Avaliação Técnica

*** Utilizar a linguagem java, nos casos que couber, utilizar recursos da API.**

1 - Acerca de sistemas de desenvolvimento web, assinale a opção correta.

A - Servlet é uma classe do Java que possibilita ampliar os recursos de servidores web, desenvolvida para permitir conteúdos dinâmicos orientados ao usuário.

B - Para utilizar bancos de dados relacionais em aplicações desenvolvidas em JSP, é obrigatória a utilização do Hibernate, que é um framework que realiza o mapeamento objeto/relacional.

C - O Ajax permite interagir com dados textuais nos formatos UTF-8 e XML, porém restringe o acesso a JSON (Java Script Object Notation) e a bancos de dados relacionais.

D - No XMLHttpRequest, utilizado para trocar dados com um servidor, com o intuito de melhorar sua usabilidade, o método open ( ) aceita somente requisições no modo asynchronous.

E - O JSF (Java Server Faces) permite usar tags customizadas limitadas a páginas JSP, com vistas a encapsular a segurança na forma nativa do acesso aos JavaBeans.

RESPOSTA CORRETA: LETRA A - Servlet é uma classe do Java que possibilita ampliar os recursos de servidores web, desenvolvida para permitir conteúdos dinâmicos orientados ao usuário.

2 - Um programador web foi contratado para desenvolver um site utilizando HTML, CSS, JSP e Servlets. Para tanto, deve usar um servidor escrito em Java, que não é contêiner EJB, mas é utilizado como servlet container, denominado

A - GlassFish.

B - JBoss.

C - WebLogic.

D - Jetty.

E - WebSphere.

RESPOSTA CORRETA: LETRA D – Jetty

3 - São apenas tipos de componentes executados em servidores Web:

A - Beans, Servlets e J2EE.

B - JVM, Servlets e JSP.

C - Beans, Servlets e JSP.

D - Beans, Swing e JSP.

E - Beans, Swing e JVM.

RESPOSTA CORRETA: LETRA C – Beans, Swing e JSP

4 - Analise os itens a seguir sobre JEE e EJB.

I. Um servidor J2EE fornece contêineres EJB e Web.

II. O contêiner EJB gerencia a execução de EJBs em aplicações J2EE.

III. O contêiner Web gerencia a execução de páginas JSP e componentes servlet em aplicações J2EE.

IV. Um session bean representa um único cliente dentro do servidor J2EE. Para acessar um aplicativo que é instalado no servidor, o cliente invoca os métodos do session bean.

Está correto o que se afirma em:

A - I, II, III e IV.

B - I e II, apenas.

C - I, III e IV, apenas.

D - I e IV, apenas.

E - III e IV, apenas.

RESPOSTA CORRETA: LETRA A – I, II, III e IV

5 - Spring Framework é uma plataforma Java completa que fornece suporte de infraestrutura para o desenvolvimento de aplicações Java. Acerca das características do framework Spring 3.0, assinale a opção correta.

A - Na arquitetura Spring MVC Web, o Validator é uma classe opcional que pode ser invocada para validar dados de formulários.

B - A injeção de dependência é feita após a criação do objeto, por meio dos métodos set de uma classe no estilo JavaBean, e não no momento da criação do objeto, tendo-se em vista que passar muitos argumentos no construtor pode tornar-se dispendioso.

C - A interface BeanFactory gerencia beans definidos em arquivos XML e trata recursos de mensagens.

D - O controlador AbstractWizardFormController, do módulo Spring MVC, permite suporte para o preenchimento de formulários a partir de determinada solicitação.

E - A porta de entrada do navegador web para a arquitetura Spring MVC Web é a componente Interface (JSP/HTML).

RESPOSTA CORRETA: LETRA A – Na arquitetura Spring MVC Web, o Validator é uma classe opcional que pode ser invocada para validar dados de formulários.

6 - No Spring, as configurações de segurança são realizadas no arquivo applicationContext-security.xml, e, para que qualquer página ou diretório seja seguro, é necessário adicionar a esse arquivo o elemento <intercept-url>.

C - Certo

E - Errado

RESPOSTA CORRETA: C - Certo

7 - Spring é um framework que suporta a publicação de mensagens para determinado tópico de mensagens para auxílio no desenvolvimento de sistemas complexos. Nesse modelo, o desenvolvedor master não sabe da existência do desenvolvedor associado e vice-versa.

