# Fondamenti di Informatica:
## *Parte 2*

**Docente**

Prof. **Serafino Cicerone**

Dip. di Ingegneria Elettrica e dell'Informazione

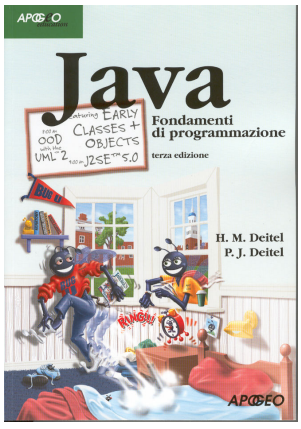
Università dell'Aquila

tel: `0862.434472`

email: `cicerone@ing.univaq.it`

web: `http://www.ing.univaq.it/~cicerone`

# Testo consigliato




H.M. Deitel, P.J. Deitel

*Java – Fondamenti di programmazione*

(terza edizione)

Apogeo, Ottobre 2005

## Sommario del corso

- Introduzione alla Programmazione Orientata agli Oggetti (OOP)
- Introduzione alle applicazioni Java
- Introduzione a classi e oggetti in Java
- Istruzioni di controllo
- I metodi
- Gli array
- Nozioni avanzate su classi e oggetti



## Introduzione alla programmazione

- Linguaggi ad alto livello
- Paradigmi di programmazione
- Problematiche moderno sviluppo software
  - software “in-the-large”
- La programmazione object-oriented

# LM vs linguaggio ad alto livello

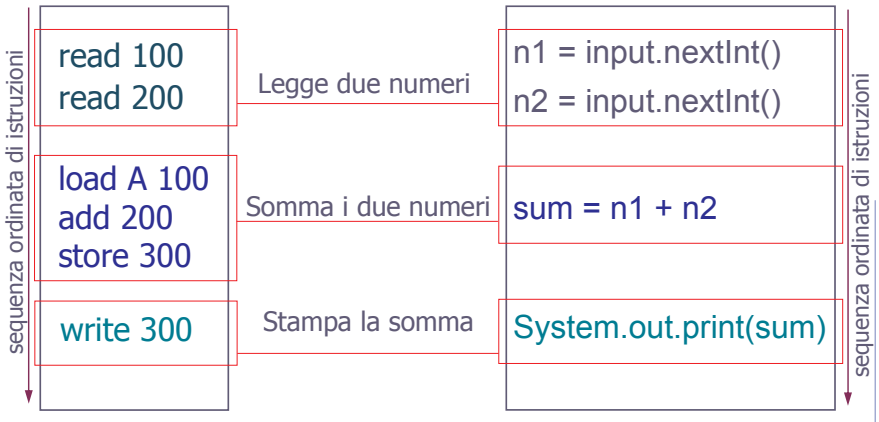

Linguaggio più vicino al programmatore

Linguaggio macchina con codici ed etichette

Unico linguaggio comprensibile ad un elaboratore



# Esempio



## Differenti tipologie di linguaggi

- Commerciali:
  - COBOL (COmmon Business Oriented Language)
- Matematici:
  - FORTRAN (FORmula TRANslator)
- Logici:
  - PROLOG (PROgramming LOGic)
- General purpose:
  - Pascal, C - *programmazione procedurale*
  - C++, C#, Java - *programmazione orientata agli oggetti, OOP*



## Moderno sviluppo software

- Moderno sviluppo software:
  - attività ingegneristica complessa
  - dimensione sempre crescente delle applicazioni
  - difficoltà della manutenzione dei programmi

- Programmazione Orientata agli Oggetti - OOP

## OOP – osservazioni dal mondo reale

- il mondo reale è costituito da **oggetti**: *persone, animali, piante, automobili, etc*
- suddivisione: oggetti animati e inanimati
- tutti gli oggetti hanno in comune:
  - **attributi**: *dimensione, forma, peso, età, colore, credito residuo, numero esami superati, etc*
  - **comportamento**: *la palla rimbalza, l'auto accelera, un telefonino spedisce SMS, lo studente supera gli esami, etc*
- oggetti distinti possono avere stessi attributi e comportamento → raggruppabili in **classi**
- oggetti possono essere definiti componendo altri oggetti: *un'automobile è composta dal motore, dalle ruote, dallo sterzo, dai freni, etc*
- oggetti di classi diverse devono conoscersi per poter comunicare tra loro: *un telefonino (attraverso la SIM) deve conoscere il provider*



