Excepciones y Aserciones

Objetivos: Que el alumno
- Gestione el tratamiento de errores en tiempo de ejecución con excepciones propias y predefinidas de Java, para desarrollar software robusto y de calidad.
- Aprenda a declarar aserciones como herramienta para depurar programas.

Excepciones predefinidas

1) Corrija la clase **PiedraPapelTijera** para que funcione correctamente. *Descargar el código de la clase PiedraPapelTijera del Aula Virtual.*
   a) Capture las posibles excepciones utilizando un bloque de región segura (**try-catch)**.
   b) Al capturar las excepciones, informe por pantalla el detalle del error.

   Tener en cuenta:
   - El Ingreso por teclado puede causar una excepción de tipo **IOException**.
   - El método Integer.parseInt() convierte números de tipo String a tipo **int**. Si la cadena no puede convertirse a entero, arroja una **NumberFormatException**.
   - Además, considerar en qué momento se debe restar la cantidad de intentos posibles.

2) Quite el manejador de excepciones para **NumberFormatException** y observe el error producido durante la ejecución, al ingresar valores inadecuados.

3) El bloque try-catch puede incluir una construcción finally. Este bloque de código siempre se ejecuta, independientemente de si se dio o no una excepción, y si se la atrapó o no. En la clase PiedraPapelTijera, incluya un bloque finally y muestre un mensaje "Fin del bloque – finally".

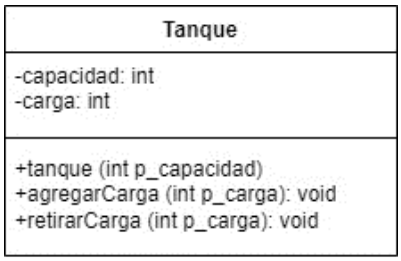
Excepciones definidas por el programador

4) Se desea agregar tratamiento de excepciones a la clase Pedido y TomaPedido desarrolladas previamente (TP4):
   a) Cree la excepción **PedidoInvalidoException** que hereda de la clase **Exception**. Esta clase sólo contiene un constructor que recibe como parámetro un mensaje descriptivo del error de tipo String.
   b) Modifique el constructor de la clase Pedidos para que lance la excepción de tipo **PedidoInvalidoException** cuando se reciba un ArrayList vacío como parámetro (trows-trow). Al lanzar la excepción se debe enviar como parámetro un mensaje descriptivo del error: "La lista de productos está vacía".
   c) Modifique la clase TomaPedidos para capturar la excepción **PedidoInvalidoException** utilizando un bloque de región segura (try-catch). En el bloque que captura la excepción mostrar por pantalla el mensaje de error:

      ****El pedido no se ha creado *****
      PedidoInvalidoException: *La lista de productos está vacía*

   d) En el método main crear un objeto Pedidos con un ArrayList vacío como parámetro.

5) Se desea manipular un tanque de usos múltiples (agua, nafta, etc). Se deben implementar los métodos para agregar/retirar líquido al tanque. Si excede la capacidad o el tanque está vacío, se debe lanzar la excepción **TanqueLlenoException** o **TanqueVacioException**, según corresponda.



Nota: No olvidar definir las clases correspondientes, que extienden a **Exception**.

6) Cree la clase ejecutable ControlarTanque, que permita ingresar por teclado la capacidad del tanque, y la carga inicial. Crear una instancia (por ejemplo: tanqueNafta). Luego solicitar cantidad a agregar/retirar, y aplicar los métodos correspondientes a la instancia creada. Manejar las posibles excepciones en un bloque try-catch.

Aserciones

7) Se desea verificar que la clase Pedido, desarrolladas previamente (TP4), cumple con la condición expresada en el diagrama UML. Esta condición establece que un pedido debe tener siempre al menos un producto. Para esto:
   a) En la clase **TomaPedidos** cree un pedido con un producto, introduciendo valores constantes. Luego intente quitar el producto del pedido.
   b) A continuación, agregue una línea de código para declarar una aserción: Si el ArrayList de productos no contiene al menos un producto debe lanzarse un **`AssertionError`** con la leyenda *"El pedido debe contener al menos 1 producto"*.

Tener en cuenta:

- La expresión en Java para declarar una aserción es:

  ```
  assert expresion : mensaje;
  ```

  Donde:

  expresion es una expresión booleana que se evalúa para verificar una condición.
  mensaje es un mensaje opcional que se muestra si la aserción falla, o sea, **expresión** es falsa.
- Cuando la aserción falla se lanzará una excepción **`AssertionError`**.