




Laboratório de Engenharia de Software

# INF1636 – PROGRAMAÇÃO ORIENTADA A OBJETOS

Departamento de Informática – PUC-Rio

Ivan Mathias Filho  
[ivan@inf.puc-rio.br](mailto:ivan@inf.puc-rio.br)




Laboratório de Engenharia de Software

## Programa – Capítulo 8

- Entrada e Saída na Console
- A Classe `String`
- Pacotes Java
- Como o class loader Encontra Classes

© LES/PUC-Rio 2

Programa – Capítulo 8




Laboratório de Engenharia de Software

- **Entrada e Saída na Console**
- A Classe `String`
- Pacotes Java
- Como o class loader Encontra Classes

© LES/PUC-Rio

3

Leitura de Dados da Console (1)



Laboratório de Engenharia de Software

- Até a edição JSE5 (JDK 1.5) a linguagem Java não oferecia suporte adequado a leitura de dados do console (teclado);
- Isso foi corrigido com o advento da classe `java.util.Scanner`;
- Para que se possa ler dados do console deve-se primeiro criar uma instância de `Scanner` e associá-la ao *input stream* padrão (`System.in`):


```
Scanner ent=new Scanner(System.in);
```

© LES/PUC-Rio

4

Laboratório de Engenharia de Software

## Leitura de Dados da Console (2)



- Após a instanciação de `Scanner` pode-se usar vários métodos desta classe para ler dados do console;
- Por exemplo, o método `nextLine` possibilita a leitura de uma linha do teclado:

```
System.out.println("Informe o nome");
String nome=ent.nextLine();
```


- Para ler uma cadeia de caracteres que não contenha espaços pode-se usar o método `next`:

```
System.out.println("Informe o codigo");
String cod=ent.next();
```

© LES/PUC-Rio
5

Laboratório de Engenharia de Software

## Exemplo (1)



```
import java.util.*;

public class Main {
    public static void main(String[] args) {

        Disciplina[] lst=new Disciplina[100];
        int i=0;
        Scanner ent=new Scanner(System.in);

        System.out.println("Continua? 0-termina");
        byte cont=ent.nextByte();
        while(cont!=0) {
            ent.nextLine();
            System.out.println("Informe o codigo");
            String cod=ent.nextLine();
            System.out.println("Informe o nome");
            String nome=ent.nextLine();
            System.out.println("Informe o numero de creditos");
            int numCred=ent.nextInt();
```

© LES/PUC-Rio
6

## Exemplo (2)



```

System.out.println("Informe o numero de horas");
int umHoras=ent.nextInt();

lst[i]=new Disciplina();
lst[i].cod=cod;
lst[i].nome=nome;
lst[i].numCred=numCred;
lst[i].numHoras=numHoras;
i++;
System.out.println("Continua? 0-termina");
cont=ent.nextByte();
}

for(int j=0;j<i;j++)
    System.out.println(lst[j].cod+" "+lst[j].nome+" "+
        lst[j].numCred+" "+lst[j].numHoras);
}

```

© LES/PUC-Rio

7

## Exibição de Dados da Console (1)




- Até agora apenas o método `System.out.println` vem sendo utilizado para exibir dados no console (monitor);
- Esse método exibe os dados e salta para linha seguinte;
- Entretanto, existem várias sobrecargas do método `println`:
  - `void println()` – salta para linha seguinte
  - `void println(char x)` – exibe um caractere e salta para linha seguinte
  - `void println(char[] x)` – exibe um array de caracteres e salta para linha seguinte

© LES/PUC-Rio

8

## Exibição de Dados da Console (2)




Laboratório de Engenharia de Software

- Continuação:
  - `void println(double x)` – exibe um double e salta para linha seguinte
  - `void println(float x)` – exibe um float e salta para linha seguinte
  - `void println(int x)` – exibe um int e salta para linha seguinte
  - `void println(long x)` – exibe um long e salta para linha seguinte
  - `void println(String x)` – exibe uma String e salta para linha seguinte

© LES/PUC-Rio 9

## Exibição de Dados da Console (3)



Laboratório de Engenharia de Software

- Para cada método `println` existe um método `print` correspondente;
- De modo diferente do método `println`, o método `print` não encerra uma linha;
- Dessa forma, não há salto de linha quando dados são exibidos com o método `print`.

