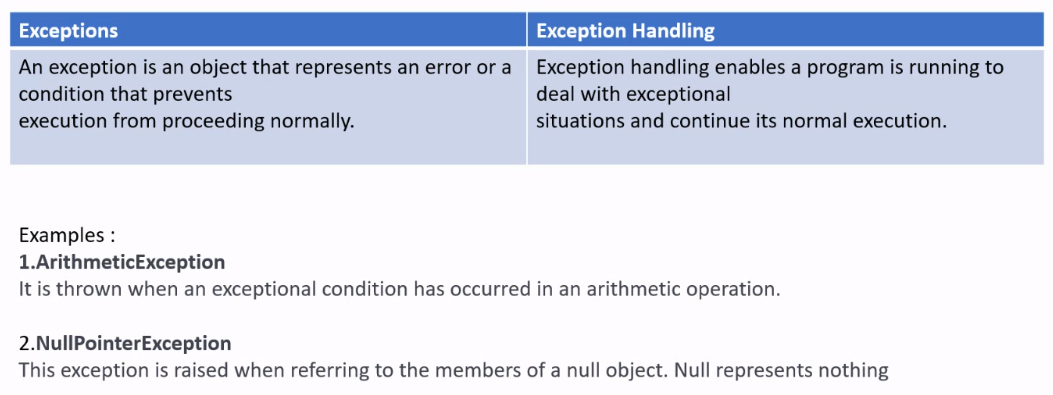
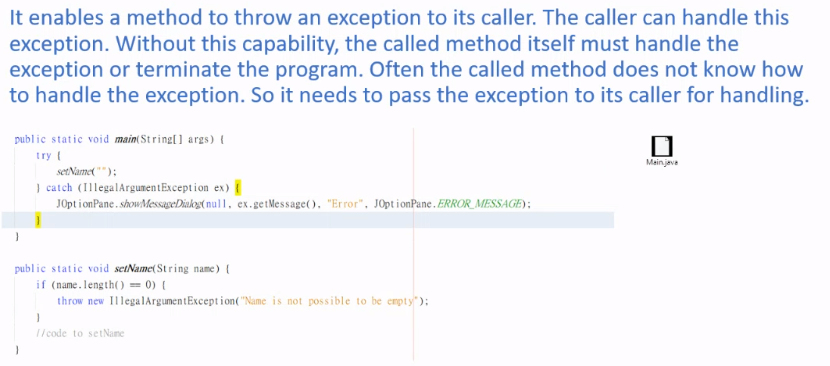
**Tutorial 3**

