1) O que é uma exceção? O que são exceções verificadas (checked) e não verificadas (unchecked)? Dê exemplos de comandos em Java que podem lançar exceções destes dois tipos

R: Uma exceção é um mecanismo utilizado para o tratamento de erros em tempo de execução de um programa, tendo assim um meio para corrigir e evitar erros no mesmo. As exceções verificadas (checked), são as de tratamento obrigatório em seu código utilizando dos comandos de try-catch ou throws, não compilando caso esse não seja feito. Já as exceções não verificadas (unchecked), são as que não precisam ser tratadas obrigatoriamente, sendo divididas em dois subtipos, Runtime e Erro. O Runtime indica um erro interno no programa, mas que o compilador não consegue detectar e, o Erro, indica algo que não é previsto, como um erro externo de disco. Você pode lançar exceções explicitamente a partir do comando throw e, utilizando-o dentro do comando try-catch.

2) O que é o bloco finally do comando try catch? Dê um exemplo em que este bloco deve ser usado.

R: Um bloco de código do tipo *finally* sempre acompanha um comando *try* e seu diferencial é sempre ser executado, diferentemente do catch, por exemplo, que depende de uma exceção ser capturada e só pode existir um bloco desse tipo para cada bloco *try*. Um exemplo clássico de utilização dessa estrutura dentro de um programa é no contexto de leitura e escrita de arquivos, que podem gerar exceções, porém, independentemente da leitura ou escrita ter sido bem sucedida, é necessário fechar o arquivo, para esse fim é que o bloco *finally* pode ser usado, inserindo-se dentro dele o código responsável pelo seu fechamento. Para citar mais um exemplo, num contexto de comunicação com um banco de dados, a conexão deve ser sempre encerrada ao final do código, sendo aplicável e recomendável o uso dessa estrutura.

3) Um bloco try e catch pode conter vários “catches”. Explique como e por que isso as vezes acontece. As exceções nos blocos catches podem ocorrer em qualquer ordem ou devem obedecer uma hierarquia? Pesquise e explique.

R: Um mesmo método pode sofrer com diferentes tipos de erro e, por isso, às vezes, é necessário mais de um catch para tratá-los individualmente da forma mais adequada para cada tipo de erro. Existe sim uma hierarquia dos catches que deve ser seguida no sentido de que os catches de erros mais específicos devem vir antes dos mais gerais, pois caso contrário, os catches específicos nunca serão acionados. Por exemplo, um catch (Exception e) antes de um catch (IOException e) impedirá que o (IOException e) trate esse erro, pois toda exceção(erro) já será pego pelo primeiro catch que é geral.