La variabile CLASSPATH

- Variabile d'ambiente del sistema operativo
  - Specifica un insieme di cartelle radice in cui cercare i file ".class" o le sottocartelle dei package
  - Può contenere direttori compressi (file ".jar")

```
set CLASSPATH=
        . ; c:\a.jar; c:\classes
```

- Cerca i file .class, in ordine:
  - Nella cartella in cui viene eseguita la JVM (".")
  - Nella cartella compressa c:\a.jar
  - Nella cartella c:\classes

Package

- Raggruppare le classi in package
- Struttura dei package delle API Java
- Il package java.lang

Nomi delle classi

- La metodologia ad oggetti favorisce il riuso delle classi
  - Il nome della classe dovrebbe suggerirne la semantica
  - A volte bisogna utilizzare nello stesso progetto classi già esistenti, di provenienza diversa ed aventi lo stesso nome
  - Occorre poter differenziare ulteriormente tali classi

Package: un cognome per le classi

- Le classi possono essere raggruppate in “package”
  - Ogni package ha un nome
  - Viene scelto in modo da essere univoco
- Una classe è denotata:
  - Dal nome proprio
  - Dal package di appartenenza

Package: appartenenza

- Le classi che appartengono ad uno stesso package formano un gruppo
- Come nel caso di una famiglia, c'è un rapporto privilegiato:
  - Accessibilità a tutti i componenti non privati (public, protected, <vuoto>)

Il nome dei package

- Sequenza di parole separate da ‘.’
  - Dovrebbe suggerire lo scopo comune del gruppo di classi
  - Per evitare collisioni, spesso inizia con il nome DNS in ordine inverso
  - it.polito.didattica.esempi

Sintassi

- La parola chiave “package” denota il gruppo di appartenenza
  - È seguita dal nome del package
  - Deve essere posta all’inizio del file sorgente

Sintassi

- Le classi dello **stesso** package si "conoscono"
  - Basta usare il nome **proprio** della classe
- Tra package **diversi** occorre usare il nome **completo**
  - Anche nei **costruttori**

Esempio

Cerchio.java

```
package forme;

public class Cerchio {
    //...
}
```

Esempio.java

```
package prova;
class Esempio {

  forme.Cerchio c;

  Esempio () {
    c=new forme.Cerchio();
  }
}
```

Importare riferimenti

- L'uso di nomi completi è scomodo
  - Gli amici si chiamano per nome
- Il costrutto "**import**" permette di definire le classi note
  - Queste vengono indicate solo per nome
  - Serve solo in fase di compilazione!

Esempio

```
package prova;
import forme.Cerchio;

class Esempio {
  Cerchio c;

  Esempio () {
    c=new Cerchio();
  }
}
```

Importare riferimenti

- Si possono includere un numero qualsiasi di clausole import
  - Devono sempre precedere la definizione della classe
- Per importare tutte le classi di un package, si usa la sintassi
  - **import *NomePackage*.* ;**

Gerarchia di package

- Il nome di un package può essere formato da molti segmenti
  - Package che condividono lo stesso prefisso, possono avere funzionalità “collegate”
    - java.awt
    - java.awt.event
  - Per Java, sono gruppi totalmente separati

Package anonimo

- Le classi che non dichiarano in modo esplicito il proprio package appartengono al package “**anonimo**”
  - A partire dalla versione 1.4, le classi del package anonimo **non possono** essere utilizzate da classi appartenenti ad altri package

Compilare ed eseguire

- Per poter utilizzare una classe all'interno di un'altra non basta "**risolverne**" il nome
  - Occorre localizzare il codice ad essa associato
  - Altrimenti viene lanciato l'errore "**NoClassDefFoundError**"