© LES/PUC-Rio 10

## Exibição de Dados da Console (4)



- A classe `PrintStream` (`System.out`) possui um método muito semelhante ao existente na biblioteca `stdio` da linguagem C;
- O método `printf` permite a exibição de dados formatados no console do sistema;
- A partir do exemplo a seguir podemos ver que seu uso é muito semelhante ao uso da função `printf` da linguagem C.


## Exemplo



```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int mat;
        double n1,n2,n3,media;
        Scanner ent=new Scanner(System.in);
        System.out.println("Informe a matricula");
        mat=ent.nextInt();
        while(mat!=0) {
            System.out.println("Informe a nota da p1");
            n1=ent.nextDouble();
            System.out.println("Informe a nota da p2");
            n2=ent.nextDouble();
            System.out.println("Informe a nota da p3");
            n3=ent.nextDouble();
            media=(n1+2*n2+3*n3)/6.0;
            System.out.printf("Matr - %7d Media - %5.2f \n",mat,media);
            System.out.println("Informe a matricula");
            mat=ent.nextInt();
        }
    }
}
```

**Programa – Capítulo 8**




Laboratório de Engenharia de Software

- Entrada e Saída na Console
- **A Classe String**
- Pacotes Java
- Como o class loader Encontra Classes

© LES/PUC-Rio

13

**Strings**



Laboratório de Engenharia de Software

- Strings são sequências de caracteres;
- Na linguagem Java, strings são instâncias da classe `String`;
- O modo mais direto de criar uma string é:


```
String s = "Departamento de Informática";
```

- Neste caso, "Departamento de Informática" é uma string literal – uma série de caracteres envolvidas por aspas duplas;
- Sempre que o compilador encontra uma string literal no código de um programa ele cria um objeto da classe `String` contendo o valor do literal.

© LES/PUC-Rio

14

## Strings - Construtores




- Como qualquer outro objeto, uma string pode ser criada pelo operador `new`;
- A classe `String` possui 11 construtores, que permitem fornecer o valor inicial da string usando diferentes fontes, tais como *arrays* de caracteres:

```
char[] str={'P','U','C','-','R','I','O'};  
String s = new String(str);
```

Laboratório de Engenharia de Software

© LES/PUC-Rio 15

## Strings - Imutabilidade



- Uma string é imutável, logo, uma vez criado, um objeto da classe `String` não pode ser modificado;
- A classe `String` possui vários métodos que parecem alterar o conteúdo de uma string;
- Dado que as strings são imutáveis, o que esses métodos na verdade fazem é criar e retornar uma nova string que contém o resultado da operação.

Laboratório de Engenharia de Software

© LES/PUC-Rio 16



## Comprimento de uma String



- Os métodos usados para se obter os dados encapsulados por um objeto são chamados de **accessors**;
- Um accessor que é muito usado com strings é o método `length()`;
- Ele retorna o número de caracteres que formam uma string;
- Após a execução do trecho de código a seguir o valor **7** será exibido no console do sistema.

```
String s1="PUC-RIO";

System.out.println(s1.length());
```

© LES/PUC-Rio

17

## Concatenação de Strings



- A classe `String` possui um método para concatenar duas strings:

```
String concat(String str);
```

- O método `concat()` retorna uma nova string, que é a concatenação da string sobre a qual ele é aplicado com a string relativa ao parâmetro `str`.

```
String s1="PUC-";
String s2="RIO";


System.out.println(s1.concat(s2));
```

© LES/PUC-Rio

18

Laboratório de Engenharia de Software

## Concatenação de Strings Literais



- Pode-se usar o método `concat()` para concatenar literais.

```
String s1="PUC-".concat("RIO");

System.out.println(s1);
```

- Entretanto, o uso do operador `+` é mais comum na concatenação de literais.


```
String s1="PUC-"+"RIO";

System.out.println(s1);
```

© LES/PUC-Rio
19

Laboratório de Engenharia de Software

## Formatação de Strings



- A classe `String` possui um método estático, chamado `format()`, que permite formatar uma string;
- Ele pode ser usado da seguinte maneira:

```
String s1;

s1=String.format("N.Alunos: %d\nMédia:%5.2f",25,7.45);

System.out.println(s1);
```


- A execução do trecho de código acima irá exibir o seguinte texto no console do sistema:

```
N.Alunos: 25
Média: 7,45
```