1. Explain **exceptions** and **exception handling**.

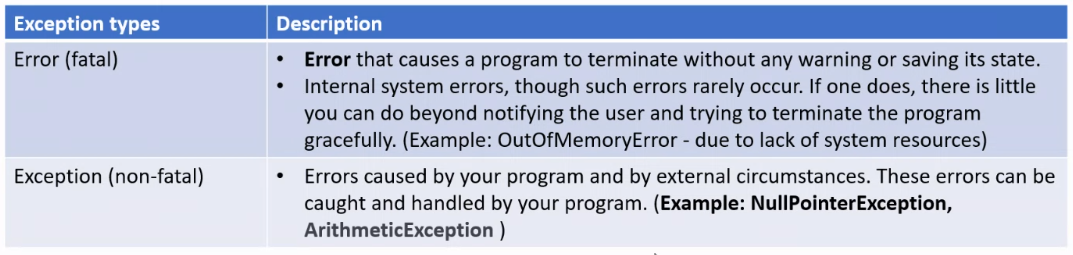


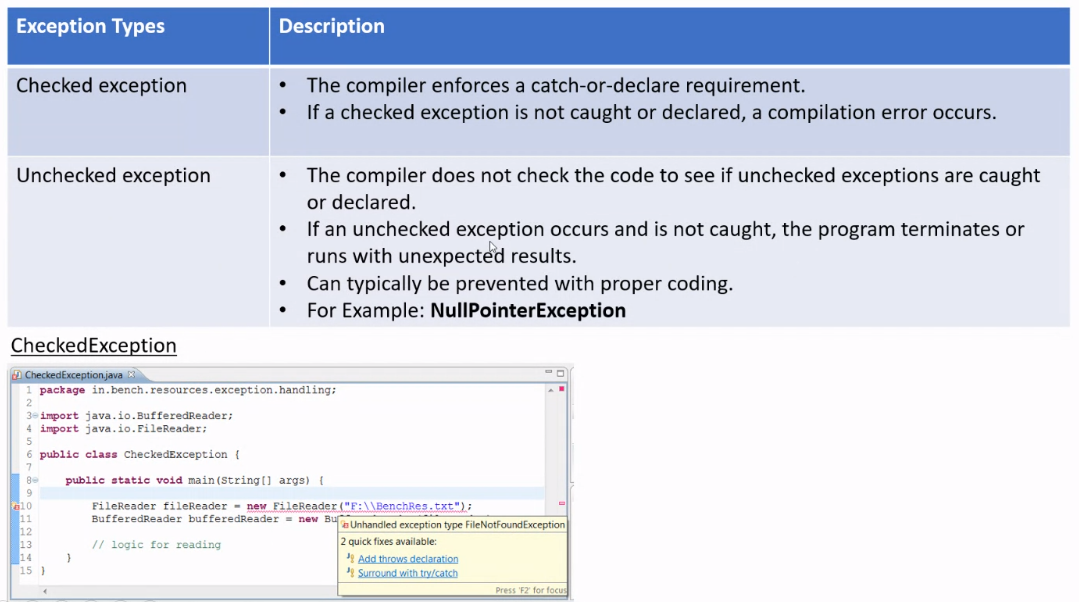
1. Describe the **advantage** of using **exception handling**.



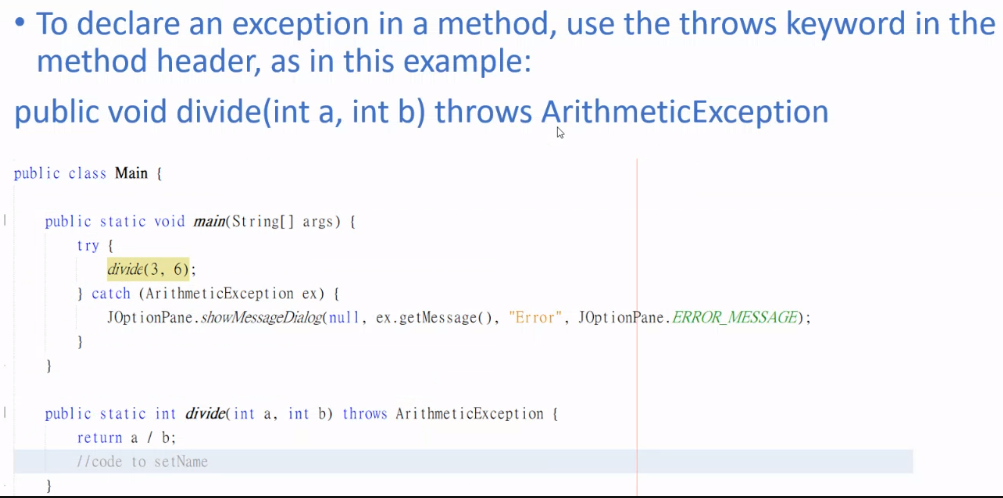
* Java separates exception handling from normal processing tasks.
* Exception handling makes it possible for the caller's caller to handle the exception.
* Exception handling simplifies programming because the error-reporting and error-handling code can be placed at the catch block.

