<http://www.java2novice.com/java_interview_questions/thread-race-condition/>

MultiThreading

Race Condition

A race condition is a situation in which two or more threads orprocesses are reading or writing some shared data, and the finalresult depends on the timing of how the threads are scheduled.Race onditions can lead to unpredictable results and subtleprogram bugs. A thread can prevent this from happening by lockingan object. When an object is locked by one thread and anotherthread tries to all a synchronized method on the same object,the second thread will be block until the object is unlocked.

The situation where multiple threads try to operate on a shared resource without proper synchronization and sequence of their operations interleave is **Race Condition**( Also see [Thread interference and Memory consistency errors](http://javasolutionsonline.blogspot.in/2013/09/thread-interference-and-memory.html)) .  
  
**Example 1 .**  
The most basic example to describe a Race condition is a Counter class that increments value of an instance variable .

[viewplaincopy to clipboardprint?](http://javasolutionsonline.blogspot.in/2013/09/race-condition-in-concurrency.html)

1. **class** Counter {
2. **privateint** c = 0;
3. **publicvoid** increment() {
4. c++;
5. }
6. }

class Counter {

private int c = 0;

public void increment() {

c++;

}

}

**Note :**  
It seems that the increment operation is atomic ( Atomic operations can't be interleaved by threads ) but it is not . It can further be divided into three steps :  
**1.** Read the value c.   
**2.** Increment the value .  
**3.** Write the incremented value.  
  
  
Lets imagine that two threads A and B increments the value of instance variable c . Ideally if there is no interleaving of thread's operations , the value of the variable should be 2 as both the threads increment it by 1 . Lets see one of the possible scenario where thread's operations interleave   
  
Thread A : Read c ( which is 0 ).  
Thread B : Read c ( Still 0 ).  
Thread A : Increments the value by 1.  
Thread A : Write the value ( Now c is 1 )  
Threab B : Increments the value   
Thread B : Write the value ( Still c is 1 )   
  
Here we can see that due to interleaving of Read operations of two threads , the final value of the variable c is 1 and not 2.   
  
**Example 2 .**  
Below is a short snippet from a singleton class . Here first the code checks whether the instance is **null** and then creates a new object . The purpose of this code is to ensure that there is only one instance of **MySingleton**class.

[viewplaincopy to clipboardprint?](http://javasolutionsonline.blogspot.in/2013/09/race-condition-in-concurrency.html)

1. **public**MySingletongetInstance(){
2. **if** (instance == **null**){
3. instance = **new**MySingleton();
4. }
5. }

public MySingleton getInstance(){

if (instance == null){

instance = new MySingleton();

}

}

Now lets consider that there are two threads trying to get an instance of **MySingleton**class by calling the method getInstance() . It might be possible , due to interleaving of thread's operations , that both the thread sees the value of instance as **null** and in that case , two objects will get created by the threads .   
  
**How to fix the problem ?**  
Race conditions can be avoided by proper synchronization of the code . As an application developer , you have to enforce locking to make sure that only one thread enters a critical section at a time and the result of the thread is visible to others once it comes out of the synchronized block .

**Lazy Initialization in Spring**

The default behavior for ApplicationContext implementations is to eagerly pre-instantiate all singleton beans at startup. Pre-instantiation means that an ApplicationContext will eagerly create and configure all of its singleton beans as part of its initialization process. Generally this is a good thing, because it means that any errors in the configuration or in the surrounding environment will be discovered immediately (as opposed to possibly hours or even days down the line).

**Spring Scoped Proxy bean:**

<http://stackoverflow.com/questions/14371335/spring-scoped-proxy-bean>

**Why is java a strongly typed language?**

This is because all the errors/warnings are shown to programmer while typing and compilation stages itself.

**Exception handling: http://www.javatpoint.com/exception-handling-in-java**

<http://beginnersbook.com/2013/04/java-exception-handling/>

**How to show a meaningful error message to user in UI when exception occurs in DAO layer? / how do you pass an exception from DAO to Service to Controller to UI?**

When an exception say EmptyDataException gets thrown in DAO layer, you must catch it in service and then throw it in service and catch it again in Controller and in the catch block in controller, you must handle it. For instance, you will add an errorMsg attribute (this will contain the meaningful error message) to model in the catch block. So now, this error message object will be available in the jsp and you can display the error message to user .

* If you are not handling an exception by using a catch block then you must declare the exception using the throws keyword.
* If your block of code inside try throws checked exception, it must be handled else you will get compile time exception.
* It isn't obligatory to include runtime exceptions or errors in throws statement.