© LES/PUC-Rio
20

Laboratório de Engenharia de Software

## Conversão de Strings para Números



- Cada classe que empacota um tipo primitivo numérico (Byte, Integer, Double, Float, Long e Short) fornece um método, chamado `valueOf()`, que converte uma string para um objeto da respectiva classe;
- Esses métodos irão levantar uma exceção do tipo `NumberFormatException` caso a string contenha caracteres inválidos:

O método `parseDouble()` retorna um valor do tipo `double`.


```
double d=Double.parseDouble("++25.54");
```

```
Double d=0.0;
try {
    d=Double.valueOf("++25.54");
}
catch (NumberFormatException e) {
    System.out.println("Erro na conversão");
}
```

© LES/PUC-Rio
21

Laboratório de Engenharia de Software

## Conversão de Números para Strings (1)



- Existem várias maneiras de converter números para strings. O exemplo a seguir mostra duas delas:

```
String s1,s2;


s1=String.valueOf(34.87);
s2=""+34.87;

System.out.println(s1);
System.out.println(s2);
```

© LES/PUC-Rio
22

Laboratório de Engenharia de Software

## Conversão de Números para Strings (2)




- Cada wrapper de tipo primitivo possui um método estático, chamado `toString()`, que converte um parâmetro do seu tipo primitivo para uma string:

```
String s1,s2;  
  
s1=Integer.toString(1234);  
s2=Double.toString(34.87);  
  
System.out.println(s1);  
System.out.println(s2);
```

© LES/PUC-Rio 23

Laboratório de Engenharia de Software

## Manipulação dos Caracteres de uma String




- Pode-se recuperar um caractere de uma determinada posição de uma string por meio do método `charAt()`;
- O índice do primeiro caractere é 0, enquanto o índice do último caractere é `length()-1`;
- O exemplo a seguir exibe o primeiro e o último caracteres de uma string:

```
String s1="PUC-RIO";  
  
System.out.println(s1.charAt(0));  
System.out.println(s1.charAt(s1.length()-1));
```

© LES/PUC-Rio 24

## Substrings




Laboratório de Engenharia de Software

- A classe `String` possui dois métodos para a manipulação de substrings:
  - `String substring(int beginIndex, int endIndex)` – o primeiro argumento é o índice do primeiro caractere, enquanto o segundo argumento é o índice do último caractere + 1.
  - `String substring(int beginIndex)` – o parâmetro define o índice do primeiro caractere. A string retornada se estende até o último caractere da string original.

© LES/PUC-Rio
25

## Substrings - Exemplo



Laboratório de Engenharia de Software

- Seja o trecho código a seguir:
 

```
String s1="Departamento de Informática";
System.out.println(s1.substring(0,12));
System.out.println(s1.substring(16));
```
- A sua execução irá exibir o seguinte na console do sistema:
 

```
Departamento
Informática
```

© LES/PUC-Rio
26

## Substrings – Funções de Busca



- Exemplos de funções de busca da classe String:
  - `int indexOf(String str)` – retorna o índice da primeira ocorrência da substring passada como parâmetro.
  - `int lastIndexOf(String str)` – retorna o índice da ocorrência mais à esquerda da substring passada como parâmetro.
  - `int indexOf(int ch)` – retorna o índice da primeira ocorrência do caractere passado como parâmetro.
  - `int lastIndexOf(int ch)` – retorna o índice da última ocorrência do caractere passado como parâmetro.

## Funções de Busca - Exemplo



- Seja o trecho código a seguir:

```
String s1="Informática Informática";


System.out.println(s1.indexOf("for"));
System.out.println(s1.lastIndexOf("for"));
System.out.println(s1.indexOf('I'));
System.out.println(s1.lastIndexOf('I'));
```

- A sua execução irá exibir o seguinte na console do sistema:

```
2
14
0
12
```

Laboratório de Engenharia de Software

## Comparação de Strings (1)




- Exemplos de funções de comparação da classe String:
  - `boolean endsWith(String suffix)` – retorna `true` se a string termina com a string passada como argumento.
  - `boolean startsWith(String prefix)` – retorna `true` se a string começa com a string passada como argumento.
  - `int compareTo(String s)` – compara duas strings lexicograficamente. Retorna um inteiro que indica se a string é maior (retorno `> 0`), igual (retorno `= 0`) ou menor (retorno `< 0`) que a string passada como argumento.

