JAVA PRACTICAL_8

1. (Catching Exceptions with Superclasses)Use inheritance to create an exception superclass (called ExceptionA) and exception subclasses ExceptionB and ExceptionC, where ExceptionB inherits from ExceptionA and ExceptionC inherits from ExceptionB. Write a program to demonstrate that the catch block for type ExceptionA catches exceptions of types ExceptionB and ExceptionC.

2. ***(Catching Exceptions Using Class Exception)*** Write a program that demonstrates how various exceptions are caught with catch (Exception exception). Define classes ExceptionA (which inherits from class Exception) and ExceptionB (which inherits from class ExceptionA). In your program, create try blocks that throw exceptions of types ExceptionA, ExceptionB, NullPointerException and IOException. All exceptions should be caught with catch blocks specifying type Exception.

3. (Order of catch Blocks) Write a program demonstrating that the order of catch blocks is important. If you try to catch a superclass exception type before a subclass type, the compiler should generate errors.

4. (Constructor Failure) Write a program that shows a constructor passing information about constructor failure to an exception handler. Define class SomeClass, which throws an Exception in the constructor. Your program should try to create an object of type SomeClass and catch the exception that's thrown from the constructor.

5. (Rethrowing Exceptions) Write a program that illustrates rethrowing an exception. Define methods someMethod and someMethod2. Method someMethod2 should initially throw an exception. Method someMethod should call someMethod2, catch the exception and rethrow it. Call someMethod from method main, and catch the rethrown exception. Print the stack trace of this exception.

6. (Catching Exceptions Using Outer Scopes) Write a program showing that a method with its own try block does not have to catch every possible error generated within the try. Some exceptions can slip through to, and be handled in, other scopes.

7. Write a program that illustrates exception propagation. Define methods propagator1 and propagator2. Method propagator1 should initially throw an ArithmeticException exception. Method propagator2 should call propagator1, re-throw the exception. Call propagator2 from method main, and catch and handle the re-thrown exception by printing the stack trace of the exception.