# Seleção Residência em TI - TST

**Professor(a):** gustavo.leitao@imd.ufrn.br

**Cód. Prova:**

**Nome:**

**Matrícula:** **Assinatura:** _____

**Questão 1** *Valor da questão: 0,33*

Sobre interfaces e classes abstratas na linguagem Java 8+, julgue os itens a seguir:

I - Tanto as interfaces como as classes podem conter métodos abstratos
II - As interfaces não podem armazenar estado
III - Apenas as classes abstratas podem conter métodos concretos
IV - As classes abstratas devem conter pelo menos um método abstrato
V - Uma classe pode implementar várias interfaces, mas só pode herdar de uma única classe abstrata

a) Apenas I, II e V
b) Apenas II e V
c) Apenas I, II, III e V
d) Apenas I, III e IV
e) Apenas II, III e IV

**Questão 2** *Valor da questão: 0,33*

*Design Patterns* ou padrões de projetos são soluções generalistas para problemas recorrentes durante o desenvolvimento de um *software*. Tais conceitos ficaram realmente conhecidos em 1994, quando os engenheiros de *software* Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides escreveram o livro "*Design Patterns: Elements of Reusable Object-Oriented Software*" com o objetivo de catalogar problemas comuns aos projetos de desenvolvimento de *software* e as formas de resolver esses problemas. Os autores do livro ficaram conhecidos como *Gang of Four* (Gangue dos quatro) ou "GoF".

Neste contexto, analise os seguintes códigos-fonte:

**Código-fonte #1:**

```
public final class ClasseSecreta {
    private static ClasseSecreta x;
    public String valor;

    private ClasseSecreta(String valor) {
        try {
            Thread.sleep(5000);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
        this.valor = valor;
    }

    public static ClasseSecreta getX(String valor) {
        if (x == null) {
            x = new ClasseSecreta(valor);
        }
        return x;
    }
}
```

**Código-fonte #2:**

No contexto de padrões de projeto, marque a alternativa que realiza a associação mais adequada entre o código-fonte exposto e o padrão de projeto indicado.

```java
public class TemperaturaEmCelsius {
    private double temperatura;
    public TemperaturaEmCelsius(double temperatura) {
        this.temperatura = temperatura;
    }
    public TemperaturaEmCelsius() { }
    public double getTemperatura() {
        return temperatura;
    }
}

public class TemperaturaEmFahrenheit {
    private double temperatura;
    public TemperaturaEmFahrenheit(double temperatura) {
        this.temperatura = temperatura;
    }
    public double getTemperatura() {
        return temperatura;
    }
}

public class DesignPatternTemperatura extends TemperaturaEmCelsius {
    private TemperaturaEmFahrenheit temp_F;

    public DesignPatternTemperatura(TemperaturaEmFahrenheit temp_F) {
        this.temp_F = temp_F;
    }

    private double convertFtoC(double temperatura) {
        return (temperatura - 32.0) * (5.0/9.0);
    }

    public double getTemperatura() {
        return convertFtoC(temp_F.getTemperatura());
    }
}
```

a) #1 - *Factory*, #2 - Prototype
b) #1 - *Singleton*, #2 - *Adapter*
c) #1 - *Factory*, #2 - Strategy
d) #1 - *Singleton*, #2 - *Builder*
e) #1 - *Abstract Factory*, #2 - *Prototype*

**Questão 3** *Valor da questão: 0,33*

O jQuery pode selecionar elementos HTML a partir de seletores, otimizando o desenvolvimento. Sobre a utilização de seletores no jQuery é correto afirmar:

a) $(‘.paragrafo’); retornará o elemento com id igual a “paragrafo”.
b) $('a[rel=’fotos’]'); retornará todos os links
c) $(span div); retornará todos os elementos div que são filhos de span
d) $(‘.p’); retornará todos os elementos <p>
e) $('#formulario'); retornará todos os elementos da classe "formulario"

**Questão 4** *Valor da questão: 0,33*

Considere a seguinte página HTML:

```html
<html>

<head>
    <title>Residência em TI - TST</title>
</head>

<body>
    <header id="titulo-1">
        <h1 id="titulo-2"><em id="em-1">Residência em Tecnologia da Informação</em></h1>
    </header>
    <p id="subtitulo-1">Atividades dos residentes:</p>
    <ul id="lista-1">
        <li id="topico-1">Desenvolver projetos de inovação</li>
        <li id="topico-2">Obter experiência de mercado</li>
        <li id="topico-3">Trabalhar em times de desenvolvimento</li>
        <li id="topico-4">
            Cursar disciplinas:
            <ol id="lista-2">
                <li id="disciplina-1">Gerência de Configuração e Testes</li>
                <li id="disciplina-2">Desenvolvimento para Dispositivos Móveis</li>
                <li id="disciplina-3">Desenvolvimento <em id="em-2">Web</em></li>
                <li id="disciplina-4">Engenharia de Requisitos Aplicada</li>
                <li id="disciplina-5">entre outras...</li>
            </ol>
        </li>
    </ul>
    <div id="div-1">
        <img id="img-1" src="metropole.jpg">Nossa logomarca!</img>
    </div>
</body>

</html>
```

Abaixo são listados seletores e os respectivos elementos (indicados pelos seus ID's) selecionados por tal seletor. Julgue se tal seleção é verdadeira ou falsa:

**I.** `p`
`#subtitulo-1`

**II.** `ol li`
`#disciplina-1, #disciplina-2, #disciplina-3, #disciplina-4, #disciplina-5`

**III.** `li em`
`#em-2`

**IV.** `ul > li`
`#topico-1, #topico-2, #topico-3, #topico-4`

**V.** `li li`
`#disciplina-1, #disciplina-2, #disciplina-3, #disciplina-4, #disciplina-5`

**VI.** `ul > ol`
`#lista-2`

**VII.** `p, img`
`#subtitulo-1, #img-1`

A ordem correta de julgamento das afirmações anteriores é:

a) I - verdadeira, II - verdadeira, III - verdadeira, IV - verdadeira, V - verdadeira, VI - falsa, VII - verdadeira
b) I - verdadeira, II - falsa, III - falsa, IV - verdadeira, V - falsa, VI - verdadeira, VII - verdadeira
c) I - verdadeira, II - verdadeira, III - verdadeira, IV - verdadeira, V - verdadeira, VI - verdadeira, VII - verdadeira
d) I - verdadeira, II - verdadeira, III - verdadeira, IV - falsa, V - verdadeira, VI - falsa, VII - verdadeira
e) I - verdadeira, II - falsa, III - falsa, IV - falsa, V - falsa, VI - falsa, VII - verdadeira

**Questão 5** *Valor da questão: 0,33*

Considere as seguintes classes implementadas em Java:

```
public class A {
public void print(){
System.out.print(1);
}
}

public class B extends A {
@Override
public void print() {
super.print();
System.out.print(2);
}
}

public class C extends B {
}
```

Com isso, assinale a alternativa que indica o que será impresso no console quando a classe Main for executada:

```
public class Main {
public static void main(String[] args) {
new A().print();
new B().print();
new C().print();
}
}
```

a) 1122
b) 11212
c) 122
d) 1212
e) 1121

**Questão 6** *Valor da questão: 0,33*

Analise o código abaixo que implementa a soma paralela da variável estática number:

```java
import java.util.ArrayList;
import java.util.List;

public class Concurrency implements Runnable{

    private static int number = 0;

    public void increment(){
        number = number + 1;
    }

    @Override
    public void run() {
        increment();
    }

    public static void main(String[] args) throws InterruptedException {

        List<Thread> threads = new ArrayList<>();
        for (int i = 0 ;i < 100000; i++){
            var t = new Thread(new Concurrency());
            t.start();
            threads.add(t);
        }

        for (Thread t : threads){
            t.join();
        }

        System.out.println(number);

    }

}

