Question and Answers → Week 10

Ques 1: What is chained exception? Explain with example.

Ans 1: Chained Exception was added to Java in JDK 1.4. This feature allows you to relate one exception with another exception, i.e one exception describes cause of another exception. For example, consider a situation in which a method throws an ArithmeticException because of an attempt to divide by zero but the actual cause of exception was an I/O error which caused the divisor to be zero. The method will throw only ArithmeticException to the caller. So, the caller would not come to know about the actual cause of exception. Chained Exception is used in such type of situations.

- Two new constructors and two new methods were added to Throwable class to support chained exception.
- Throwable (Throwable cause)
- Throwable (String str, Throwable cause)

```
import java.io.IOException;

public class practice
{
    public static void divide(int a, int b)
    {
        if(b == 0)
        {
            ArithmeticException ae = new ArithmeticException("top layer");
            ae.initCause(new IOException("cause"));
            throw ae;
        }
        else
        {
            System.out.println(a/b);
        }
    }

    public static void main(String[] args)
    {
        try
        {
            divide(5, 0);
        }
        catch(ArithmeticException ae) {
            System.out.println( "caught : " +ae);
            System.out.println("actual cause: "+ae.getCause());
        }
    }
}
```

Ques 2: Explain the difference between compile time and ruin time exception.

Ans 2: An exception that occurs at compile time is called a checked exception. This exception cannot be ignored and must be handled carefully. For example, in Java if you use FileReader class to read data from the file and the file specified in class constructor does not exist, then a FileNotFoundException occurs and you will have to manage that exception. For the purpose, you will have to write the code in a try-catch block and handle the exception. On the other hand, an exception that occurs at runtime is called unchecked-exception Note: Checked exception is not handled so it becomes an unchecked exception. This exception occurs at the time of execution.

Ques 3: Is it necessary that each try block must be followed by the catch block? Explain with example.

Ans 3: It is **not necessary** that each try block must be followed by a catch block. It should be followed by either a catch block or a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

```
import java.io.IOException;

public class practice
{
        void me() throws ArithmeticException
        {
            try
            {
                System.out.println(5/0);
            }
            finally {
                System.out.println( "Inside Finally");
            }
        }

    public static void main(String[] args) throws ArithmeticException
    {
        practice p =new practice();

        try
        {
            p.me();
        }
        catch(Exception e)
        {
            System.out.println("Caught an exception --> "+e);
        }
    }
}
```

Ques 4: Is there any cases where finally block will not be executed?

Ans 4: Yes, the finally block is always get executed unless there is an abnormal program termination either resulting from a JVM crash or from a call to System. exit() or some similar

code is written into try block then program will automatically terminate and the finally block will not be executed in this case.

Ques 5: What is OutOfMemoryError in Java?

Ans 5: ***OutOfMemoryError*** exception. Usually, this error is thrown when there is insufficient space to allocate an object in the Java heap. In this case, the garbage collector cannot make space available to accommodate a new object, and the heap cannot be expanded further.

Ques 6: What are the best practices for exception handling?

Ans 6: This section describes best practices for handling and creating exceptions.

- Use try/catch/finally blocks to recover from errors or release resources.
- Handle common conditions without throwing exceptions.
- Design classes so that exceptions can be avoided.
- Throw exceptions instead of returning an error code.
- End exception class names with the word Exception.
- Place throw statements so that the stack trace will be helpful.