**BDQ ARA0075 2024-1**

PROGRAMAÇÃO ORIENTADA A OBJETOS EM JAVA

**Implementação de Tratamento de Exceções em Java – Subtema:** TIPOS DE EXCEÇÕES

**Semana 5**

Todos os tipos de exceção em Java são subclasses da classe Throwable. A primeira separação feita agrupa as exceções em dois subtipos. Marque a opção correta que mostra esses dois tipos.

A [ ] Error e IOException

B [ ] IOException e Exception

C [ ] Exception e AWTError

D [ ] Error e Exception

E [ ] Error e Try

**Gabarito**. Opção D

Explicação:

A classe **Throwable** tem duas subclasses:

* + **Exception** (java.lang.Exception) – É a raiz das classes originárias da classe **Throwable**, onde mostra as situações em que a aplicação pode querer capturar e realizar um tratamento para conseguir realizar o processamento.
  + **Error** (java.lang.Error) – Também é raiz das classes originárias da classe  **Throwable**, indicando as situações em que a aplicação não deve tentar tratar, como ocorrências que não deveriam acontecer.

**Herança e Polimorfismo – Subtema:** HIERARQUIA DE HERANÇA EM JAVA

**Semana 3**

Em relação à Herança em java, quando temos uma classe “Aluno” estendendo uma classe “Pessoa”, podemos dizer que: “Aluno” é uma subclasse de “Pessoa”, formando assim uma relação hierárquica (“Aluno” deriva de “Pessoa”).

A [ ] “Aluno” **não** é uma subclasse de “Pessoa”, formando assim uma relação hierárquica (“Aluno” deriva de “Pessoa”).

B [ ] “Aluno”é uma **super**classe de “Pessoa”, formando assim uma relação hierárquica (“Aluno” deriva de “Pessoa”).

C [ ] “Aluno”é uma subclasse de “Pessoa”, formando assim uma relação hierárquica (“Aluno”**não** deriva de “Pessoa”).

D [ ] “Aluno”é uma subclasse de “Pessoa”, formando assim uma relação hierárquica (“Aluno” deriva de “Pessoa”).

E [ ] “Aluno”é uma subclasse de “**Funcionário**”, formando assim uma relação hierárquica (“Aluno” deriva de “Pessoa”).

**Gabarito**. Opção D

**Explicação**.

A classe “Aluno” estende a classe “Pessoa”, ou seja, “Aluno” é uma subclasse de “Pessoa”, formando assim uma relação hierárquica (“Aluno” deriva de “Pessoa”). Podemos aplicar a ideia introduzida com o conceito de herança para deixar mais claro o que estamos falando: “Aluno” é um tipo de/é uma “Pessoa”.

**Integração com Banco de Dados em Java – Subtema:** O MAPEAMENTO OBJETO RELACIONAL

**Semana 9**

Na Integração com Banco de Dados em Java, usamos a camada front-end e o conjunto de tecnologias back-end. Dentre as opções abaixo, marque a opção correta.

A [ ] front-end, estamos **não** nos referindo à camada de software responsável pelo interfaceamento do sistema, com o uso de uma linguagem de programação.

back-end compreende o conjunto de tecnologias com a finalidade de fornecer recursos específicos, devendo ser acessadas a partir de nosso front-end, embora não façam parte do mesmo ambiente, como os bancos de dados e as mensagerias.

B [ ] front-end, estamos nos referindo à camada de software responsável pelo interfaceamento do sistema, com o uso de uma linguagem de programação.

back-end compreende o conjunto de tecnologias com a finalidade de fornecer recursos específicos, devendo ser acessadas a partir de nosso front-end, embora não façam parte do mesmo ambiente, como os bancos de dados e as mensagerias.

C [ ] front-end, estamos nos referindo à camada de software responsável pelo interfaceamento do sistema, com o uso de uma linguagem de programação.

back-end compreende o conjunto de tecnologias com a finalidade de fornecer recursos específicos, **não** devendo ser acessadas a partir de nosso front-end, embora não façam parte do mesmo ambiente, como os bancos de dados e as mensagerias.

