Exercícios sobre exceções

1) Analise a classe *ContaCorrente* abaixo. Acrescente o tratamento de operações inválidas (saldo inicial, depósito ou saques nulos ou negativos e saque maior que o saldo) lançando exceções. Crie exceções verificadas para tratar problemas com o saldo inicial e depósito ou retiradas menores ou iguais a zero. Crie uma exceção não verificada para os problemas relativos a retiradas maiores que o saldo.

```
public class ContaCorrente {
    private double saldo;

    public ContaCorrente(double saldoInicial){
        saldo = saldoInicial;
    }

    public void deposito(double valor){
        saldo += valor;
    }

    public void retirada(double valor){
        saldo -= valor;
    }

    public double getSaldo(){
        return(saldo);
    }
}
```

2) Escreva um exemplo de uso que trate as exceções possíveis de serem lançadas pelo exemplo criado no exercício 1.

3) Crie um exemplo que explore exceções verificadas e não verificadas lançadas pelo próprio Java. Pesquise diferentes tipos de exceções (de memória, aritméticas, de índice de arranjo, de coerção de tipo, de entrada e saída etc).

4) Qual a diferença entre exceções verificadas e não verificadas? *NullPointerException* é verificada ou não verificada? Porque? Que tipos de exceções devo declarar com a palavra reservada *throws*?

5) Porque não é necessário declarar que um método pode lançar uma *NullPointerException*?

6) O que acontece quando uma exceção não encontra uma clausula capaz de captura-la?

7) O que um programa pode fazer com a exceção que recebe em uma clausula *catch*?

8) O tipo de um objeto de exceção é sempre o mesmo que o declarado na clausula *catch*? Invente uma maneira de testar.

9) Analise o código da classe *Aluno*, a seguir. O construtor dessa classe não prevê nenhum tipo de consistência. Trabalhando no contexto de programação defensiva, escreva código para fazer as consistências necessárias e lançar exceções prevendo cada tipo de erro. Responda: existe alguma vantagem no uso de exceções em métodos construtores para avisar sobre dados inconsistentes?

```
public class Aluno {
    private int matricula;
    private String nome;
    private int anoNascimento;

    public Aluno(int umaMtr,String umNom,int umANasc){
        nome = umNom;
        matricula = umaMatr;
        anoNascimento = umAnoNasc;
    }

    public int getMatricula() {  return matricula;   }

    public String getNome() { return nome;  }

    public int getAnoNascimento() {
        return anoNascimento;
    }

    @Override
    public String toString(){
       return(matricula+", "+nome+", "+anoNascimento);
    }
}
```

10) Pode-se ler o conteúdo de uma página Web com a seguinte sequência de comandos:

```
String endereco = "http://java.sun.com//index.html"
URL url = new URL(endereco);
BufferedReader reader = new BufferedReader(new
InputStreamReader(url.openStream()));
boolean done = false;
while(!done){
    String input = reader.readLine();
    if (input == null) done = true;
    else faça alguma coisa com a entrada ...
}
```

Projete uma classe chamada *WebPageReader* cujo construtor recebe uma string com o endereço da página a ser lida. Esta classe deve ter um método chamado *readLine* que retorna a próxima linha de entrada da página Web sendo lida ou null se foi atingido o final da página. Os métodos dessa classe não devem capturar exceções, mas sim usar especificadores *throws* para indicar os tipos de exceções que podem ocorrer. Consulte a documentação da API para saber que exceções podem ocorrer. Escreva um programa exemplo que usa esta classe (mais detalhes sobre a leitura de uma página web em Java em http://www.mballem.com/post/capturando-html-de-pagina-web-com-java).