**Bradley Justice**

**Question 1**

The differences between Error and Exception

Errors are caused by environmental factors whereas issues with the code/application itself cause Exceptions. Errors are, by definition, also exceptions, but only of the unchecked type as they occur only at run time and are not known to the compiler. Exceptions can be checked or unchecked, sometimes known by compiler, sometimes not. Error’s are impossible to recover from, whereas Exceptions are possible to recover from.

Examples:

A device runs out of memory, and so there is a memory error, or a stack overflow, both of which are caused by environmental factors, versus Exceptions, which are caused by the code itself, like trying to call or access a null object and getting back the NullPointerException.

The difference between Exception classes that are sub-classes of RuntimeException and those that are not lies in the fact that those of RuntimeException only occur at run time, and as such, are not identified by the compiler ahead of time, whereas checked exceptions must be handled with a catch, or passed on in a throw.

**Question 2**

Exception catching works by catching the exception thrown in specific handlers. When an exception occurs that is not handled, the method where the error occurs throws the exception to its parent class, and checks for a handler for this instead. This propagates upward throughout the exception classes and sub-classes. If the runtime system cannot find a method on the call stack with an appropriate exception handler, it terminates.

If doSomething() throws a FileNotFoundException, the compiler will look for and find the catch (FileNotFoundException ex) method and execute whatever code is within it. If doSomething() throws an IOException, it will look for handlers for IOExceptions, and will find its handler and execute that code. If doSomething() throws some other type of exception, the compiler will look for and not find a handler for it, and the exception will be thrown to its parent class, eventually leading to the catch (exception ex) method being called, as the exception is thrown upwards at each stage. Following whatever catch method is called, the finally method is always ensured to be executed before the try block exits.

**Question 3**

In the previous example, if one of the catch blocks throws an exception, it is once again, immediately dealt with, as the compiler will look for the exception handler for the second exception. In these circumstances, the try block for the second exception will be fully executed, and its exception handled before the first exceptions try block is fully run through.

**Question 4**

In this same example, if the finally block in the first exception handler throws a second exception, then the compiler will look to address the second exception first, looking for a handler of whatever type it is and executing that and its finally block, all before the finally block of the first exception is executed.