Conceitos iniciais

● Definição do Paradigma de Orientação a Objetos

O paradigma de orientação a objetos traz um enfoque diferente da programação estruturada, no sentido de adotar formas mais próximas do mecanismo humano para gerenciar a complexidade de um sistema. Nesse paradigma, o mundo real é visto como sendo constituído de objetos autônomos, concorrentes, que interagem entre si, e cada objeto tem seu próprio estado (atributos) e comportamento (métodos), semelhante a seu correspondente no mundo real. (MENDES, 2009, pg. 18).

Como o próprio nome indica, os “atores” principais do paradigma de projetos orientados a objetos são chamados de objetos. Um objeto se origina de uma classe, que é uma especificação tanto dos campos de dados, também chamados de variáveis de instância, que um objeto contém, como dos métodos (operações) que pode executar. Cada classe apresenta para o mundo exterior uma visão concisa e consistente dos objetos que são instâncias dessa classe, sem detalhes desnecessários ou acesso às estruturas internas dos objetos. (GOODRICH; TAMASSIA, 2013, pg. 60).

● Comparações com o paradigma estruturado

Neste modelo de programação, a reutilização é pequena, e a redundância é grande. No paradigma procedural, quando existe alguma modificação, muitas vezes pelo alto acoplamento entre os programas, é necessário alterar vários programas e novamente testar todo o sistema. No modelo estruturado ficamos limitados a limitar os programas necessários e, no máximo, a criar funções que possam ser reutilizadas em outros programas. No paradigma da orientação a objetos temos outras possibilidades, tais como o uso de encapsulamento para oferecer segurança à classe, herança que permita a reutilização de código, polimorfismo e padrões de projeto, todos explorados de forma detalhada neste livro. (MENDES, 2009, pg. 19).

Na Programação estruturada observamos algumas vantagens como um controle mais eficaz quanto ao fluxo de execução do programa e a facilidade de compreender o código quando o mesmo é analisado. Como principal desvantagem temos a facilidade em desenvolver um código confuso caso o desenvolvedor faça o tratamento dos dados juntamente com o restante da execução do programa, além do que o reuso de código se torna um pouco complicado devido a não definição da tarefa. Já na programação orientada a objetos temos como vantagens a reutilização de código e a melhor organização do código do programa. Porém, alguns detalhes tornam essa programação um pouco prejudicada quando comparada a programação estruturada como, por exemplo, o desempenho do código e a confusão na hora de aplicar os conceitos de orientação a objetos. (DEVMEDIA, 2022).

● Definição de:

● Classe

Uma classe define o modelo de um objeto, ou seja, todas as características que o objeto contém foram definidas pela classe. É importante considerar que uma classe não representa nenhum objeto em particular, pois é só um modelo. (MENDES, 2009, pg. 29).

Em java, cada objeto pertence a uma classe. A classe define os métodos para os objetos. Por exemplo, a classe String define os métodos length e toUpperCase [...]. Os métodos formam a interface pública da classe e determinam o que você pode fazer com os objetos dela. Uma classe também define uma implementação privada, que descreve os dados dentro dos seus objetos e as instruções para seus métodos. Esses detalhes permanecem ocultos dos programadores, os quais utilizam objetos e métodos de chamada. (HORSTMANN, 2009, pg. 65).

● Atributo

Pode haver várias instâncias de um objeto. Por exemplo, um objeto chamado estudante pode possuir 60 instâncias, implicando que há 60 estudantes em uma classe. Portanto, cada instância de um objeto pode ter sua própria informação, como seu próprio número, nome, notas, frequência, etc. Nós podemos dizer então que os atributos, por exemplo informação, de um objeto são de propriedade daquela instância do objeto. Consequentemente, atributos também são chamados de estados de um objeto. (Vasappanavara, Vasappanavara e Vasappanavara, 2011, pg. 1-7, tradução nossa).

“Propriedade identificada que um objeto deve manter.” (HORSTMANN, 2009).