C - Certo

E – Errado

RESPOSTA CORRETA: E – Errado

8 - Dados dois numeros inteiros A e B, crie um terceiro inteiro C seguindo as seguintes regras:

- O primeiro número de C é o primeiro número de A;

- O segundo número de C é o primeiro número de B;

- O terceiro número de C é o segundo número de A;

- O quarto número de C é o segundo número de B;

Assim sucessivamente…

- Caso os números de A ou B sejam de tamanhos diferentes, completar C com o restante dos números do inteiro maior. Ex: A = 10256, B = 512, C deve ser 15012256.

- Caso C seja maior que 1.000.000, retornar -1

Desenvolva um algoritmo que atenda a todos os requisitos acima.

RESPOSTA EXERCÍCIO 8

```
package br.com.desafios2it.domain;

/**
 *
 * @author Emello
 */
public class elementos {

    private Integer a;
    private Integer b;
    private Integer c;


    public Integer getA() {
        return a;
    }


    public void setA(Integer a) {
        this.a = a;
    }


    public Integer getB() {
        return b;
    }

    public void setB(Integer b) {
        this.b = b;
    }


    public Integer getC() {
        return c;
    }


    public void setC(Integer c) {
        this.c = c;
    }



}
```

```
package br.com.desafios2it.bean;

import br.com.desafios2it.domain.elementos;

/**
 *
 * @author Emello
 */
public class ElementosBean {

   public String gerarC(elementos elementoAeB) {

        String elementoC = "";
       try {

          //Compara os 2 elementos e popula a variável com a quantidade do elemento que tem a maior
quantidade de caracteres
          //Caso a quantidade de caracteres de ambos forem iguais, retornará a quantidade do
elementoAeB.getA().toString()
          int elementoMaiorCaracteres = elementoAeB.getA().toString().length() >=
elementoAeB.getB().toString().length() ?  elementoAeB.getA().toString().length() :
elementoAeB.getB().toString().length() ;

          //Percorre de zero até o valor da variável elementoMaior
          for (int i = 0; i <= elementoMaiorCaracteres; i++) {

            //Caso i for menor que a quantidade de caracteres do objeto elementoAeB.getA().toString(),
concatena
            //elementoC com posição do char atual de elementoAeB.getA().toString(), caso seja maior,
não pegará mais nada do elementoAeB.getA().toString()
            if (i < elementoAeB.getA().toString().length()) {
                    elementoC = elementoC + elementoAeB.getA().toString().charAt(i);
            }

            //Caso i for menor que a quantidade de caracteres do objeto elementoAeB.getB().toString(),
concatena
            //elementoC com posição do char atual de elementoAeB.getB().toString(), caso seja maior,
não pegará mais nada do elementoAeB.getB().toString()
            if (i < elementoAeB.getB().toString().length()) {
                    elementoC = elementoC + elementoAeB.getB().toString().charAt(i);
            }

          }

       } catch (Exception e) {

       }

       return elementoC;
   }

}
```

```
package desafios2it;

import br.com.desafios2it.bean.ElementosBean;
import br.com.desafios2it.domain.elementos;

/**
 *
 * @author Emello
 */
public class DesafioS2IT {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        //Início do Exercício 8 Gerar C

        elementos elemen = new elementos();
        elemen.setA(75);
        elemen.setB(1256);
        ElementosBean elemenBean = new ElementosBean();

        if (elemen.getA() != null && elemen.getB() != null) {
            //Elemento C recebe valor retornado do método gerarC que está no pacote Bean na classe
elemenBean
            elemen.setC(Integer.valueOf(elemenBean.gerarC(elemen)));

            //Verifica se o elemento C é maior que 1000000, caso for elemento C recebe -1 senão, receberá o
valor dele mesmo
            elemen.setC(elemen.getC() > 1000000 ? -1 : elemen.getC());

            System.out.println("\n Exercício 8 - " + "O valor do elemento C é " + elemen.getC());

        } else {
            System.out.println("\n Exercício 8 - " + "Os elementos A e B não podem serem nulos!");
        }

        //Fim do Exercício 8


    }

}
```

9 - Considerando a estrutura de uma árvore binária:

public class BinaryTree {

int valor;

BinaryTree left;

BinaryTree right;

}

Desenvolva um método que dado um nó da árvore calcule a soma de todos os nós subsequentes.

