Universidade de São Paulo
Brasil

Aula 05 – Programando com Classe e Pacotes

MAC0321 - Laboratório de Programação Orientada a Objetos

Professor: Marcelo Finger (mfinger@ime.usp.br)

Departamento de Ciência da Computação
Instituto de Matemática e Estatística

Tópicos

1. Controle de Acesso
2. Pacotes
3. Final
4. Interfaces
5. Interfaces Funcionais

Controle de Acesso

Atributos para membros e métodos

Existem quatro tipos de controle de acesso

(mais restrito)

- private - acesso restrito aos métodos da classe
- protected (herdeiros também têm acesso)
- friendly (default) - visível para todo o package
- public - visível a todos

(menos restrito)

Pacotes de Java (Packages)

O que é um package ?

Assim como bibliotecas são conjuntos de funções, packages são conjuntos de classes

Ex: o package java.math contém as classes

```
java.math.BigDecimal
java.math.BigInteger
```

Logo, após o comando import java.math.*; podemos nos referir às classes BigDecimal e BigInteger.

Caso contrário, para criar um objeto deste tipo:

```
java.math.BigInteger bi = new java.math.Biginteger();
```

Como criar um package ?

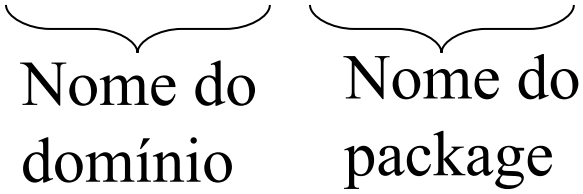
- Deve se colocar como a primeira linha não comentada o comando:

```
package nomedopackage; // em letras minusculas
```

- Escolhendo o nome do package:

  usar, se possível, o nome do domínio:

```
package br.usp.ime.mac0321Aula05;
```



Onde ficam os packages?

- Um package pode ser formado por vários (muitos) arquivos, cada um com uma classe pública
- Os arquivos de um package podem estar em:
  - um mesmo diretório
  - ou em um único arquivo compactado (.jar)

Como usar os packages? Método Preferido

- **<sdkTool> -classpath classpath1:classpath2..**
- SdkTool: java, javac, javadoc, ant, etc
- *Classpath: Caminhos para arquivos JAR, zip ou classe. Cada caminho deve terminar com um nome de arquivo ou diretório:*
  - *Para um arquivo JAR ou zip, o caminho termina com o nome do arquivo zip ou JAR*
  - *Para arquivos de classe em um pacote sem nome, o caminho da classe termina com o diretório que contém os arquivos de classe.*
  - *Para arquivos de classe em um pacote nomeado, o caminho da classe termina com o diretório que contém o pacote raiz, que é o primeiro pacote no nome completo do pacote.*

Como usar os packages? No Eclipse

- *Por padrão, a pasta {project} / src é a pasta classpath*
- *Para adicionar arquivos ao classpath:*
  - *Clique com o botão direito em um projeto do eclipse*
  - *Selecione Propriedades*
  - *Selecione o link Caminho de Construção Java/Java Build Path*
  - *Selecione a guia Origem/Source.*

Como usar os packages ? Outra Alternativa

- A localização dos arquivos package depende da variável de ambiente CLASSPATH, que contém os caminhos separados por ';', por exemplo:

```
.;~mfinger/minhasclasses;/usr/local/jdk1.17/lib/tools.zip;
/usr/local/jdk1.17/jre/lib/rt.jar
```

Como usar os packages ?

- Os arquivos .jar devem estar com o caminho completo no CLASSPATH
- A partir dos caminhos no CLASSPATH as classes são buscadas
- Quando “import marcelo.utils;” é encontrado as classes serão buscadas nos subdiretórios marcelo/utils

Ex: `set CLASSPATH=.;/usr/java`

A JVM vai procurar também os objetos no subdiretório `marcelo/utils` do diretório atual e no diretório `/usr/java/marcelo/utils`

Observações sobre Packages

- Caso existam duas classes com o mesmo nome, sempre use o nome completo
  - Ex: `java.math.BigInteger` ao invés de `BigInteger`
- Classes no mesmo diretório pertencem ao mesmo package
- Crie um package com as classes que podem ser úteis

Dicas

- Os membros devem ser, na medida do possível, `private`;
- Se necessário crie métodos com nomes get/set para consultar, alterar membros;
- Métodos que também podem vir a ser modificados devem ser `private`;

Lembrete com relação às classes

- Criação de objeto com construtor
  - Pode ser privado em ocasiões especiais
- Quando não há mais referência a um objeto
  - Ele pode ser pego pelo coletor de lixo
    - Mas, não se sabe quando...
    - Pode se solicitar isso com `System.gc()`
- Referências para objetos são apontadores

Java Final

Java final

- Variável final: definição de constantes. Uma vez inicializada ela não pode ser alterada
- Classe final: não pode ser estendida (herdada)
- Método final: não pode ser sobrescrito

Java Interfaces
(Classes totalmente abstratas)

Interfaces

- Motivação: Contrato
  - Dizer que serviços a classe deve prover
    - Métodos e suas assinaturas
  - Não se especifica o comportamento
    - Nem nos possíveis atributos
    - A não ser os estáticos, que serão final.

Forma genérica

```
/* File name : NameOfInterface.java */
import java.lang.*;
//Any number of import statements

public interface NameOfInterface
{
   //Any number of final, static fields
   //Any number of abstract method declarations
}
```

Exemplo

```
interface Animal {
   public void eat();
   public void travel();
}
```

```
public class UmAnimal implements Animal{
   public void eat(){
      System.out.println("Come");
   }
   public void travel(){
      System.out.println("Viaje");
   }
   public int noOfLegs(){
      return 0;
   }
}
```

Várias interfaces

- Uma classe pode implementar várias interfaces
  - Desde que todos os métodos sejam implementados
    - Caso contrário erro de compilação
- Se uma classe A implementa as interfaces I1 e I2
  - Os objetos de A podem ser vistos como:
    - Da classe A;
    - Como objetos do tipo I1
    - Como objetos do tipo I2
  - Um exemplo prático

Interfaces Funcionais

- Contém exatamente um método abstrato.
- Pode conter métodos estáticos ou default, mas só um método abstrato.
- Pode declarar métodos de classe Object (ex. **int** hashCode(); **String** toString(); **boolean** equals(**Object** obj);
- Single Abstract Method Interfaces (SAM) Interfaces.
- Iniciou em Java 8, ajuda a alcançar uma abordagem de **programação funcional**.

Interfaces Funcionais

- Existem muitas interfaces funcionais pré-definidas.
- Antes de definir uma nova, verifique se não existe já uma interface funcional pré-definida.
- Ver exemplo no programa anexo

Interfaces não aceitam defaults

- Numa interface, não é possível definir um “comportamento padrão” para algum dos métodos definidos
- Para prover estes “métodos com comportamento padrão” existem as **classes abstratas**
- Classes abstratas serão tratadas na próxima aula

Lista de exercícios

No computador com o Eclipse

Entrega até o final do dia

MAC321
Lab POO

- Professor: Marcelo Finger

  E-mail: mfinger@ime.usp.br