```

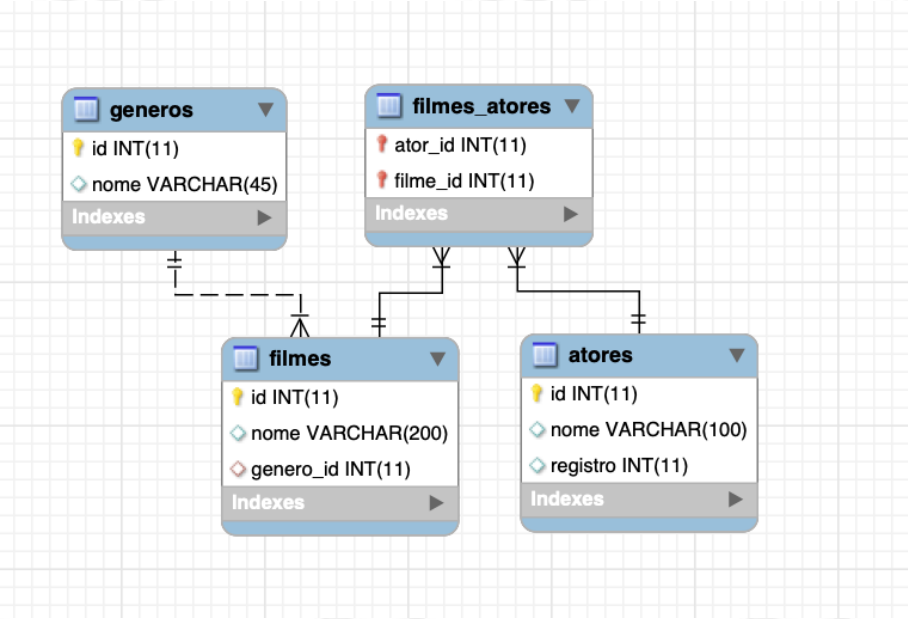
Da forma que está escrito, o programa possui comportamento não determinístico. Ou seja, a saída pode ser diferente para execuções diferentes. Esse comportamento não é desejado e tem origem em uma falha de concorrência.

Sabendo disso, escolha opção que implementa uma solução válida no método increment para resolver o problema mencionado:

a)
```
public void increment(){
synchronized (Currency.class){
number = number + 1;
}
}
```
b)
```
public synchronized void increment(){
number = number + 1;
}
```
c)
```
public atomic void increment(){
number = number + 1;
}
```
d)
```
public void increment(){
synchronized {
number = number + 1;
}
}
```
e)
```
public void increment(){
AtomicInteger atomicInteger = new AtomicInteger(number);
number = atomicInteger.incrementAndGet();
}
```

Considere o seguinte enunciado para responder às questões de 7 a 8.

Dado o seguinte diagrama de EER (Enhanced Entity-Relationship) extraído de um banco de dados PostgreSQL, responda as questões que se seguem:




**Questão 7** *Valor da questão: 0,33*

Assinale a alternativa que contém consulta válida para contabilizar a quantidade de filmes registrados no banco de dados cujo o gênero possui nome 'Drama'.

a) `SELECT count(*) from filmes f JOIN generos ON f.id = generos.genero_id WHERE generos.nome = 'Drama'`
b) `SELECT count(*) from filmes WHERE genero_id = 'Drama'`
c) `SELECT count(*) from filmes f JOIN generos ON f.id = generos.genero_id HAVING generos.nome = 'Drama'`
d) `SELECT count(*) from filmes f JOIN generos ON f.genero_id = generos.id HAVING generos.nome = 'Drama'`
e) `SELECT count(*) from filmes f JOIN generos ON f.genero_id = generos.id WHERE generos.nome = 'Drama'`

**Questão 8** *Valor da questão: 0,33*

Assinale a alternativa que contém consulta válida para buscar todos os nomes de filmes registrados no banco de dados e interpretados pela atriz de nome 'Angelina Jolie'.

a)
```
SELECT f.nome from filmes_atores fa
  JOIN filmes f ON f.id = fa.filme_id
  JOIN atores a ON a.id = fa.ator_id
  WHERE f.nome = 'Angelina Jolie'
```
b)
```
SELECT f.nome from filmes f
  LEFT OUTER JOIN filmes_atores ON f.id = filmes_atores.filme_id
  LEFT OUTER JOIN atores a ON a.id = filmes_atores.ator_id
  WHERE a.nome = 'Angelina Jolie'
```
c)
```
SELECT f.nome from filmes f
  LEFT OUTER JOIN filmes_atores ON f.filme_id = filmes_atores.id
  LEFT OUTER JOIN atores a ON a.id = filmes_atores.ator_id
  WHERE a.nome = 'Angelina Jolie'
```
d)
```
SELECT f.nome from filmes_atores fa
  JOIN filmes f ON f.filme_id = fa.filme_id
  JOIN atores a ON a.ator_id = fa.id
  WHERE a.nome = 'Angelina Jolie'
```
e)
```
SELECT f.nome from filmes f
  LEFT OUTER JOIN filmes_atores ON f.filme_id = filmes_atores.id
  LEFT OUTER JOIN atores a ON a.ator_id = filmes_atores.id
  WHERE a.nome = 'Angelina Jolie'
```

**Questão 9** *Valor da questão: 0,33*

Sobre o desenvolvimento nativo iOS, considere as assertivas sobre **IBOutlets**, **IBActions**.

I - Podemos usar o Interface Builder para associar os elementos da interface do usuário com nosso próprio código conectando os objetos e métodos em nosso código como IBOutlet e IBAction, respectivamente.

II - Por meio do Interface Builder, um IBOutlet está associado a um componente e é capaz de modificar tal componente, como a sua visualização e o seu estado, quando um evento ocorre e uma ação é disparada pelo IBActions. Assim, IBOutlet ou IBAction afetam como o código é compilado.

III - Outlets são uma forma de composição de objetos, que é um padrão dinâmico que requer que um objeto adquira, de alguma forma, referências a seus objetos subsequentes para que possa enviar mensagens a eles.

IV - IBAction permite computador alguma tarefa associada a um evento ocorrido. Assim, IBAction é o retorno de um método que recebe um evento enviado pelo Interface Builder.

Assinale a alternativa que indica as sentenças CORRETAS:

a) II, III
b) I, III, IV
c) I, II, IV
d) III, IV
e) I, II, III

Considere o trecho de código abaixo para responder às questões 10 a 11.

Considere o trecho de código abaixo:

```java
public class FilmeListaFragment extends ListFragment {

    List<Filme> mFilmes;
    ArrayAdapter<Filme> mAdapter;


    @Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);

        mFilmes = carregaFilmes();

        mAdapter = new ArrayAdapter<Filme>(
                getActivity(),
                android.R.layout.simple_list_item_1,
                mFilmes
        );
        setListAdapter(mAdapter);
    }

    private List<Filme> carregaFilmes(){
        List<Filme> filmes = new ArrayList<Filme>();
        filmes.add(new Filme("Filme 1"));
        filmes.add(new Filme("Filme 2"));
        filmes.add(new Filme("Filme 3"));
        filmes.add(new Filme("Filme 4"));

        return filmes;
    }

    public FilmeListaFragment() {

    }
}
```

**Questão 10** *Valor da questão: 0,33*

Ainda sobre o código descrito, considere que o método de clicar nos elementos da lista foi implementado e que uma interface foi utilizada como mecanismo para notificar a Activity de FilmeListaFragment, chamada de **FilmeActivity**, que um item foi clicado. Segue o código da interface.

```java
public interface AoClicarNoFilme{
        void clicouNoFilme(Filme filme);
    }

```

Assinale a alternativa que indica o trecho de código de **FilmeActivity** que, ao identificar um clique em um elemento da lista, abre uma nova tela, chamada Detalhe, passando o filme como parâmetro.

a)

```
@Override
    public void clicouNoFilme(Filme filme) {
        Intent it = new Intent(getCurrentContext(),DetalheActivity.class);
        it.putExtra("filme",filme);
        it.getContext().inflate();
    }
```

b)

```
@Override
    public void clicouNoFilme(Filme filme) {
        Intent it = new Intent(this,DetalheActivity.class);
        it.put("filme",filme);
        it.inflate();
    }
```

c)

```
@Override
    public void clicouNoFilme(Filme filme) {
        Intent it = new Intent(this,DetalheActivity.class);
        it.putExtra("filme",filme);
        startActivityForResult(it);
    }
```

d)

```
@Override
    public void clicouNoFilme(Filme filme) {
        Intent it = new Intent(getCurrentContext(),DetalheActivity.class);
        it.put("filme",filme);
        startActivity(it);
    }
```

e)

```
@Override
    public void clicouNoFilme(Filme filme) {
        Intent it = new Intent(this,DetalheActivity.class);
        it.putExtra("filme",filme);
        startActivity(it);
    }
```

**Questão 11** *Valor da questão: 0,33*

O codigo ilustrado implementa um fragmento com uma listagem de filmes. Julgue as assertivas a seguir.

