Depuración (Debugging)

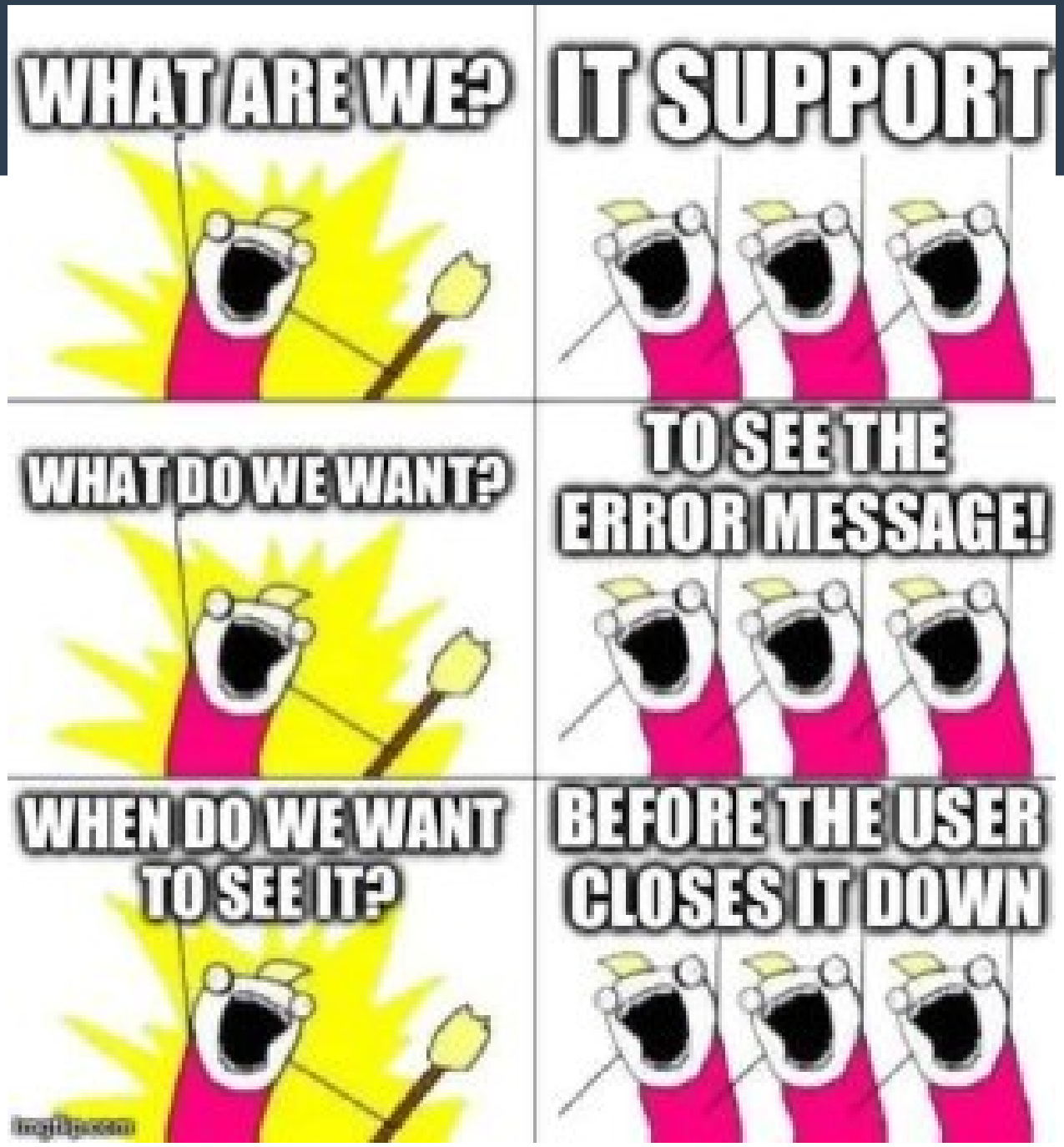
DEBUGGING

THE CLASSIC MYSTERY GAME WHERE YOU ARE THE DETECTIVE, THE VICTIM, AND THE MURDERER!