Rappresentazione su disco

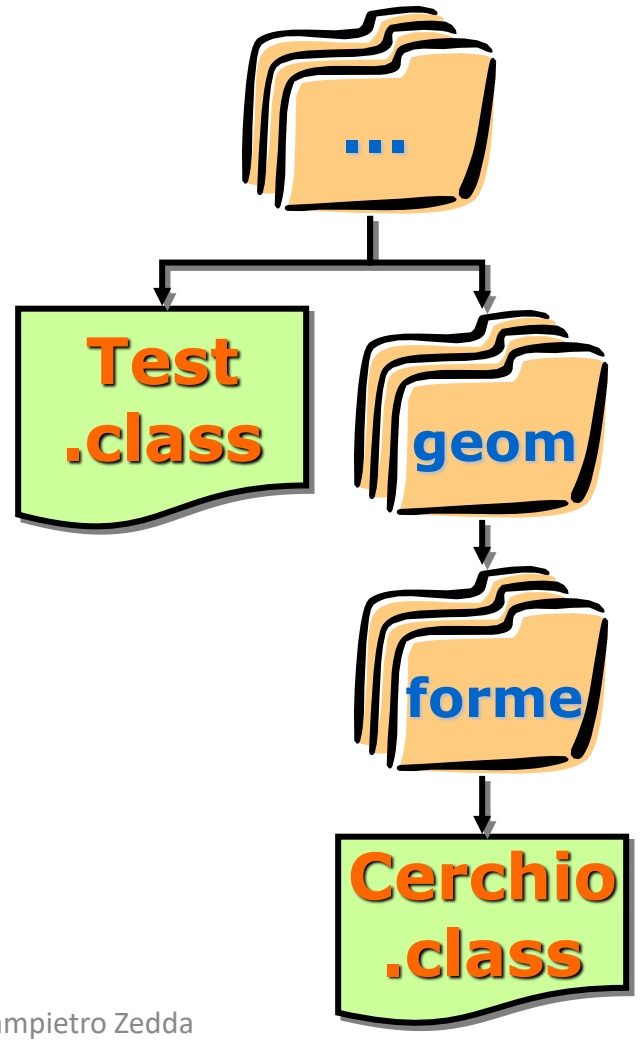
- Ad ogni classe, corrisponde un file “.class” che contiene il codice eseguibile
  - Questo deve risiedere in una (gerarchia di) **cartella** il cui nome coincide con quello del **package**
  - Le classi del package anonimo si trovano nella **cartella radice**

Rappresentazione su disco

```
public class Test {
    //...
}
```

```
package geom.forme;

public class Cerchio {
    //...
}
```



File jar

- **J**ava **AR**chive
- Gerarchie di cartelle e file compressi
  - Analoghi a file ".zip"
  - Possono essere manipolati con il comando "jar"
- Facilitano la distribuzione di interi package

API Java

- Application Programming Interface
  - Insieme di meccanismi per interagire con il sistema ospitante
  - Progettati interamente ad oggetti
- Offrono
  - Funzioni di libreria
  - Interfaccia verso il sistema operativo
- Versione 8:
  - > 200 package
  - Oltre 3000 classi

Package principali

- java.awt
  - **Abstract Windowing Toolkit**
  - Classi per creare interfacce utente di tipo grafico
- java.io
  - **Input/Output**
  - Classi per accedere a a flussi di dati, file e altri meccanismi di comunicazione

Package principali

- java.lang
  - Contiene le **classi fondamentali** del linguaggio
- java.math
  - **Estensioni matematiche**
  - Classi per modellare numeri interi e reali con precisione illimitata
- java.net
  - **Meccanismi di accesso alla rete**
  - Socket, URL, connessioni HTTP, …

Package principali

- java.nio
  - **New Input/Output**
  - Meccanismi di basso livello per interfacciarsi con il mondo esterno
- java.security
  - Classi che implementano il **modello di sicurezza** di Java
- java.sql
  - **Accesso a basi dati relazionali**

Package principali

- java.text
  - Trattamento **multiculturale** di numeri, date, testo
- java.util
  - Insieme variegato di **classi ad uso generale**

java.lang: l'ABC delle applicazioni Java

- Fornisce le classi fondamentali per la programmazione Java
  - Importato automaticamente dal compilatore in tutti i programmi
- Contiene – tra le altre – le classi `Object`, `Throwable`, `String`

java.lang.StringBuffer

- Analoga alla classe String
  - Ma permette di modificare i caratteri contenuti
  - thread-safe e synchronized
- Principali operazioni
  - append(…)
  - insert(…)
  - replace(…)
  - toString()

