UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 12: Excepciones.

Profesora: Amparo López Gaona
Ayudante: Ramsés Antonio López Soto
Ayudantes de Laboratorio:
Adrián Aguilera Moreno
Kevin Jair Torres Valencia

Objetivos

A continuación se presentan los objetivos de esta práctica de laboratorio:

- Que el alumno se ejercite en el manejo de excepciones como un mecanismo para escribir programas robustos. Estas excepciones pueden ser de las proporcionadas por Java o bien desarrolladas para un programa particular. El manejo de las excepciones incluye su lanzamiento, atrapado y recuperación.

Introducción

Una excepción es un evento que ocurre en cualquier momento de ejecución de un programa y que modifica el flujo normal de éste. Las excepciones son objetos de la clase `Exception` que almacenan información que se regresa en caso de que ocurra una anormalidad. Todos los métodos que hayan llamado al método en que se produce la excepción pueden darse por enterados y alguno de ellos o todos tomar las medidas apropiadas.

La clase `Exception`, que se encuentra en el paquete `java.lang`, es la raız de una jerarquía de clases para los errores más comunes. En esta jerarqua se tiene la clase `RuntimeException` de la que se derivan varias clases de uso frecuente, por ejemplo, `NullPointerException`. Una excepción se activa (dispara) para indicar que ocurrió una falla durante la ejecución de un método. La excepción se propaga hasta encontrar un método en el cual se indica (atrapa) qué se debe hacer en circunstancias anómalas.

Para tratar con las excepciones es necesario escribir un manejador de excepciones utilizando la instrucción try que tiene la siguiente sintaxis:

```
try {
   instrucciones
}

catch (Excepción e) {
   instrucciones
}
...
catch (Excepción e) {
   instrucciones
}
finally {
   instrucciones
}
```

Cada cláusula de la instrucción `try` es un bloque que incluye las instrucciones que pueden disparar la(s) excepción(es). Las cláusulas `catch` tienen como parámetro un objeto de alguna clase de excepción. En una instrucción try puede haber varias cláusulas catch; en el bloque de cada una se coloca el código que implementa la acción a realizar en caso de que ocurra una excepción del tipo de su parámetro. Por último, la cláusula opcional `finally` contiene

el código para establecer un estado adecuado para continuar la ejecución del método donde aparece la instrucción `try`.

En ocasiones puede suceder que las clases de excepciones existentes no describan la naturaleza del error que se tiene; en ese caso y para dar mayor claridad a los programas es posible crear excepciones propias, esto se logra extendiendo la clase `Exception`. Todas las nuevas clases requieren que se proporcione una cadena de diagnóstico al constructor.

Desarrollo

1.-Crearán un programa simple que simule un sistema de gestión de cuentas bancarias. Deberán implementar el manejo de excepciones para situaciones como intentos de retirar más dinero del que hay en la cuenta o intentar acceder a una cuenta que no existe.

1. Crear la clase CuentaBancaria:
   (a) Donde tendrá los atributos: numeroCuenta, saldo.
   (b) Métodos:
   - depositar(double cantidad): Incrementa el saldo.
   - retirar(double cantidad): Disminuye el saldo, lanzando una excepción si la cantidad es mayor que el saldo disponible.
   - obtenerSaldo(): Retorna el saldo actual.
2. Crear la clase Banco:
   - Atributos: CuentaBancaria[] cuentas (un arreglo de cuentas bancarias de tamaño fijo).
   - Métodos:
     – agregarCuenta(CuentaBancaria cuenta, int indice): Agrega una nueva cuenta en un índice específico del arreglo.
     – retirarDeCuenta(int indice, double cantidad): Intenta realizar un retiro en la cuenta en el índice dado, lanzando ArrayIndexOutOfBoundsException si el índice no es válido.
3. Por último, crear una clase Main:

   Donde implementarán un menú simple que permita al usuario: Crear cuentas, Realizar depósitos y Realizar retiros.

Manejar las excepciones adecuadamente al realizar operaciones.

2.- Crearán un sistema que almacene las calificaciones de un número fijo de estudiantes en un arreglo. El programa permitirá agregar, modificar y mostrar las calificaciones, y deberá manejar excepciones para situaciones como el acceso a índices fuera de los límites del arreglo o la entrada de notas inválidas.

1. Crear la clase Estudiante:
   - Como atributos: nombre y nota.
   - Métodos:
     - asignarCalificacion(double calificacion): Establece la calificación del estudiante, lanzando una excepción si la calificación está fuera del rango (0-100).
     - obtenerCalificacion(): Retorna la calificación del estudiante.
     - obtenerNombre(): Retorna el nombre del estudiante.
2. Crear la clase Gestion:
   - Atributo: Estudiante[] estudiantes (un arreglo de estudiantes de tamaño fijo).
   - Métodos:
     - agregarEstudiante(Estudiante estudiante, int indice): Agrega un estudiante en un índice específico del arreglo, lanzando una excepción si el índice está fuera de los límites.
     - modificarCalificacion(int indice, double nuevaCalificacion): Modifica la calificacion del estudiante en el índice dado, lanzando excepciones si el índice no es válido o si la calificacion es inválida.
     - mostrarCalificaciones(): Muestra las calificaciones de todos los estudiantes
3. Por último, crear una clase Main:

   Donde implementarán un menú simple que permita al usuario: Agregar estudiantes, Modificar calificaciones y Mostrar calificaciones.

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. Cada práctica (sus archivos y directorios) deberá estar contenida en un directorio llamado `EquipoX_pY`, donde:
   (a) X es el número de equipo correspondiente.
   (b) Y es el número de la práctica.

   Por ejemplo: `Equipo09_p01`
3. NO incluir los archivos .class dentro de la carpeta.
4. Los archivos de código fuente deben estar documentados.
5. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
6. La práctica se debe subir al Github Classroom correspondiente.
7. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
8. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.