D [ ] front-end, estamos nos referindo à camada de **hardware** responsável pelo interfaceamento do sistema, com o uso de uma linguagem de programação.

back-end compreende o conjunto de tecnologias com a finalidade de fornecer recursos específicos, devendo ser acessadas a partir de nosso front-end, embora não façam parte do mesmo ambiente, como os bancos de dados e as mensagerias.

E [ ] front-end, estamos nos referindo à camada de hardware responsável pelo interfaceamento do sistema, com o uso de uma linguagem de programação.

back-end **não** compreende o conjunto de tecnologias com a finalidade de fornecer recursos específicos, devendo ser acessadas a partir de nosso front-end, embora não façam parte do mesmo ambiente, como os bancos de dados e as mensagerias.

**Gabarito**. Opção B

**Explicação**.

Integração Com Banco de Dados em Java.

Quando falamos de front-end, estamos nos referindo à camada de software responsável pelo interfaceamento do sistema, com o uso de uma linguagem de programação. Aqui, utilizaremos os aplicativos Java como opção de front-end.

Já o back-end compreende o conjunto de tecnologias com a finalidade de fornecer recursos específicos, devendo ser acessadas a partir de nosso front-end, embora não façam parte do mesmo ambiente, como os bancos de dados e as mensagerias. Para nossos exemplos, adotaremos o banco de dados Derby como back-end.

**Programação Paralela em Java : Threads**

**Subtema:** O CONCEITO DE THREADS E SUA IMPORTÂNCIA PARA O PROCESSAMENTO PARALELO

**– Semana 7**

Sabemos que uma thread é uma forma de implementar múltiplos caminhos de execução em uma aplicação. Dentre as opções abaixo, marque a correta.

A [ ] Toda thread **não** possui uma prioridade. A prioridade de uma thread é utilizada pelo escalonador da MVJ para decidir o agendamento de que thread vai utilizar a CPU.

B [ ] Toda thread possui uma prioridade. A prioridade de uma thread **não é** utilizada pelo escalonador da MVJ para decidir o agendamento de que thread vai utilizar a CPU.

C [ ] Toda thread possui uma prioridade. A prioridade de uma thread é utilizada pelo **processador** da MVJ para decidir o agendamento de que thread vai utilizar a CPU.

D [ ] Toda thread possui uma prioridade. A prioridade de uma thread é utilizada pelo escalonador da MVJ para decidir o agendamento de que thread vai utilizar a CPU.

E [ ] Toda thread possui uma prioridade. A prioridade de uma thread é utilizada pelo escalonador da MVJ para **excluir** o agendamento de que thread vai utilizar a CPU.

**Gabarito**. Opção D

**Explicação**.

Toda thread possui uma prioridade. A prioridade de uma thread é utilizada pelo escalonador da MVJ para decidir o agendamento de que thread vai utilizar a CPU. Threads com maior prioridade têm preferência na execução, porém é importante notar que ter preferência não é ter controle total.

**Introdução à Programação Orientada a Objetos em Java – Subtema:** CLASSES E OBJETOS

**Semana 1**

Em relação à Programação Orientada a Objetos, sabe-se que uma classe é uma maneira de se criar objetos que possuem mesmo comportamento e mesma estrutura. Diante disso, marque a opção abaixo que descreve o que é definido na estrutura de uma classe.

A [ ] Os dados, Os métodos e o Mecanismo de instanciação dos objetos.

B [ ] Os dados, A Herança e o Mecanismo de instanciação dos objetos.

C [ ] Os dados, **O Polimorfismo** e o Mecanismo de instanciação dos objetos.

D [ ] Os dados, Os métodos e **o Polimorfismo**.

E [ ] **O Encapsulamento**, Os métodos e o Mecanismo de instanciação dos objetos.

**Gabarito**. Opção A

**Explicação**.

Em POO (Programação Orientada a Objetos), uma classe é uma maneira de se criar objetos que possuem mesmo comportamento e mesma estrutura.

Formalmente falando, uma classe é uma estrutura que define:

Os dados.

Os métodos que operam sobre esses dados e formam o comportamento do objeto.

O mecanismo de instanciação dos objetos.