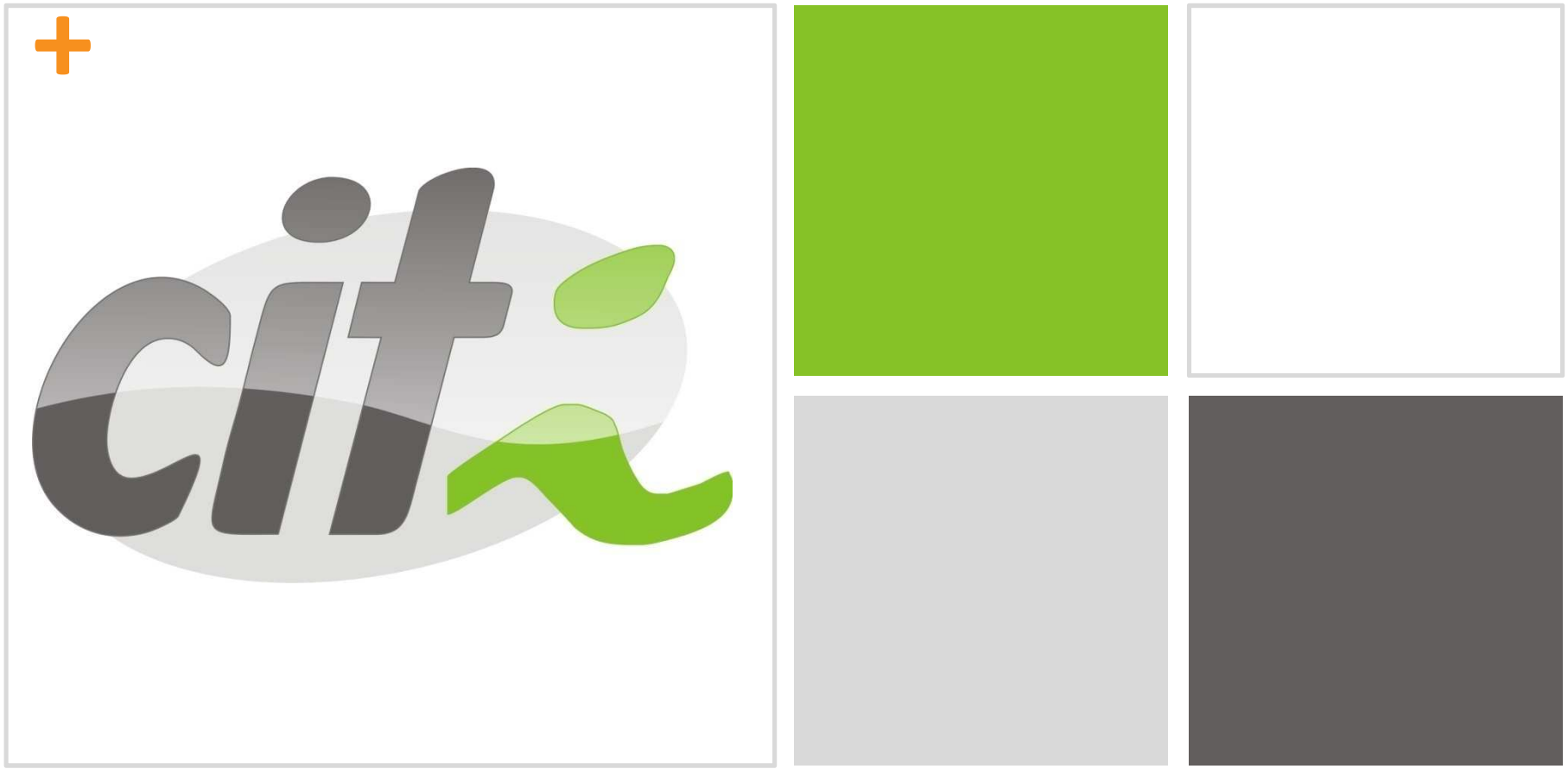

Java Básico

Igor Ebrahim (ies@cin.ufpe.br)

Módulo 6

Classes Abstratas e Interfaces

Interfaces

- Uma interface em Java é uma coleção de métodos abstratos e constantes;
- Um método abstrato é um cabeçalho de método sem o seu corpo;
- Um método abstrato pode ser declarado usando o modificador abstract, mas como todos os métodos de uma interface são abstratos, normalmente não usa-se o modificador;
- Uma interface é usada para estabelecer um conjunto de métodos que uma classe irá implementar;

Interfaces

interface é palavra reservada

```
public interface Factivel
{
   public void facaIsto();
   public int facaIsto2();
   public void facaIsto2 (float valor, char ch);
   public boolean facaOutro (int num);
}
```