## OOP – l'approccio

- La progettazione orientata agli oggetti modella il software in termini simili a quelli che le persone usano per descrivere oggetti del mondo reale
- Si progettano classi per poter definire oggetti con certi attributi e comportamento
  - attributi → strutture dati
  - comportamento → procedure
- Le classi hanno relazioni con altre classi
  - per la composizione di oggetti e loro reciproca conoscenza
- Oggetti comunicano tramite messaggi:
  - esempio: *un oggetto conto corrente riceve il messaggio di sottrarre dal totale l'importo prelevato dal cliente*

## Vantaggi di OOP

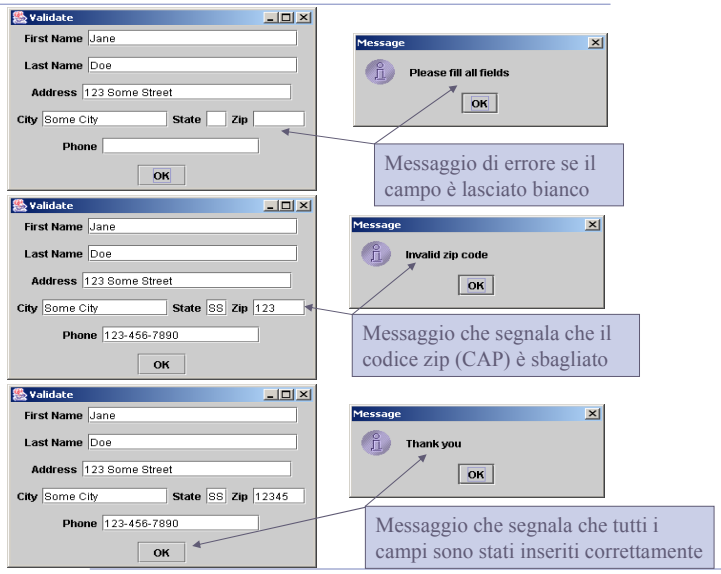
- Meccanismo base:
  - dati e operazioni incapsulati in un oggetto
  - la classe specifica oggetti simili
- Con la OOP si costruisce software combinando classi:
  - le classi sono parti intercambiabili
  - programmi di più semplice manutenzione
    - la modifica è localizzata in una o più specifiche classi
- Progettare grandi sistemi software in modo efficiente
- Riuso: le classi sono riusabili in diversi progetti software



## Progettazione OOP

- Progettare le classi
  - e stabilire le relazioni tra le classi
- Implementare le classi
  - alcune di queste classi possono essere già disponibili (in librerie) poiché realizzate in progetti software precedenti
  - alcune di queste classi differiscono di poco da classi già disponibili in libreria
- Tipologie di classi:
  - Entità Base: *Studente, ContoCorrente, Libro, Automobile, …*
  - Interfaccia: *Finestra, CasellaInput, Bottone, …*
  - Controllo: *FaseIniziale, …*
  - Test: *TestContoCorrente, TestApplicazione,…*

# Librerie: classi per GUI






# Librerie

- Classi per GUI (Graphical User Interface)
- Classi per operazioni matematiche
- Classi per gestione input e output
- Classi per connessione a database
- Classi che definiscono strutture dati complesse
- Classi per realizzazione applicazioni grafiche
- Classi per la gestione degli eventi
- …

# UML

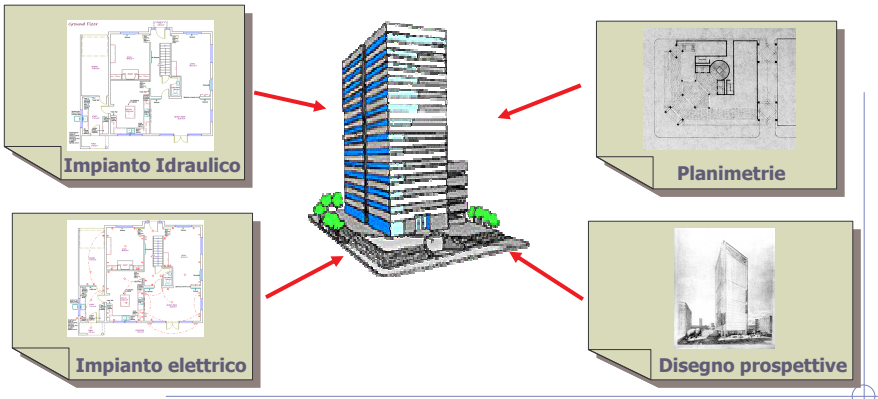
- Unified Modeling Language (UML): linguaggio di modellazione di tipo general-purpose usato per specificare, visualizzare, costruire e documentare i risultati della progettazione di un sistema software

