MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DO ESPÍRITO SANTO
REITORIA
Avenida Rio Branco, 50 – Santa Lúcia – 29056-255 – Vitória – ES
27 3357-7500

CONCURSO PÚBLICO
EDITAL Nº 03 / 2015

Professor do Magistério do Ensino Básico, Técnico e Tecnológico

ÍNDICE DE INSCRIÇÃO	327/328
CAMPUS	Serra, Cachoeiro
ÁREA/SUBÁREA	Metodologia e Técnicas da Computação

PROVA DE CONHECIMENTOS ESPECÍFICOS | DISCURSIVA
MATRIZ DE CORREÇÃO

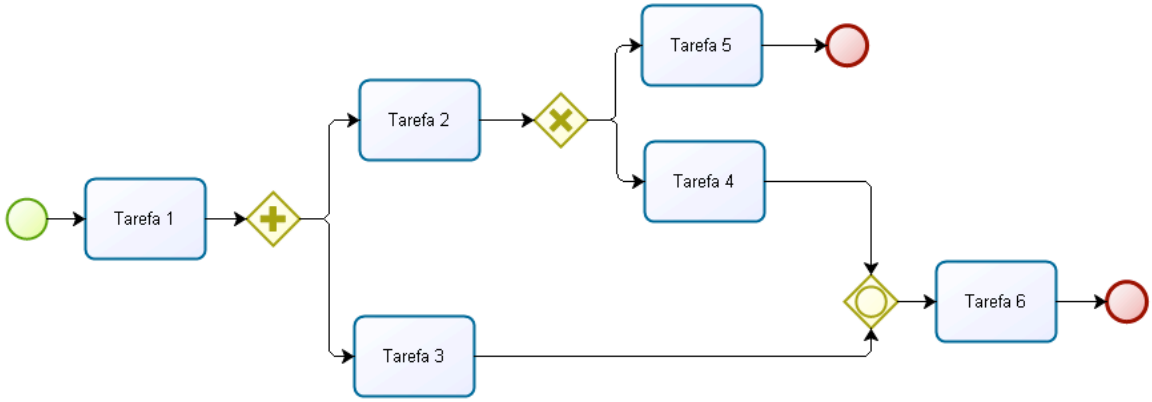
QUESTÃO 01

O problema que se apresenta neste fluxo é a possibilidade da ocorrência de um *deadlock* (travamento). Se o fluxo transcorrer para a Tarefa 5, a Tarefa 6 nunca ocorrerá, pois, o último *gateway* (*AND-Join*) aguardará a chegada dos *tokens* vindos de cada ramo entrante, neste caso, Tarefa 3 e Tarefa 4. Nessa situação, o *gateway* ficará aguardando indefinidamente a chegada do *token* da Tarefa 4, o que não ocorrerá.

Para resolver este problema, mantendo-se o fluxo atual, é preciso alterar o último *gateway* (antes da Tarefa 6). Existem duas possibilidades para resolver o problema do *deadlock*. Caso a escolha recaia sobre a substituição do *AND-Join* por um *XOR-Join*, resolve-se o problema do *deadlock*, no entanto, cria-se outro, a execução dupla da Tarefa 6, algo que se deve evitar.