© LES/PUC-Rio

29

Laboratório de Engenharia de Software

## Comparação de Strings (2)




- `int compareToIgnoreCase(String str)` – compara duas strings lexicograficamente, ignorando diferenças entre caracteres maiúsculos e minúsculos. Retorna um inteiro que indica se a string é maior (retorno `> 0`), igual (retorno `= 0`) ou menor (retorno `< 0`) que a string passada como argumento.
- `boolean equals(Object anObject)` – retorna `true` se, e somente se, a string representa a mesma sequência de caracteres que o objeto passado como argumento.
- `boolean equalsIgnoreCase(String s)` – retorna `true` se, e somente se, a string representa a mesma sequência de caracteres que o objeto passado como argumento, ignorando diferenças entre caracteres maiúsculos e minúsculos.

© LES/PUC-Rio

30

Laboratório de Engenharia de Software

## Comparação de Strings - Exemplo



- Seja o trecho código a seguir:

```
String s1="Informática";

System.out.println(s1.startsWith("Info"));
System.out.println(s1.endsWith("ica"));
System.out.println(s1.compareToIgnoreCase("INFORMÁTICA"));
```


- A sua execução irá exibir o seguinte na console do sistema:

```
true
true
0
```

© LES/PUC-Rio31

Laboratório de Engenharia de Software


## Programa – Capítulo 8




- Entrada e Saída na Console
- A Classe String
- Pacotes Java**
- Como o class loader Encontra Classes

© LES/PUC-Rio



Laboratório de Engenharia de Software	<h2>Pacotes Java</h2>	
	<ul style="list-style-type: none"><li>• Um pacote é uma coleção de classes, interfaces e enumerados relacionados, que formam uma biblioteca;</li><li>• Os arquivos dessa biblioteca são mantidos em um mesmo diretório (pasta);</li><li>• A especificação da linguagem Java (JLS) não determina que a organização dos pacotes tenha de obedecer à estrutura do sistema de arquivos de um sistema operacional;</li><li>• Várias ferramentas usam repositórios, com estrutura própria, para organizar os pacotes Java.</li></ul>	
© LES/PUC-Rio		

Laboratório de Engenharia de Software	<h2>Propósitos dos Pacotes</h2>	
	<ul style="list-style-type: none"><li>• Existem três propósitos básicos no uso de pacotes:<ul style="list-style-type: none"><li>– Organizar espaços de nomes em um programa de modo que não haja colisão de nomes de classes;</li><li>– Organizar espaços de nomes para toda uma organização de modo que não haja colisão de nomes entre diferentes aplicações;</li><li>– Controlar a visibilidade das classes.</li></ul></li></ul>	
© LES/PUC-Rio		

## Regras para a Nomeação de Pacotes



- O nome do pacote deve casar com o nome do diretório (pasta) em que residem o código fonte (.java) e o código binário (.class) de uma classe;
- Por exemplo, o pacote `java.lang` mantém o código binário das suas classes em um diretório em que a última parte do nome é `java\lang` (Windows) ou `java/lang` (Unix, Linux, MacOS e etc);
- Dessa forma, se o nome completo de uma classe é conhecido, a sua localização também o é.

© LES/PUC-Rio

## Empacotando Classes



- Para dizer ao compilador que a classe **EX01** é parte do pacote `a.b.c` deve-se colocar o arquivo `EX01.java` em um diretório cuja última parte seja `a\b\c`;
- O código fonte de **EX01** deverá se parecer com o seguinte:

```
package a.b.c;

public class EX01 {
    public static void main(String[] args) {
        int x=10;
        while(x>0) {
            System.out.println(x);
            x--;
        }
    }
}
```

© LES/PUC-Rio

## O Nome Completo de uma Classe (1)



- O nome do pacote é usado para criar o nome completo de uma classe;
- O uso do nome completo de uma classe evita a colisão de nomes – duas classes com o mesmo nome, mas residentes em pacotes diferentes;
- Todas as ferramentas do compilador Java reconhecem o nome completo de uma classe;
- Dessa forma, para compilar e executar uma classe de nome `a.b.c.EX01`, deve-se utilizar o seguinte comando, a partir do diretório imediatamente acima de **a**:

© LES/PUC-Rio


## O Nome Completo de uma Classe (2)



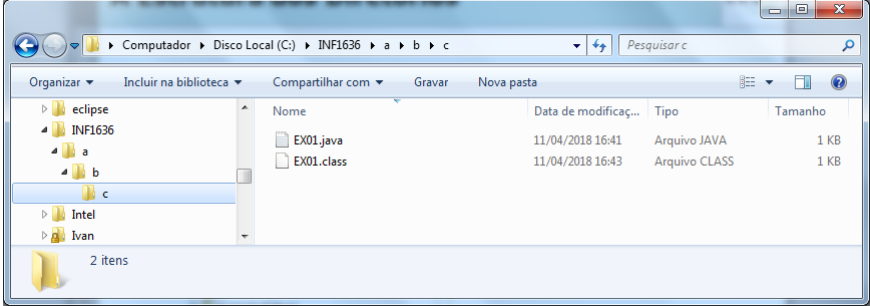
```
C:\windows\system32\cmd.exe
c:\INF1636>javac a/b/c/EX01.java
c:\INF1636>java a.b.c.EX01
c:\INF1636>
```

© LES/PUC-Rio

## A Estrutura dos Diretórios




Laboratório de Engenharia de Software



© LES/PUC-Rio

## Como Criar Nomes Únicos para Pacotes?




Laboratório de Engenharia de Software

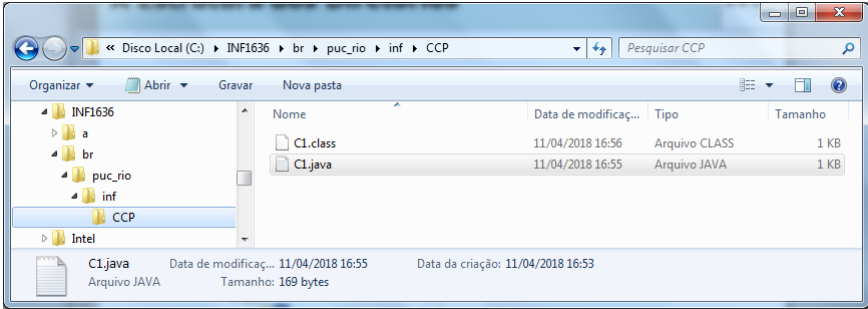
- Como os nomes de domínios são únicos por toda a Internet, pode-se usá-los para criar nomes de pacotes únicos na rede;
- Deve-se inverter os nomes dos domínios para que se possa caminhar do diretório mais geral para o mais específico;
- Dessa forma, todos os pacotes desenvolvidos no Departamento de Informática deveriam ser guardados em subdiretórios de `br\puc_rio\inf` ;
- Por exemplo, uma classe chamada C1, de um projeto chamado CCP, teria o nome completo `br.puc_rio.inf.CCP.C1`

© LES/PUC-Rio

Laboratório de Engenharia de Software

A Estrutura dos Diretórios




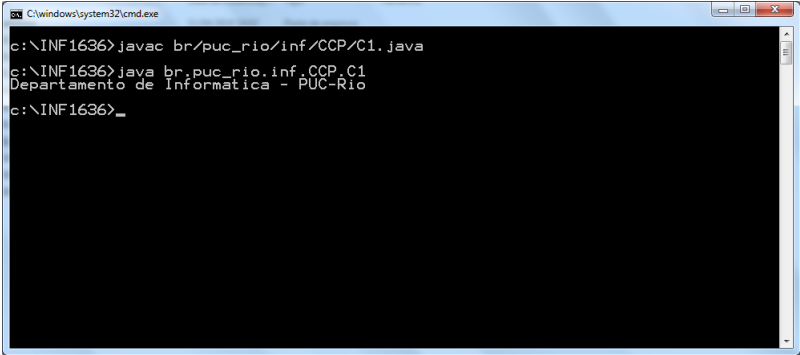


© LES/PUC-Rio

Laboratório de Engenharia de Software


Compilação e Execução da Classe C1





© LES/PUC-Rio

Programa – Capítulo 8




Laboratório de Engenharia de Software

- Entrada e Saída na Console
- A Classe String
- Pacotes Java
- **Como o class loader Encontra Classes**

© LES/PUC-Rio

Como o class loader Encontra Classes



Laboratório de Engenharia de Software

- De modo diferente ao de outras linguagens, Java não liga todo o código binário em um único arquivo de programa;
- Em vez disso, cada classe gera o seu próprio arquivo binário (.class), que é mantido separado dos outros;
- A organização de classes em pacotes impõe certa ordem de busca das classes em tempo de execução;
- O JRE sabe como achar as classes das bibliotecas nativas do Java, mas o **class loader** precisa saber como encontrar as classes específicas de uma aplicação.