I. Similar a ListActivity, a classe FilmeListaFragmet herda de ListFragment, onde será exibido apenas um componente ListView.
II. mAdapter é o componente responsável por exibir a lista na tela. Por sua vez, setListAdapter é o método de que recebe o mAdapter e, dessa forma, gera uma instância de ArrayAdapter.
III. Para processar o clique em itens da lista, pode-se sobreescrever o método onClickListItem do ListFragment, que recebe, entre outros parâmetros, a própria lista e a posição do elemento clicado.
IV. Para adicionar o Fragmento FilmeListaFragment a uma Activity, pode-se adicionar a tag <fragment>, com a propriedade nome preenchida, no arquivo de layout da respectiva Activity.

Assinale a alternativa que indica as assertivas CORRETAS

a) II, III
b) I, IV
c) III, IV
d) II, IV
e) I, II

**Questão 12** *Valor da questão: 0,33*

O padrão *async/await* é um recurso sintático de muitas linguagens de programação que permite que uma função assíncrona e sem bloqueio (*non-blocking*) seja estruturada de maneira semelhante a uma função síncrona comum. Diante deste contexto, Indique a opção que representa a saída (*output*) do console após a execução deste programa em Javascript:

```
function init() {
    console.log("macaxeira");
    ola();
    funcaoSync();
}

async function ola() {
    await resolveEmDoisSegundos();
    console.log("carne de sol");
}

async function resolveEmDoisSegundos() {
    // considere que essa função "resolve" em 2 segundos, mas não tem output.
}

function funcaoSync() {
    setTimeout(() => {
        console.log("farofa");
    }, 1000);
}

init();
console.log("arroz");
```

a) `macaxeira`
   `carne de sol`
   `farofa`
   `arroz`

b) `macaxeira`
   `arroz`
   `carne de sol`
   `farofa`

c) `macaxeira`
   `arroz`
   `farofa`
   `carne de sol`

d) `arroz`
   `macaxeira`
   `farofa`
   `carne de sol`

e) `macaxeira`
   `farofa`
   `carne de sol`
   `arroz`

**Questão 13** *Valor da questão: 0,33*

Considere o seguinte programa Javascript:

```
(function () {
  window.addEventListener("load", init);

  function init() {
    id("meu-btn").addEventListener("click", gato);
    id("meu-outro-btn").addEventListener("click", cachorro);
    id("meu-btn").addEventListener("click", galinha());
  }

  function cachorro() {
    console.log("au au!");
    id("meu-outro-btn").addEventListener("click", function () {
      console.log("muuuu!");
    });
  }

  function gato() {
    console.log("miau!");
    cachorro();
  }

  function galinha() {
    console.log("cocorocó!");
    id("meu-outro-btn").removeEventListener("click", cachorro);
  }

  function id(id) {
    return document.getElementById(id);
  }
})();
```

Assumindo que os botões `#meu-btn` e `#meu-outro-btn` existem no HTML correspondente, marque a opção que indica a saída correta no console quando a página carregar e os seguintes eventos ocorrerem:

1. Usuário clica no botão `#meu-btn`
2. Usuário clica no botão `#meu-outro-btn`
3. Usuário clica no botão `#meu-btn`

a)
```
cocorocó!
cocorocó!
```

b)
```
cocorocó!
  miau!
  au au!
  muuuu!
  miau!
  au au!
```

c)
```
miau!
  au au!
  cocorocó!
  muuuu!
  miau!
  au au!
  cocorocó!
```

d)
```
cocorocó!
  au au!
```

e)
```
miau!
  au au!
  au au!
  muuuu!
  miau!
  au au!
```

**Questão 14** *Valor da questão: 0,33*

O Git é um importante sistema que auxilia a no controle de versão. Sabendo disso, analise as sentenças abaixo e assinale a alternativa correta.

I - Para o armazenamento dos arquivos, deve possuir um servidor central
II - Ao comitar os arquivos saem do *work directory* e vão para o *staging area*.
III - O comando de git *rebase* permite atualizar a base de uma *branch* aplicando os *commits* locais em cima de uma nova base
IV - Para enviar mudanças a um repositório remoto, deve-se utilizar o comando git push

a) Apenas II, III e IV
b) Apenas III e IV
c) Apenas I e IV
d) Apenas II e III
e) Apenas I e II

**Questão 15** *Valor da questão: 0,33*

Considere o seguinte techo de código de um ***Controller*** em uma **Aplicação Spring REST** completamente configurada:

```java
@RestController
@ANOTACAO1
public class EstudanteRestController {

    private List<Estudante> listaEstudantes;

    @ANOTACAO2
    public void carregaDados() {

        listaEstudantes = new ArrayList<>();

        listaEstudantes.add(new Estudante("Poornima", "Patel"));
        listaEstudantes.add(new Estudante("Mario", "Rossi"));
        listaEstudantes.add(new Estudante("Mary", "Smith"));
    }

    @ANOTACAO3
    public List<Estudante> getEstudantes() {

        return listaEstudantes;
    }

    @ANOTACAO4
    public Estudante getEstudante(@PathVariable int estudanteId) {

        if ( (estudanteId >= listaEstudantes.size()) || (estudanteId < 0) ) {
            throw new EstudanteNotFoundException("Estudante id not found - " + estudanteId);
        }

        return listaEstudantes.get(estudanteId);

    }

    @ANOTACAO5
    public ResponseEntity<EstudanteErrorResponse> handleException(EstudanteNotFoundException exc) {

        EstudanteErrorResponse error = new EstudanteErrorResponse();

        error.setStatus(HttpStatus.NOT_FOUND.value());
        error.setMessage(exc.getMessage());
        error.setTimeStamp(System.currentTimeMillis());

        return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
     }
}
// ========================================================================== //

public class EstudanteNotFoundException extends RuntimeException {

    public EstudanteNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }

    public EstudanteNotFoundException(String message) {
        super(message);
    }

    public EstudanteNotFoundException(Throwable cause) {
        super(cause);
    }

}

```

Para que as requisições feitas possam mostrar, de maneira correta, a lista de estudantes, um estudante específico ou uma mensagem de erro tratada, caso não ache o estudante, precisamos utilizar substituir os termos

ANOTACAO1, ANOTACAO2, ANOTACAO3, ANOTACAO4 e ANOTACAO5 por anotações apropriadas. Assinale a alternativa que apresenta as anotações corretas, respectivamente.

a)
```
@RequestMap("/api")

@Construct

@GetMapping("/Estudantes")

@PostMapping("/Estudantes/{EstudanteId}")

@ExceptionHandler
```
b)
```
@Request("/api")

@PostConstruct

@PutRequest("/Estudantes")

@PatchRequest("/Estudantes/{EstudanteId}")

@ExceptionHandler
```
c)
```
@RequestMap("/api")

@InitConstruct

@Get("/Estudantes")

@Patch("/Estudantes/{EstudanteId}")

@ExceptionRunTime
```
d)
```
@RequestMapping("/api")

@InitConstruct

@PostMapping("/Estudantes")

@PostMapping("/Estudantes/{EstudanteId}")

@ExceptionRunTime
```
e)
```
@RequestMapping("/api")
@PostConstruct
@GetMapping("/Estudantes")
@GetMapping("/Estudantes/{EstudanteId}")
@ExceptionHandler
```

**Questão 16** *Valor da questão: 0,33*

Considere um sistema de um banco, em que temos clientes e funcionários bancários realizando diversas funcionalidades no *software*. Durante o levantamento de requisitos, foi identificado o seguinte requisito:

O cliente pode fazer um pedido de crédito no sistema, e o banco tem um período de 2 meses para aprovar ou não esse crédito. A aprovação pode ser feita apenas através de uma análise de crédito do cliente requisitante. No entanto, dependendo do valor do crédito e da renda do cliente, pode ser necessário fazer uma atualização cadastral do cliente para uma reavaliação de crédito.

