Nome: Allan Samuel Alves do Monte
Matrícula: 2410279
Curso: Ciência da Computação

Introdução aos conceitos de orientação a objetos (Resumo)

- Com o avanço da internet, a orientação a objetos passou a ser amplamente utilizada e os seus conceitos são tão atuais como eram 25 anos atrás.
- Quando a orientação a objetos começou a ser implementada, havia uma questão em como usar o OO (orientação a objetos) com a linguagem estruturada já existente, a verdade é que as duas são complementares e não devem ser tratadas com distinção em relação à exclusividade de uso.
- Não se deve substituir programas que utilizam ~~OO~~ linguagem estruturada por programas que utilizam OO; cada um tem a sua relevância.
- Um objeto necessita ter atributos e comportamento

"
- No OO, os atributos e comportamentos estão dentro (contidos) em um único objeto, enquanto na linguagem procedural ou estruturada, os atributos e comportamentos são normalmente separados."

- Mas por que trocar as linguagens já estruturadas que funcionam muito bem, por linguagem OO?

- "Não há dados ofuscos em OO, essa foi provavelmente uma grande quantidade de integridade de dados em sistemas OO"

- "Na terminologia OO, dados são referidos como atributos, e comportamentos são referidos como métodos. Restringir acesso a certos atributos ou métodos é chamado de "esconder dados"."

- Encapsulação é combinar atributos e métodos em uma mesma entidade.

- "É normalmente ~~comum~~ melhor construir pequenos objetos com tarefas específicas do que grandes objetos que têm muitas tarefas."

- Na linguagem procedural, o envio de dado é feito de uma forma separada, nas linguagem OO a data e a código (operação que manipula a data) são enviados (encapsulados) no objeto.

- "Objetos são as tijolos da programação OO".

- "Getters e Setters: o conceito de getters e setters ajudam o conceito de 'esconder dados'. Por que outros objetos não deveriam manipular diretamente dados dentro de outro objeto, os getters e setters providenciam acesso controlado aos dados do objeto. Getters e Setters os vezes são chamados de "métodos de acessa" e "métodos de mudança", respectivamente."

- Cada objeto em uma classe tem sua própria cópia dos atributos e métodos presentes na classe.

- "~~Um objeto~~ Uma classe é uma planta de um objeto. Quando você criar/instância

Parte 2

um objeto, você cria uma classe como base para como esse objeto é construída."

- Um objeto não pode ser criado (instanciado) sem uma classe
- "Você precisa projetar uma classe antes de criar um objeto
- Exemplo de uma classe "pessoa":

```
public class Person {

    //Atributos
    private String name;
    private String address;

    //métodos
    public String getName() {
        return name;
    }
public void setName(String n) {
        name = n;
    }
    public String getAddress() {
        return address;
    }
    public void setAdress(String adr) {
        address = adr;
    }
}
```

- Uma tipo de dado "pública" pode ser acessada diretamente por outros objetos, um tipo de dado "privada" só pode ser acessada por aquele objeto específico.

- "Métodos podem implementar comportamentos que são chamados por outros objetos (mensagem), ou providenciar o funcionamento interno comportamental da classe".

- "Encapsulação é definida pelo fato de que objetos possuem tanto os atributos quanto os comportamentos. Esconder dados é uma parte considerável da encapsulação".

- "Para esconder dados, é necessário que todos os atributos sejam declarados como privados".

- Apenas os métodos e atributos PÚBLICOS são considerados a interface, ou seja, tudo que é privado não deve ser acessado pelo usuário.

- Você pode mudar dados de um arquivo para um banco de dados sem precisar reescrever e escrevê-los novamente.

- "Herança retorna uma classe a herdar os atributos e métodos de outra classe. Isso providencia a habilidade de criar novas classes abstraindo atributos e comportamentos comuns de outra classe".

- As superclasses (pai) tem todos os atributos que as subclasses (filhos) vão herdar (mamífero é uma superclasse, cães e gatos são subclasses).

- Por sua vez subclasses podem ser superclasses e dela mais subclasses surgem.

Parte 3

- "Heranças múltiplas: considere uma criança que herda de dois parentes. Qual cor de olho a criança vai herdar? Isso é um problema significante quando se trata de escrever um compilador. C++ autoriza múltiplas heranças, muitas linguagens (como Java) não".

- Explicando polimorfismo com exemplo: "Por exemplo, suponha que você tem um ator de três formas - Círculo, Quadrado e Estrela. Mesmo que você os trate como objetos "forma", e mande uma mensagem "desenhar" para cada "forma" objeto, ~~o fim resulta~~ o resultado final é diferente para cada um, porque Círculo, Quadrado e Estrela providenciam a implementação real. Em resumo, cada classe é "pessoal" de responder diferente a um mesmo método "desenhar" e desenhar a si mesmo, isso é o sentido de polimorfismo".

"- Quando o nome de um método é o mesmo de sua classe e não tem tipo de retorno, o método é um método especial, chamado de construtor. Considere um construtor como a porta de entrada para a classe, onde o objeto é construído. O construtor é um bom lugar para performar inicializações e tarefas de inicialização".

"- Objetos são frequentemente construídos por compostos, de outros objetos: Isso é composição"

- ("Enquanto herança é considerada uma 'é-uma' relação, a composição é denominada como 'Tem-uma' relação. O computador tem um disco rígido, tem uma placa de vídeo, mas uma placa de vídeo não é um computador. (Essa é a diferença entre herança e composição, herança é dependente, composição é 'independente)".