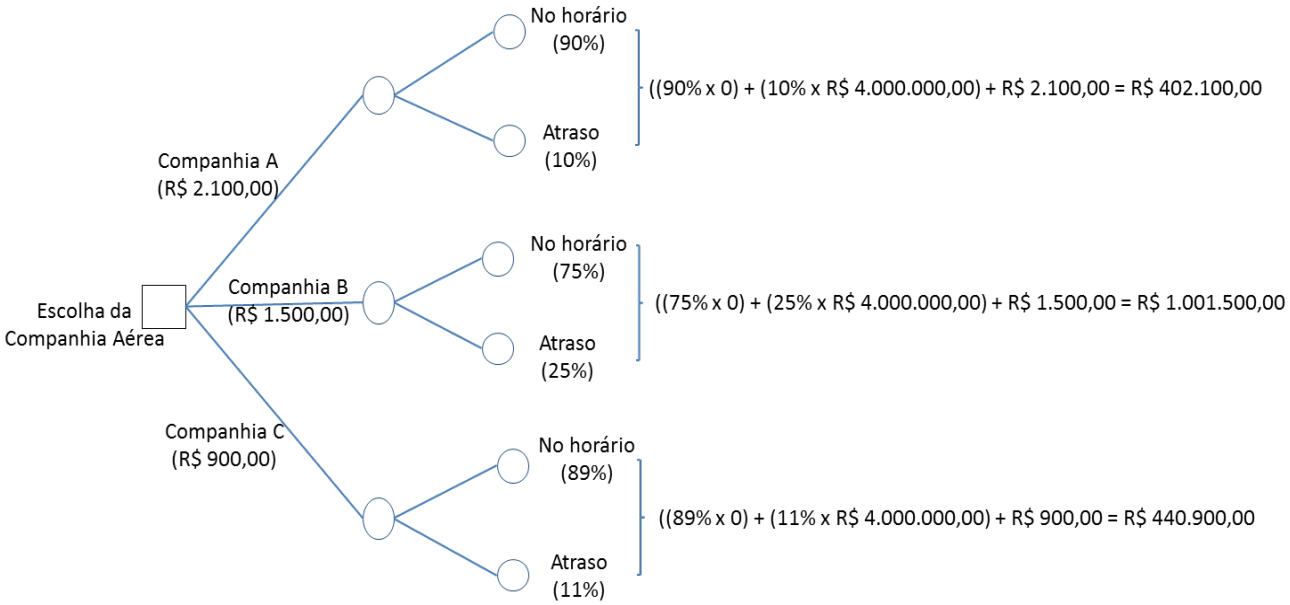
A solução definitiva é o uso do *OR-Join* no último *gateway*. Caso o fluxo se desvie para a Tarefa 5, o *OR-Join* não aguardará o *token* vindo da Tarefa 4, uma vez que ele nunca chegará. Caso ambos os *tokens* cheguem (das tarefas 4 e 3) no *gateway*, esses serão fundidos em apenas um *token*, fazendo com que a Tarefa 6 seja executada apenas uma única vez.

O diagrama final ficaria com a seguinte configuração:



QUESTÃO 02

Para a escolha da companhia aérea, deve-se considerar o prejuízo total que a empresa irá sofrer caso ocorra atraso do voo. Utilizando a técnica da árvore de decisão a escolha final recai sobre a Companhia A, pois é a que menor prejuízo causará à empresa, caso ocorra atraso. Veja o diagrama da árvore de decisão abaixo.



QUESTÃO 03

A) As linhas de código em que estão os comandos responsáveis por inserir cada linha do elemento de tabela identificado por "lista_alunos" e suas respectivas células são as de número 19, 20, 21, 22 e 23. Nestas linhas estão as chamadas aos métodos "insertRow", do objeto Table, e "insertCell", do objeto TableRow, que fazem a inserção de uma linha na tabela e a inserção de uma célula em uma linha da tabela, respectivamente. É aceitável que se indique também a linha 17, na qual uma referência para o objeto Table é obtida.

B) O preenchimento das células inseridas na nova linha da tabela é feito através do acesso à propriedade "innerHTML", do objeto TableData. Este acesso é feito nas linhas 24, 25, 26 e 27.

C) O evento programado para disparar a execução da função "carregarDados" é o de carga do objeto Body do documento HTML, conforme declaração feita na linha 32.