Considere as seguintes afirmações sobre a modelagem UML deste sistema:

**I** - No diagrama de casos de uso, o caso de uso "*Analisar pedido de crédito*" terá um relacionamento de inclusão com o caso de uso "*Atualizar cadastro do cliente*".
**II** - No diagrama de sequência, o pedido de crédito feito pelo cliente para o banco pode ser representado através de uma mensagem síncrona entre as classes *Conta* e *Agencia*.
**III** - No diagrama de classes, a classe *PedidoCredito* pode se relacionar com a classe *Cliente* através de uma composição ou uma associação, independente da multiplicidade.
A análise das afirmativas permite concluir que:

a) Todas as afirmações estão corretas.
b) Apenas a afirmação I está correta.
c) As afirmações I e II estão corretas.
d) Nenhuma afirmação está correta.
e) As afirmações II e III estão corretas.

**Questão 17** *Valor da questão: 0,33*

Considere que você é o atual responsável por desenvolver o código Javascript do *front-end* de uma aplicação *web* chamada ***FastDietas!***. Veja um segmento da página HTML desta aplicação:

```
<body>
    <h1>FastDieta!</h1>
    <p>
        Clique no botão para gerar um planejamento diário de refeiçoes aleatoriamente!
        <button id="dieta-btn">Solicitar!</button>
    </p>
    <section id="planejamento-diario" class="hidden">
        <article id="cafe">
            <h2><span class="nome"></span> (Café da manhã)</h2>
            <p class="descricao"></p>
            <p>Categorias:</p>
            <ul class="categorias"></ul>
        </article>
        <article id="almoco">
            <h2><span class="nome"></span> (Almoço)</h2>
            <p class="descricao"></p>
            <p>Categorias:</p>
            <ul class="categorias"></ul>
        </article>
        <article id="jantar">
            <h2><span class="nome"></span> (Jantar)</h2>
            <p class="descricao"></p>
            <p>Categorias:</p>
            <ul class="categorias"></ul>
        </article>
    </section>
</body>
```

Ao clicar no botão `#dieta-btn` nossa página irá realizar uma requisição *fetch* para o *end-point* GET `/dieta` da nossa API, retornando como resposta um planejamento diário de refeições aleatoriamente do nosso banco de dados. A resposta é retornada no formato JSON como no exemplo abaixo:

```
{
"cafe": {
"nome": "Cereal",
"descricao": "200g de cereal com 100ml de iogurte natural.",
"categorias": ["Grãos", "Iogurte"]
},
"almoco": {
"nome": "Macarrão com molho de tomate",
"descricao": "Macarrão integral com molho de tomate natural caseiro.",
"categorias": ["Carboidratos"]
},
"jantar": {
"nome": "Sopa com torradas",
"descricao": "Porção de sopa acompanhado de 2 torradas de pão integral.",
"categorias": ["Carboidratos", "Outros"]
}
}
```

Caso a requisição retorne com sucesso, nossa página deve reagir da seguinte forma:

- A classe `.hidden` deve ser removida do elemento `#planejamento-diario`;
- A resposta JSON deve ser usada para definir os valores de cada um dos elementos `#cafe`, `#almoco` e `#jantar`, de tal forma que:
  - O elemento com `.nome` é definido com o nome da refeição correspondente;
  - O elemento com `.descricao` é definido com a descrição da refeição correspondente;
  - O elemento com `.categorias` é definido com a lista de categorias correspondente daquela refeição (uma ou mais categorias podem estar presentes no *array* "categorias" contido no JSON).

Deve ser permitido ao usuário clicar no botão `#dieta-btn` múltiplas vezes, sendo assim gerado novamente o seu planejamento (resultados anteriores devem ser substituídos). Nosso desenvolvedor sênior já deu início a implementação do código Javascript, sendo necessário somente implementar a função `defineRefeicoes`, conforme detalhado no código abaixo:

```
(function () {
    window.addEventListener("load", () => {
        id("dieta-btn").addEventListener("click", solicitaRefeicoes);
    });

    function solicitaRefeicoes() {
        fetch("/dieta")
            .then(checkStatus)
            .then((resp) => resp.json())
            .then(defineRefeicoes)
            .catch(console.log);
    }

    function defineRefeicoes(resposta) {
        // TODO
    }

    function id(id) {
        return document.getElementById(id);
    }

    function qs(selector) {
        return document.querySelector(selector);
    }

    function create(tag) {
        return document.createElement(tag);
    }

})();
```

Neste contexto, marque a opção que contenha a implementação correta da função `defineRefeicoes` para atender todos os requisitos previamente descritos.

*Observação: Não é necessário conhecer os detalhes dos estilos CSS usados na página.*

a)

```
function defineRefeicoes(resposta) {
    id("planejamento-diario").classList.removeClass("hidden");
    for (let key in resposta) {
        let item = resposta[key];
        qs(".nome #" + key).textContent = item.nome;
        qs(".descricao #" + key).textContent = item.descricao;
        let ul = qs(".categorias #" + key);
        ul.innerHTML = "";
        for (let i = 0; i < item["categorias"].length; i++) {
            let li = create("li");
            li.textContent = item["categorias"][i];
            ul.insertElement(li);
        }
    }
}
```

b)

```
function defineRefeicoes(resposta) {
    id("planejamento-diario").classList.remove("hidden");
    for (let key in resposta) {
        let item = resposta[key];
        qs("#" + key + " .nome").textContent = item.nome;
        qs("#" + key + " .descricao").textContent = item.descricao;
        let ul = qs("#" + key + " .categorias");
        ul.innerHTML = "";
        for (let i = 0; i < item["categorias"].length; i++) {
            let li = create("li");
            li.textContent = item["categorias"][i];
            ul.appendChild(li);
        }
    }
}
```

c)

```
function defineRefeicoes(resposta) {
    for (let key in resposta) {
        let item = resposta[key];
        qs("#" + key + " .nome").textContent = item.nome;
        qs("#" + key + " .descricao").textContent = item.descricao;
        let ul = qs("#" + key + " .categorias");
        for (let i = 0; i < item["categorias"].length; i++) {
            let li = create("li");
            li.textContent = item["categorias"][i];
            ul.appendChild(li);
        }
    }
}
```

d)

```
function defineRefeicoes(resposta) {
    id("planejamento-diario").classList.remove("hidden");
    for (let key in resposta) {
        let item = resposta[key];
        qs(".nome").textContent = item.nome;
        qs(".descricao").textContent = item.descricao;
        let ul = qs(".categorias");
        ul.innerHTML = "";
        for (let i = 0; i < item["categorias"].length; i++) {
            let li = create("li");
            li.textContent = item["categorias"][i];
            ul.appendChild(li);
        }
    }
}
```

e)

```
function defineRefeicoes(resposta) {
    id("planejamento-diario").classList.remove("hidden");
    for (let key in resposta) {
        let item = resposta[key];
        qs(".nome").textContent = item.nome;
        qs(".descricao").textContent = item.descricao;
        let ul = qs(".categorias");
        ul.innerHTML = "";
        for (let i = 0; i < item["categorias"].length; i++) {
            let new_ul = create("ul");
            new_ul.textContent = item["categorias"][i];
            ul.appendChild(new_ul);
        }
    }
}
```

**Questão 18** *Valor da questão: 0,33*

Analise o código Javascript abaixo:

```
function requisitar_1() {
    var requisicao = new XMLHttpRequest();
    requisicao.onload = () => console.log(JSON.parse(requisicao.responseText));
    requisicao.open("GET", apiUrl, true);
    requisicao.send();
}

function requisitar_2() {
    fetch(apiUrl)
        .then(function (response) {
            response.json().then((data) => console.log(data));
        })
}

function requisitar_3() {
    $.getJSON(apiUrl, (data) => console.log(data));
}
```

