UNIP – UNIVERSIDADE PAULISTA
Ribeirão Preto - 2023
CIÊNCIA DA COMPUTAÇÃO - 3º Semestre
CC3P18

APS – ‘Monte o Seu Computador’

Evando Dos Reis Barbosa Filho – G4757D2
Igor Davi De Souza – N862726
João Lucas Roldão Rocha – N9416H4
Luccas Zanatta - G449EF3

Link para o vídeo:
https://drive.google.com/file/d/1Bxqa9zuI6cd0NxWKE3Q1kwuraqw7DSJh/view?usp=sharing
Link para o trabalho:

Índice:

1 Objetivo do trabalho

O objetivo deste trabalho é proporcionar uma experiência prática no desenvolvimento de um programa orientado a objetos, utilizando o conceito de polimorfismo, por meio da criação de código intitulado "Monte seu PC". Neste projeto, nós fomos desafiados a construir um programa capaz de simular a montagem de um computador personalizado, permitindo ao usuário escolher os componentes.

2 Introdução

2.1 Conceitos de Programação Orientada a Objetos:

A programação orientada a objetos (POO) é um paradigma de programação que organiza o código em torno de objetos, que são entidades que representam algo concreto ou abstrato do mundo real. Esses objetos possuem características (atributos) e comportamentos (métodos) que podem interagir entre si.

A POO tem como principal objetivo modelar o mundo real de forma mais próxima, permitindo a criação de sistemas mais modulares, flexíveis e reutilizáveis. Ela se baseia em quatro pilares fundamentais: encapsulamento, herança, polimorfismo e abstração.

O encapsulamento é um princípio que permite ocultar os detalhes internos de um objeto, fornecendo uma interface controlada para interagir com ele. Isso garante a integridade dos dados e promove a segurança e a manutenção do código.

A herança é um mecanismo que permite a criação de novas classes a partir de classes existentes, aproveitando as características e comportamentos já implementados. A classe derivada (subclasse) herda os atributos e métodos da classe base (superclasse), permitindo a reutilização de código e a criação de hierarquias de classes.

O polimorfismo permite que um objeto seja tratado de diferentes formas, dependendo do contexto em que é utilizado. Isso significa que objetos de classes diferentes podem responder a uma mesma mensagem de maneiras distintas. O polimorfismo aumenta a flexibilidade e a extensibilidade do código, permitindo lidar com diferentes tipos de objetos de forma genérica.

A abstração é o processo de identificar as características essenciais de um objeto e representá-las em uma classe. Ela permite abstrair os detalhes complexos e irrelevantes, focando apenas no que é necessário para o funcionamento do sistema. A abstração facilita a compreensão e a manutenção do código, tornando-o mais legível e conciso.

A programação orientada a objetos utiliza conceitos como classes, objetos, herança, polimorfismo, encapsulamento, abstração, entre outros, para criar estruturas de

código mais organizadas e modularizadas. Ela é amplamente utilizada em diversas linguagens de programação, como Java, C++, Python e C#, e é especialmente adequada para o desenvolvimento de sistemas complexos e de grande escala.

Em resumo, a programação orientada a objetos é um paradigma que organiza o código em torno de objetos, que possuem características e comportamentos. Ela permite a criação de sistemas mais modulares, flexíveis e reutilizáveis, por meio dos pilares do encapsulamento, herança, polimorfismo e abstração. A POO é amplamente adotada na indústria de desenvolvimento de software e oferece diversas vantagens, como a reutilização de código, a manutenção facilitada e a modelagem mais próxima do mundo real.

2.2 Computadores personalizados

No contexto atual, a tecnologia desempenha um papel central em nossas vidas, e o computador se tornou uma ferramenta indispensável em diversas atividades cotidianas. No entanto, muitos usuários se sentem limitados pelas opções pré-configuradas disponíveis no mercado. O trabalho "Monte o Seu Computador" tem como objetivo capacitar e empoderar os usuários, fornecendo um guia abrangente para a montagem personalizada de um sistema tecnológico, permitindo que cada indivíduo tenha um computador adaptado às suas necessidades e preferências.

A montagem de computadores é uma atividade que tem conquistado cada vez mais adeptos no mundo da tecnologia. Mais do que apenas uma tarefa mecânica, a montagem de um computador envolve conhecimento técnico, habilidades práticas e uma dose de criatividade. Nesta introdução, exploraremos o processo de montagem de computadores, destacando seu objetivo, benefícios e o impacto que essa habilidade pode ter tanto para usuários domésticos quanto para profissionais da área.

A montagem de computadores exige um conhecimento técnico sólido dos diferentes componentes que compõem um sistema. Desde a escolha da placa-mãe até a seleção da fonte de alimentação, cada peça desempenha um papel crucial no funcionamento do computador. Ao se aventurar nesse processo, os montadores adquirem uma compreensão profunda da arquitetura interna dos sistemas e das interações entre os componentes.

Uma das grandes vantagens da montagem de computadores é a possibilidade de personalização. Cada indivíduo possui necessidades e preferências específicas,

seja para jogos, trabalho, edição de mídia ou outras atividades. Ao montar seu próprio computador, os usuários têm a liberdade de escolher cuidadosamente cada componente, garantindo um sistema que atenda perfeitamente às suas exigências e ao orçamento disponível.

A montagem personalizada de um computador permite a otimização do desempenho. Ao selecionar componentes de alta qualidade e compatíveis entre si, é possível obter um sistema mais rápido, eficiente e estável. Além disso, a capacidade de realizar atualizações graduais ao longo do tempo é uma vantagem significativa. Os montadores podem substituir componentes obsoletos, aumentar a capacidade de armazenamento, atualizar a placa de vídeo e adicionar recursos extras de acordo com suas necessidades em constante evolução.

A montagem de computadores não se limita apenas ao aspecto teórico. Ela envolve habilidades práticas, como encaixar corretamente os componentes, conectar cabos e configurar o sistema operacional. Essa experiência prática não só desenvolve habilidades técnicas, mas também capacita os montadores a solucionar problemas e lidar com situações imprevistas. A capacidade de diagnosticar e corrigir falhas ou incompatibilidades é uma valiosa competência para qualquer entusiasta de tecnologia.