¿Como reproducir un error?

Reproducir el error

- Comience por encontrar un caso de prueba pequeño y repetible que produzca la falla. Si el error fue encontrado por la prueba de regresión, entonces ya tiene un caso de prueba que falla en su conjunto de pruebas. Si el error fue reportado por un usuario, puede tomar algún (bastante) esfuerzo reproducirlo.


WHAT ARE WE?
IT SUPPORT
WHAT DO WE WANT?
TO SEE THE ERROR MESSAGE!
WHEN DO WE WANT TO SEE IT?
BEFORE THE USER CLOSES IT DOWN

Reproducir el error

- Sin embargo, cualquier esfuerzo que haga para que el caso de prueba sea pequeño y repetible valdrá la pena, ya que tendrá que ejecutarlo una y otra vez mientras busca el error y desarrolla una solución para él. Además, una vez que hayas solucionado el error, querrás agregar el caso de prueba a tu conjunto de pruebas de regresión, para que el error nunca vuelva a aparecer. Una vez que tenga un caso de prueba para el error, **hacer que esta prueba funcione se convierte en su objetivo.**

Ejemplo

```
/**
 * Find the most common word in a string.
 * @param text string containing zero or more words, where a word
 *     is a string of alphanumeric characters bounded by nonalphanumerics.
 * @return a word that occurs maximally often in text, ignoring alphabetic case.
 */
public static String mostCommonWord(String text) {
    ...
}
```

Un usuario pasa todo el texto de las obras de Shakespeare a su método, algo así como mostCommonWord(allShakespearesPlaysConcatenated) , y descubre que, en lugar de devolver una palabra inglesa predeciblemente común como "**the**" o "**a**" , el método devuelve algo inesperado, tal vez "**e**" .

¿Como encuentro el error?

- Las obras de Shakespeare tienen **100,000 líneas** que contienen más de **800,000 palabras**, por lo que esta entrada sería muy dolorosa de depurar mediante métodos normales. La depuración será más fácil si primero trabajas en reducir el tamaño de la entrada con errores a algo manejable que aún muestre el mismo error (o muy similar):

¿Como encuentro el error?

- ¿La primera mitad de Shakespeare muestra el mismo error? (¡Búsqueda binaria! Siempre una buena técnica.)
- ¿Una sola obra tiene el mismo error?
- ¿Un solo discurso tiene el mismo error?

- Una vez que se encontró un caso de prueba pequeño, busca y corrige el error utilizando ese caso de prueba más pequeño, y luego vuelve a la entrada original y confirma que se corrigió el error inicial.

Entender la ubicación y la causa del error

Para localizar el error y su causa, puede utilizar el método científico:

Entender la ubicación y la causa del error

Para localizar el error y su causa, puede utilizar el método científico:

- **Estudia los datos**. Observe la entrada de prueba que causa el error y los resultados incorrectos, las aserciones fallidas y los seguimientos de la pila que resultan de ella.
- **Hipotetizar**. Proponer una hipótesis, consistente con todos los datos, sobre dónde puede estar el error o dónde no puede estar. Es bueno hacer esta hipótesis general al principio.

Entender la ubicación y la causa del error

- **Experimentar**. Crea un experimento que pruebe tu hipótesis. Es bueno hacer que el experimento sea una observación al principio, una sonda que recopila información pero altera el sistema lo menos posible.
- **Repetir**. Agregue los datos que recopiló de su experimento a lo que sabía antes y formule una nueva hipótesis. Esperemos que haya descartado algunas posibilidades y reducido el conjunto de posibles ubicaciones y las razones del error.

