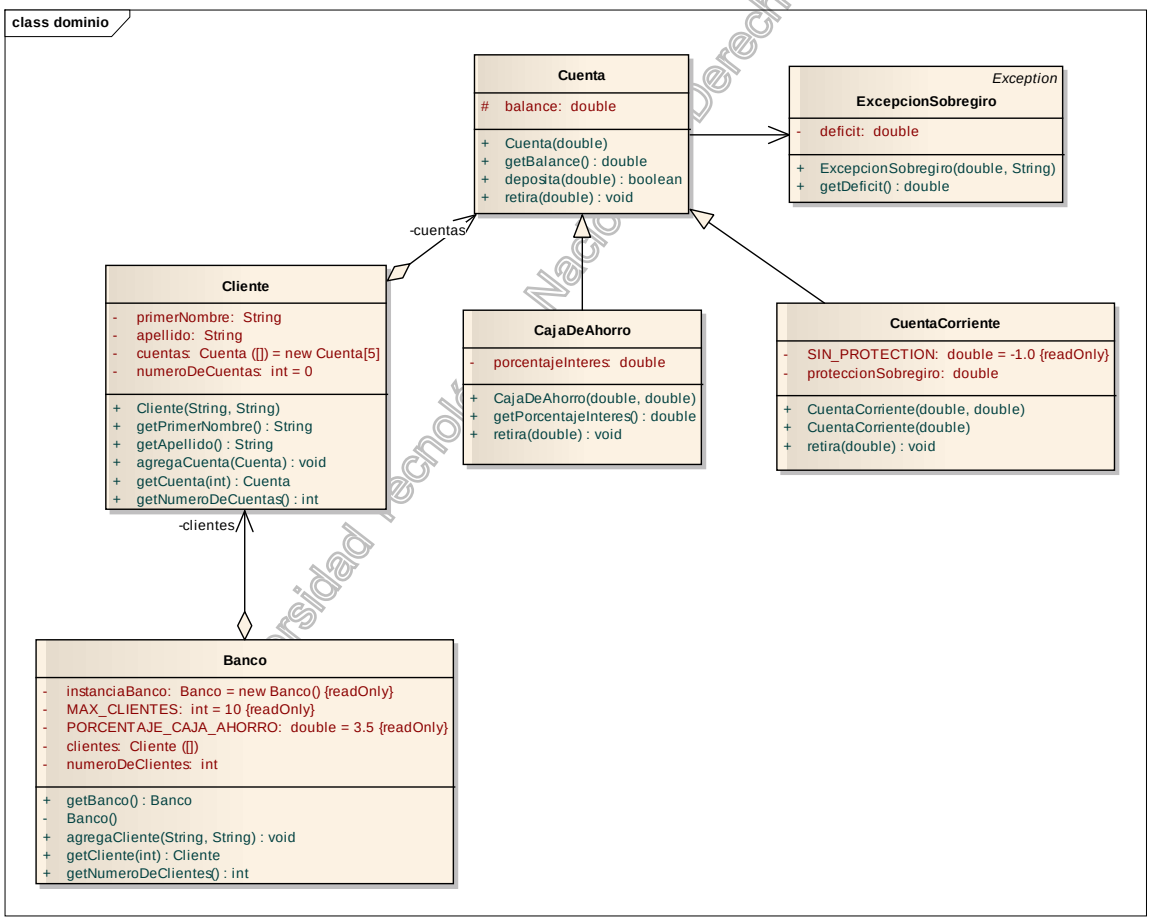
```
vec[0] es '1'
```

```
vec[1] es '2'
vec[2] es '3'
Exception in thread "main" vec[3] es '4'
vec[4] es '5'
vec[5] es '6'
vec[6] es '7'
vec[7] es '8'
java.lang.AssertionError
        at VerificaExcepciones.main(VerificaExcepciones.java:5)
```

17. Probar modificar la aseveración para que se produzca primero la excepción antes de fallar al validar la misma.

## *Ejercicio 2*

En este ejercicio se creará una excepción llamada ExcepcionSobregiro la cual es lanzada por el método retira de la clase Cuenta. Las clases a obtener figuran en el siguiente diagrama UML



1. Abrir el espacio de trabajo del directorio Capìtulo 6\Ejercicio 2. Los errores que figuran luego de importar los archivos se deben a las modificaciones que deberán realizarse.

## Crear la clase ExcepcionSobregiro

2. Crear una clase pública llamada ExcepcionSobregiro en el paquete operacionesBancarias.dominio. Esta clase hereda de la clase Exception.
3. Agregar un atributo privado llamado déficit que pueda almacenar un `double`
4. Agregar un método de acceso público llamado getDeficit() .
5. Agregar un constructor público que reciba dos parámetros: mensaje y deficit. El parámetro mensaje deberá pasarse al constructor de la superclase. El parámetro déficit inicializa el atributo deficit.

## Modificar la clase Cuenta

6. Modificar el método retira() :
    a. Volver a escribir el método para que no devuelva valores, (esto es, que sea `void`).
    b. Declarar que este método lanza una excepción llamada ExcepcionSobregiro.
7. Modificar el código para lanzar la nueva excepción especificando "Fondos Insuficientes" en el mensaje y el déficit (lo que le falta para estar dentro del acuerdo de giro descubierto).

## Modificar la clase Caja de Ahorros

8. Modificar el método retira
    a. Volver a escribir el método para que no devuelva valores, (esto es, que sea `void`).
    b. Declarar que este método lanza una excepción llamada ExcepcionSobregiro. La misma se debe lanzar cuando la cantidad a retirar supera al balance. En dicho caso lanzar la excepción con el mensaje "Fondos insuficientes" y conservar el valor del balance

## Modificar la clase CuentaCorriente

9. Modificar el método retira() :
    a. Volver a escribir el método para que no devuelva valores, (esto es, que sea `void`).
    b. Declarar que este método lanza una excepción llamada ExcepcionSobregiro.
10. Modificar el código para lanzar una excepción si es necesario. Hay dos casos que necesitan ser manejados. Primero, si hay un déficit pero no se puede cubrir con la protección de giro en descubierto para la cuenta de ahorro. En este caso utilizar un

mensaje "No hay protección por sobregiro". Segundo, la cantidad que almacena proteccionSobregiro es suficiente para cubrir el déficit, utilizar el mensaje "Fondos insuficientes para proteger el sobregiro"

11. Compilar y ejecutar el programa
12. Verificar que la salida sea:

```
El Cliente [Perez, Juan] tiene un balance en cuenta corriente de 200.0
con una protección por sobregiro de 500.00.
Cuenta Corriente [Juan Perez]: retira 150.00
Cuenta Corriente [Juan Perez]: deposita 22.50
Cuenta Corriente [Juan Perez]: retira 147.62
Cuenta Corriente [Juan Perez]: retira 470.00
Excepción: Fondos insuficientes para proteger el sobregiro   Deficit: -
470.0
El Cliente [Perez, Juan] tiene un balance en cuenta corriente de 0.0

El Cliente [Toma, Oscar] tiene un balance en cuenta corriente de 200.0
Cuenta Corriente [Oscar Toma]: retira 100.00
Cuenta Corriente [Oscar Toma]: deposita 25.00
Cuenta Corriente [Oscar Toma]: retira 175.00
Excepción: No hay protección por sobregiro   Deficit: -50.0
El Cliente [Toma, Oscar] tiene un balance en cuenta corriente de 125.0
```