A montagem de computadores também pode proporcionar benefícios financeiros e ambientais. Ao montar um computador personalizado, os usuários podem evitar custos extras associados à compra de sistemas pré-configurados com recursos desnecessários. Além disso, a escolha consciente de componentes de qualidade contribui para a durabilidade do sistema, reduzindo a necessidade de substituição prematura. Isso resulta em um menor impacto ambiental, promovendo a sustentabilidade.

A montagem de computadores é uma atividade enriquecedora que combina conhecimento técnico, habilidades práticas e personalização. Ao dominar essa habilidade, os usuários têm a oportunidade de criar sistemas que atendam perfeitamente às suas necessidades, além de desenvolver competências técnicas valiosas e a capacidade de solucionar problemas. A montagem de computadores também oferece benefícios financeiros e ambientais, além de proporcionar uma experiência gratificante. Portanto, encoraja-se a todos os entusiastas de tecnologia a se aventurarem nesse processo, explorando a arte de montar seus próprios computadores.

2.3 Qual é a importância de um aplicativo que permita o usuário montar um computador personalizado?

A importância de um aplicativo que permita ao usuário montar um computador personalizado é bastante significativa nos dias de hoje. Com a rápida evolução da tecnologia e a diversidade de componentes disponíveis no mercado, os usuários têm cada vez mais opções para personalizar e otimizar suas máquinas de acordo com suas necessidades e preferências específicas.

Um aplicativo desse tipo oferece uma série de benefícios tanto para usuários leigos quanto para entusiastas de hardware. Vejamos algumas das principais vantagens:

Personalização: Um aplicativo de montagem de computadores personalizados permite que os usuários escolham os componentes específicos que desejam usar em seu sistema, como processador, placa-mãe, memória RAM, placa de vídeo, armazenamento, fonte de alimentação, entre outros. Isso possibilita a construção de um computador sob medida, adequado às necessidades e orçamento de cada pessoa.

Desempenho otimizado: Ao montar um computador personalizado, os usuários podem selecionar os componentes de hardware que atendam às suas exigências de desempenho. Seja para jogos, edição de vídeo, programação ou qualquer outra finalidade específica, a escolha cuidadosa dos componentes pode resultar em um sistema mais poderoso e eficiente.

Compatibilidade garantida: Um aplicativo de montagem de computadores personalizados geralmente fornece informações sobre a compatibilidade entre os diferentes componentes. Isso evita a escolha de peças incompatíveis e garante que todos os elementos do computador funcionem corretamente em conjunto. Dessa forma, os usuários evitam dores de cabeça e desperdício de recursos ao adquirir componentes que não são adequados para o seu sistema.

Educação e aprendizado: Esse tipo de aplicativo também pode ser uma excelente ferramenta educacional. Ele permite que os usuários conheçam melhor os componentes de um computador, entendam suas funções e aprendam sobre as melhores práticas de montagem. Isso contribui para a disseminação do conhecimento técnico e promove a autonomia dos usuários em relação à configuração e manutenção de seus sistemas.

Economia de custos: Montar um computador personalizado pode ser uma alternativa mais econômica em comparação à compra de um computador pré-montado. Os usuários têm a liberdade de escolher componentes com melhor custo-benefício e evitar pagar por recursos ou especificações que não são necessários

para suas necessidades específicas. Além disso, a possibilidade de atualização e substituição individual de componentes permite uma maior vida útil e redução de custos a longo prazo.

Satisfação e senso de conquista: Montar seu próprio computador personalizado é uma experiência gratificante para muitos usuários. A sensação de realizar um projeto personalizado, ver o sistema funcionando corretamente e alcançar um desempenho de acordo com as expectativas traz uma sensação de satisfação e realização. Isso também promove um maior envolvimento e interesse pelo mundo da tecnologia.

Em suma, um aplicativo que permite ao usuário montar um computador personalizado oferece uma série de benefícios, como personalização, desempenho otimizado, compatibilidade garantida, educação, economia de custos e satisfação pessoal. Com a crescente demanda por computadores personalizados e a diversidade de opções disponíveis no mercado, essa ferramenta se torna cada vez mais relevante para os usuários que desejam ter controle total sobre sua máquina e obter o máximo de desempenho e personalização possível.

3 Tema escolhido

Escolhemos esse temos pois sabemos a importância da montagem personalizada de computadores. A montagem de um PC sob medida é uma prática cada vez mais relevante no mundo da tecnologia. É uma forma de criar um sistema que atenda às nossas necessidades específicas, tanto em termos de desempenho quanto de funcionalidades.

A montagem personalizada nos permite escolher cuidadosamente cada componente do computador, desde o processador e placa-mãe até a memória RAM, placa de vídeo e armazenamento. Essa liberdade de escolha nos permite otimizar o desempenho do sistema para atender às nossas demandas específicas, seja para jogos, edição de vídeos, desenvolvimento de software ou outras tarefas intensivas de computação.

Além disso, montar nosso próprio PC nos dá a oportunidade de aprender sobre os diferentes componentes do computador e como eles interagem entre si. Isso nos ajuda a desenvolver um conhecimento mais aprofundado sobre hardware e software, além de nos capacitar a solucionar problemas e realizar upgrades futuros com mais facilidade.

Outro ponto relevante é o aspecto econômico. Ao montar nosso próprio computador, podemos buscar componentes de melhor custo-benefício e evitar pagar por recursos que não utilizaremos. Isso nos permite economizar dinheiro e direcionar nossos investimentos para os aspectos mais importantes do sistema.

Por fim, a montagem personalizada de computadores é uma atividade empolgante e gratificante. Poder escolher cada peça, criar um sistema único e vê-lo funcionando perfeitamente é uma experiência satisfatória para qualquer entusiasta da tecnologia.

Nesse contexto, o projeto "Monte seu Computador" se torna uma ferramenta valiosa para aqueles que desejam explorar a montagem personalizada, oferecendo um ambiente interativo e intuitivo para selecionar componentes e visualizar as especificações do sistema resultante.

Como aluno, a relevância de desenvolver o projeto "Monte seu Computador" utilizando programação orientada a objetos foi imensurável. A programação orientada a objetos é uma abordagem poderosa que nos permite organizar e estruturar nosso código de forma mais eficiente e modular. Ao aplicar os conceitos de classes, objetos, herança, polimorfismo e encapsulamento, pudemos criar um sistema flexível e de fácil manutenção.