So you either

1. catch the exception (OR)
2. throw it using the throw keyword so that the previous block of code (say method A) from where the call came to this exception throwing method (method B) will handle it i.e method A must handle it. (OR)
3. Declare the exceptions using throws keyword in the method where exception could occur (method B)

Important link for throw and throws:

<https://weblogs.java.net/blog/manningpubs/archive/2013/06/13/using-throws-and-throw-statements-java>

**Ways to handle exception and backtrack- different combinations:**

**Example1 – Type1**

Controller method calls service method which in turn calls dao method. DAO method throws FileNotFoundException andJAXBException. This can either be handled in DAO or this exception can be thrown to the calling method i.e the service method. Here these exceptions are thrown to the service method. The service method does not handle these exceptions and throws it back to the calling method i.e the controller method. In the controller method, we have catch blocks to handle these exceptions. In these blocks we can also set error messages to be used in the jsp.

**Controller:**

@RequestMapping (value=”/hello1”)

ControllerMethod() {

publicModelAndViewControllerMethod(

@ModelAttribute("model") ModelMap model) {

try{

//calling method in service

Something something =serviceMethod();

//based on the value of something, do something and add the results to model and //return the //model

}

catch(FileNotFoundException f){

//Lines of code handleFileNotFoundException

}

catch(JAXBException j){

//Lines of code handleJAXBException

}

return new ModelAndView("hello1", model);

}

**Service:**

serviceMethod()throws FileNotFoundException, JAXBException{

//calling method in DAO

DAOMethod();

//write lines of code to check conditions based on which FileNotFoundException, JAXBException

//can get thrown and throw them.

if(some condition){

throw new FileNotFoundException();

}

if(some condition2){

throw new JAXBException ();

}

return something;

}

**DAO:**

DAOMethod() throws FileNotFoundException, JAXBException{

//Lines of code that can throw any of the above exception

//-----

try{

//Lines of code that can IOexception

}

catch(IOException e){

//Lines of code handleIOException

}

}

**Example1 – Type2**

Here the service method has catch blocks but throws the exception again in catch block and these will get thrown to the calling method i.e the controller method. **A method can handle the exception and still declare it to be thrown**. When you are handling exception using catch block, u can also declare what exception is thrown using throws keyword in method signature. So what will happen when FileNotFoundException or JAXBException is thrown. Will the respective catch blocks will handle the exception or will this method throw these exceptions to the calling method? 🡪 This is usually done by methods whose exception handlers might throw the same exception.When exception is thrown in the try block, it will be caught in the catch block of course, but if catch block throws the same exception, the exception will be handed over to the calling method.

**Controller:**

@RequestMapping (value=”/hello1”)

ControllerMethod() {

publicModelAndViewControllerMethod(

@ModelAttribute("model") ModelMap model) {

try{

//calling method in service

Something something =serviceMethod();

//based on the value of something, do something and add the results to model and //return the //model

}

catch(FileNotFoundException f){

//Lines of code handleFileNotFoundException

}

catch(JAXBException j){

//Lines of code handleJAXBException

}

return new ModelAndView("hello1", model);

}

**Service:**

serviceMethod()throws FileNotFoundException, JAXBException{

//calling method in DAO

try(){

DAOMethod();

}

catch(FileNotFoundException f){

throw new FileNotFoundException();

}

catch(JAXBException j){

throw new JAXBException ();

}

return something;

}

**DAO:**

DAOMethod() throws FileNotFoundException, JAXBException{

//Lines of code that can throw any of the above exception

//-----

try{

//Lines of code that can IOexception

}

catch(IOException e){

//Lines of code handleIOException

}

}

**Example2**

In this example, the exception is thrown from dao to service and handled in the service and not thrown to the controller layer.

**Controller:**

@RequestMapping (value=”/hello1”)

ControllerMethod() {

publicModelAndViewControllerMethod(

@ModelAttribute("model") ModelMap model) {

//calling method in service

Something something =serviceMethod();

//based on the value of something, do something and add the results to model and //return the //model

return new ModelAndView("hello1", model);

}

**Service:**

serviceMethod() {

//calling method in DAO

try(){

DAOMethod();

}

catch(FileNotFoundException f){

//Lines of code to handle FileNotFoundException

}

catch(JAXBException j){

//Lines of code to handle JAXBException

}

return something;

}

**DAO:**

DAOMethod() throws FileNotFoundException, JAXBException{

//Lines of code that can throw any of the above exception

//-----

try{

//Lines of code that can IOexception

}

catch(IOException e){

//Lines of code handleIOException

}

}