RESPOSTA EXERCÍCIO 9

```
package br.com.desafios2it.domain;
/**
 *
 * @author Emello
 */
public class binaryTree {
   public int valor;
   public binaryTree left;
   public binaryTree right;

   public int getValor() {
      return valor;
   }
   public void setValor(int valor) {
      this.valor = valor;
   }
   public binaryTree getLeft() {
      return left;
   }
   public void setLeft(binaryTree left) {
      this.left = left;
   }
   public binaryTree getRight() {
      return right;
   }
   public void setRight(binaryTree right) {
      this.right = right;
   }
}
```

```
package br.com.desafios2it.bean;
import br.com.desafios2it.domain.binaryTree;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author Emello
 */
public class BinaryTreeBean {

   List<binaryTree> binaryTreeList = new ArrayList<>();

   public String somaDosNosSubsequentesDoNo(binaryTree raizParam) {

      if (raizParam == null) {
         //Caso a raízParam for nula retornará a mensagem abaixo
         return "A árvore está vazia!";
      }

      //Caso a raízParam não for nula adiciona na list binaryTreeList para poder iniciar o for
      binaryTreeList.add(raizParam);

      //Seta o valor 0 na variável somaTotalNosSubs, que será o retorno para o usuário dos nós
subsequentes
      int somaTotalNosSubs = 0;

//A lista torna-se crescente, conforme o left ou right do nó atual for diferente de nullo os mesmo são
adiconados na lista
      for (int i = 0; i < binaryTreeList.size(); i++) {

         //Como o primeiro nó será o nó raiz parametizado pelo usuário ele não será contado, somente
os subsequentes
         if(raizParam != binaryTreeList.get(i)){
           somaTotalNosSubs += binaryTreeList.get(i).getValor();
         }

         //Caso o nó left(esquerdo) for diferente de nulo será adicionado a lista binaryTreeList
         if (binaryTreeList.get(i).getLeft() != null) {
            binaryTreeList.add(binaryTreeList.get(i).getLeft());
         }

         //Caso o nó left(direito) for diferente de nulo será adicionado a lista binaryTreeList
         if (binaryTreeList.get(i).getRight() != null) {
            binaryTreeList.add(binaryTreeList.get(i).getRight());
         }
      }
      return ("A soma dos nós subsequentes do nó de valor " + raizParam.valor + " é: " +
somaTotalNosSubs);
   }
}
```

```
package desafios2it;
import br.com.desafios2it.bean.BinaryTreeBean;
import br.com.desafios2it.domain.binaryTree;
/**
 *
 * @author Emello
 */
public class DesafioS2IT {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        //Início do Exercício 9 Somar valores subsequentes de um determinado nó

        //Poderia também criar e utilizar um método construtor para setar os valores de cada objeto
        binaryTree A = new binaryTree();
        A.setValor(15);

        binaryTree B = new binaryTree();
        B.setValor(10);

        binaryTree C = new binaryTree();
        C.setValor(20);

        binaryTree D = new binaryTree();
        D.setValor(8);

       binaryTree E = new binaryTree();
        E.setValor(12);

        binaryTree F = new binaryTree();
        F.setValor(16);

        binaryTree G = new binaryTree();
        G.setValor(25);

        binaryTree H = new binaryTree();
        H.setValor(6);

        binaryTree I = new binaryTree();
        I.setValor(9);
```

```
        A.left = B;
         A.right = C;
         B.left = D;
         B.right = E;
         C.left = F;
         C.right = G;
         D.left = H;
         D.right = I;

         BinaryTreeBean binaryBean = new BinaryTreeBean();

/**
 * Após instanciar a classe BinaryTreeBean chamamos o método somaDosNosSubsequentesDoNo *
 * passando o nó que deverá ser somados os nós subsequentes do mesmo.
 *
 */

         //
         System.out.println("\n Exercício 9 - " + binaryBean.somaDosNosSubsequentesDoNo(D));

    }

}
```

Obs:- Tanto o exercício 8 quanto o 9 elaborei ambos em um único projeto no netbeans e subi no repositório, para fins de teste.

https://github.com/edsooon/desafioS2IT