Durante o desenvolvimento do projeto, aprendemos a identificar as entidades principais envolvidas, como Processador, Placa-mãe, Memória RAM, Placa de Vídeo, entre outros. Essas entidades se tornaram nossas classes, com atributos e métodos específicos que representam as características e comportamentos de cada componente.

Através da herança, pudemos estabelecer uma hierarquia de classes, onde a classe Componente é a superclasse e as classes Processador, Placa-mãe, Memória RAM, Placa de Vídeo, entre outras, são subclasses que herdam seus atributos e métodos. Essa hierarquia nos permitiu reutilizar código e organizar de forma coerente as relações entre os diferentes componentes do computador.

O polimorfismo desempenhou um papel fundamental no projeto. Com ele, pudemos tratar diferentes objetos de forma uniforme, permitindo a adição de componentes ao computador de forma flexível e dinâmica. Por exemplo, utilizando o método adicionar Componente da classe Computador, podemos adicionar um Processador, uma Placa-mãe ou qualquer outro componente sem a necessidade de tratar cada um individualmente. Isso tornou o sistema mais modular, facilitando a expansão e manutenção futuras.

Além disso, o encapsulamento garantiu que os atributos e métodos das classes fossem adequadamente protegidos e acessados somente quando necessário. Isso promoveu a segurança e integridade dos dados em nosso sistema.

Ao aplicar todos esses conceitos, fomos capazes de criar um sistema robusto e escalável para a montagem personalizada de computadores. A programação orientada a objetos nos ajudou a compreender a importância da modelagem correta e da estruturação do código, resultando em um projeto mais organizado e de fácil entendimento.

Em resumo, o desenvolvimento do projeto "Monte seu Computador" utilizando programação orientada a objetos nos proporcionou uma experiência valiosa, permitindo-nos aplicar os conceitos teóricos aprendidos em sala de aula em um projeto real. Foi uma oportunidade enriquecedora para aprimorar nossas habilidades de programação e compreender a importância da abordagem orientada a objetos no desenvolvimento de sistemas mais eficientes e flexíveis.

4. Dissertação:

4.1 Programação Orientada a Objetos:

A programação orientada a objetos é um paradigma de programação amplamente utilizado no código do trabalho "Monte o seu computador". Essa abordagem se baseia na criação de objetos que possuem propriedades e comportamentos, permitindo modelar o sistema de forma mais intuitiva e modular.

No código do trabalho, a programação orientada a objetos é aplicada de maneira abrangente. Cada componente do computador, como processador, memória RAM, placa de vídeo, entre outros, é representado por uma classe específica. Cada classe define as propriedades do componente, como marca, modelo, capacidade, e também os métodos relacionados a ele, como instalação, configuração e verificação de compatibilidade.

A programação orientada a objetos permite uma organização clara e estruturada do código. As classes são criadas com base em conceitos de herança, encapsulamento e polimorfismo. A herança permite que as classes compartilhem características comuns, evitando a repetição de código. Por exemplo, a classe Processador pode herdar atributos e métodos da classe Componente, pois ambos têm características em comum.

O encapsulamento garante que os detalhes internos de cada classe sejam ocultados, fornecendo uma interface clara e definida para o uso das funcionalidades do componente. Isso permite que as interações entre os diferentes componentes sejam controladas e padronizadas, aumentando a segurança e a modularidade do código.

Além disso, a programação orientada a objetos possibilita o uso do polimorfismo, como mencionado anteriormente. O polimorfismo permite tratar objetos de diferentes classes de forma uniforme, por meio de interfaces comuns e métodos com a mesma assinatura. Isso facilita a escrita de código genérico, que pode lidar com diferentes tipos de componentes sem a necessidade de considerar cada caso separadamente.

No trabalho "Monte o seu computador", a programação orientada a objetos é fundamental para criar uma estrutura modular e flexível, que permite a interação entre os diferentes componentes de forma simplificada. Ela também facilita a

manutenção e a extensão do código, uma vez que novos componentes podem ser adicionados com facilidade, e o sistema como um todo é mais organizado e reutilizável.

Em suma, a programação orientada a objetos é amplamente utilizada no código do trabalho "Monte o seu computador" para criar uma estrutura modular e flexível, permitindo a modelagem dos componentes do computador por meio de classes, herança, encapsulamento e polimorfismo. Essa abordagem proporciona uma melhor organização, reutilização de código e facilidade na manutenção e expansão do sistema.

4.2 Herança:

No código do trabalho "Monte o seu computador", a herança é amplamente utilizada como um conceito fundamental da programação orientada a objetos. A herança permite que as classes compartilhem características e comportamentos comuns, evitando a duplicação de código e promovendo a reutilização.

No contexto da montagem de computadores, a herança é aplicada para modelar a relação entre os diferentes componentes. Por exemplo, podemos ter uma classe base chamada "Componente" que contém as propriedades e métodos comuns a todos os componentes, como a marca, o modelo e a capacidade. Essa classe servirá como base para outras classes mais específicas, como "Processador", "Placa de Vídeo" e "Memória RAM".

Ao usar herança, as classes específicas podem herdar as propriedades e métodos da classe base, eliminando a necessidade de reescrever código similar em cada classe individualmente. Por exemplo, a classe "Processador" pode herdar as propriedades da classe "Componente", como marca e modelo, e também adicionar suas próprias propriedades e métodos específicos, como a velocidade de clock e o número de núcleos.

Dessa forma, a herança ajuda a organizar e estruturar o código de forma mais eficiente. Ela permite que as classes compartilhem características em comum, promovendo a reutilização de código e facilitando a manutenção e a extensão do sistema. Além disso, a herança ajuda a estabelecer uma hierarquia entre as classes, permitindo que o código seja mais intuitivo e fácil de entender.

No trabalho "Monte o seu computador", a herança é utilizada para estabelecer a relação entre os diferentes componentes, agrupando-os de acordo com suas características em comum. Essa abordagem permite criar uma estrutura hierárquica, em que classes mais específicas herdam as propriedades e métodos das classes mais genéricas, promovendo a reutilização de código e simplificando o desenvolvimento do sistema.