1. Distinguish exception types: **Error (fatal) vs. Exception (non-fatal)**, and **checked vs. unchecked**.

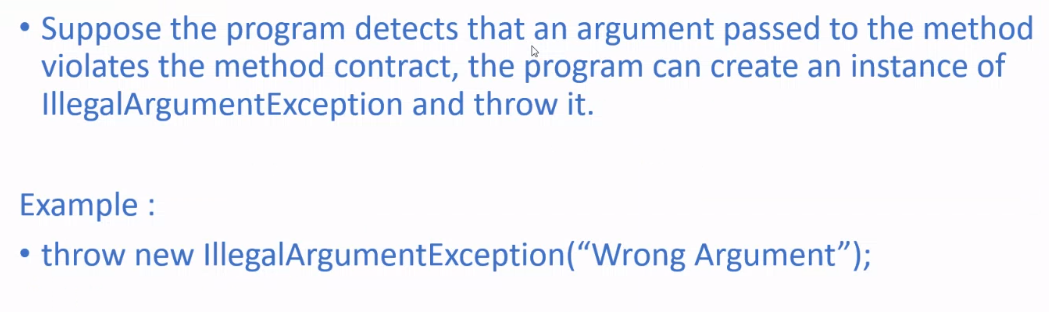




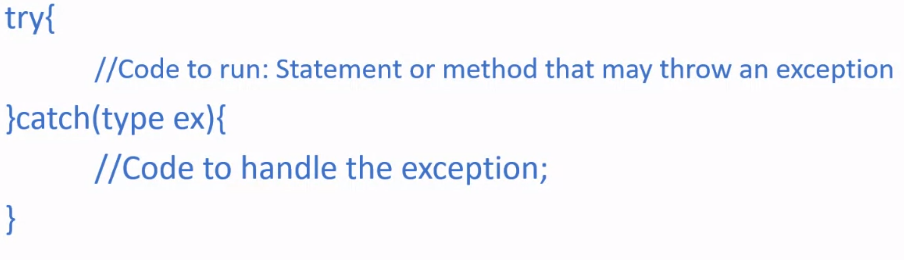
1. With example, **declare exceptions** in a method header



1. With an example, how do you **throw exceptions** in a method?



1. Write a **try-catch** block to handle exceptions.

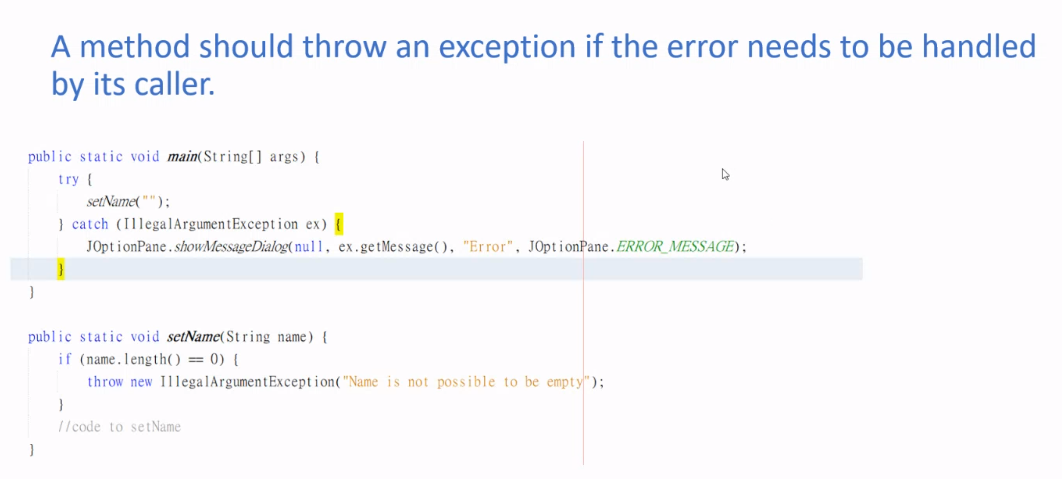


1. Write a **finally** clause in a try-catch block

| try {  // statements;  } catch(TheException ex) {  // handling ex;  } finally {  // finalStatements;  } |
| --- |

1. Determine when to use exceptions

* Exceptions are used when we are dealing with unexpected error conditions in our program code. For example, the user is trying to access out of range index in an array. For example, when it encounters broken or invalid input or when a resource (e.g. a file) is unavailable



1. Write syntax for **rethrowing** an **exception** in a catch block

| try {  //statements;  } catch(TypeOfException ex) {  //perform operations before exits;  throw ex;  } |
| --- |