- NB: *uso limitato ed informale all'interno del corso*



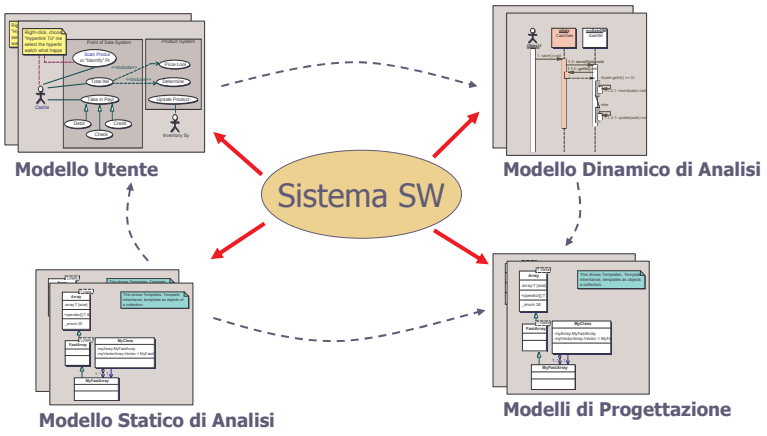
# Progettazione e Modelli

- Rappresentazione semplificata di una realtà complessa
- Evidenza di aspetti rilevanti e rimozione di dettagli
- Modelli diversi per diverse viste



# Progettazione SW e modelli UML

- Ogni diagramma UML evidenzia un particolare aspetto
- L'insieme dei diagrammi UML raccoglie le diverse viste di un sistema sw






# Obiettivo del corso

- Introduzione alla programmazione
  - studio di Java
  - realizzazione di semplici applicazioni con uso di classi di libreria
  - realizzazione di semplici applicazioni con progetto di classi Entità Base

## Il linguaggio Java

Java è un linguaggio orientato agli oggetti molto diffuso in quanto potente, facile da imparare ed estremamente utile per la realizzazione di applicazioni basate su Internet

## Sommario

- Primi passi nell'ambiente di programmazione Java
  - l'ambiente Java
  - primo programma in Java
  - test di un'applicazione Java

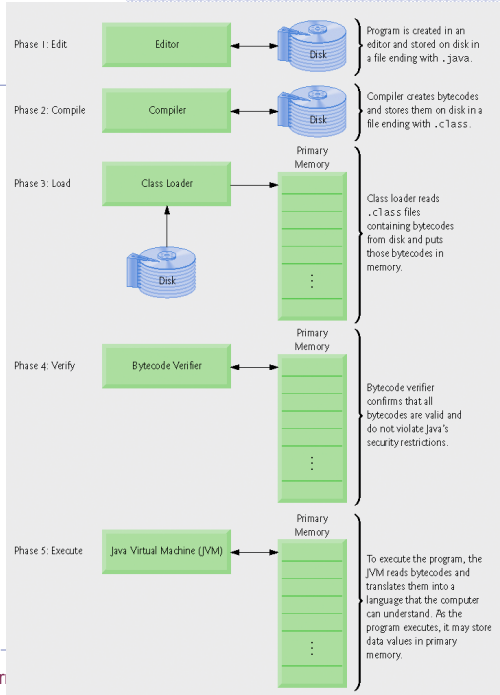
## Tipico ambiente Java (1)

- Il corso è basato sulla piattaforma Java 2 Standard Edition (J2SE) che descrive il linguaggio, le librerie e gli strumenti Java
- Sulla base delle specifiche della piattaforma ogni produttore implementa un proprio Java development kit che contiene gli strumenti necessari per realizzare programmi software con Java
- SUN offre un kit denominato Java 2 Software Development Kit (JDK): java.sun.com/j2se/5.0



## Tipico ambiente Java (2)

- Un programma Java attraversa cinque fasi prima di essere eseguito:
  1. Scrittura/Modifica
  2. Compilazione
  3. Caricamento
  4. Verifica
  5. Esecuzione






# Scrittura/Modifica

- Il programmatore scrive il programma (codice sorgente) attraverso un editor
- Il programma viene salvato su disco e il suo nome termina con l'estensione `.java`
- Un tipico editor è il Blocco Note di Windows
- Esistono ambienti di sviluppo Java integrati (Integrated Development Environment, IDE) che mettono a disposizione strumenti per lo sviluppo di programmi e la loro messa a punto
  - NetBeans (www.netbeans.org)
  - JEdit (www.jedit.org)
  - JCreator (www.jcreator.com)