Em resumo, a herança desempenha um papel essencial no código do trabalho "Monte o seu computador", permitindo que as classes compartilhem características em comum, promovendo a reutilização de código e facilitando a organização e a estruturação do sistema. Ela ajuda a estabelecer uma hierarquia entre as classes, proporcionando uma abordagem mais intuitiva e eficiente para modelar os diferentes componentes do computador.

4.3 Encapsulamento:

No código do trabalho "Monte o seu computador", o encapsulamento é um princípio importante da programação orientada a objetos que é amplamente utilizado para proteger os dados e funcionalidades de uma classe. O encapsulamento ajuda a garantir a integridade dos objetos e promove a segurança e a manutenção do código.

No contexto da montagem de computadores, o encapsulamento é aplicado para controlar o acesso aos atributos e métodos das classes. Isso significa que os detalhes internos de implementação de uma classe são ocultados e só podem ser acessados por meio de interfaces públicas definidas.

As classes no código do trabalho "Monte o seu computador" definem atributos e métodos que encapsulam as informações relacionadas aos componentes do computador. Os atributos são declarados como privados (private), o que significa que eles não podem ser acessados diretamente fora da classe. Em vez disso, métodos públicos (public) são fornecidos para permitir o acesso e a manipulação desses atributos.

Por exemplo, a classe "Componente" pode ter atributos privados como marca, modelo e capacidade. O encapsulamento impede que esses atributos sejam modificados ou acessados diretamente de fora da classe. Em vez disso, métodos públicos como "getMarca()" e "getModelo()" são definidos para fornecer acesso controlado aos valores desses atributos.

O encapsulamento no código do trabalho "Monte o seu computador" oferece várias vantagens. Em primeiro lugar, ele protege os dados da classe, evitando que sejam alterados de forma inadequada ou acidental. Isso ajuda a manter a consistência e a integridade dos objetos.

Além disso, o encapsulamento permite que a implementação interna de uma classe seja modificada sem afetar o código que a utiliza. Os detalhes internos podem ser alterados, como a forma como os atributos são armazenados ou os cálculos realizados nos métodos, sem que os usuários da classe precisem saber ou se preocupar com essas mudanças.

O encapsulamento também contribui para a segurança do código, pois impede que partes externas tenham acesso direto e não autorizado aos dados sensíveis. Somente os métodos públicos definidos na classe permitem interações controladas com os atributos privados.

Em resumo, o encapsulamento desempenha um papel crucial no código do trabalho "Monte o seu computador". Ele protege os dados e funcionalidades das classes, promovendo a integridade, a segurança e a manutenção do código. O encapsulamento é alcançado através da definição de atributos como privados e fornecendo métodos públicos para acessar e manipular esses atributos. Essa abordagem garante que os detalhes internos de implementação de uma classe permaneçam ocultos e que o código seja mais seguro, flexível e fácil de manter.

4.4 Polimorfismo

No código do trabalho "Monte o seu computador", o polimorfismo é uma característica fundamental. O polimorfismo é um conceito da programação orientada a objetos que permite que objetos de diferentes classes sejam tratados de maneira uniforme, por meio de interfaces comuns e métodos com a mesma assinatura.

No contexto da montagem de computadores, o polimorfismo pode ser aplicado em diversas situações. Por exemplo, imagine que existam diferentes tipos de componentes, como processadores, memórias RAM, placas de vídeo, entre outros. Cada um desses componentes pode ser representado por uma classe específica, com suas próprias propriedades e comportamentos.

No entanto, ao utilizar o polimorfismo, é possível criar uma interface comum para todos esses componentes, definindo métodos como "instalar", "verificar

compatibilidade", "configurar", entre outros. Esses métodos terão a mesma assinatura em todas as classes de componentes, mas cada classe irá implementá-los de acordo com suas características específicas.

Dessa forma, ao escrever o código para montar o computador, é possível tratar todos os componentes de maneira uniforme, sem se preocupar com as diferenças entre eles. Por exemplo, ao chamar o método "instalar" em um objeto do tipo Processador ou em um objeto do tipo Memória RAM, o código executará a implementação específica de cada classe sem a necessidade de tratar cada caso de forma separada.

O polimorfismo traz benefícios significativos para o código do trabalho "Monte o seu computador". Ele simplifica a lógica de programação, tornando-a mais flexível e extensível. Além disso, o polimorfismo favorece a reutilização de código, pois as classes de componentes podem ser facilmente estendidas ou substituídas por novas classes sem afetar o funcionamento das demais partes do sistema.

Em resumo, o polimorfismo no código do trabalho "Monte o seu computador" permite tratar os diferentes componentes de maneira uniforme, simplificando a lógica de programação, favorecendo a reutilização de código e tornando o sistema mais flexível e extensível.

4.5 Métodos Especiais:

No código do trabalho "Monte o seu computador", foram implementados alguns métodos especiais, também conhecidos como métodos mágicos ou dunder methods (double underscore methods). Esses métodos têm a finalidade de personalizar o comportamento das classes e fornecer funcionalidades específicas para as operações relacionadas a objetos.

Aqui estão alguns dos métodos especiais presentes no código do trabalho "Monte o seu computador":

Método __init__(self, nome, preco): Esse método é chamado automaticamente quando um objeto da classe Componente é criado. Ele recebe como parâmetros o nome e o preço do componente e é responsável por inicializar os atributos da classe com esses valores.

Método __str__(self): Esse método retorna uma representação em string do objeto Componente quando ele é convertido em uma string. Ele é usado para exibir informações significativas do componente, como o nome e o preço, quando necessário, por exemplo, na impressão do objeto.

Método __eq__(self, other): Esse método permite comparar se dois objetos Componente são iguais. Ele recebe como parâmetro outro objeto Componente e verifica se ambos possuem o mesmo nome e preço. É utilizado quando utilizamos o operador de igualdade (==) entre dois objetos da classe Componente.

Método __lt__(self, other): Esse método é utilizado para determinar se um objeto Componente é menor que outro objeto Componente. Ele é acionado quando utilizamos o operador de menor que (<) entre dois objetos da classe Componente. No código, ele compara os preços dos componentes para determinar a ordem.

