FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

Introdução aos Sistemas e Redes de Computadores 20010/11

10ª Folha de Exercícios

Assunto: Ciclo de desenvolvimento de um programa simples em *linguagem Java*; variáveis do ambiente; desenvolvimento de programas que exercitam a API do Java relacionada com ficheiros e directorias.

A. O ciclo de desenvolvimento de um programa simples na *linguagem Java*

À semelhança do ciclo de desenvolvimento de um programa escrito na linguagem *C*, apresentado na aula anterior, o objectivo desta secção é compreender agora o ciclo de desenvolvimento de um programa escrito na linguagem de programação de alto nível *Java*. Enquanto a *linguagem C* é uma *linguagem compilada*, a *linguagem Java* é, geralmente, *interpretada*.

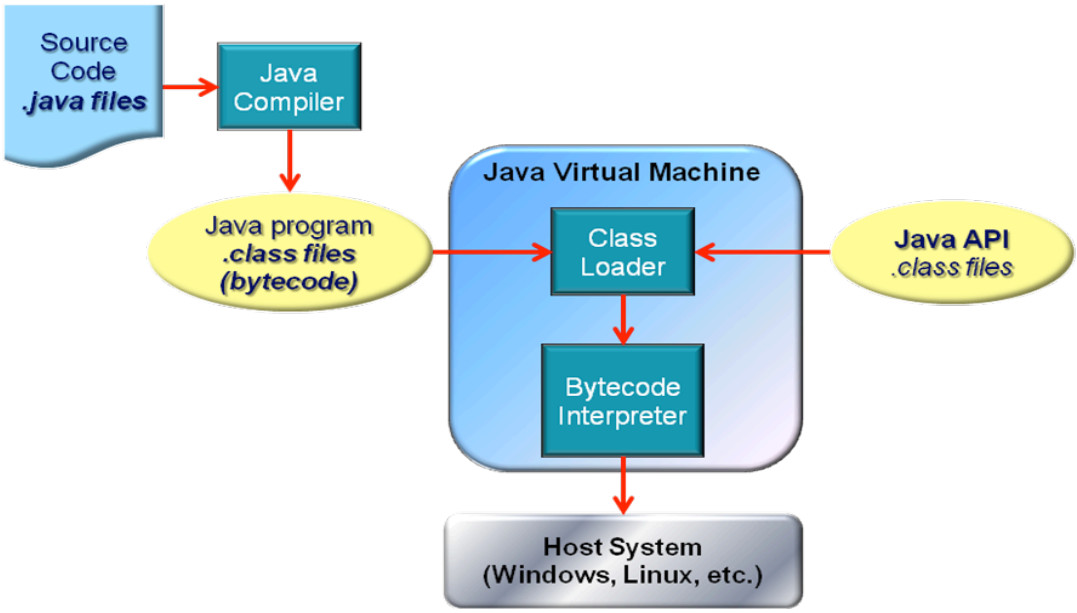

Figura 1

Os passos necessários para traduzir e executar as instruções de um programa escrito em *Java* estão ilustrados na **Figura 1**, tal como apresentado na aula teórica: um ficheiro em Java é primeiro traduzido para uma linguagem intermédia designada por *bytecode*, i.e. a linguagem máquina reconhecida pela "*Java Virtual Machine*" ("*JVM*"); de seguida, esse ficheiro (com extensão *.class*) é combinado com ficheiros da biblioteca *standard* do Java; finalmente, o código resultante é interpretado pela *JVM*. Descrevem-se a seguir os vários passos para criar e executar uma aplicação escrita em Java.

O ambiente de trabalho é o Windows.

1. Nesta fase deve ser usado um editor de texto como o *notepad* (disponível em **Programs→ Acessories**) ou o *notepad++* (disponível em **Programas→ Tools**). Utilizando o editor crie um ficheiro de nome "HelloWorld.java" com o código

```
import java.lang.System;

public class HelloWorld
{
     public static void main( String[] args) {
      System.out.println( "Hello world!" );
    }
}
```

