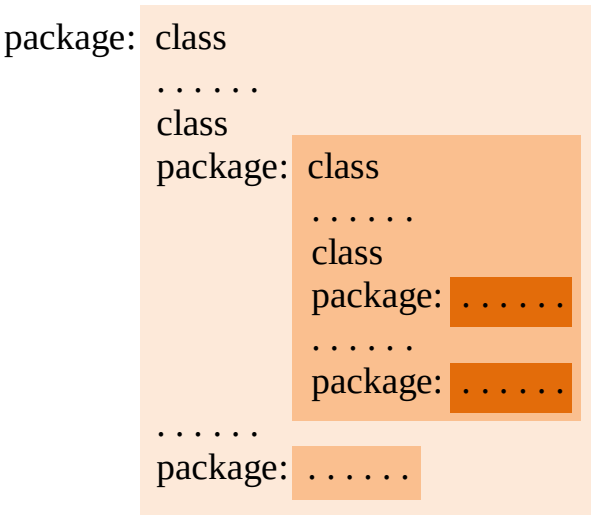
Packages of classes and JDK

Packages
Predefined packages
File organization
Java Development Kit (JDK)

Packages

package = group of classes or interfaces and, eventually, other sub packages:



by default, the classes of the file belong to an unnamed default package

declaring classes of a file as part of a certain package:

specify the optional following statement at the FIRST LINE of the file:

package package_name;

→ declares that all the classes of the file belong to the package package_name

package package_name$_1$ . . . . package_name$_n$;

→ declares that all the classes of the file belong to the package **package_name$_n$** which is a sub package of **package_name$_1$**

fully qualified names:

of a sub package:

package_name$_1$ . . . . package_name$_n$

of a class:

package_name$_1$ . . . . package_name$_n$. class_name

of a static variable of a class:

package_name$_1$ . . . . package_name$_n$. class_name . variable_name

of a static method of a class:

package_name$_1$ . . . . package_name$_n$. class_name . method_name

shortcuts:

up to several shortcut statements at the beginning of the file,
JUST AFTER the optional package statement

import package_name . class_name ; → this class will be accessible directly by its single **class_name**

import package_name . * ; → all the classes of this package will be accessible directly by their single names

Predefined packages

we will use the following packages and classes (many more are available):

java.lang
Object → *root class, superclass of all the classes*
System
Thread
Math
String
Boolean , Byte , Short , Integer , Long , Float , Double

java.io
InputStream , PrintStream
FileOuputStream , FileInputStream
DataOutputStream , DataInputStream

java.awt
Graphics , Color , Font

java.awt.event
ActionEvent , ActionListener
FocusEvent , FocusAdapter
KeyEvent , KeyAdapter
MouseEvent , MouseAdapter , MouseMotionAdapter
ComponentEvent , ComponentAdapter
WindowEvent , WindowAdapter

javax.swing
JFrame , JApplet , JPanel
JLabel
JButton , JMenuBar , JMenu , JMenuItem , JSeparator , JPopupMenu
JSlider , JTextField , JTextArea , JComboBox , JOptionPane

javax.swing.event
ChangeEvent , ChangeListener

java.util
Hashtable

we will also use several predefined classes related to the handling of exceptions

- at the beginning of every .java file, always write:

```
import  java . lang . * ;
import  java . io . * ;
```

(if not present, added automatically by the compiler)

- if the file is related to definition of a GUI or to graphics, write also:

```
import  java . awt . * ;
import  java . awt . event . * ;
import  javax . swing . * ;
import  javax . swing . event . * ;
```

File organization

the java source code of a program consists of several classes and interfaces

→ their description is dispatched between one or several **filename.java** files

.java files:

- one .java file may contain one or several classes or interfaces
- if a .java file contains several classes or interfaces, at most one of these classes or interfaces may be declared **public**; in which case the file name must be the name of that public class or interface (plus extension **.java**)

ONLY the class of same name as the file can be referred outside it,
the other classes must be used locally within the file!

- the file containing the main method must have the same name (plus **.java**) as the class enclosing the main method this class must be declared **public** and must contain only the main method and eventually some class variables (**static final** in particular)

.class files:

- each class or interface is compiled into a separate .class file
- **.class** files are stored in an arborescence of directories that has the same structure as the package names of the classes.
- the root directory of this arborescence corresponds to the default package
  the root directory is, by default, the current directory
  (unless specified with **CLASSPATH**)

Java Development Kit (JDK)

compiler:

javac options one or several source files(.java)

options: **–g** (debug) , **–O** (optimize)

when a source file references a class that is not defined in another source file on the command line, **javac** searches for that class according to the class path

if the compilation of that class is out of date,
javac automatically recompiles that class

interpreter:

java options name of the class enclosing main arguments of main

→ variant of the interpreter = debugger **jdb**

debugger: = special interpreter

note: before calling the debugger, compile with option **-g**

jdb options name of the class enclosing main arguments of main

- **run**

. . . the program is executed until an error occurs

- **list**

. . . debugger lists the source code around the line where error occured

- **print variable_name**

. . . debugger prints the value of the variable just before error

- **exit**

. . . debugger exits

example:

file default_dir \ Myprogram.java

```
import java.lang.*;
import java.io.*;
import mypackage.*;

public class Myprogram
{
   static final int n=3

   public static void main(String[] arg)
   {  if (arg.length<n)  {  D d = new D();  d.f();  }
                      else {  A a = new A();  a.f(arg);
                              B b = new B();  b.f();  }  }
}

class A
{  void f(String[] arg)
  {  for (int i=0 ; i<Myprogram.n ; i++)
        System.out.println(arg[i]);  }  }
```

file default_dir \ D.java

```
import java.lang.*;
import java.io.*;
class D  {  void f()  {  System.out.println("D…");  }  }
```

file default_dir \ B.java

```
package mypackage;
import java.lang.*;
import java.io.*;

public class B
{ public void f() { System.out.println("B…");
                    C c = new C();  c.f(); } }

class C { void f() System.out.println("C…"); }
```

example continued:

javac –O Myprogram.java

default_dir:

Myprogram.java

D.java

B.java

→ default_dir:

Myprogram.class

A.class

D.class

mypackage dir:

B.class

C.class

java Myprogram aa bb

→ D…

java Myprogram aa bb cc dd

→ aa
bb
cc
B…
C…