Nenhum dos métodos de uma interface possuem uma definição

Um ponto-e-vírgula termina cada um dos cabeçalhos dos métodos

Interfaces

- Uma interface não pode ser instanciada;
- Métodos em uma interface têm que ser public “por default”;
- Uma classe implementa uma interface:
  - Declarando isto no cabeçalho da classe;
  - Implementando cada um dos métodos abstratos da interface.
- Uma classe que implementa uma interface deve definir TODOS os métodos da interface;

Interfaces

```
public class Faz implements Factivel
{
    public void facaIsto ()
    {
        // qualquer coisa
    }

    public int facaIsto2 ()
    {
        // qualquer coisa
    }

    // etc.
}
```

implements é palavra reservada de Java

Cada método listado em Factivel recebe uma definição

Interfaces

- Uma classe que implementa uma interface pode definir outros métodos também;
- Além de métodos abstratos, interfaces podem conter constantes;
- Quando uma classe implementa uma interface, ela ganha acesso a todas essas constantes.

Interfaces

- Uma classe pode implementar múltiplas interfaces;
- As interfaces são listadas após o implements;
- A classe deve implementar todos os métodos de todas as interfaces listadas.

```
class MuitasCoisas
   implements interface1, interface2 {
    // todos os métodos de ambas as interfaces
}
```

Interfaces

- A biblioteca padrão de Java contem uma série de interfaces úteis;
- A interface Comparable contém um método abstrato chamado compareTo, que é usado para comparar dois objetos;
- A classe String implementa Comparable, dando a ela a habilidade de colocar strings em ordem lexicográfica;
- Olhem a API de Java para mais detalhes...
- Dêem uma olhada na interface Iterator. Qualquer dúvida procurem seus monitores.

Classes Abstratas

- Uma classe abstrata é uma classe que representa uma idéia (conceito) genérica;
- Uma classe abstrata não pode ser instanciada;
- Usa-se o modificador abstract no cabeçalho da classe para declará-la como uma classe abstrata:

```
public abstract class Produto
{
    // conteúdo
}
```

Classes Abstratas

- Uma classe abstrata geralmente contém métodos abstratos (sem definição);
- Ao contrário das interfaces, o modificador abstract tem que ser aplicado a todos os métodos abstratos;
- As classes abstratas também podem possuir métodos não abstratos;
- As classes abstratas não precisam necessariamente possuir métodos abstratos.

Classes Abstratas

- Uma sub-classe de uma classe abstrata tem que sobrescrever seus métodos abstratos ou eles continuarão sendo considerados abstratos;
- Um método abstrato não pode ser definido como final ou static;

Herança para Interfaces

- Herança também pode ser aplicada a interfaces da mesma forma que nas classes;
- Isto é, uma interface pode derivar de uma outra já criada;
- A interface filha herda todos os métodos abstratos da interface herdada;
- A classe que implementar a interface filha deve definir todos os métodos;
- Uma classe pode implementar uma ou mais interfaces
  - Herança múltipla de interfaces

Polimorfismo via Interfaces

- Um nome de uma interface pode ser usada como tipo de uma referência de uma variável de objeto:

```
Speaker atual;
```

- A referência atual pode ser usada para apontar para qualquer objeto de qualquer classe que implemente a interface Speaker;
- A versão do método speak que a seguinte linha chama depende do tipo de objeto que atual refere-se:

```
atual.speak();
```

Polimorfismo via Interfaces

- Suponha que duas classes, Filosofo e Cao, ambas implementam a interface Speaker, fornecendo versões distintas do método speak;
- Analisemos o seguinte código:

```
Speaker s = new Filosofo();

s.speak();

s = new Cao();

s.speak();
```

Prática

1. Implemente os tipos ContaAbstrata, Poupanca, ContaImposto e ContaBonificada, levando em consideração a relação hierárquica entre eles. Segue descrição:
   - ContaAbstrata: Tipo de maior abstração na hierarquia;

Prática

- Poupança: Além das características de conta, possui um método renderJuros(double taxa) que credita o valor (taxa * saldo). Lembre-se que a taxa deve ser um número entre 0 e 1;
- ContaImposto: Este tipo debita um dado valor mais imposto. O imposto é calculado aplicando-se a taxa ao valor a ser debitado. Lembre-se de verificar se a conta possui saldo suficiente.

Prática

- ContaBonificada: O método creditar deste tipo, acumula um bônus à instância a cada operação de crédito. Este bônus é igual a 1% do valor creditado. O cliente poderá decidir quando utilizar o bônus acumulado, chamando a funcionalidade renderBonus.