Apliquemos el MC al problema

```
public static String mostCommonWord(String text) {
    ... words = splitIntoWords(text); ...
    ... frequencies = countOccurrences(words); ...
    ... winner = findMostCommon(frequencies); ...
    ... return winner;
}

/** Split a string into words ... */
private static List<String> splitIntoWords(String text) {
    ...
}

/** Count how many times each word appears ... */
private static Map<String,Integer> countOccurrences(List<String> words) {
    ...
}

/** Find the word with the highest frequency count ... */
private static String findMostCommon(Map<String,Integer> frequencies) {
    ...
}
```

1. Estudia los datos

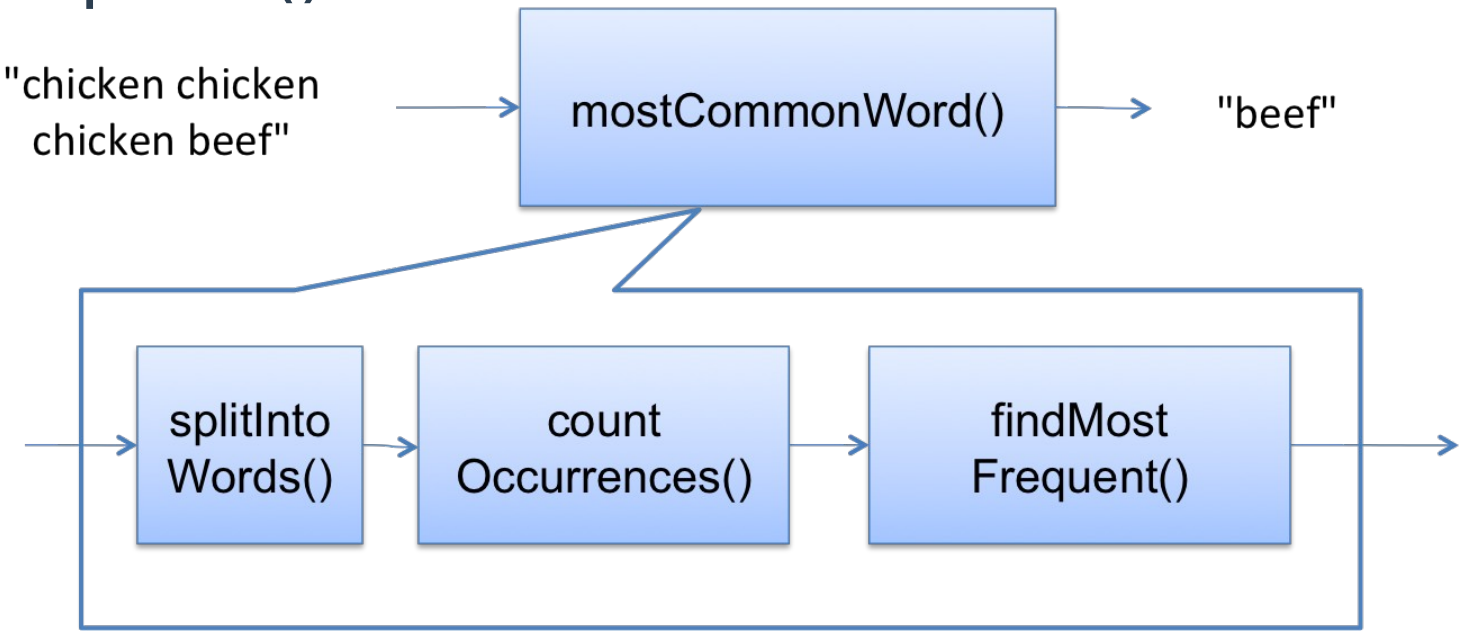
- Una forma importante de datos es el seguimiento de la pila de una excepción. Practica leyendo los rastros de pila que obtienes, porque te darán enormes cantidades de información sobre dónde y cuál podría ser el error.

2. Hacer hipótesis

- Es útil pensar en su programa como módulos o pasos en un algoritmo, y tratar de descartar secciones enteras del programa a la vez.

2. Hacer hipótesis

- Si el síntoma del error es una excepción en countOccurrences(), entonces puede descartar todo lo que está después, específicamente findMostFrequent().



2. Hacer hipótesis

- Luego, elegiría una hipótesis que intente localizar el error aún más. Podría suponer que el error está dentro splitIntoWords() , corrompiendo sus resultados, lo que provocará la excepción en countOccurrences() . Luego usarías un experimento para probar esa hipótesis. Si la hipótesis es cierta, entonces se habría descartado countOccurrences() como la fuente del problema. Si es falso, entonces descartarías splitIntoWords() .

