Agenda

ALPOO

- Revisão Conceitos Sobre Orientação a Objetos

Revisão Conceitos Orientação a Objetos

Construtores

Construtores são métodos especiais que são chamados automaticamente quando instâncias são criadas através da palavra-chave new.

Características de um construtor:

Ter o mesmo nome da classe, podem ter modificadores public, protected e private (certos casos), não possui tipo de retorno nem mesmo void.

Revisão Conceitos Orientação a Objetos

Sobrecarga de métodos

Métodos com nomes iguais mas a assinatura é diferente. A assinatura de um método é composta de seu nome mais os tipos de argumentos que são passados para esse método, independentemente dos nomes

de variáveis usadas na declaração do método. O tipo de retorno não é considerado parte da assinatura.

Revisão Conceitos Orientação a Objetos

A palavra-chave this

Não se pode chamar um construtor diretamente. Java cria, internamente para cada instância, uma "auto-referência", ou seja, uma referência à própria instância. Essa referência é representada pela palavra-chave this.

Para chamar um construtor de dentro do outro, basta usar a palavra-chave this substituindo o nome do construtor.

Existem algumas regras para a chamada de construtores que são:

Somente construtores podem chamar construtores como sub-rotinas, a chamada deve ser a primeira linha de código dentro do corpo do construtor, não são chamados pelos seus nomes, e sim por this, Construtores podem chamar outros métodos, Métodos não podem chamar construtores, nem mesmo com this, Construtores não podem ser chamados recursivamente.

Revisão Conceitos Orientação a Objetos

Campos estáticos em classes

Campos estáticos de uma classe são compartilhados por todas as instâncias dessa classe – em outras palavras, somente um valor será armazenado em um campo estático, e caso esse valor seja modificado por uma das instâncias da classe, a modificação será refletida em todas as outras instâncias dessa classe.

Revisão Conceitos Orientação a Objetos

Métodos estáticos em classes

Métodos estáticos em classes também são declarados com o modificador static, que deve preceder o tipo de retorno do método e que pode ser combinado com modificadores de acesso ao método.

A diferença principal entre métodos estáticos e não estáticos é que métodos estáticos podem ser chamados sem a necessidade de criação de instâncias das classes às quais pertencem.

Revisão Conceitos Orientação a Objetos

Herança

Java oferece outra maneira de reutilizar classes, através do mecanismo de herança, que permite que

criemos uma classe usando outra como base e descrevendo ou implementando as diferenças e adições da classe usada como base, reutilizando os campos e métodos não privados da classe base.

O mecanismo de herança é o mais apropriado para criar relações é-um-tipo-de entre classes.

Revisão Conceitos Orientação a Objetos

A palavra-chave super

As classes derivadas ou subclasses podem ter acesso a métodos das superclasses usando a palavra-chave super.

O acesso a métodos de classes ancestrais é útil para aumentar a reutilização de código:

- Se existem métodos na classe ancestral que podem efetuar parte do processamento necessário, devemos usar o código que já existe em vez de reescrevê-lo.

Revisão Conceitos Orientação a Objetos

Polimorfismo

O mecanismo de herança permite a criação de classes a partir de outras já existentes com relações "é-um-tipo-de", de forma que, a partir de uma classe genérica, classes mais especializadas possam ser criadas.

A relação "é-um-tipo-de" entre classes permite a existência de outra característica fundamental de linguagens de programação orientadas a objetos: polimorfismo. Polimorfismo permite a manipulação de instâncias de classes que herdam de uma mesma classe ancestral de forma unificada.

Revisão Conceitos Orientação a Objetos

Classes abstratas

- Mecanismo de criação de superclasses com declarações mas sem definições de métodos já que estes são declarados como abstratos.
- Métodos abstratos são somente declarados (com seu nome, modificadores, tipo de retorno e lista de argumentos), não tendo um corpo que contenha os comandos da linguagem que este método deva executar.
- Se uma classe contém um método declarado como abstrato, as classes que herdarem desta deverão obrigatoriamente implementar o método abstrato com o nome, modificador, tipo de retorno e argumentos declarados na classe ancestral.
- Métodos abstratos são declarados com o modificador abstract. Se uma classe tiver algum método abstrato, a classe também deverá obrigatoriamente ser declarada com o modificador abstract.
- Uma classe também pode ser declarada abstrata mesmo que nenhum método tenha sido declarado como abstrato.
- Uma classe que herde de uma classe abstrata deverá, obrigatoriamente, implementar todos os métodos declarados como abstratos na classe ancestral, se houver métodos abstratos na classe ancestral. Caso não haja, nenhuma implementação será obrigatória.

Revisão Conceitos Orientação a Objetos

Interfaces

- Classes abstratas podem conter métodos não abstratos que serão herdados e poderão ser utilizados por instâncias de classes herdeiras.
- Se a classe não tiver nenhum método não abstrato, podemos criá-la como uma interface, que segue um modelo de declaração diferente do usado para classes mas tem funcionalidade similar à de classes abstratas.
- Assim como uma classe abstrata, uma interface não pode ser instanciada. Todos os métodos na interface são implicitamente abstract e public e não podem ser declarados com seus corpos.
- Campos, se houver, serão implicitamente considerados static e final devendo, portanto, ser iniciados na sua declaração.
- A diferença essencial entre classes abstratas e interfaces é que uma classe herdeira somente pode herdar de uma única classe (abstrata ou não), enquanto qualquer classe pode implementar várias interfaces simultaneamente.
- Interfaces são, então, um mecanismo simplificado de implementação de herança múltipla em Java, permitindo que mais de uma interface determine os métodos que uma classe herdeira deve implementar.