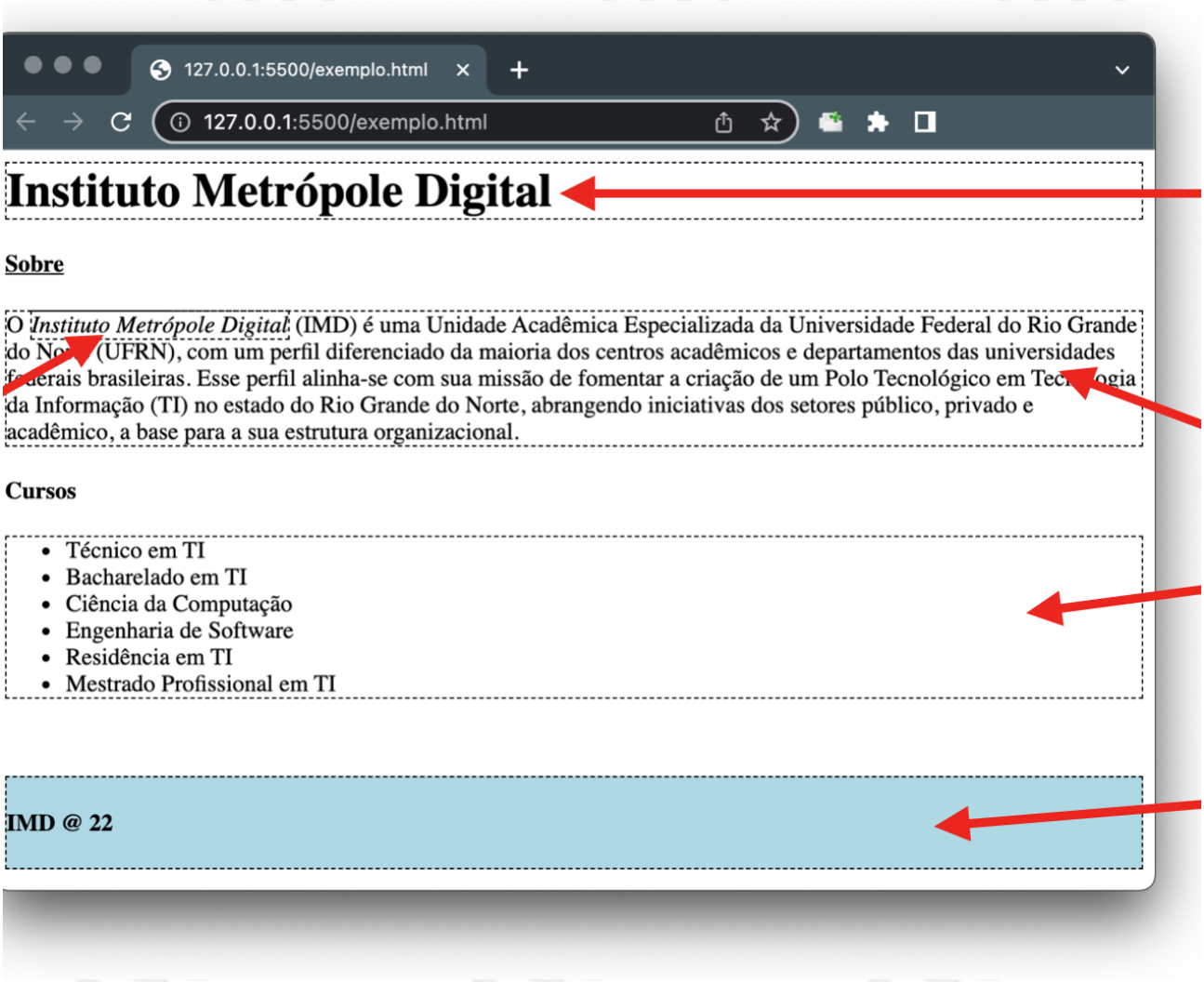
Após analisar o código acima, marque a alternativa **correta**:

a) A sintaxe de implementação da função `requisitar_3()` é inválida.
b) Todos os métodos indicados fazem requisições GET síncronas ao servidor com endereço indicado na variável `apiUrl`.
c) O método `requisitar_2()` faz uso da biblioteca jQuery.
d) O método `requisitar_3()` faz uso da API Fetch.
e) A execução de qualquer uma das três funções resulta na mesma saída no console.

**Questão 19** *Valor da questão: 0,33*

Considere o layout da página *web* abaixo. Para cada uma das 5 setas (cada uma apontando para um único elemento envolvido por uma borda tracejada), identifique a *tag* HTML mais apropriada da lista de *tags* a seguir.

**`<head> <header> <main> <footer> <nav> <title> <section> <input> <h1> <p> <ol> <ul> <hr> <code> <em> <strong> <button>`**




Marque a opção que faz a associação mais apropriada entre os elementos enumerados no *layout* e as *tags* listadas.

a) `1 - <strong>, 2 - <em>, 3 - <section>, 4 - <ol>, 5 - <footer>`
b) `1 - <h1>, 2 - <em>, 3 - <p>, 4 - <ul>, 5 - <footer>`
c) `1 - <h1>, 2 - <em>, 3 - <p>, 4 - <ul>, 5 - <header>`
d) `1 - <head>, 2 - <strong>, 3 - <p>, 4 - <ul>, 5 - <footer>`
e) `1 - <h1>, 2 - <strong>, 3 - <nav>, 4 - <ul>, 5 - <p>`

**Questão 20** *Valor da questão: 0,33*

Considere o seguinte conjunto de dados armazenados na variável Javascript de nome products:

```
const products = [{"sku":43900,"name":"Duracell - AAA Batteries (4-Pack)","price":5.49},
{"sku":1065859,"name":"Digital Cordless Expansion Handset - Black","price":49.99},
{"sku":1153016,"name":"Nikon - Digital Camera Body Cap","price":11.99},
{"sku":1192182,"name":"Elite Screens - YardMaster 200", "price":1599.99},
{"sku":1446341,"name":"Lowepro - Magnum 400 AW Camera Bag - Black","price":259.99}]
```

De posse dos dados, você foi convidado a escrever um código em Javascript para retornar a média de preço, daqueles produtos com valor acima de 10 reais. Assinale a alternativa que corretamente implementa tal requisito em uma função de nome calculateAvg.

a)
```
const calculateAvg = (p) => {
  const f = p.filter(i => i.price > 10)
  return f.sum(i => i.price).avg();
  }
```

b)
```
const calculateAvg = (p) => {
  const f = p.map(i => i.price)
  return f.filter(i => i.price > 10)
  .reduce((a,v) => a+=v) / f.length
  }
```

c)
```
const calculateAvg = (p) => {
  const f = p.filter(price > 10)
  return f.map(i)
  .reduce((a,v) => a+=v) / p.length
  }
```

d)
```
const calculateAvg = (p) => {
  const f = p.filter(i => i.price > 10)
  return f.map(i => i.price)
  .reduce((a,v) => a+=v) / f.length
  }
```

e)
```
const calculateAvg = (p) => {
  const f = p.filter(i => i.price > 10)
  return f.avg(i => i.price)
  }
```

**Questão 21** *Valor da questão: 0,33*

Considere o seguinte código Java:

```java
class Box
{
    double largura;
    double altura;
    double profundidade;

    Box(){
        this.largura   = 0;
        this.altura = 0;
        this.profundidade = 0;
    }

    Box(Box b){
        this.largura = b.largura;
        this.altura = b.altura;
        this.profundidade = b.profundidade;
    }
}

public class Teste
{
    public static void main(String[] args)
    {
        Box b1 = new Box();

        Box b2 = b1;

        Box b3 = new Box(b2);

        b1.largura = 10;
        b2.altura = 20;
        b3.profundidade = 5;

        System.out.println(b1.altura);
        System.out.println(b2.profundidade);
        System.out.println(b3.largura );

        b1.altura = 20;

        System.out.println(b1.altura);
        System.out.println(b2.altura);
        System.out.println(b3.altura);
    }

}
```

