Introdução a Programação – IF669
http://www.cin.ufpe.br/~if669

Garbage Collection e Pacotes

AULA 10

Ricardo Massa F. Lima
rmfl@cin.ufpe.br

Sérgio C. B. Soares
scbs@cin.ufpe.br

Centro de Informática — Universidade Federal de Pernambuco — CIn.ufpe.br

Até aqui . . .

- Conceitos de programação
  - tipos, entrada e saída de dados, operadores, comandos)
- Introdução a orientação a objetos
- Recursão e passagem de parâmetros
- Array

Já sabemos como criar objetos, mas

como posso destruir objetos?

você não destrói!

Java cuida disso para você!

Concentre-se em coisas mais importantes, como o algoritmo que está implementando!

Garbage Collection

A técnica que Java usa para destruir objetos quando eles não são mais necessários, chama-se Garbage Collection

Usada em linguagens como Lisp, Haskell, etc

O interpretador Java sabe quais objetos foram alocados

Ele também sabe quais variáveis se referem a quais objetos e quais objetos se referenciam a quais objetos

Assim, o interpretador pode descobrir quando um objeto alocado não for mais referenciado por outros objetos ou variáveis e , portanto, pode ser destruído sem perigo

Java faz isso AUTOMATICAMENTE

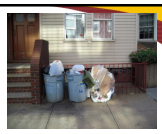
Garbage Collection

Embora os GCs possam ser bastante eficientes, nenhum deles é mais rápido que um código bem escrito para desalocar objetos de maneira explícita

Por outro lado...

- GC torna a programação mais simples e os programas menos propensos a erros
- Para muitos programas do mundo real, desenvolvimento rápido, sem bugs e fáceis de manter são mais importantes do que a eficiência

Puting the trash out!



Já usou o objeto e não precisa mais dele?

Ignore-o!!!

```
String processString(String s){

 // Create a StringBuffer object to process the string in.
 StringBuffer b = new StringBuffer(s);

 // Process it somehow…
 // return it as a String. Just forget about the StringBuffer object:
 // it will be automatically garbage collected.
 return b.toString();
}
```

Puting the trash out!

Tentando forçar o "abandono" de um objeto!

```
public static void main(String argv[]) {
    int big_array[] = new int[100000];

    // Do some computations with big_array and get a result.
    int result = compute(big_array);

    // We no longer need big_array.  It will get garbage collected when
    // there are no more references.  Since big_array is a local variable,
    // it refers to the array until this method returns.  But this
    // method doesn't return.  So we've got to get rid of the reference
    // ourselves, just to help out the garbage collector.
    big_array = null;

    // Loop forever, handling the user's input.
    for(;;) handle_input();
}
```

Em Java o Garbage Collector é quem decide quando executar, por isso não temos certeza de quando os objetos serão destruídos, não da para forçar!

Podemos sinalizar para se e quando o GC executar, o objeto seja destruído!

O método finalize

É o oposto de um construtor

O GC não consegue liberar determinados recursos de um objeto:

- descritor de arquivos - fechar arquivo
- sockets - terminar conexão

Nesses casos, você deve escrever um método finalize para tratar desses casos

Pode haver apenas um `finalize` por classe

```
// Checks the file descriptor first to make sure
// it is not already closed.
protected void finalize() throws IOException {
    if (fd != null) close();
}
```

o método `finalize`

1. `finalize` é chamado antes que o sistema colete o objeto
2. interpretador Java pode terminar sem que todos os objetos tenham sido eliminados. Nesse caso, os recursos pendentes são removidos pelo sistema operacional
3. Java não oferece garantias sobre quando o GC irá executar ou sobre a ordem de coleta dos objetos - o mesmo ocorre com a ordem de execução dos finalize pendentes

PACOTES

Centro de Informática — Universidade Federal de Pernambuco — CIn.ufpe.br

Objetivo

Depois desta aula você será capaz de desenvolver sistemas mais **reusáveis** e **extensíveis**, organizando as classes do sistema em "módulos" que podem ser analisados, reusados e modificados isoladamente ou com o auxílio de outros poucos "módulos".

Tipos de Módulos em Java

- Classes
  - agrupam definições de métodos, atributos, inicializadores, etc.
  - definem tipos
- Pacotes
  - agrupam definições de **classes** relacionadas
  - estruturam sistemas de grande porte, facilitando a localização das classes
  - oferece um nível mais alto de abstração

Pacotes em Java

- Um pacote é uma coleção de nomes de classes
- Um pacote serve para relacionar classes e definir um namespace para as classes do pacote
- Pacotes da plataforma Java:
  - Começam o a palavra java, javax e orgm.omg
  - javax - extensões padronizadas da plataforma java
  - java.lang – classes fundamentais da linguagem
  - java.util – classes utilitárias
  - java.io – classes para entrada e saída
  - java.net – classes para gerenciar rede

`java.util`

Arrays	*This class contains various methods for manipulating arrays (such as sorting and searching).*
Calendar	*Calendar is an abstract base class for converting between a Date object and a set of integer fields such as YEAR, MONTH, DAY, HOUR, and so on.*
Hashtable	*This class implements a hashtable, which maps keys to values.*
Random	*An instance of this class is used to generate a stream of pseudorandom numbers.*
Stack	*The Stack class represents a last-in-first-out (LIFO) stack of objects.*
Vector	*The Vector class implements a growable array of objects.*
...	...