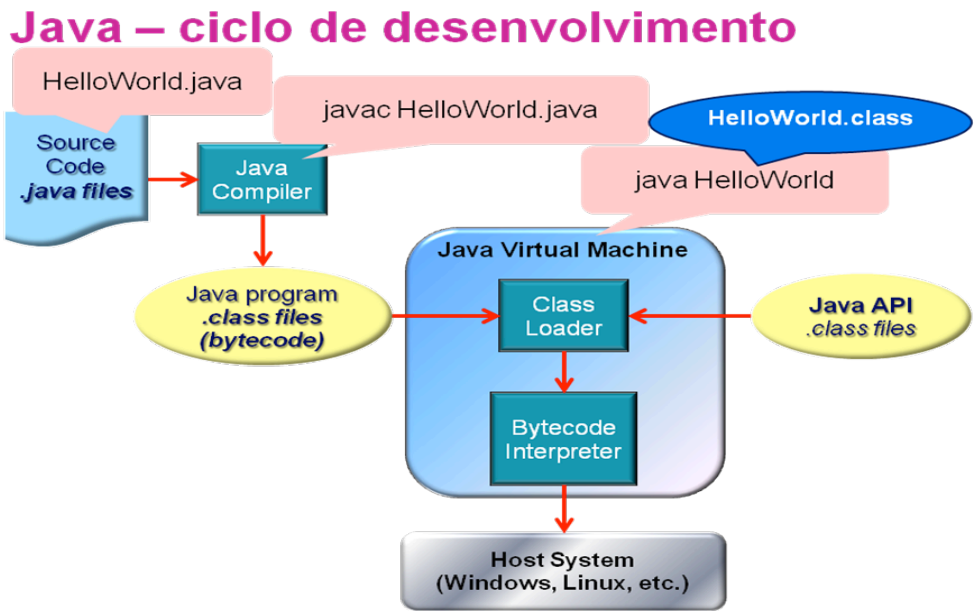

Figura 2

2. Proceda agora à tradução e interpretação do ficheiro por si criado, tal como ilustrado na **Figura 2**. Esta parte do ciclo de desenvolvimento deve ser feito usando a interface de linha de comando do Windows. Para lançar esta interface use o menu **Run** e escreva na caixa de diálogo *cmd*. Para que o interpretador de comandos encontre o executável com o compilador de Java para *bytecode* dê o seguinte comando:

```
PATH=%PATH%;C:\Program Files\Java\jdk1.6.0_07\bin
```

Também poderá ser necessário definir a variável de ambiente *CLASSPATH*. Essa variável indica as directorias onde o carregador dinâmico da JVM procura ficheiros com bytecodes – isto é os que têm a extensão *.class*. É preciso dizer que também devem ser procurados na directoria corrente (representada por .). Para conseguir o efeito pretendido faça

```
set CLASSPATH=.
```

2.1. Fase de tradução/compilação para *bytecode* usando o compilador de Java – *javac*:

```
$ javac HelloWorld.java
```

Nota: Verifique a criação do novo ficheiro *HelloWorld.class*

```
$ dir HelloWorld*
(…) HelloWorld.java  HelloWorld.class (…)
```

2.2. Fase de interpretação invocando a *JVM* sobre o ficheiro em *bytecode*:

```
$ java HelloWorld
Hello world!
```

Observação: Implicitamente, o interpretador de Java usa o ficheiro *HelloWorld.class* para a execução, e faz a ligação com todos os ficheiros de biblioteca (*.class*) necessários.

B. Desenvolvimento de programas Java que usam a API de acesso a ficheiros e directorias

Nesta parte da aula vemos a API (*Application Programmer´s Interface*) que corresponde aos *packages* genericamente conhecidos por *java.io*. Detalhes sobre esta API podem ser consultados em http://download.oracle.com/javase/tutorial/essential/io/

Note, que no tutorial acima referido, quando se referem as classes suportadas por *java.io.File*, o que aparece tem a ver o JDK7 que ainda não está disponível nos laboratórios. Sobre a "versão tradicional" do *java.io.File* pode consultar estes dois tutoriais disponíveis *online*:
http://tutorials.jenkov.com/java-io/file.html
http://www.roseindia.net/java/example/java/io/
Também pode consultar o exemplo apresentado na aula teórico-prática.

1. Escreva um programa em Java que recebe como argumento um nome. O programa deve indicar se trata de uma directoria ou de um ficheiro. No caso de se tratar de um ficheiro, deve indicar o seu comprimento em bytes

2. Escreva um programa em Java que lista o conteúdo de uma directoria cujo nome é recebido como argumento. O programa deve escrever uma linha para cada entrada na directoria. A linha deve conter o nome do ficheiro e o tipo da entrada (DIR ou FILE). No caso de ser um ficheiro deve indicar o seu comprimento em bytes. Exemplo:

```
IP   <dir>
decimal.pep8  <file>  357 bytes
x.html  <file>  2100 bytes
AM1   <dir>
exercícios.pdf  <file>  80357 bytes
ISRC   <dir>
menu.java  <file>  357 bytes
```

3. Escreva um programa em Java que copia o conteúdo de um ficheiro para o ecrã. O programa deve verificar se o nome existe e se não é uma directoria. Este programa tem de lidar com o aparecimento de *excepções*. Ver o exemplo seguinte:

```
try{
                File f2 = new File(arg2);
                InputStream in = new FileInputStream(f1);
                OutputStream out = new FileOutputStream(f2);

                /* código que lê de um ficheiro e escreve no outro */

                 ...

                in.close();
                out.close();
}
catch(FileNotFoundException ex){
  System.out.println(ex.getMessage() + " in the directory.");
  System.exit(0);
}
catch(IOException e){
          System.out.println(e.getMessage());
}
```

4. Escreva um programa em Java que recebe dois argumentos na linha de comando. O programa deve verificar se o 1º argumento é um ficheiro existente e se o segundo não existe. Caso as condições anteriores se verifiquem deve ser feita a cópia. Note que tem de incluir o tratamento de excepções como no programa anterior

5. Faça um interpretador de comandos que inclua os programas desenvolvidos nos pontos anteriores. O interpretador de comandos deve obedecer ao seguinte esquema:

```
import java.io.*;
import java.util.Scanner;

public class MyShell {
  public static void main( String args[]){
          boolean cont = true;
          Scanner input = new Scanner( System.in );
          while(cont){
                  System.out.println("Prompt >");
                  String command = input.next();
                  if( command.equals("type"))   /* código desenvolvido no ponto 3 */
                          . . .

                  else if (command.equals("dir"))  /* código desenvolvido no ponto 2 */
                                  . . .
                  else if (command.equals("copy"))  /* código desenvolvido no ponto 4 */
                                  . . .
                 else if (command.equals(“quit”)) cont = false
                  else System.out.printf(“Comando desconhecido”);
          }
  }
}
```