3. Experimento

Un buen experimento es una observación suave del sistema sin molestarlo mucho. Puede ser:

- Ejecutar un caso de prueba diferente.
- Inserte una declaración o afirmación de impresión en el programa en ejecución, para verificar algo sobre su estado interno.

¿Y si no hago más pruebas?

- Es muy engorro, pierdo mucho tiempo, ¿que pasa si no lo hago?¿y si mejor introduzco todo el código en un try catch?

When tu programa no marca errores... pero no hace ni madres

¿Y si no hago más pruebas?

- Muy seguramente tus correcciones pueden enmascarar el verdadero error sin eliminarlo realmente.
- Por ejemplo, si obtienes una ArrayOutOfBoundsException, trata de entender qué está pasando primero. No solo agregue código que evite o atrape la excepción, sin solucionar el problema real.

Otros Consejos

- **Localización de errores por búsqueda binaria.** La depuración es un proceso de búsqueda y, a veces, puede utilizar la búsqueda binaria para acelerar el proceso.

Otros Consejos

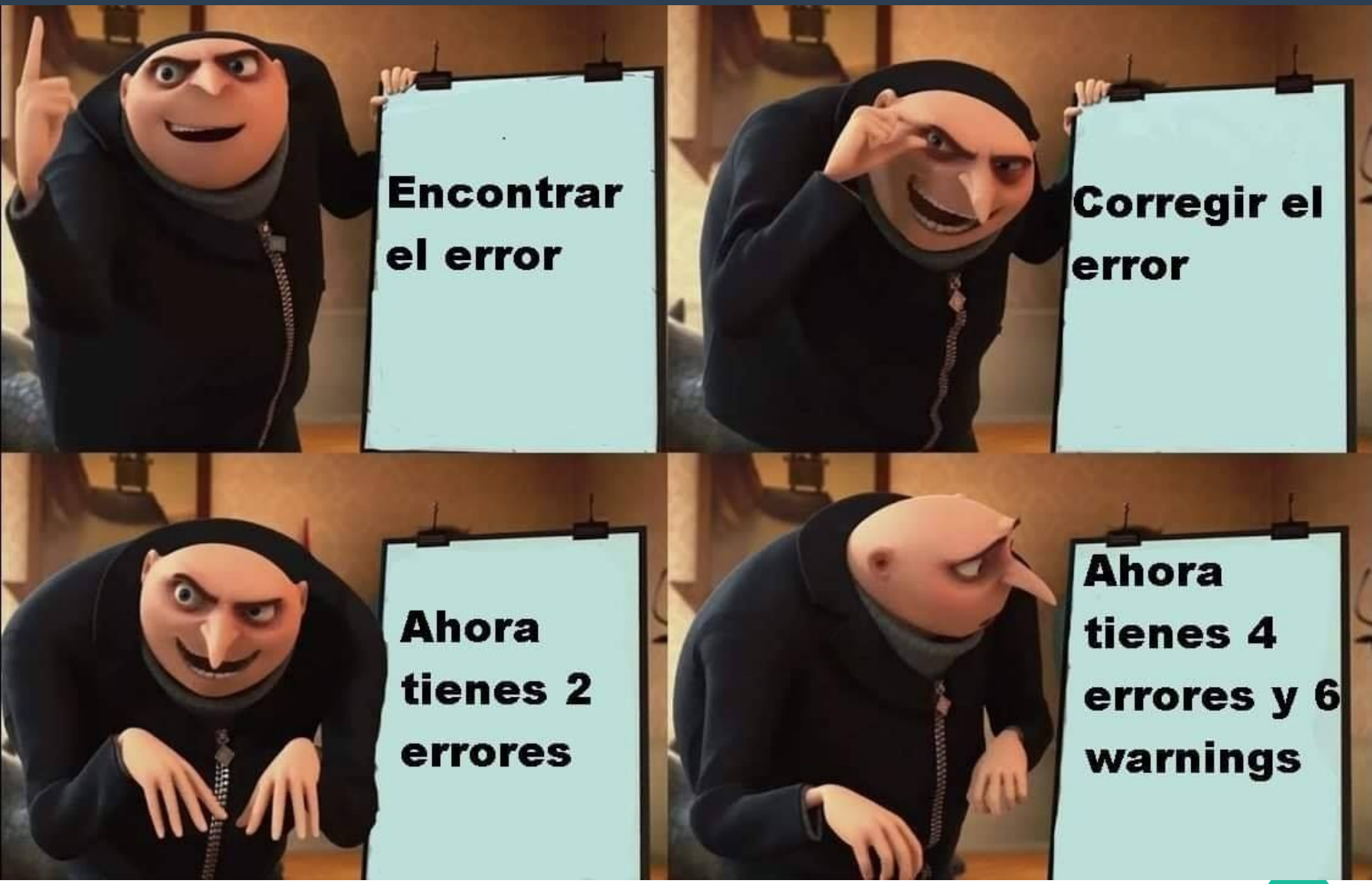
- **Prioriza tus hipótesis.** Al formular su hipótesis, es posible que desee tener en cuenta que las diferentes partes del sistema tienen diferentes probabilidades de fallar. Por ejemplo, el **código antiguo y bien probado es probablemente más confiable** que el código recientemente agregado. El código de la **biblioteca de Java es** probablemente **más confiable que el tuyo**. Debes confiar en estos niveles inferiores hasta que encuentres una buena razón para no hacerlo.