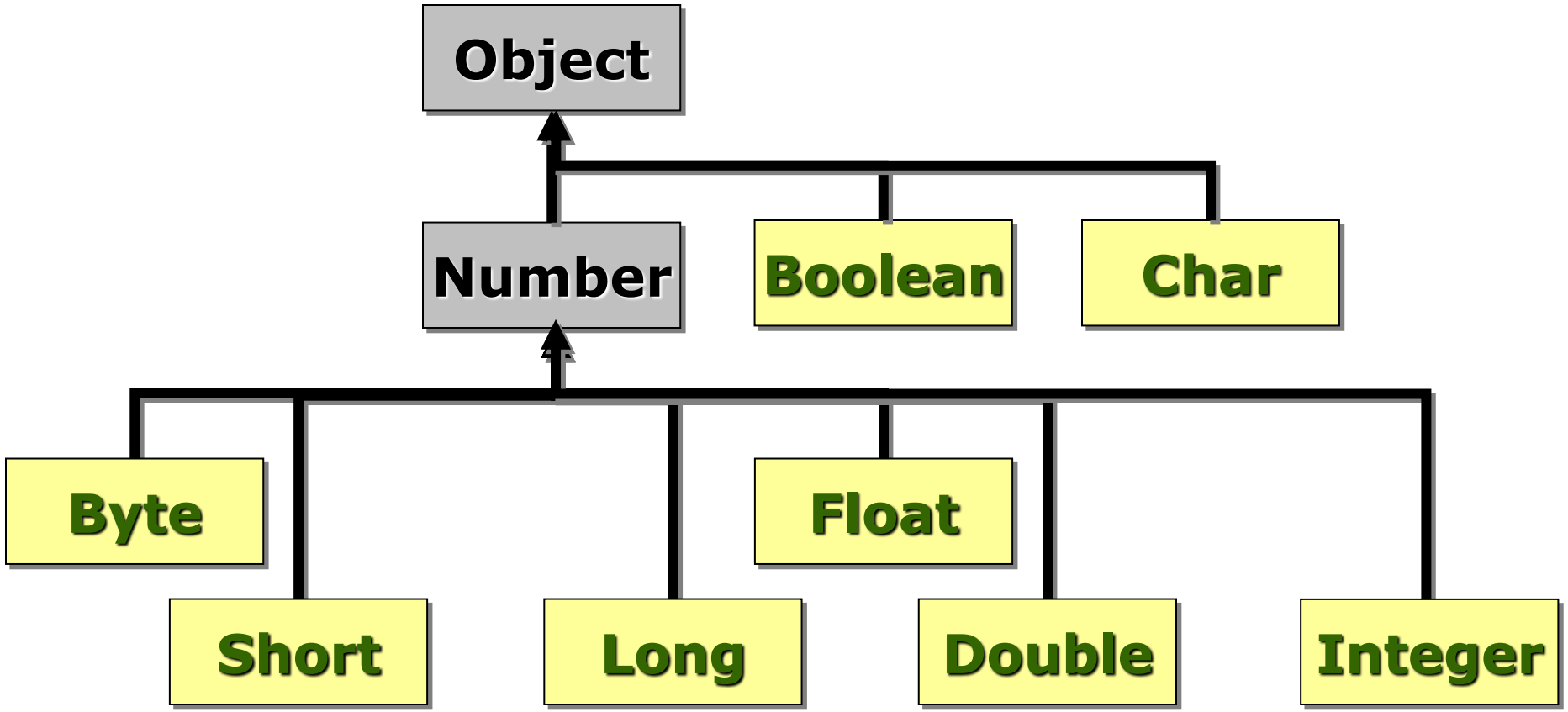
java.lang.StringBuilder

- Analoga alla classe StringBuffer
  - NON è thread-safe e synchronized
- Principali operazioni (come StringBuffer)
  - append(…)
  - insert(…)
  - replace(…)
  - toString()

Classi “wrapper”

- Utilizzate per trasformare in oggetti dati elementari
  - Il dato contenuto è immutabile
- Pattern generale dell’ingegneria del software

Classi “wrapper”



Wrapper numerici

- Sottoclassi di **Number**
- Metodi per
  - Trasformare una stringa in un numero e viceversa
  - Trasformare un numero in formati diversi (con possibile troncamento)
  - Rappresentazione testuale ottale, esadecimale, binaria

Character, Boolean

- Character
  - Maiuscolo / minuscolo
  - Valore Unicode
  - Confronto
  - …
- Boolean
  - Conversione da/verso stringa
  - Confronto
  - …

java.lang.System

- Contiene attributi e metodi statici, utilizzati per:
  - Interazione con il sistema
  - Acquisizione di informazioni
  - Caricamento di librerie
  - Accesso a standard input e output
  - …
- Non può essere istanziata

System: i metodi (1)

- `exit(…)`
  - terminazione della JVM
- `currentTimeMillis()`
  - Numero di millisecondi trascorsi dal 1 gennaio 1970
- `setProperties(…)`/`getProperties(…)`
  - assegnazione e acquisizione delle proprietà di sistema
- `in`
  - ‘Standard’ input stream (Console)
- `out`
  - ‘Standard’ output stream (Console)

System: i metodi e proprieta (2)

- `gc()`
  - invocazione del garbage collector
- `load(…)` / `loadLibrary(…)`
  - carica dinamicamente un file o una libreria
- `setIn(…)`, `setOut(…)`, `setErr(…)`
  - riassegnazione dei flussi standard
- `lineSeparator()`
  - Restituisce la system dependent line separator come da system property line.separator
- …

java.lang.Math

- Mette a disposizione gli strumenti necessari per le operazioni matematiche base
  - Contiene solo metodi e attributi statici
  - valore assoluto, logaritmo, potenza, trigonometrici, minimo, massimo, …
- `Math.pow(2,10); //`2^{10}

java.lang.Runtime

- Singola istanza per ogni applicazione
  - Consente di interfacciarsi con il sistema operativo
- Metodi
  - Alcuni analoghi a System: exit, gc, …
  - Esecuzione di un processo, memoria disponibile,…
- Non può essere istanziata