Introdução à Programação
Orientada a Objetos com Java

Interfaces

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

Auditor de Banco

```
public class AuditorB {
    private final static double MINIMO = 500.00;
    private String nome;
    /* ... */
    public boolean auditarBanco(Banco banco) {
        double saldoTotal, saldoMedio;
        int numeroContas;
        saldoTotal = banco.saldoTotal()
        numeroContas = banco.numeroContas();
        saldoMedio = saldoTotal/numeroContas;
        return (saldoMedio < MINIMO);
    }
}
```

Auditor de Banco Modular

```
public class AuditorBM {
    private final static double MINIMO = 500.00;
    private String nome;
    /* ... */
    public boolean auditarBanco(BancoModular banco)
    {
        double saldoTotal, saldoMedio;
        int numeroContas;
        saldoTotal = banco.saldoTotal()
        numeroContas = banco.numeroContas();
        saldoMedio = saldoTotal/numeroContas;
        return (saldoMedio < MINIMO);
    }
}
```

Problema

- Duplicação desnecessária de código
- O mesmo auditor deveria ser capaz de investigar qualquer tipo de banco *que possua operações para calcular*
  - o número de contas, e
  - o saldo total de todas as contas.

Auditor Genérico

```
public class AuditorGenerico {
    private final static double MINIMO = 500.00;
    private String nome;
    /* ... */
    public boolean auditarBanco(QualquerBanco banco)
    {
        double saldoTotal, saldoMedio;
        int numeroContas;
        saldoTotal = banco.saldoTotal()
        numeroContas = banco.numeroContas();
        saldoMedio = saldoTotal/numeroContas;
        return (saldoMedio < MINIMO);
    }
}
```

Definindo Interfaces

```
public interface QualquerBanco {
    double saldoTotal();
    int numContas();
}
```

Interfaces

- Caso especial de classes abstratas...
  - todos os métodos são abstratos
    - —provêem uma interface para serviços e comportamentos
    - —são qualificados como `public` por default
  - não definem atributos
    - —definem constantes
    - —por default todos os "atributos" definidos em uma interface são qualificados como `public`, `static` e `final`
  - não definem construtores

Interfaces

- Não pode-se criar objetos
- Definem tipo de forma abstrata, apenas indicando a assinatura dos métodos
- Os métodos são implementados pelos subtipos (subclasses)

Subtipos sem Herança de Código

```
public class Banco implements QualquerBanco {
    /* ... */
}


public class BancoModular
        implements QualquerBanco {
    /* ... */
}
```

implements

- *classe* implements
  interface1, interface2, ...
- *subtipo* implements
  supertipo1, supertipo2, ...
- Múltiplos supertipos:
  - uma classe pode implementar mais de uma interface (contraste com classes abstratas...)

implements

- Classe que implementa uma interface deve *definir* os métodos da interface:
  - classes concretas têm que implementar os métodos
  - classes abstratas podem simplesmente conter métodos abstratos correspondentes aos métodos da interface

Usando Auditores

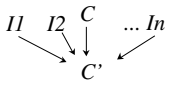
```
Banco b = new Banco();
BancoModular bm = new BancoModular();
Auditor a = new Auditor();
/* ... */
boolean r = a.auditarBanco(b);
boolean r' = a. auditarBanco(bm);
/* ... */
```

Interfaces e Reusabilidade

- Evita duplicação de código através da definição de um tipo genérico, tendo como subtipos várias classes não relacionadas
- Tipo genérico pode agrupar objetos de várias classes definidas de forma independente, sem compartilhar código via herança, tendo implementações totalmente diferentes
- Classes podem até ter mesma semântica...

Definição de Classes: Forma Geral

```
class C'
  extends C
  implements I1, I2, ..., In {
    /* ... */
  }
```



Subtipos com Herança Múltipla de Assinatura

```
interface I
  extends I1, I2, ..., In {
  /*... assinaturas de novos
        métodos ...
  */
}
```

O que usar? Quando?

Classes (abstratas)

- Agrupa objetos com implementações compartilhadas
- Define novas classes através de herança (simples) de código
- Só uma pode ser supertipo de outra classe

Interfaces

- Agrupa objetos com implementações diferentes
- Define novas interfaces através de herança (múltipla) de assinaturas
- Várias podem ser supertipo do mesmo tipo

Cadastro de Contas: Parametrização

```
public class CadastroContas {
    private RepositorioContas contas;
    public CadastroContas (RepositorioContas r) {
        if (r != null) contas = r;
    }
    /* ... */
}
```

A estrutura para armazenamento das contas é fornecida na inicialização do cadastro, e pode depois ser trocada!

Repositório: Definição

```
public interface RepositorioContas {
    void inserir(Conta conta);
    Conta procurar(String numero);
    boolean existe(String numero);
}
```

Repositório: Implementações

```
public class ConjuntoContas
        implements RepositorioContas {...}

public class ListaContas
        implements RepositorioContas {...}

public class ArrayContas
        implements RepositorioContas {...}

public class VectorContas
        implements RepositorioContas {...}
```

Cadastro de Contas: Parametrização

```
public void cadastrar(Conta conta) {
    if (conta != null) {
        String numero = conta.getNumero();
        if (!contas.existe(numero)) {
            contas.inserir(conta);
        }
    }
}
```

Cadastro de Contas: Parametrização

```
public void debitar(String numero, double valor){
    Conta conta;
    conta = contas.procurar(numero);
    if (conta != null) {
        conta.debitar(val);
    }
}
```