Pacotes e Interfaces

Pacotes e Interfaces

- *Pacotes* são grupos de classe relacionadas. Os pacotes ajudam a organizar o código e fornecem outra camada de encapsulamento.
- *Interface* define um conjunto de métodos que será implementado por uma classe.
- A interface propriamente dita não implementa método algum, é uma estrutura puramente lógica.
- Os pacotes e as interfaces proporcionam um controle maior sobre a organização do programa.

Pacotes

- Em programação, com frequência é útil agurpar partes relacionadas de um programa.
- Em Java, isso é feito com o uso de um pacote. O pacote serve a duas finalidades.
- Em primeiro lugar, fornece um mecanismo pelo qual parte relacionadas de um programa podem ser acessadas com o uso do nome de seu pacote. Logo, um pacote fornece uma maneira de nomear um conjunto de classes.

Pacotes

- Em segundo lugar, o pacote participa do mecanismo de controle de acesso Java. Classes definidas dentro de um pacote podem se tornar privadas desse pacote sem poder ser acessadas por códigos de fora dele.
- Portanto, o pacote fornece um meio pelo qual classes podem ser encapsuladas. Um espaço de nome define uma região declarativa.
- Em Java, duas classe não podem usar nomes iguais do mesmo espaço de nome.
- Todos os exemplos que vimos, usaram o espaço de nome padrão ou global.

Pacotes

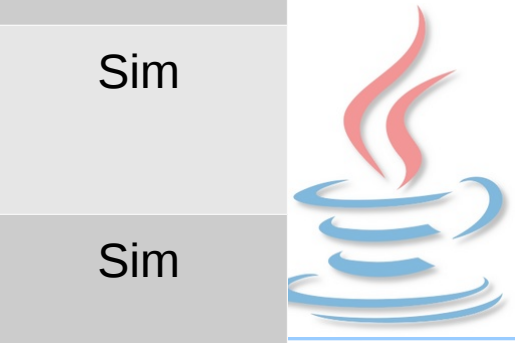
- Embora isso seja adequado para exemplos de programas curtos, torna-se um problema à medida que os programas crescem e o espaço de nome fica abarrotado.
- Em programas grandes, encontrar nomes exclusivos para cada classe pode ser difícil.
- Já que geralmente um pacote contém classes relacionadas, Java define direitos de acesso especiais para os códigos do pacote. Em um pacote, podemos definir um código acessado por outro código do mesmo pacote.

Definindo um Pacote

- Todas as classes em Java pertencem a algum pacote. Quando não é especificada uma instrução **package**, o pacote padrão é usado.
- O pacote padrão não tem nome, o que o torna transparente
- Por isso, até agora você não precisou se preocupar com os pacotes. Embora o pacote padrão seja adequado para exemplos de programas curtos, não é apropriado para aplicativos reais.

Acesso a membros

	Membro privado	Membro padrão	Membro protegido	Membro público
Visível dentro da mesma classe	Sim	Sim	Sim	Sim
Visível dentro do mesmo pacote pela subclasse	Não	Sim	Sim	Sim
Visível dentro do mesmo pacote por não subclasse	Não	Sim	Sim	Sim
Visível dentro de pacote diferente pela subclasse	Não	Não	Sim	Sim
Visível dentro de pacote diferente por não subclasse	Não	Não	Não	Sim



Interface

- Na programação orientada a objetos, às vezes é útil definir o que uma classe deve fazer, mas não como ela fará.
- Já vimos um exemplo disso: o método abstrato. Um método abstrato define a assinatura de um método, mas não fornece implementação.
- Uma subclasse deve fornecer sua própria implementação de cada método abstrato definido por sua superclasse.
- Portanto, um método abstrato especifica a *interface* do método, mas não a implementação.

Interface

- As interfaces são sintaticamente semelhantes às classe abstratas. Porém, na interface, os métodos não podem incluir um corpo.
- Isto é, uma interface não fornece implementação, ela especifica o que deve ser feito, mas não como.
- Para implementar uma interface, a classe deve fornecer corpos(implementações) para os métodos descritos nela.
- Cada classe é livre para determinar os detalhes de sua própria implementação.

Implementando Interfaces

- Quando uma **interface** tiver sido definida, uma ou mais classe poderão implementá-la.
- Para implementar uma interface, inclua a cláusula **implements** em uma definição de classe e então crie os métodos definidos pela interface.
- A forma geral de representação é:

class *nomeclasse* extends *superclasse* implements *interface{*

//corpo-classe

}

Variáveis em Interfaces

- Podemos declarar variáveis em um interface, mas elas serão implicitamente **public, static** e **final.**
- A primeira vista, você poderia pensar que, para essas variáveis, haveria uma aplicação muito limitada, mas é o contrário que acontece.
- Já que um programa grande costuma ser mantido em muitos arquivos-fonte separados, é preciso haver uma maneira conveniente de disponibilizar essas constantes para cada arquivo.
- Exercício da pg. 291, mostra como estender interfaces.