UNIP
UNIVERSIDADE PAULISTA

Linguagem de Programação Orientada a Objeto

Abstração - Encapsulamento

Professora Sheila Cáceres

Variáveis locais

- Campos são um tipo de variável. Eles:
  - armazenam valores por toda a vida de um objeto; e
  - são acessíveis por meio da classe.
- Métodos podem incluir variáveis de vida mais curta. Eles:
  - existem apenas enquanto o método está em execução; e
  - são acessíveis de dentro do método.

Uma variável local

```
public int refundBalance()
{
    int amountToRefund;
    amountToRefund = balance;
    balance = 0;
    return amountToRefund;
}
```

Modificadores de Acesso

Modificadores de Acesso

- Usados para alterar o escopo dos membros da classe pois é conveniente proibir o acesso a alguns atributos o métodos de uma classe.
- Utilizados antes das declarações de atributos e métodos.
- Na orientação a objetos, é prática quase que obrigatória proteger os atributos.
- Cada classe é responsável por controlar seus atributos. Esta validação não deve ser controlada por quem está usando a classe que pode desconhecer fatores internos.
- Java fornece:
  - Public
  - Private
  - Protected
  - Default (ausencia de um modificador de acesso)

O modificador Público (public)

- Permite o maior grau de visibilidade.
- Permite acesso a partir de qualquer classe sem restrições.
- Pode ser aplicado a classes, atributos de classes, construtores e métodos.
  - Exemplo: public class Aluno{ }

O modificador Privado (private)

- É o mais restritivo de todos
- Permite acesso apenas na própria classe.
- Nem mesmo subclasses podem acessar elementos private da sua superclasse. (conceitos explicados em próximos módulos)
- Os elementos privados são ocultos para o programador usuário que for usar instâncias da classe.
- Usualmente é utilizado para os atributos de uma classe (ocultamento de dados).
- Pode ser aplicado a atributos, métodos (não pode ser aplicado a classes). Ex dentro de uma classe: private String nome
- O componente privado não é acessado de fora da classe. Para tornar um atributo privado acessível, deve-se definir um método público na própria classe, que retorne o valor do atributo (get...) e se quisermos modificar o atributo precisariamos definir outro método (set...).

O modificador Protegido (protected)

- Apenas as subclasses e classes do mesmo pacote tem acesso.
- Pode ser aplicado normalmente a variáveis e métodos (não se aplica a classes)

Ausência de um modificador de acesso

- Essa ausência indica um grau de acessibilidade considerado ”default”.
- Torna os elementos visíveis somente para a própria classe e para as outras classes contidas na mesma package.
- Elementos declarados como default possuem **menor** grau de acesso do que os declarados como protected (elementos default não podem ser usados pelas subclasses caso pertençam a outras packages).

Modificadores de Acesso

- A tabela indica quais elementos da classe podem ter quais modificadores de acesso.

	Classe	Atributos	Construtores	Métodos
public	sim	sim	sim	sim
protected	não	sim	sim*	sim
default	sim	sim	sim	sim
private	não	sim	sim*	sim

* Pouco utilizado

Exemplo: Uma classe não pode ter o modificador de acesso private

- A tabela indica os níveis de acesso (ou graus de visibilidades) definidos por cada modificador

	Classe	Package	Subclasse*	Global**
public	sim	sim	sim	sim
protected	sim	sim	sim	não
default	sim	sim	não	não
private	sim	não	não	não

* Subclasse contida em outra package.
** Acesso por classes de outros projetos ou sistemas.

Pilares da POO

- Abstração
- Encapsulamento
- Herança (será visto em módulos futuros)
- Polimorfismo (será visto em módulos futuros)

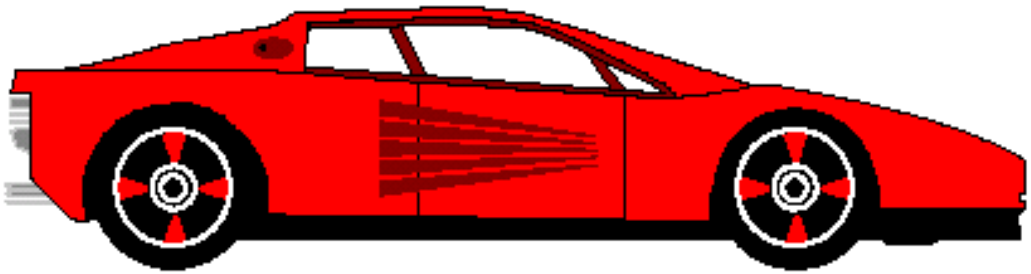
Abstração

- **Ignorar aspectos não relevantes.**
- É a capacidade de focalizar o essencial e ignorar detalhes menos importantes ou acidentais.
- Abstração é utilizada para a definição de entidades do mundo real tendo como consideração as suas características e ações relevantes.
- Para abstrair um objeto do mundo real criamos as classes a partir da qual obteremos os objetos.