Assinale a alternativa que representa a saída do código:

a)
0.0
0.0
10.0
20.0
20.0
0.0

b)
10.0
5.0
20.0
20.0
20.0
20.0

c)
20.0
0.0
0.0
20.0
20.0
0.0

d)
0.0
0.0
0.0
20.0
0.0
0.0

e)
20.0
5.0
0.0
20.0
0.0
0.0

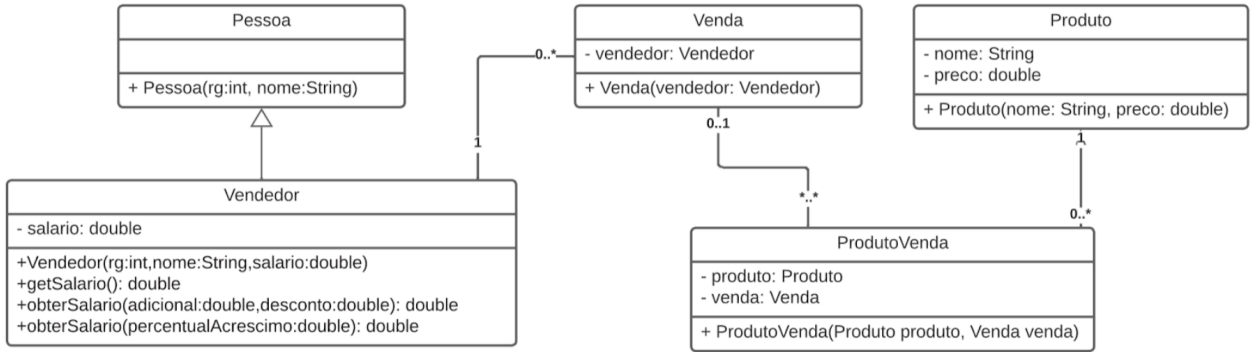
**Questão 22** *Valor da questão: 0,33*

Se um sistema encontra-se com a documentação desatualizada, é possível realizar uma engenharia reversa que permite mapear o código para um diagrama UML. Neste contexto, considere as seguintes classes Java no código abaixo:

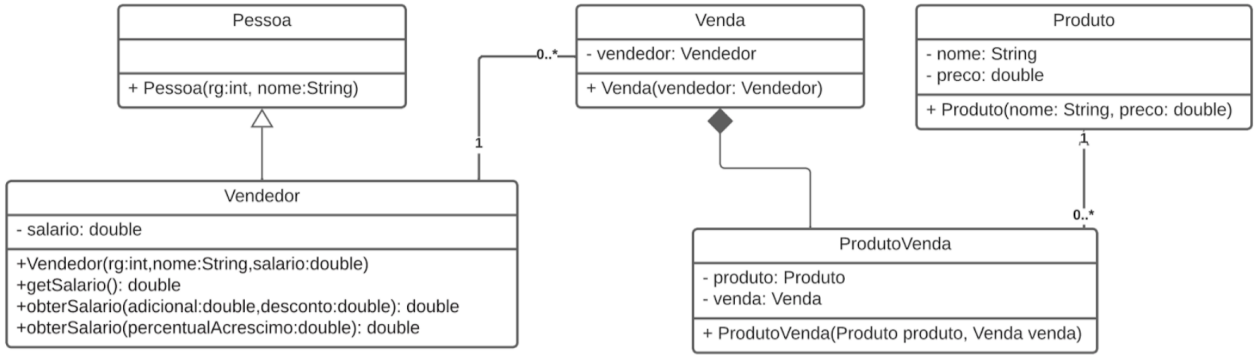
```java
public class Vendedor extends Pessoa {
    private double salario;

    public Vendedor(int rg, String nome, double salario) {
        super(rg, nome);
        this.salario= salario;
    }

    public double getSalario() {
        return salario;
    }

    public double obterSalario(double percentualAcrescimo) {
        double salarioReajustado = salario + salario*percentualAcrescimo / 100;
        return salarioReajustado;
    }

    public double obterSalario(double adicional, double desconto){
        return this.getSalario() + adicional - desconto;
    }
}

public class Venda {
    private Vendedor vendedor;

    public Venda(Vendedor vendedor) {
        this.vendedor = vendedor;
    }
}

public class Produto {
    private String nome;
    private double preco;

    public Produto(String nome, double preco) {
        this.nome = nome;
        this.preco = preco;
    }
}

public class ProdutoVenda {
    private Produto produto;
    private Venda venda;

    public ProdutoVenda(Produto produto, Venda venda) {
        this.produto = produto;
        this.venda = venda;
    }
}
```

Utilizando a engenharia reversa, o diagrama UML de classes correspondente é:

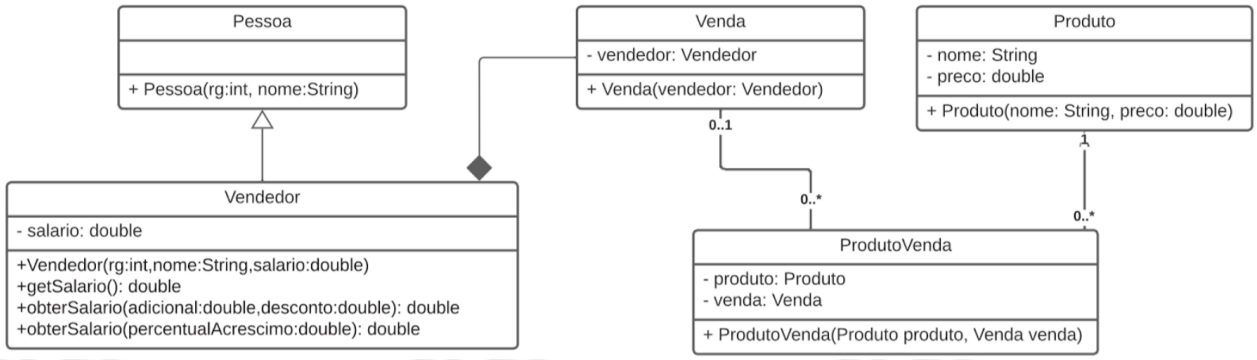
a)


Pessoa
+ Pessoa(rg:int, nome:String)
Venda
- vendedor: Vendedor
+ Venda(vendedor: Vendedor)
Produto
- nome: String
- preco: double
+ Produto(nome: String, preco: double)
Vendedor
- salario: double
+Vendedor(rg:int,nome:String,salario:double)
+getSalario(): double
+obterSalario(adicional:double,desconto:double): double
+obterSalario(percentualAcrescimo:double): double
ProdutoVenda
- produto: Produto
- venda: Venda
+ ProdutoVenda(Produto produto, Venda venda)
0..*
1
0..1
*..*

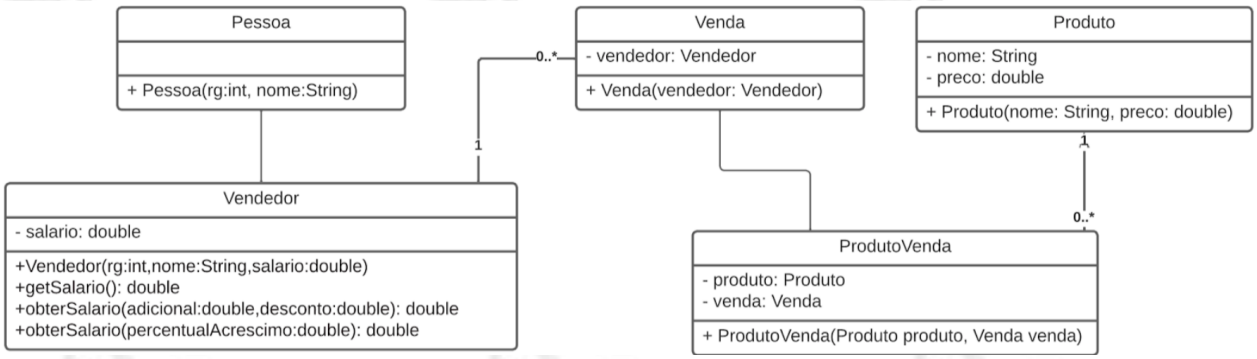
b)


