Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Campus Maracanaú
Coordenadoria de Computação
Curso de Bacharelado em Ciência da Computação
Disciplina: Programação Orientada a Objetos
Professor: Igor Rafael Silva Valente

ATIVIDADE

Assunto:
Polimorfismo.

Orientações:
A atividade deve ser executada individualmente e entregue através do ambiente *Google Classroom*.

Regras de criação dos programas:
Crie um novo projeto Java denominado **AtividadePolimorfismo**. As classes devem possuir os nomes informados no texto. Ao final, o projeto deve ser exportado para um arquivo em formato ZIP.

Nome completo:

Yan Pedro Façanha Brasileiro

1. Explique o que é polimorfismo e em quais situações é útil.
   O polimorfismo permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum, normalmente de uma superclasse ou interface em comum. Um mesmo método pode ter comportamentos distintos dependendo do objeto que o invoca.

   Situações úteis:

   - Reutilização de Código: Código genérico que opera em múltiplos tipos.
   - Extensibilidade: Adicionar novas funcionalidades sem alterar código existente..
   - Redução da Complexidade: Elimina `if/else if` ou `switch` para cada tipo de objeto.
   - Interface Unificada: Diferentes classes se apresentam por uma interface comum.

2. O polimorfismo possui desvantagens? Explique.
   Sim;
   Dificuldade na Leitura e Rastreamento: Em sistemas complexos, pode ser difícil rastrear qual implementação de método será executada.
   Complexidade Excessiva: Uso desnecessário pode levar a um design complexo e difícil de manter.
   Depuração Mais Complexa: O fluxo de execução pode saltar entre classes, tornando a depuração mais desafiadora.

3. O que é amarração tardia? Explique.
   Amarração tardia é o mecanismo pelo qual a decisão de qual método específico será executado é adiada para o tempo de execução do programa, e não em tempo de compilação.

   Isso significa que, quando um método polimórfico é chamado em uma variável de superclasse que referencia uma subclasse, a Java Virtual Machine determina dinamicamente qual implementação do método (da superclasse ou da subclasse) será invocada com base no tipo real do objeto. É fundamental para o polimorfismo de sobrescrita.

4. Para demonstrar o uso do polimorfismo, siga os passos a seguir:

   a. Crie um projeto no Eclipse denominado AtividadePolimorfismo.
   b. Crie a classe Produto com os atributos privados: nome, descrição e preço. Esta classe deve possuir os métodos get e set para cada atributo, bem como um construtor que receba todos

os argumentos. Crie também os métodos equals e toString. O método toString deve imprimir uma frase parecida com a seguir: “Produto: <nome>, <descrição>, <preço>”. Não deve existir construtor padrão.

c. Crie a classe Livro, que herda de Produto, adicionando os atributos privados: autores, ISBN e editora. Esta classe deve possuir os métodos get e set para cada atributo da classe Livro, bem como um construtor que receba todos os argumentos (incluindo os da classe Produto, que devem ser repassados para a classe pai no construtor). Os métodos equals e toString devem ser sobrescritos. O método toString deve imprimir uma frase parecida com a seguir: “Livro: <nome>, <descrição>, <preço>, <autores>, <ISBN>, <editora>”.
d. Crie a classe Principal, cujo objetivo é realizar a interação com o usuário. Esta classe deve possuir um vetor (ou ArrayList) da classe Produto. Você deve criar um menu em modo texto em que o usuário pode selecionar entre as opções:
    i. Cadastrar produto
    ii. Listar produtos (deve utilizar o toString para imprimir)
    iii. Cadastrar livro
    iv. Listar livros (deve utilizar o toString para imprimir)
    v. Imprimir tudo (deve utilizar o toString para imprimir)
    vi. Sair
e. Note que apesar do vetor/lista ser do tipo Produto, você conseguirá adicionar objetos do tipo Livro. Adicionalmente, quando chamar o método toString() dos objetos, algumas vezes será chamado o método da classe Produto e outras vezes o método da classe Livro, quando o objeto armazenado tiver sido um livro. Você economizou código? A estrutura do seu código ficou mais simples? Isto é possível graças ao polimorfismo. Quais suas conclusões sobre o polimorfismo?

Conclusões sobre o Polimorfismo:

Economia de Código e Simplicidade: A `ArrayList<Produto>` armazena tanto `Produto` quanto `Livro`. O loop que chama `p.toString()` não precisa de `if/else` para determinar o tipo, pois o Java chama a implementação correta automaticamente.

Extensibilidade: Adicionar novos tipos de produtos, ex: `Eletronico`, não exigiria modificação no método `imprimirTudo()`, apenas a criação de uma nova classe que herde de `Produto` e sobrescreva `toString()`.

Manutenibilidade: A lógica de impressão é encapsulada em cada classe (`Produto` ou `Livro`), facilitando alterações futuras sem afetar outras partes do sistema.

Boa sorte! Prof. Igor.