● Método

Toda classe Java contém pelo menos um método chamado de método construtor (método com o mesmo nome da classe e sem parâmetros de entrada). Podem existir vários outros construtores e métodos, mas o método construtor sem parâmetros, se não for explicitamente criado pelo programador, será implicitamente criado pelo compilador Java sempre que nenhum outro tiver sido criado. Caso o programador crie um construtor diferente do construtor sem parâmetros, o compilador Java não mais criará esse construtor de forma automática. Neste caso o programador também deverá criar o construtor sem parâmetros explicitamente. (MENDES, 2009, pg. 32).

Um método consiste em uma sequência de instruções que acessam os dados internos. quando você chama o método, não há como saber exatamente quais são essas instruções, mas você sabe o propósito do método. (HORSTMANN, 2009, pg. 63).

● Objeto

Um objeto é uma entidade que você pode manipular no seu programa. Normalmente, você não sabe como o objeto é organizado internamente, mas ele tem um comportamento bem-definido e é isso o que nos importa quando o utilizamos. Você manipula um objeto chamando um ou mais dos seus métodos. (HORSTMANN, 2009, pg. 63).

Os principais “atores” em um programa Java são os objetos. Os objetos armazenam dados e fornecem os métodos para acessar e modificar esses dados. Todo objeto é instância de uma classe que define o tipo de objeto, bem como os tipos de operações que executa. (GOODRICH; TAMASSIA, 2013, pg. 2).

● O que significa instância de um objeto

Um objeto armazena seus dados em campos de instância. Um campo é um termo técnico para uma posição de memória dentro de um bloco de memória. Uma instância de uma classe é um objeto dessa classe. Portanto, um campo de instância é uma posição de memória presente em cada objeto da classe. (HORSTMANN, 2009, pg. 110).

Dados de objetos Java são armazenados em variáveis de instância (também chamadas de campos). Por essa razão, se um objeto de uma classe deve armazenar dados, então sua classe deve especificar variáveis de instância para esse fim. As variáveis de instância podem ser de tipos básicos (tais como inteiros, números de ponto flutuante ou booleanos) ou podem se referir a objetos de outras classes. (GOODRICH; TAMASSIA, 2013, pg. 2).

● Modificadores de acesso

● public

“Qualquer um pode acessar variáveis de instância públicas.” (GOODRICH; TAMASSIA, 2013).

“Os membros declarados public são acessíveis em qualquer lugar que a classe seja acessível”(tradução livre). (BUYYA; SELVI; CHU, 2009).

● private

“Apenas métodos da mesma classe (excluindo métodos de uma subclasse) podem acessar variáveis de instâncias privadas.” (GOODRICH; TAMASSIA, 2013).

Os campos de instância geralmente são declarados com o especificador de acesso private. Esse especificador significa que os campos só podem ser acessados pelo métodos da mesma classe e por nenhum outro método. (HORSTMANN, 2009, pg. 111).

● protected

“Apenas métodos do mesmo pacote ou subclasse podem acessar variáveis de instância protegidas.” (GOODRICH; TAMASSIA, 2013).

“O controle de acesso dos membros protected é similar ao dos membros private e possui mais significância na herança. Contudo, os membros da classe são acessíveis fora da classe dentro do mesmo pacote”(tradução livre). (BUYYA; SELVI; CHU, 2009).

Referências

BUYYA, Rajkumar; SELVI, S Thamarai; CHU, Xingchen. **Object-Oriented Programming with Java**: essentials and applications. New Delhi: Tata McGraw Hill, 2009.

DEVMEDIA. **Programação Orientada a Objetos e Programação Estruturada**. Disponível em: https://www.devmedia.com.br/programacao-orientada-a-objetos-e-programacao-estruturada/32813. Acesso em: 09 maio 2022.

GOODRICH, Michael T.; TAMASSIA, Roberto. **Estruturas de Dados & Algoritmos**. 5. ed. Porto Alegre: Bookman, 2013.

HORSTMANN, Cay. **Conceitos de Computação com Java**. Porto Alegre: Bookman, 2009.

MENDES, Douglas Rocha. **Programação Java com ênfase em orientação a objetos**. São Paulo: Novatec, 2009

VASAPPANAVARA, Ramesh; VASAPPANAVARA, Anand; VASAPPANAVARA, Gautam. **Object-oriented Programming Using C++ and Java**. New Delhi: Pearson, 2011.