D) Após a execução da função "carregarDados", o elemento identificado por "lista_alunos" terá quatro linhas com os dados que foram adicionados ao vetor "listaAlunos" nas cinco primeiras linhas do código da função "carregarDados". O código HTML ficaria como abaixo:

```
<table id="lista_alunos">
<tbody>
<tr><td>32490</td><td>Aaaaaa Bbbbbb Ccccc</td><td>17</td><td>55027900001111</td></tr>
<tr><td>29871</td><td>Ddddddddd Eeee Fffffff</td><td>19</td><td>55031900001234</td></tr>
<tr><td>10982</td><td>Gggggggg Hhhh Ii Jjjjjjjj</td><td>18</td><td>55028900001010</td></tr>
<tr><td>10129</td><td>Llllllll Mmmmmm Nnnnnnn</td><td>19</td><td>55011900004321</td></tr>
</tbody>
</table>
```

QUESTÃO 04

Como não houve especificação do comportamento dos métodos das classes ControleCadastro e Aluno, os códigos desses métodos é de livre implementação pelos candidatos. No entanto, a estrutura de atributos e métodos das classes deve seguir à implementação abaixo:

```
import java.util.Date;
import java.util.ArrayList;

interface Registro {
    public Object obterIdentificador();
}

interface ControleCadastro {
    public int inserir(Registro item);
    public boolean alterar(Registro item);
    public boolean excluir(Registro item);
    public ArrayList selecionarTodos();
    public Registro selecionar();
}

class ControleCadastroAluno implements ControleCadastro {
    private ArrayList<Aluno> colecaoAlunos;

    public ControleCadastroAluno() {
        this.colecaoAlunos = new ArrayList<>();
    }

    @Override
    public int inserir(Registro item) {
        return 0;
    }

    @Override
```

```
    public boolean alterar(Registro item) {
        return true;
    }

    @Override
    public boolean excluir(Registro item) {
        return true;
    }

    @Override
    public ArrayList selecionarTodos() {
        return colecaoAlunos;
    }

    @Override
    public Registro selecionar() {
        if (colecaoAlunos.size() > 0)
            return colecaoAlunos.get(0);
        return null;
    }
}

class Aluno implements Registro {
    private int idAluno;
    private String nome;
    private Date dataNascimento;
    private String email;

    @Override
    public Object obterIdentificador() {
        return this.getIdAluno();
    }

    public Aluno(int idAluno, String nome, Date dataNascimento, String email) {
        this.idAluno = idAluno;
        this.nome = nome;
        this.dataNascimento = dataNascimento;
        this.email = email;
    }

    public int getIdAluno() {
        return this.idAluno;
    }

    public void setIdAluno(int idAluno) {
        this.idAluno = idAluno;
    }

    public String getNome() {
        return this.nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Date getDataNascimento() {
        return this.dataNascimento;
    }

    public void setDataNascimento(Date dataNascimento) {
        this.dataNascimento = dataNascimento;
    }

    public String getEmail() {
        return this.email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

QUESTÃO 05

Os casos de uso preliminares geralmente são escritos na forma de narrativas informais. Essas narrativas podem ser apresentadas como um texto simples ou como uma sequência enumerada de ações de usuário, sendo cada ação representada como uma sentença declarativa.

Os casos de uso preliminares refinados incluem a descrição de cenários secundários, que representam comportamentos alternativos ao cenário primário apresentado a princípio. Dessa forma, nesses casos de uso é comum encontrar, além do fluxo básico do cenário primário, descrições de condições e fluxos alternativos de ações.

Já nos casos de uso formais é encontrada uma estrutura composta por elementos como:

- Objetivo no contexto, que identifica o escopo geral do caso de uso;
- Precondição, que descreve o que é conhecido como verdadeiro antes de o caso de uso ser iniciado;
- Disparador, que identifica o evento ou a condição que faz iniciar o caso de uso;
- Cenário, que descreve a sequência básica de ações específicas que o ator deve tomar e as respostas apropriadas do sistema;
- Exceções, que identificam as situações reveladas à medida que o caso de uso preliminar é refinado.

Nos casos de uso formais também podem ser encontrados outros elementos, como prioridade, pós-condição, frequência de uso, atores secundários, etc. Os termos que denominam os elementos da estrutura do caso de uso formal também pode ser diferente, desde que haja uma relação de equivalência ou forte semelhança entre os significados desses elementos com os da estrutura descrita acima.

_____ / _____ / 2015

Assinatura Presidente

Assinatura Membro