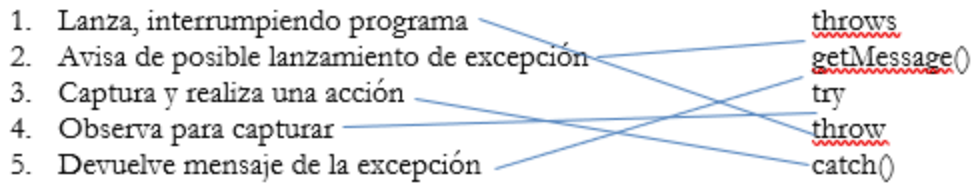
Excepciones

A. Una con flechas lo que corresponda



1) **Que es una excepción**
   Una excepción es un evento que ocurre durante la ejecución de un programa y que interrumpe el flujo normal de las instrucciones. Representa un error o situación inesperada, como dividir por cero o intentar acceder a un archivo que no existe. En Java, cuando ocurre una excepción, se crea un objeto de tipo Exception (o una subclase) que puede ser manejado por un bloque catch. Si no se maneja, la JVM termina el programa mostrando un mensaje de error.

2) **¿Cuál es la diferencia entre capturar y lanzar una excepción?**
   Lanzar (throw) una excepción significa crear y enviar un objeto de excepción cuando ocurre un error. Esto se hace con la palabra clave throw.
   Capturar (catch) una excepción significa detectar y manejar esa excepción en un bloque try-catch, evitando que el programa se detenga

3) **¿En qué orden se deben capturar varias excepciones con un solo try? Cite un ejemplo.**
   Las excepciones deben capturarse desde la más específica hasta la más general. Si se captura primero una clase padre (más general), las hijas (más específicas) se vuelven inalcanzables y causarán error de compilación.

5) **a. Leer un archivo que no se encuentra en el disco**
   Excepción: FileNotFoundException
Pertenece al paquete java.io
Es una **checked exception** (debe tratarse obligatoriamente con try-catch o throws)

b. Acceder a un elemento de un arreglo con un índice fuera de rango
Excepción: ArrayIndexOutOfBoundsException

Es una **unchecked exception**
Se lanza en tiempo de ejecución si el índice es inválido

c. Castear un objeto a un tipo incorrecto
Excepción: ClassCastException

Es una **unchecked exception**

Se lanza cuando el tipo real del objeto no coincide con el tipo al que se lo intenta convertir

d. Llamar a un método con un parámetro del tipo equivocado
Excepción: No aplica directamente como excepción en ejecución, porque:

Java es un lenguaje **fuertemente tipado**, así que esto genera un **error de compilación**, no una excepción en tiempo de ejecución.

6) **¿Qué excepción debería capturarse?**
ArithmeticException
Se lanza al hacer una división por cero, que ocurre cuando i == 0.

¿Dónde se lanza el objeto de la excepción?
f /= i;

¿En qué línea debería ubicarse `try`?

```
try {
 f /= i;
 System.out.println("f: " + f);
} catch (ArithmeticException e) {
 System.out.println("Error: división por cero");
}
```

¿Es posible que la ejecución continúe con las divisiones, habiendo capturado la excepción?

Sí, es posible, si colocamos el try-catch dentro del bucle for, de modo que sólo se salte la iteración con error y continúe con el resto.