**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: Francisco Aldenor Silva Neto**

1. Explique o que é polimorfismo e em quais situações é útil.

é um conceito importante em programação orientada a objetos, que se refere à capacidade de um objeto de assumir diferentes formas ou comportamentos. Em outras palavras, um objeto pode ser usado de diferentes maneiras, podemos instanciar um objeto seja interface ou classe em um atributo de um método e com isso conseguimos passar como parâmetro desse método qualquer objeto que implemente ou herde do objeto que foi instanciado.

1. O polimorfismo possui desvantagens? Explique.

Sim podemos listar como exemplo:

1. Complexidade: O uso excessivo de polimorfismo pode tornar o código mais complexo e difícil de entender. Isso pode ser especialmente verdadeiro em sistemas grandes e complexos, onde há muitas classes e métodos polimórficos.

2. Desempenho: O polimorfismo pode ter um impacto negativo no desempenho do sistema, especialmente quando métodos polimórficos são chamados muitas vezes em um loop. Isso ocorre porque o compilador precisa determinar dinamicamente qual método deve ser chamado em tempo de execução, o que pode ser mais lento do que a chamada direta de um método.

3. Dificuldade de depuração: Quando ocorrem erros em código polimórfico, pode ser mais difícil de depurar porque o objeto pode ter diferentes formas, dependendo do contexto em que é usado.

4. Dificuldade de manutenção: O polimorfismo pode tornar o código mais difícil de manter, especialmente se as subclasses são adicionadas ou alteradas com frequência. Isso ocorre porque as alterações em uma classe podem afetar outras classes que dependem dela.

1. O que é amarração tardia? Explique.

é um conceito da programação orientada a objetos em que a decisão sobre qual método deve ser chamado é tomada em tempo de execução em vez de tempo de compilação. Isso significa que a escolha do método a ser executado é adiada até que o programa esteja sendo executado.

1. Para demonstrar o uso do polimorfismo, siga os passos a seguir:
   1. Crie um projeto no Eclipse denominado AtividadePolimorfismo.
   2. 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.
   3. 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>”.
   4. 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:
      1. Cadastrar produto
      2. Listar produtos (deve utilizar o toString para imprimir)
      3. Cadastrar livro
      4. Listar livros (deve utilizar o toString para imprimir)
      5. Imprimir tudo (deve utilizar o toString para imprimir)
      6. Sair
   5. 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?

Boa sorte! Prof. Igor.