Session: Exceptions in Java

1. What is wrong with this code?

```
public void Foo ()  }
     try {
       lock() // lock some resource
       // open some resource
       // try to change some things
       // fool around a bit
     }
     catch (Exception e)   {
     }
     catch (ConcurrentModificationException e) {
          System.out.println(„ bad stuff going on today!“)
     }
     finally {
          return;
     }
}
```

- Nothing happens upon an Exception
- ConcurrentModificationException won’t be catched separately as Exception is catched before
- Bad logging („bad stuff going on today!“)
- return in finally is bad practice

2. What will be the output of the program?

```
public class TestException
{
    public static void badCall()
    {
        System.out.print("throwing it ");
        throw new RuntimeException();
    }
    public static void main(String [] args)
    {
        try
        {
            System.out.print("hello ");
            badCall();
        }
        catch (Exception re )
        {
            System.out.print("caught ");
        }
        finally
        {
            System.out.print("finally ");
        }
        System.out.println("after ");
    }
```

As it is:

„java: reached end of file while parsing“

How it should be (with enough brackets):

„hello throwing it caught finally after “

3. Output?

```
public class TestException1
{
          public static void main(String [] args)
          {
              try
              {
                  badMethod();
                  System.out.print("A");
              }
              catch (Exception ex)
              {
                  System.out.print("B");
              }
              finally
              {
                  System.out.print("C");
              }
              System.out.print("D");
          }
          public static void badMethod()
          {
              throw new Error();
          }
}
```

```
Exception in thread "main" java.lang.Error
        at org.example.TestException1.badMethod(TestException1.java:24)
        at org.example.TestException1.main(TestException1.java:9)
```

4. Make it compile!

```
public class TestException2
     {
         class TestException extends Exception {}  // inner class
          public void runTest() throws TestException {}

          public void test()
              {
                  runTest();
              }
     }
```

Our solution:

```
public class TestException2
{
    class TestException extends Exception {}  // inner class
    public void runTest() throws TestException {}

    public void test() throws TestException {
        runTest();
    }
}
```

5. When should you re-throw a caught exception?

If you want to have the possibility to do exception handling in the method calling the exception-throwing method as well as in the catch-clause.

6. A banking software detects, that a certain customer ID is not in the database. Is this a) a system exception, b) a custom exception, c) no exception.

It depends on the implementation. It could be a) if e.g. IllegalArgumentException is used (we would not suggest to use that here...), but could be a custom exception („IllegalCustomerIDException“). If it is coded that way it could also throw no exception but rather print something to the screen without any exception handling.

7. Was ist der Vorteil von Exceptions gegenüber dem Auswerten von Fehlerwerten im Return?

It can be dealt with immediately at the point it gets thrown via try/catch. Besides that, it makes code way more readable and prevents a mess.