Pessoa
+ Pessoa(rg:int, nome:String)
Venda
- vendedor: Vendedor
+ Venda(vendedor: Vendedor)
Produto
- nome: String
- preco: double
+ Produto(nome: String, preco: double)
Vendedor
- salario: double
+Vendedor(rg:int,nome:String,salario:double)
+getSalario(): double
+obterSalario(adicional:double,desconto:double): double
+obterSalario(percentualAcrescimo:double): double
ProdutoVenda
- produto: Produto
- venda: Venda
+ ProdutoVenda(Produto produto, Venda venda)
0..*
1

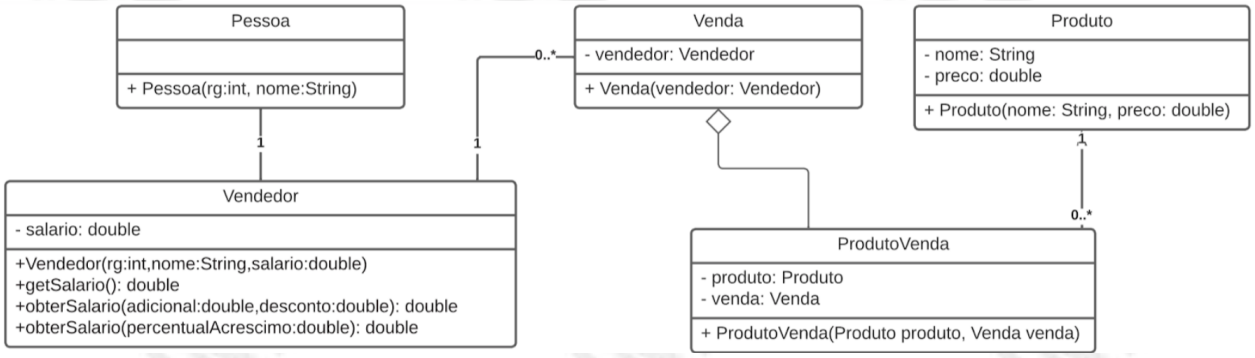
c)


Pessoa
+ Pessoa(rg:int, nome:String)
Venda
- vendedor: Vendedor
+ Venda(vendedor: Vendedor)
Produto
- nome: String
- preco: double
+ Produto(nome: String, preco: double)
Vendedor
- salario: double
+Vendedor(rg:int,nome:String,salario:double)
+getSalario(): double
+obterSalario(adicional:double,desconto:double): double
+obterSalario(percentualAcrescimo:double): double
ProdutoVenda
- produto: Produto
- venda: Venda
+ ProdutoVenda(Produto produto, Venda venda)
0..1
0..*
1

d)


Pessoa
+ Pessoa(rg:int, nome:String)
Venda
- vendedor: Vendedor
+ Venda(vendedor: Vendedor)
Produto
- nome: String
- preco: double
+ Produto(nome: String, preco: double)
Vendedor
- salario: double
+Vendedor(rg:int,nome:String,salario:double)
+getSalario(): double
+obterSalario(adicional:double,desconto:double): double
+obterSalario(percentualAcrescimo:double): double
ProdutoVenda
- produto: Produto
- venda: Venda
+ ProdutoVenda(Produto produto, Venda venda)
0..*
1

e)


Pessoa
+ Pessoa(rg:int, nome:String)
Venda
- vendedor: Vendedor
+ Venda(vendedor: Vendedor)
Produto
- nome: String
- preco: double
+ Produto(nome: String, preco: double)
Vendedor
- salario: double
+Vendedor(rg:int,nome:String,salario:double)
+getSalario(): double
+obterSalario(adicional:double,desconto:double): double
+obterSalario(percentualAcrescimo:double): double
ProdutoVenda
- produto: Produto
- venda: Venda
+ ProdutoVenda(Produto produto, Venda venda)
0..*
1

**Questão 23** *Valor da questão: 0,33*

Sobre tratamento de erro no JavaScript analise o código abaixo e assinale a alternativa correta.

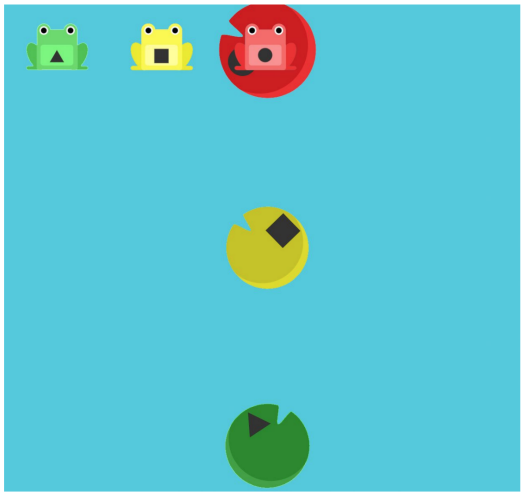
```
try{
doSomething();
next();
}catch (err) {
handleError();
}finally {
handleFinally();
}
```

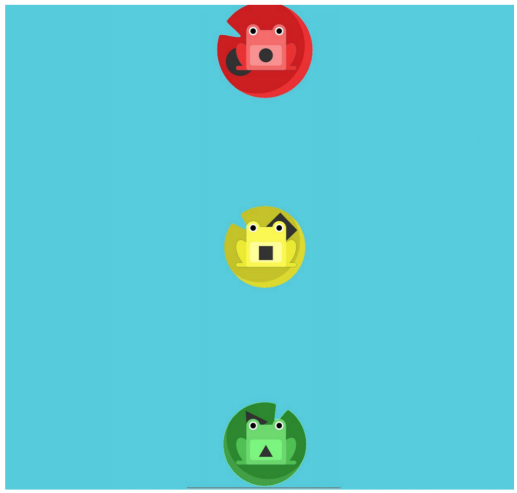
a) Caso haja um erro na execução da função doSomething(), a função handleFinally não será chamada.
b) Caso haja um erro na execução da função doSomething(), a função next() não é chamada. Nesse caso, a função handleError é chamada após a falha.
c) Caso haja um erro na execução da função doSomething(), a função next() não é chamada nem a função handleFinally.
d) Caso haja um erro na execução da função doSomething(), a função next() é chamada em seguida e depois a função handleFinally é chamada normalmente.
e) Caso haja um erro na execução da função doSomething(), a função next() é chamada após a execução da função handleError()
Nenhuma das alternativas anteriores

**Questão 24** *Valor da questão: 0,33*

Imagine que você esteja jogando o jogo conhecido como *FlexboxFroggy* (muito interessante para aprender a usar o Flexbox do CSS!), e você precisa colocar os sapos da posição inicial (ilustrado a esquerda) para a posição final (ilustrado a direita). Quais regras do Flexbox você adicionaria ao seletor CSS pond abaixo?

```
#pond {
display: flex;


}
```





*Site do FlexboxFroggy: flexboxfroggy.com*

a)
```
#pond {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column-reverse;
  }
```

b)
```
#pond {
  display: flex;
  justify-content: space-around;
  align-items: flex-start;
  flex-direction: column-reverse;
  }
```

c)
```
#pond {
  display: flex;
  justify-content: space-between;
  align-items: center;
  flex-direction: column-reverse;
  }
```

d)
```
#pond {
  display: flex;
  justify-content: space-around;
  align-items: center;
  flex-direction: column;
  }
```

e)
```
#pond {
  display: flex;
  justify-content: space-between;
  align-items: stretch;
  flex-direction: column;
  }
```

**Questão 25** *Valor da questão: 0,33*

Considere o seguinte código Java:

```java
public class Exemplo1 {
    public static void main(String[] args)
    {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                if (i != j) {
                    continue;
                } else if(j > i){
                    break;
                }
                System.out.println(i + " * " + j);
            }
        }
    }
}
```

Assinale a alternativa que representa a saída do código acima:

a)
1 * 1
1 * 2
1 * 3
2 * 1
2 * 2
2 * 3
3 * 1
3 * 2
3 * 3

b)
1 * 2
1 * 3
2 * 1
2 * 3
3 * 1
3 * 2

c)
1 * 1
2 * 1
2 * 2
3 * 1
3 * 2
3 * 3

d)
1 * 1
2 * 2
3 * 3

e)
1 * 1
1 * 2
1 * 3
2 * 2
2 * 3
3 * 3

**Questão 26** *Valor da questão: 0,33*

No desenvolvimento de aplicativos móveis nativos da plataforma Android, há vários componentes visuais que exibem listas de informações sobre um determinado objeto, como, por exemplo, os ListView. Para preencher esse tipo de componente, utilizamos um Adapter. Sobre a interação dos componentes ListView e Adapter customizados, julgue as assertivas a seguir:

I. Um lista personalizada, com mais de um widget.View, pode ser implementada por adapter customizado. Tal adapter é representado por uma classe que implemente a interface BaseAdapter, sobreescrevendo seus métodos de manipulação das informações.