Entidade	Características	Ações
Carro, Moto	tamanho, cor, peso, altura	acelerar, parar, ligar, desligar
Elevador	tamanho, peso máximo	subir, descer, escolher andar
Conta Banco	saldo, limite, número	depositar, sacar, ver extrato

Abstração

Motorista dirige um carro através dos pedais, alavanca de marchas e volante. Questões a respeito de motor estão escondidas para ele.

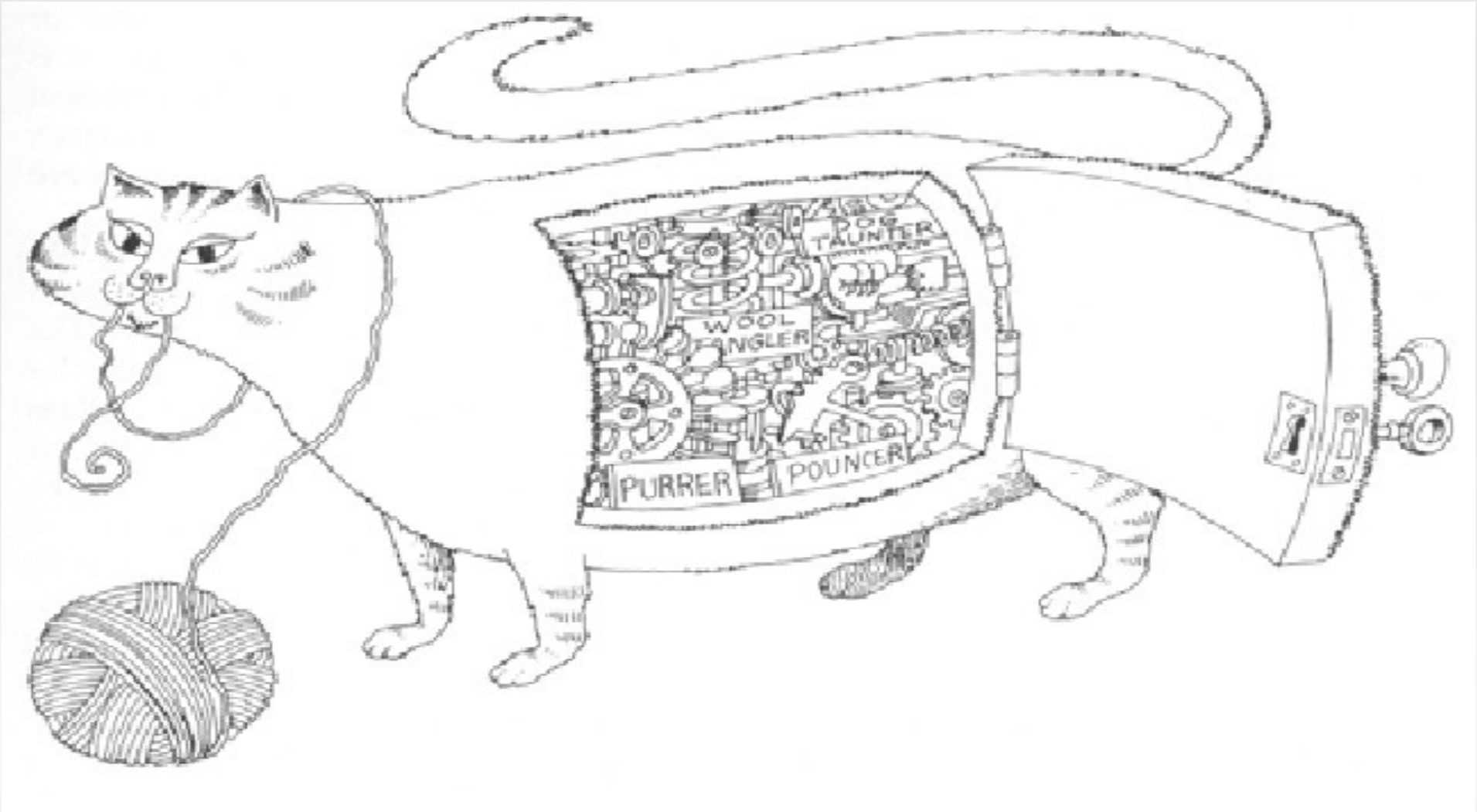


Encapsulamento

- **Ocultar aspectos não relevantes.**
- Técnica para esconder uma ideia para o usuário, tornando partes do sistema o mais independentes possível.
- Um dos grande trunfo da POO em relação a prog. Tradicional: Os dados e processos estão em uma única entidade, permite alterações sem afetar demais partes do sistema.
- **Exemplo**, Quando um controle remoto estraga apenas é trocado ou consertado o controle e não a televisão inteira. Nesse exemplo do controle remoto, acontece a forma clássica de encapsulamento, pois quando o usuário muda de canal não se sabe que programação acontece entre a televisão e o controle para efetuar tal ação.

