# Aula 05 – Criando classes e objetos

Baseado nos slides oficiais do livro Java – Como programar – Deitel e Deitel – 10ª edição

©1992–2015 by Pearson Education, Inc. All Rights Reserved.

# Criando as próprias classes

- Cada classe representa um novo tipo de dados de um conjunto de objetos semelhantes
- O processo de criação geralmente tem as seguintes etapas:
  - Definição de um pacote para a classe
  - Definição do nome da classe
  - Criação dos atributos
  - Criação dos métodos construtores
  - Criação de métodos de acesso (get/set)
  - Criação dos demais métodos

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Pacotes

- Um pacote é onde se encontram um conjunto de classes com propósitos semelhantes.
- Em Java, fisicamente um pacote é um diretório onde se encontram as definições de várias classes. Se um pacote nao é definido o diretório corrente é o pacote padrão.
- Cada pacote tem um nome (geralmente em minúsculo) com pontos separando os níveis de acordo com um padrão :
  - java.??? - padrão da linguagem java (java.io, java.lang)
  - javax.??? - extensão incorporada à linguagem (javax.swing, javax.xml)
  - Domínio reverso - pacotes de terceiros (org.omg, com.google.api.services, br.uern.di.poo)

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Criando um novo pacote

- Palavra reservada package
  - package br.uern.di.poo.exemplos;
- Compilação a partir do diretório base:
  - javac br/uern/di/poo/exemplos/Conta.java
- Variável de ambiente CLASSPATH: indica o diretório onde os pacotes podem ser achados
  - java -cp projetos br.uern.di.poo.exemplos.Conta

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Importando pacotes

- Se uma classe não está definida no mesmo pacote ela precisa ter seu caminho completo definido
  - java.util.Scanner entrada;
- Uma alternativa é importar as classes desejadas
  - import java.util.Scanner;
  - Scanner entrada;
- Pode-se importar todas as classes de um pacote, mas ela não é recursiva
  - import java.util.*;

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Definindo o nome da classe

- Devem ter nomes relacionados com os objetos que aquela classe define
- Os nomes começam com letras maiúsculas, assim como os nomes intermediários
  - public class Conta {...}
  - public class TestaConta {...}

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Criação dos atributos

- São os dados que representam as propriedades dos objetos daquela classe
- São declaradas como variáveis dentro do corpo da classe, podendo ser tipos primitivos ou objetos de outras classes
- Por padrão recomenda-se que tenha um encapsulamento com visibilidade privada

```
public class Conta{
    private int número;
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Métodos da classe

- São métodos definidos dentro da própria classe que manipulam os atributos definidos dentro dela
- São definidos com uma sintaxe semelhante a uma função em C/C++ “tipo_retorno nome (parametros)”
- Uma função definida em uma classe deve ser chamada a partir de um objeto daquela classe
- São deixados públicos para permitir o acesso a partir de outras classes

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Métodos construtores

- Todo atributo de um objeto de uma classe deve ser inicializado
- Isso é feito no momento da criação de cada objeto de acordo com uma função especial chamada construtor
- Define a maneira como se cria um objeto
- Características de um construtor:
  - Não tem tipo de retorno especificado
  - Tem o mesmo nome da classe
  - Normalmente é público

  public Conta ( ) { … }

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Criando construtores

- Se nenhum construtor é definido, o próprio Java cria um construtor vazio (implicitamente) onde os objetos são atibuídos a null, tipos primitivos são atribuídos a zero ou falso
  - Conta conta1 = new Conta ( );
- Se algum construtor é definido, os objetos tem que ser criados de acordo com seus parâmetros
  - public Conta (String proprietario) { … }
  - Conta conta1 = new Conta ( ); // Erro
  - Conta conta1 = new Conta ( “Sebastião” );

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Palavra reservada this

- É uma referência ao próprio objeto que pode ser usada dentro do código da classe
- Pode ser usado para distinguir parâmetros de variáveis da classe

```
public class Conta {
    private String proprietário;
    private double saldo;
    public Conta (String proprietário) {
        this.proprietário = proprietário;
        this.saldo = 0.0;
    }
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Sobrecarga de construtores

- É possível criar mais de um construtor para a mesma classe, mudando os parâmetros

```
public Conta (String proprietário, double saldo) {
    this.proprietário = proprietário;
    this.saldo = saldo;
}
```

- É possível chamar um construtor a partir de outro usando a palavra this

```
public Conta (String proprietário) {
    this(proprietário, 0.0);
}
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Métodos acessores get/set

- Definem o acesso aos atributos privados da classe
- Acompanham o nome do atributo para o qual estão dando acesso
  - get: obtém o valor do atributo retornando-o

```
public String getProprietário ( ) {
    return this.proprietário;
}
```

  - set: modifica o valor do atributo e retorna void

```
public void setSaldo (double saldo){
    this.saldo = saldo;
}
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Comportamento de um objeto

- A chamada de métodos de classe a partir dos objetos mostra o seu comportamento
- Uma mesma chamada a um mesmo método realizado a partir de objetos diferentes pode dar resultados diferentes

```
Conta c1 = new Conta (“Sebastião”);
Conta c2 = new Conta (“Emidio”, 1000.0);
System.out.println(c1.getSaldo ( ) );
System.out.println(c2.getSaldo ( ) );
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Método toString

- Usado quando queremos retornar uma string representando um objeto da classe
- Chamado implicitamente quando se quer exibir um objeto para exibí-lo como String

```
public String toString () { … }

Conta c1 = new Conta( “Sebastiao”);
System.out.println ( c1 );
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.

# Outros métodos

- Além de construtores e acessores, é possível implementar quantos métodos se quiser
- Geralmente usam verbos que indicam a ação a ser realizada, get/set, ou is para booleanos

```
public void fazDepósito (double valor ) {
    this.saldo += valor;   }
public boolean realizaSaque (double valor){
    if ( this.saldo >= valor){
        this.saldo -= valor;
        return true;
    } else return false;
```

© Copyright 1992–2015 by Pearson Education, Inc.
All Rights Reserved.