II. Mediante solicitação, o Adapter informa ao componente ListView quantas linhas ele terá que exibir. Assim, para cada objeto da Lista, o ListView também solicita sua representação visual ao Adapter, que retornará uma View com as informações do objeto.

III. Um atributo da classe Context pode ser utilizado no Adapter customizado para que tenhamos acesso a arquivos de layout e recursos da aplicação.

IV. O método getView, de um BaseAdapter, é capaz de obter um objeto pela posição do mesmo na lista. Assim, pode-se preencher os componentes com os atributos do objeto e inflar o layout correspondente.
Assinale a alternativa que indica as assertivas CORRETAS:

a) I, II
b) II, III
c) I, IV
d) II, IV
e) III, IV

**Questão 27** *Valor da questão: 0,33*

Sobre os recursos do Spring Framework, considere as assertivas:

I. O recurso O Spring MVC é o núcleo do Spring Framework. Ele cria os objetos, chamados de Beans, configura e monta suas dependências, gerencia todo o seu ciclo de vida.
II. O Spring IoC geralmente se refere diretamente a um contêiner principal que usa o padrão DI/DC para fornecer implicitamente uma referência de objeto em uma classe durante o tempo de execução. O contêiner IoC contém código que trata do gerenciamento de configuração de objetos de aplicativo.

III. O Spring Boot Starter Web é usado para construir aplicativos RESTful usando Spring MVC e está ligado, de maneira intrínseca, à utilização serviços da web por parte da aplicação. Além disso, ele configura automaticamente o Dispatcher Servlet, Error Page, Embedded servlet container e Web JARs para gerenciar as dependências estáticas para desenvolvimento web.

IV. Spring Boot Starter Tomcat é o contêiner incorporado padrão para Spring Boot Starter Web. Atua como servidor web embarcado, mas podemos excluí-lo quando quisermos usar outro contêiner incorporado.

V. No Spring Data JPA, a interface principal de abstração de um repositório é Repository. Repository atua, principalmente, como uma interface de marcador para capturar os tipos com os quais trabalhar. Para essa atuação, tal interface precisa ser informada apenas da classe de domínio a se gerenciar.

Assinale a alternativa que indica as assertivas INCORRETAS

a) IV, V
b) I, V
c) I, III
d) II, IV
e) II, III

**Questão 28** *Valor da questão: 0,33*

Considere o seguinte código HTML e assinale a alternativa que contém seletor CSS que seleciona apenas os elementos <h2> precedidos imediatamente por elemento <p>.

```
<div id="container">
<p>Algum parágrafo</p>
<h2>Selecionar aqui</h2>
<section>
<p>Algum parágrafo</p>
<div>
<h2>Aqui não</h2>
</div>
</section>
</div>
```

a) p - h2
b) p h2
c) p ~ h2
d) p + h2
e) p > h2

**Questão 29** *Valor da questão: 0,33*

Considere os trechos de código a seguir:

I - Código do Controller de uma Aplicação Java Spring Web

```java
import javax.servlet.http.HttpServletRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class OlaMundoController {

    @RequestMapping("/mostraFormulario")
    public String mostraFormulario() {
        return "olamundo-form";
    }

    @RequestMapping("/processForm")
    public String processaFormulario() {
        return "olamundo";
    }

    @RequestMapping("/processaFormulario")
    public String processaFormulario(
            @RequestParam("nome") String nome,
            Model modelo) {

        nome = nome.toUpperCase();

        String mensagem = "Olá, " + nome;

        modelo.addAttribute("message", mensagem);

        return "olamundo";
    }
}

```

II - Códigos HTML de View da Aplicação Java Spring Web: **olamundo-form**

```html
<!--olamundo-form-->
<!DOCTYPE html>
<html>
<head>
    <title>Olá Mundo - Formulário de Entrada</title>
</head>
<body>
    <form action="processaFormulariio" method="GET">
        <input type="text" name="nome"
            placeholder="Qual o seu nome?" />
        <input type="submit" />
    </form>
</body>
</html>
```

III - Códigos HTML de View da Aplicação Java Spring Web: **olamundo**

```
<!-- olamundo -->
<!DOCTYPE html>
<html>
    <body>
    Olá Mundo do Spring!
    Nome: ${param.nome}
    Mensagem: ${mensagem}
    </body>
</html>
```

Assinale a alternativa que representa a saída da View **olamundo**, quando o usuário submeter o formulário **olamundo-form**, com o nome João no o campo de entrada.

a)
Olá Mundo do Spring!
Nome: João
Mensagem: null

b)
Olá Mundo do Spring!
Nome: null
Mensagem: Olá, João

c)
Olá Mundo do Spring!
Nome: null
Mensagem: null

d)
Olá Mundo do Spring!
Nome: João
Mensagem: Olá,

e)
Olá Mundo do Spring!
Nome: João
Mensagem: Olá, João

**Questão 30**

*Valor da questão: 0,33*

Considere o seguinte código Java

```java
interface Saudacoes {
    void saudar();
}
class Pai implements Saudacoes{
    int a = 10;
    public void print()
    {
        System.out.println("Dentro da classe pai");
    }
    @Override
    public void saudar() {
        System.out.println("Saudações da classe pai!");
    }
}
class Filho extends Pai {

    int a = 20;
    public void print()
    {
        System.out.println("Dentro da classe filho!");
    }

    @Override
    public void saudar() {
        System.out.println("Saudações da classe filho!");
    }


}

public class Application {

    public static void main(String[] args)
    {
        Pai obj = new Filho();

        obj.print();

        System.out.println(obj.a);

        obj.saudar();

        Filho obj2 = new Filho();

        System.out.println(obj2.a);

        obj2.saudar();


    }
}
```

Assinale a alternativa que representa a saída quando o código é executado:

a)
Dentro da classe Pai!

20

Saudações da classe filho!

20

Saudações da classe filho!

b)
Dentro da classe pai!

10

Saudações da classe filho!

20

Saudações da classe filho!

c)
Dentro da classe filho!
10
Saudações da classe filho!
20
Saudações da classe filho!

d)
Dentro da classe pai!

10

Saudações da classe pai!

20

Saudações da classe filho!

e)
Dentro da classe Pai!

20

Saudações da classe Pai!

20

Saudações da classe filho!

**PROGRAMA DE RESIDÊNCIA EM TECNOLOGIA DA INFORMAÇÃO**
**TRIBUNAL SUPERIOR DO TRABALHO**
EDITAL 001/2022 – Residência em TI (TST)

**PROVA DE CONHECIMENTOS ESPECÍFICOS**
**ÁREA DE CONCENTRAÇÃO 1: DESENVOLVIMENTO DE SOFTWARE**
**05/06/2022**

## GABARITO

| Questão | Alternativa |
|---|---|
| 1 | A |
| 2 | A |
| 3 | A |
| 4 | A |
| 5 | A |
| 6 | A |
| 7 | A |
| 8 | A |
| 9 | A |
| 10 | A |
| 11 | A |
| 12 | A |
| 13 | A |
| 14 | A |
| 15 | A |

| Questão | Alternativa |
|---|---|
| 16 | A |
| 17 | A |
| 18 | A |
| 19 | A |
| 20 | A |
| 21 | A |
| 22 | A |
| 23 | A |
| 24 | A |
| 25 | A |
| 26 | A |
| 27 | A |
| 28 | A |
| 29 | A |
| 30 | A |