Pacotes em Java

- Exemplo
  - `java.util.zip` contém classes para trabalhar com arquivos comprimidos
- Toda classe tem um nome simples e um nome completamente qualificado, que inclui o nome do pacote
  - Ex: `String` (nome simples)

    `java.lang.String` (nome qualificado)

Pacotes e Subdiretórios

- Subdiretórios (subpastas) não correspondem a "subpacotes", são pacotes como outros quaisquer
- Por exemplo, não existe nenhuma relação, além de lógica, entre `exemplos` e `exemplos.banco`:

```
package exemplos;
public class /*...*/
```

```
package exemplos.banco;
public class /*...*/
```

Definindo pacotes

- Cada pacote é associado a um diretório/pasta:
  - os arquivos .class das classes do pacote são colocados neste diretório
  - é recomendável que o código fonte das classes do pacote também esteja neste diretório
- O pacote `exemplos.banco` deve estar em

  `c:\ricardo\exemplos\banco`
- assumindo que o compilador Java foi informado para procurar classes em

  `c:\ricardo\`
- Este caminho é o classpath!

Pacotes e Visibilidade de Declarações

- `public`
  - atributos, métodos, inicializadores e classes
  - declaração pode ser utilizada (é visível) em qualquer lugar
- `private`
  - atributos, métodos e inicializadores
  - declaração só pode ser utilizada na classe onde ela é introduzida

Pacotes e e Visibilidade de Declarações

- `protected`
  - atributos, métodos e inicializadores
  - declaração só pode ser utilizada no pacote onde ela é introduzida, ou nas subclasses da classe onde ela é introduzida
- Ausência de modificador
  - atributos, métodos, inicializadores e classes
  - declaração só pode ser utilizada no pacote onde ela é introduzida

Atenção

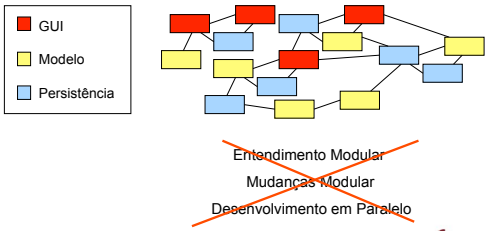
Se duas classes com nomes idênticos, pertencentes a pacotes diferentes, forem referenciadas, deve-se usar o nome completamente qualificado para resolver o conflito

Pacotes em Java

- Uma das funções importantes dos pacotes é prevenir "colisão" de nomes
- Ex: `java.util.List` e `java.awt.List` são classes distintas
- Para que o esquema funcione, os nomes dos pacotes têm que ser diferentes

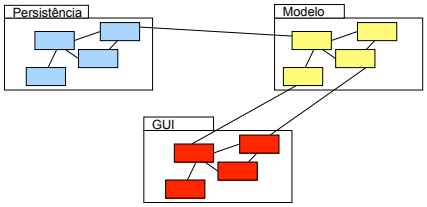
Dicas de bons sistemas O.O.

- Modularizando o sistema



Dicas de bons sistemas O.O.

- Modularidade OK!



Pacotes e Importação

- Cada classe na API Java está em um pacote específico que contém um grupo de classes relacionadas
- Para usar classes de uma API Java, o operador `import` deve ser usado
- Para definir pacotes para classes, o operador `package` deve ser usado

Pacotes

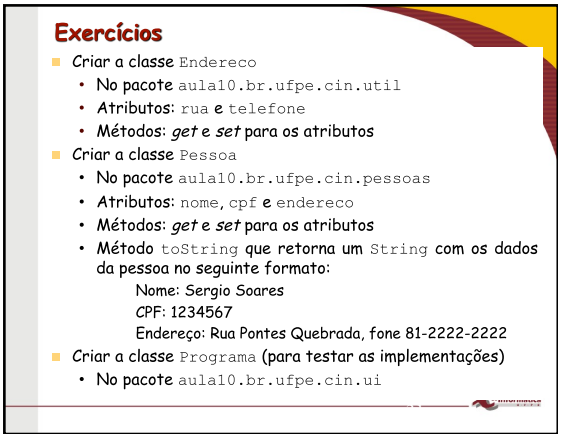
- Um padrão de Nomenclatura de pacotes
- Site da organização invertido

http://www.cin.ufpe.br/

Pacote **br.ufpe.cin**

Implementação em Java

```
package br.ufpe.cin;

import br.ufpe.cin.util.Endereco;

public class Pessoa {

        private Endereco endereco;
        private String nome;

        public Pessoa(String nome) {
                this.nome = nome;
        }

}
```


Exercícios
Criar a classe Endereco
No pacote aula10.br.ufpe.cin.util
Atributos: rua e telefone
Métodos: get e set para os atributos
Criar a classe Pessoa
No pacote aula10.br.ufpe.cin.pessoas
Atributos: nome, cpf e endereco
Métodos: get e set para os atributos
Método toString que retorna um String com os dados da pessoa no seguinte formato:
Nome: Sergio Soares
CPF: 1234567
Endereço: Rua Pontes Quebrada, fone 81-2222-2222
Criar a classe Programa (para testar as implementações)
No pacote aula10.br.ufpe.cin.ui