Método __add__(self, other): Esse método permite a adição de dois objetos Componente. Ele é acionado quando utilizamos o operador de adição (+) entre dois objetos da classe Componente. No código, ele retorna um novo objeto Componente com o nome concatenado dos componentes e o preço total.

Método __iter__(self): Esse método permite que o objeto Componente seja iterado. Ele é acionado quando utilizamos um loop de iteração em um objeto da classe Componente. No código, ele retorna um iterador que percorre os componentes na ordem em que foram adicionados.

Esses são alguns exemplos dos métodos especiais implementados no código do trabalho "Monte o seu computador". Eles permitem personalizar o comportamento das classes Componente, fornecendo funcionalidades específicas e facilitando a manipulação e interação com os objetos. Esses métodos especiais contribuem para um código mais elegante, legível e flexível, proporcionando uma melhor experiência para o usuário ao montar seu computador personalizado.

5 – Relatório com as linhas de código

Classe principal:

// CLASSE PRINCIPAL

//Fizemos um layout para tela do usuário final onde criamos 1 objeto chamado computador da superclasse computador e realizamos a comunicação das classes de cada componente com para montar o computador, atribuindo os métodos escolherProcessador e adicionarComponente em todos as subclasses componentes, no final exibimos as especificações do computador juntamente com seu valor total.

```
package monteseupc;

public class MonteseuPc {
    public static void main(String[] args) {
        Computador computador = new Computador();

        System.out.println("-------------------------------------------------");
        System.out.println("          Bora montar seu computador!?          ");
        System.out.println("        Rapido e Pratico, sem enrolacao!        ");
        System.out.println("-------------------------------------------------");

        //Componentes do computador:
        Processador processador = Processador.escolherProcessador();
        computador.adicionarComponente(processador);

        PlacaMae placamae = PlacaMae.escolherPlacaMae();
        computador.adicionarComponente(placamae);

        PlacaVideo placavideo = PlacaVideo.escolherPlacaVideo();
        computador.adicionarComponente(placavideo);

        MemoriaRam memoriaRam = MemoriaRam.escolherMemoriaRam();
```

```
        computador.adicionarComponente(memoriaRam);

        Fonte fonte = Fonte.escolherFonte();
        computador.adicionarComponente(fonte);

        Armazenamento hd = Armazenamento.escolherArmazenamento();
        computador.adicionarComponente(hd);

        //Exibir as especificações do computador
        computador.exibirEspecificacoes();

        // Calcular o preço total do computador
        double precoTotal = computador.calcularTotal();
        System.out.println("Preco Total: R$ " + precoTotal);
    }
}
```

//SUBCLASSES DE COMPONENTES

//Na parte de cada classe de cada componente do computador que são: processador, placaMae, placaVideo, memoriaram, Fonte, Armazenamento. Seguimos uma lógica padrão definindo 2 atributos exclusivos de cada classe a mais do que os 3 atributos da superclasse componentes. Criamos um método que permite que o usuário escolha duas opções de cada componente por vez, podendo digitar sua escolha (1 ou 2), e caso ele informe algum número diferente (inválido) o algoritmo seleciona uma opção defina como padrão do componente que o usuário estiver escolhendo, para implementar essa função optamos por utilizar o comando SwitchCase que é a parecido com o IF else. No final de cada classe componente chamamos o método exibirEspecificacoes colocando um system para exibir a descrição do componente, utilizando o comando get de cada atributo.

Classe Componente:

//SUPERCLASSE+SUBCLASSE COMPONENTE

//Agora criamos uma subclasse da superclasse computador, que será também uma superclasse e irá interagir com suas subclasses, compartilhando seus atributos como (marca, modelo e preço) que são atributos em comum com as próximas subclasses. Todos os atributos do nosso código foram programados como tipo privado e utilizamos os métodos especiais (get e set) para conseguir visualizar e manipular seus valores.

```
package monteseupc;
public abstract class Componente {
    //Atributos padrão dos componentes
    private String marca;
    private String modelo;
    private double preco;

    public Componente(String marca, String modelo, double preco) {
        this.marca = marca;
        this.modelo = modelo;
        this.preco = preco;
    }

    //métodos construtores que permite alterar valor
    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public String getModelo() {
        return modelo;
    }
```

```
    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }

    // Método abstrato para ser implementado nas subclasses
    public abstract void exibirEspecificacoes();
}
```

Classe Processador:

```
package monteseupc;
import java.util.Scanner;
public class Processador extends Componente {
    //Atributos do processador
    private double frequencia;
    private int nucleos;

    public Processador(String marca, String modelo, double preco, double frequencia,
int nucleos) {
        super(marca, modelo, preco);
        this.frequencia = frequencia;
        this.nucleos = nucleos;
```

```
}

//métodos construtores que permite alterar valor
public double getFrequencia() {
    return frequencia;
}

public void setFrequencia(double frequencia) {
    this.frequencia = frequencia;
}

public int getNucleos() {
    return nucleos;
}

public void setNucleos(int nucleos) {
    this.nucleos = nucleos;
}

//Opção de escolha para usuario
public static Processador escolherProcessador() {
System.out.println("Opcoes de Processadores:");
System.out.println("1. Intel Core i5 - R$ 1000.00");
System.out.println("2. AMD Ryzen 7 - R$ 1500.00");
System.out.print("Escolha uma opcao (1 ou 2): ");
Scanner scanner = new Scanner(System.in);
int escolha = scanner.nextInt();
scanner.nextLine(); // Consumir a quebra de linha pendente

System.out.println("-------------------------------------------------");
```

```
    switch (escolha) {
        case 1 -> {
            return new Processador("Intel", "Core i5", 1000.0, 3.2, 6);
            }
        case 2 -> {
            return new Processador("AMD", "Ryzen 7", 1500.0, 3.8, 8);
            }
        default -> {
            System.out.println("Opcao invalida. Processador padrao selecionado.");
            return new Processador("Intel", "Core i3", 800.0, 3.0, 4);
            }
        }

 }

    @Override
    public void exibirEspecificacoes() {
        String especificacoes = "Processador: " + getMarca() + getModelo() + " |
Frequencia: " + getFrequencia() + " GHz | Nucleos: " + getNucleos();
        System.out.println(especificacoes);
        System.out.println("");
    }
}
```