Otros Consejos

- **Intercambiar componentes.** Si tiene otra implementación de un módulo que satisface la misma interfaz y sospecha que el módulo, entonces un experimento que puede hacer es intentar intercambiar la alternativa. Por ejemplo, si sospecha que su implementación **binarySearch()**, entonces sustituya un simple **linearSearch()** en su lugar. Si sospechas de **java.util.ArrayList**, podrías intercambiar j**ava.util.LinkedList** en su lugar. Si sospecha del **tiempo de ejecución de Java**, ejecute con una **versión diferente de Java**. Si sospecha del **sistema operativo**, ejecute su programa en un sistema operativo diferente. Sin embargo, puede perder mucho tiempo intercambiando componentes que no fallan, así que no haga esto a menos que tenga una buena razón para sospechar un componente.

Otros Consejos

- **Asegúrese de que su código fuente y su código objeto estén actualizados.** Extraiga la última versión del repositorio, elimine todos sus archivos binarios y vuelva a compilar todo


Encontrar el error
Corregir el error
Ahora tienes 2 errores
Ahora tienes 4 errores y 6 warnings

Otros Consejos

- **Consigue ayuda.** A menudo es útil explicar su problema a otra persona, incluso si la persona con la que está hablando no tiene idea de lo que está hablando.
- Los ayudantes de laboratorio y otros estudiantes por lo general saben de qué están hablando.

Otros Consejos

- **Haz con tiempo tu proyecto y duerme bien.** Si estás demasiado cansado, no serás un depurador efectivo.

Arreglar el error

- **Una vez que sea encontrado el error y que entendidas su causa, el tercer paso es idear una solución para él.** Pregúntate si el error fue un error de codificación, como una variable mal escrita o parámetros de métodos intercambiados, o un error de diseño, como una interfaz insuficiente mente especificada o insuficiente para lo que se esta usando.

Arreglar el error

- Piensa también **si el error tiene algún pariente**. Si acabo de encontrar un error de división por cero aquí, ¿lo hice en otro lugar en el código? Trate de hacer el código a salvo de futuros errores como este. También considera qué efectos tendrá tu corrección. ¿Romperá alguna otra parte del código?

Arreglar el error

- Finalmente, una vez que aplicado su corrección, agregue el caso de prueba del error a su conjunto de pruebas de regresión y ejecute todas las pruebas para asegurarse de que (a) el error está solucionado y (b) no se han introducido nuevos errores.


DEBUGGING
I DON'T KNOW WHERE YOU ARE, I DON'T KNOW HOW YOU WORK, BUT I WILL FIND YOU, AND
I WILL FIX YOU