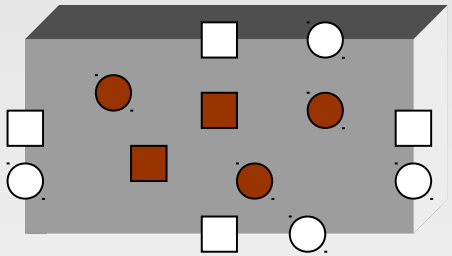
Encapsulamento

Posso ocultar detalhes não relevantes (com chave na figura).



Encapsulamento

O encapsulamento objetiva esconder detalhes de implementação



Métodos Públicos

Métodos Privados

Atributos Públicos

Atributos Privados

Encapsulamento

- Em um processo de encapsulamento:
  - os atributos das classes são do tipo **private**.
  - Para acessar esses tipos de modificadores, é necessário criar métodos **setters** e **getters**.

```
public class Funcionario {
	private double salario;
	private String nome;

	public String getNome() { return nome; }
	public void setNome(String nome) { this.nome = nome; }
	public void setSalario(double salario) { this.salario = salario; }
	public double getSalario() { return salario; }
}
```

Encapsulamento

- *Benefícios*
  - **Modularidade** - (O código é independente de outros objetos)
  - **Informações Privadas** (um objeto tem uma interface pública que outros objetos podem utilizar. As informações do objeto são privadas) [**Information Hiding]**
  - **Permitir a criação de programas com menos erros e mais clareza.**
  - **Segurança no acesso ao objeto;**
  - Melhor consistência no estado interno, pois tem o intuito de diminuir as **alterações incorretas** nos valores das propriedades.

    Ex: no método set podemos adicionar condiciones para garantir a integridade dos dados.

Aplicação

```
public class Triangulo {
	int lado1;
	int lado2;
	int lado3;
	int calculaPerimetro() {
		return lado1 + lado2 + lado3;
	}
}
public class Programa {
	public static void main(String args[]) {
		Triangulo t = new Triangulo();
		t.lado1 = 3;
		t.lado2 = 5;
		t.lado3 = 7;
		System.out.println(t.calculaPerimetro());
	}
}
```

Qual pilar da OO está ferindo?

Aplicação

Encapsulando a classe Triangulo, a classe main da erro:

```
public class Triangulo {
	private int lado1;
	private int lado2;
	private int lado3;
	public int calculaPerimetro() {
		return lado1 + lado2 + lado3;
	}
}
public class Programa {
	public static void main(String args[]) {
		Triangulo t = new Triangulo();
		t.lado1 = 3;  // erro
		t.lado2 = 5;  // erro
		t.lado3 = 7;  // erro
		System.out.println(t.calculaPerimetro());
	}
}
```

Como soluciono?

Aplicação

```
public class Triangulo {
    private int lado1;
    private int lado2;
    private int lado3;
    public int calculaPerimetro() {
        return lado1 + lado2 + lado3;
    }
    //Métodos de acesso
    public int getLado1() {
        return lado1;
    }
    public void setLado1(int novoValor) {
        lado1 = novoValor;
    }
}
```

Adicionando métodos get e set

Agora SIM!!!

```
public class Programa {
    public static void main(String args[]) {
        Triangulo t = new Triangulo();
        t.setLado1(3);
        t.setLado2(5);
        t.setLado3(7);
        System.out.println(t.calculaPerimetro());
    }
}
```

Referências

- Deitel, H.M; Java Como Programar. Ed. Bookman, 2005.
- Programação Orientada a Objetos com Java, David J. Barnes and Michael Kolling. Pearson 2004.
- Material do professor Marco Fagundes, UFPa, 2003.
- http://www.caelum.com.br/apostila-java-orientacao-objetos/modificadores-de-acesso-e-atributos-de-classe/#6-1-controlando-o-acesso
- Material de Encapsulamento pela professora Ludimila Monjardim Casagrande.
-
- Material do Professor Marcio Golçalves

- Nota: O material da apresentação foi extraido de algumas das fontes aqui apresentadas