Classe Armazenamento:

```
package monteseupc;
import java.util.Scanner;
public class Armazenamento extends Componente {
    //Atributos do HD de armazenamento
```

```
    private int capacidade;
    private String tipo;

    public Armazenamento(String marca, String modelo, double preco, int
capacidade, String tipo) {
        super(marca, modelo, preco);
        this.capacidade = capacidade;
        this.tipo = tipo;
    }

    //métodos construtores que permite alterar valor
    public int getcapacidade(){
        return capacidade;
    }

    public void setcapacidade(int capacidade){
        this.capacidade = capacidade;
    }

     public String gettipo(){
        return tipo;
    }

    public void settipo(String tipo){
        this.tipo = tipo;
    }

    //Opção de escolha para usuario
    public static Armazenamento escolherArmazenamento() {
        System.out.println("Opcoes de Armazenamento:");
        System.out.println("1. Samsung SSD 500GB - R$ 400.00");
```

```
        System.out.println("2. Western Digital HDD 1TB - R$ 200.00");
        System.out.print("Escolha uma opcao (1 ou 2): ");
        Scanner scanner = new Scanner(System.in);
        int escolha = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha pendente

        System.out.println("------------------------------------------------");

        switch (escolha) {
            case 1 -> {
                return new Armazenamento("Samsung", "SSD", 400.0, 500, "SSD");
            }
            case 2 -> {
                return new Armazenamento("Western Digital", "HDD", 200.0, 1000,
"HDD");
            }
            default -> {
                System.out.println("Opcao invalida. Armazenamento padrao
selecionado.");
                return new Armazenamento("Samsung", "SSD", 400.0, 500, "SSD");
            }
        }
    }

    @Override
    public void exibirEspecificacoes() {
        String especificacoes = "Armazenamento: " + getMarca() + getModelo() + " |
Capacidade: " + getcapacidade() + "GB | Tipo: " + gettipo();
        System.out.println(especificacoes);
        System.out.println("");
    }
```

```
}
```

Classe Fonte:

```
package monteseupc;
import java.util.Scanner;
public class Fonte extends Componente {
    //Atributos da fonte
    private int potencia;
    private String certificacao;

    public Fonte (String marca, String modelo, double preco, int potencia, String
certificacao) {
        super(marca, modelo, preco);
        this.potencia = potencia;
        this.certificacao = certificacao;
    }

    //métodos construtores que permite alterar valor
    public int getPotencia() {
        return potencia;
    }

    public void setPotencia(int potencia) {
        this.potencia = potencia;
    }

    public String getCertificacao() {
        return certificacao;
    }
```

```
    public void setCertificacao(String certificacao) {
       this.certificacao = certificacao;
    }

    public static Fonte escolherFonte() {
        System.out.println("Opcoes de Fonte de Energia:");
        System.out.println("1. Corsair CX550M 550W - R$ 300.00");
        System.out.println("2. EVGA SuperNOVA 750 G3 750W - R$ 500.00");
        System.out.print("Escolha uma opcao (1 ou 2): ");
        Scanner scanner = new Scanner(System.in);
        int escolha = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha pendente

        System.out.println("-------------------------------------------------");

        switch (escolha) {
            case 1 -> {
                return new Fonte("Corsair", "CX550M", 300.0, 550, "80 Plus Bronze");
            }
            case 2 -> {
                return new Fonte("EVGA", "SuperNOVA 750 G3", 500.0, 750, "80 Plus
Gold");
            }
            default -> {
                System.out.println("Opcao invalida. Fonte de Energia padrao
selecionada.");
                return new Fonte("Corsair", "CX550M", 300.0, 550, "80 Plus Bronze");
            }
        }
    }
```

```
    @Override
    public void exibirEspecificacoes() {
        String especificacoes = "Fonte de Energia: " + getMarca() + getModelo() + " |
Potencia: " + getPotencia() + "W | Eficiencia: " + getCertificacao();
        System.out.println(especificacoes);
        System.out.println("");
    }
}
```

Classe MemoriaRam:

```
package monteseupc;
import java.util.Scanner;
public class MemoriaRam extends Componente {
    //Atributos da Memoria Ram
    private int capacidade;
    private String tipo;

    public MemoriaRam(String marca, String modelo, double preco, int capacidade,
String tipo) {
        super(marca, modelo, preco);
        this.capacidade = capacidade;
        this.tipo = tipo;
    }

    //métodos construtores que permite alterar valor
    public int getCapacidade(){
        return capacidade;
    }
```

```
public void setCapacidade(int capacidade){
    this.capacidade = capacidade;
}

 public String gettipo(){
    return tipo;
}

public void settipo(String tipo){
    this.tipo = tipo;
}

//Opção de escolha para usuario
public static MemoriaRam escolherMemoriaRam() {
    System.out.println("Opcoes de Memoria RAM:");
    System.out.println("1. Corsair Vengeance 8GB - R$ 300.00");
    System.out.println("2. Kingston HyperX 16GB - R$ 500.00");
    System.out.print("Escolha uma opcao (1 ou 2): ");
    Scanner scanner = new Scanner(System.in);
    int escolha = scanner.nextInt();
    scanner.nextLine(); // Consumir a quebra de linha pendente

    System.out.println("-----------------------------------------------");

    switch (escolha) {
        case 1 -> {
            return new MemoriaRam("Corsair", "Vengeance", 300.0, 8, "DDR4");
        }
        case 2 -> {
            return new MemoriaRam("Kingston", "HyperX", 500.0, 16, "DDR4");
```

```
            }
            default -> {
                System.out.println("Opcao invalida. Memoria RAM padrao selecionada.");
                return new MemoriaRam("Corsair", "Vengeance", 300.0, 8, "DDR4");
            }
        }
    }

    @Override
    public void exibirEspecificacoes() {
        String especificacoes = "Memoria RAM: " + getMarca() + getModelo() + " |
Capacidade: " + getCapacidade() + "GB | Frequencia: " + gettipo();
        System.out.println(especificacoes);
        System.out.println("");
    }
}
```