© LES/PUC-Rio

## A Variável de Ambiente CLASSPATH (1)



- O JDK utiliza a variável de ambiente `CLASSPATH` para procurar os pacotes (diretórios);
- A variável `CLASSPATH` informa ao **class loader** todos os possíveis pontos de partida (raízes) para a importação de pacotes, na compilação, e para a carga de classes, em tempo de execução;
- O `CLASSPATH` deve conter a lista de todos os diretórios por onde o JDK deve começar a busca por pacotes e classes.

© LES/PUC-Rio

## A Variável de Ambiente CLASSPATH (2)



- No exemplo anterior, a variável `CLASSPATH` deveria conter o nome do diretório pai de `br\puc_rio\inf\CCP` ;
- Isto é, deveria conter o seguinte:

```
C:\windows\system32\cmd.exe
c:\>set classpath
classpath=c:\INF1636
c:\>
```

© LES/PUC-Rio

## Arquivos JAR



- Pode-se, opcionalmente, distribuir uma aplicação por meio de um arquivo `.jar` (JAR);
- Um arquivo JAR (Java Archive) funciona exatamente como um arquivo `.zip`;
- Logo, pode-se criar um arquivo JAR que compacte arquivos usando a mesma estrutura dos pacotes de uma aplicação.

© LES/PUC-Rio

## Criação de um Arquivo JAR



- O exemplo a seguir cria o arquivo `br.puc-rio.inf.CCP.jar` contendo o arquivo binário `br\puc_rio\inf\CCP\C1.class` ;

```
C:\windows\system32\cmd.exe
c:\INF1636>set classpath
classpath=br\puc_rio\inf\CCP
c:\INF1636>jar -cvf br.puc-rio.inf.CCP.jar br\puc_rio\inf\CCP\C1.class
manifesto adicionado
adicionando: br/puc_rio/inf/CCP/C1.class(dentro = 454) (fora= 322)(vazio 29%)
c:\INF1636>
```

© LES/PUC-Rio



## Definição do CLASSPATH



- Para executar a aplicação pode-se criar um arquivo BAT como o mostrado a seguir:

```
teste.bat - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
SET CLASSPATH=C:\INF1636;C:\INF1636\br.puc_rio.inf.CCP.jar
java br.puc_rio.inf.CCP.C1
```

© LES/PUC-Rio

## O Uso do CLASSPATH



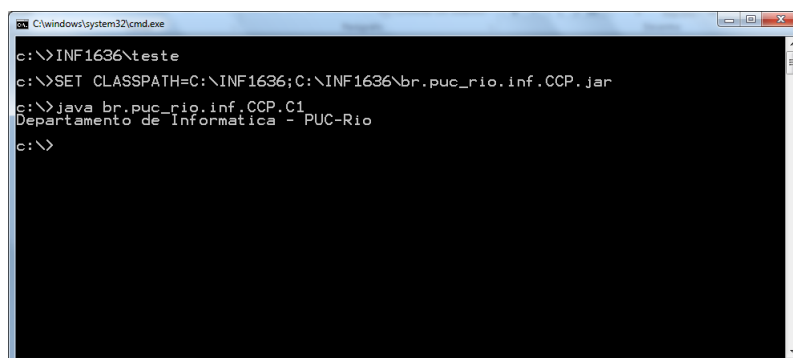
- Nesse ponto é importante entender como o **class loader** utiliza a informação contida no CLASSPATH para encontrar e carregar uma classe:
  - Cada caminho listado no CLASSPATH é concatenado com o nome completo da classe para formar um nome completo de arquivo;
  - No exemplo, serão gerados os seguintes caminhos:
    - C:\INF1636\br\puc\_rio\inf\CCP\C1.class
    - br\puc\_rio\inf\CCP1\C1.class em c:\INF1636\br.puc-rio.inf.CCP.jar

© LES/PUC-Rio

## Execução da Aplicação



- Nesse caso, o **class loader** irá encontrar o arquivo binário no segundo caminho gerado.



```
C:\windows\system32\cmd.exe
c:\>INF1636\teste
c:\>SET CLASSPATH=C:\INF1636;C:\INF1636\br.puc_rio.inf.CCP.jar
c:\>java br.puc_rio.inf.CCP.C1
Departamento de Informatica - PUC-Rio
c:\>
```