Classe PlacaVideo:

```
package monteseupc;
import java.util.Scanner;
public class PlacaVideo extends Componente {
    //Atributos da placa de vídeo
    private int memoria;
    private String fabricante;

    public PlacaVideo(String marca, String modelo, double preco, int memoria, String
fabricante) {
        super(marca, modelo, preco);
        this.memoria = memoria;
        this.fabricante = fabricante;
```

```
}

//métodos construtores que permite alterar valor
public int getMemoria() {
    return memoria;
}

public void setMemoria(int memoria) {
    this.memoria = memoria;
}

public String getFabricante() {
    return fabricante;
}

public void setFabricante(String fabricante) {
   this.fabricante = fabricante;
}

public static PlacaVideo escolherPlacaVideo() {
    System.out.println("Opcoes de Placa de Video:");
    System.out.println("1. NVIDIA GeForce GTX 1660 Super - R$ 1500.00");
    System.out.println("2. AMD Radeon RX 5700 XT - R$ 2000.00");
    System.out.print("Escolha uma opcao (1 ou 2): ");
    Scanner scanner = new Scanner(System.in);
    int escolha = scanner.nextInt();
    scanner.nextLine(); // Consumir a quebra de linha pendente

    System.out.println("-----------------------------------------------");
```

```
        switch (escolha) {
            case 1 -> {
                return new PlacaVideo("NVIDIA", "GeForce GTX 1660 Super", 1500.0, 6,
"ASUS");
            }
            case 2 -> {
                return new PlacaVideo("AMD", "Radeon RX 5700 XT", 2000.0, 8,
"Gigabyte");
            }
            default -> {
                System.out.println("Opcao invalida. Placa de Vídeo padrao selecionada.");
                return new PlacaVideo("NVIDIA", "GeForce GTX 1660 Super", 1500.0, 6,
"ASUS");
            }
        }
    }

    @Override
    public void exibirEspecificacoes() {
        String especificacoes = "Placa de Video: "+ getMarca() + getModelo() + " |
VRAM: " + getMemoria() + "GB | Tipo de Conexao: " + getFabricante();
        System.out.println(especificacoes);
        System.out.println("");
    }
}
```

Classe PlacaMae:

```
package monteseupc;
import java.util.Scanner;
public class PlacaMae extends Componente {
    //Atributos da placa mãe
```

```
    private String formato;
    private int slotsMemoria;

    public PlacaMae(String marca, String modelo, double preco, String formato, int
slotsMemoria) {
        super(marca, modelo, preco);
        this.formato = formato;
        this.slotsMemoria = slotsMemoria;
    }

    //métodos construtores que permite alterar valor
    public String getFormato() {
        return formato;
    }

    public void setFormato(String formato) {
        this.formato = formato;
    }

    public int getSlotsMemoria() {
        return slotsMemoria;
    }

    public void setSlotsMemoria(int slots) {
       this.slotsMemoria = slots;
    }

    public static PlacaMae escolherPlacaMae() {
        System.out.println("Opcoes de Placa Mae:");
        System.out.println("1. ASUS PRIME B450M-GAMING/BR - R$ 500.00");
        System.out.println("2. Gigabyte B550 AORUS ELITE - R$ 800.00");
```

```
        System.out.print("Escolha uma opcao (1 ou 2): ");
        Scanner scanner = new Scanner(System.in);
        int escolha = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha pendente

        System.out.println("-------------------------------------------------");

        switch (escolha) {
            case 1 -> {
                return new PlacaMae("ASUS", "PRIME B450M-GAMING/BR", 500.0,
"Micro ATX", 4);
            }
            case 2 -> {
                return new PlacaMae("Gigabyte", "B550 AORUS ELITE", 800.0, "ATX", 4);
            }
            default -> {
                System.out.println("Opcao invalida. Placa Mae padrao selecionada.");
                return new PlacaMae("ASUS", "PRIME B450M-GAMING/BR", 500.0,
"Micro ATX", 4);
            }
        }
    }

    @Override
    public void exibirEspecificacoes() {
        String especificacoes = "Placa Mae: " + getMarca() + getModelo() + " | Slots de
Memoria: " + getSlotsMemoria();
        System.out.println(especificacoes);
        System.out.println("");
    }
}
```

Classe Computador:

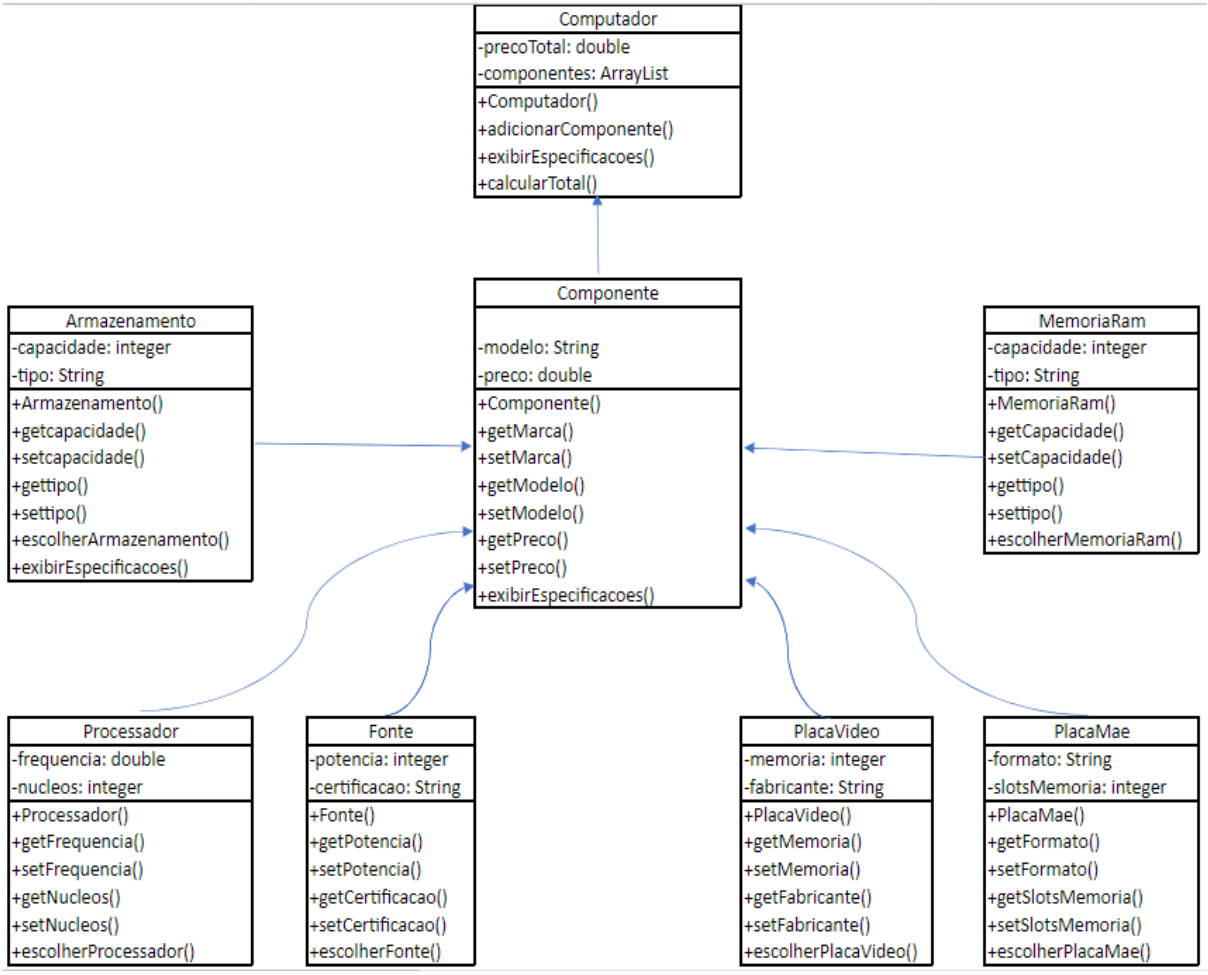
//SUPERCLASSE COMPUTADOR

//Criamos uma superclasse (Computador) onde utilizamos um atributo ArrayList que irá armazenar todos os componentes do nosso computador juntamente com o seu preço, para isso criamos um método (adicionarComponente). Também implementamos um método que irá descrever a configuração do computador exibindo as informações de cada componente (exibirEspecificacoes).

```
package monteseupc;

import java.util.ArrayList; //Importação de uma função para listas

public class Computador {

    //Atributos do computador

    private ArrayList<Componente> componentes; //Lista que irá armazenar os componentes do computador

    private double precoTotal; //Variável que irá somar o preço de todos os componentes


    public Computador() {

        componentes = new ArrayList<>();

        precoTotal = 0.0;

    }


    //Método para adicionar o componente a lista e também realizar a soma do produto para o valor total

    public void adicionarComponente(Componente componente) {

        componentes.add(componente);

        precoTotal += componente.getPreco();

    }


    //Método para exibir as informações do computador escolhido pelo usuário

    public void exibirEspecificacoes() {
```

```
        System.out.println("          Configuracao do Computador          ");
        System.out.println("----------------------------------------------");
        for (Componente componente : componentes) {
            componente.exibirEspecificacoes();
        }
    }

    //Método para retornar o valor total
    public double calcularTotal() {
        return precoTotal;
    }
}
```

6 – Diagrama UML das classes



7 – Bibliografia:

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

https://www.devmedia.com.br/conceitos-e-exemplos-polimorfismo-programacao-orientada-a-objetos/18701#:~:text=O%20Polimorfismo%20%C3%A9%20um%20mecanismo,por%C3%A9m%2C%20com%20implementa%C3%A7%C3%B5es%20l%C3%B3gicas%20diferentes.

https://www.devmedia.com.br/conceitos-e-exemplos-heranca-programacao-orientada-a-objetos-parte-1/18579

https://www.alura.com.br/artigos/o-que-e-encapsulamento

https://www.tomshardware.com/

https://www.javatpoint.com/

https://www.geeksforgeeks.org/

https://stackoverflow.com/

https://razor.com.br/blog/hardware/configuracao-de-computadores-a-importancia-de-montar-a-maquina-ideal-para-voce/

https://www.javaprogressivo.net/2012/09/o-comando-switch-fazendo-escolhas-em_6667.html

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, trabalhos Individuais e em grupo, práticas de ensino e outras)

NOME: Igor Souza, Evando Barbosa, João Lucas, Luccas Zanatta

RA:N86272-6, G4757D-2, N9416H-4, G449EF-3 CURSO:Ciência da Computação

CAMPUS: RIB.PRETO-VARGAS SEMESTRE:3º Semestre TURNO:Noturno

			ASSIN
DATA	ATIVIDADE	TOTAL DE HORAS	ALUNO
03/04/2023	Reunião para conversar sobre possíveis temas a ser escolhidos	1:00:00	TODOS
05/04/2023	Tema escolhido, conversamos como fariamos, quais classes/métodos deveriam ser criados	1:30:00	TODOS
10/04/2023	Iniciamos o código na classe "Computador"	2:00:00	IGOR, EVANDO
13/04/2023	Terminamos a classe "Computador" e iniciamos o código na classe "Componente"	1:30:00	EVANDO, RUIVO
16/04/2023	Continuamos a discutir e codando na classe "Componente"	0:45:00	JOÃO, LUCCAS
16/04/2023	Terminamos a classe "Componente" e fomos para a criação das subclasses de componente	0:30:00	TODOS
20/04/2023	Criamos as classes: "Fonte", "MemoriaRam", "PlacaMae"	0:40:00	EVANDO, JOÃO, IGOR
23/04/2023	Criamos as classes: "PlacaVideo" e "Processador"	0:30:00	EVANDO, LUCCAS
13/05/2023	Criamos a classe "MonteseuPc" e testamos o algoritimo	0:20:00	TODOS
14/05/2023	Fomos mudando partes do código e testando para que chegasse em uma finalização melhor	1:00:00	IGOR, JOÃO
18/05/2023	Fazendo a parte escrita...	1:30:00	TODOS
20/05/2023	continuação da parte escrita...	0:50:00	EVANDO, JOÃO
21/05/2023	continuação e finalização da parte escrita	1:15:00	TODOS
25/05/2023	Criação dos slides e do vídeo de apresentação	1:00:00	TODOS

TOTAL DE HORAS: 14:20:00