## Compilazione

- Il compilatore trasforma il codice sorgente Java (il file con l'estensione `.java`) in bytecode producendo un file con lo stesso nome e l'estensione `.class`
- Il bytecode rappresenta le operazioni che verranno eseguite nella fase 5
- Il bytecode viene eseguito da una Java Virtual Machine (JVM) che simula un computer nascondendo al programma l'hardware e il SO della particolare piattaforma
- A differenza del linguaggio macchina che è specifico dal hardware il bytecode è indipendente dalla piattaforma
- Il bytecode Java è portabile



## Caricamento/Verifica/Esecuzione

- Caricamento
  il programma sorgente viene caricato nella RAM dal class loader
- Verifica
  il verificatore di bytecode garantisce che i bytecode delle classi siano validi e rispettino tutte le norme di sicurezza di Java per evitare che ad esempio file scaricati dalla rete possano causare problemi
- Esecuzione
  L'interprete Java interpreta ed esegue il programma un bytecode per volta

# Problemi

- Non sempre i programmi funzionano a primo colpo
- Ciascuna delle fasi descritte può fallire a causa di errori
- Ad esempio un programma può tentare di eseguire una divisione per zero
- In questi casi l'ambiente Java segnala un errore tramite un apposito messaggio
- Il programmatore torna alla fase di scrittura/modifica per correggere l'errore e ripetere le fasi successive



# Il primo programma Java

Visualizza sullo schermo la stringa
"Welcome to Java Programming"

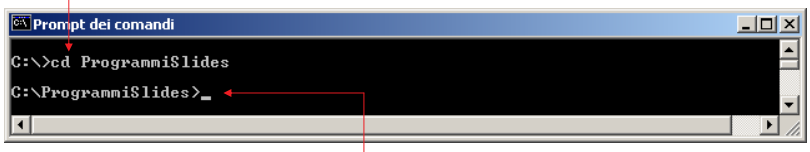
```
/* Il primo programma Java */
public class Welcome1
{
   public static void main( String args[] )
   {
      System.out.println( "Welcome to Java Programming!" );
   } // --- fine metodo main
} // --- fine classe Welcome1
```

Il metodo main inizia l'esecuzione di qualsiasi applicazione Java

Visualizza sul video la scritta "Welcome to Java Programming"

# Il primo programma Java (5)


Uso del comando *cd* per cambiare directory



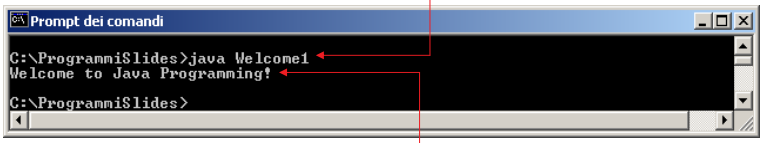

Posizione del file da testare: Welcome1.java



# Il primo programma Java (6)


Uso del comando *javac* per compilare Welcome1.java
il risultato è il file Welcome1.class

Uso del comando *java* per eseguire Welcome1.class




Risultato dell'esecuzione di Welcome1.class

# Test di un'applicazione Java

- Test dell'applicazione ATM (bancomat)
  - Check system setup
  - Locate the ATM application
  - Run the ATM application
  - Enter an account number
  - Enter a PIN
  - View the account balance
  - Withdraw money from the account
  - Confirm that the account information has been updated
  - End the transaction
  - Exit the ATM application

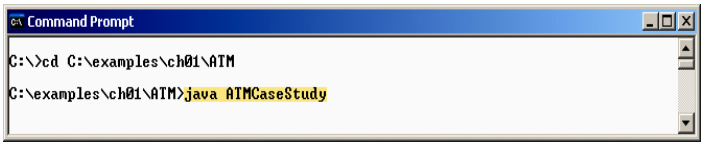


# ATM application (1)

Using the cd command to change directories

File location of the ATM application

# ATM application (2)

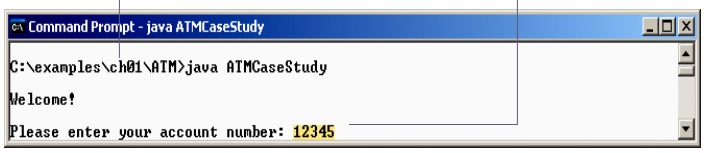





# ATM application (3)

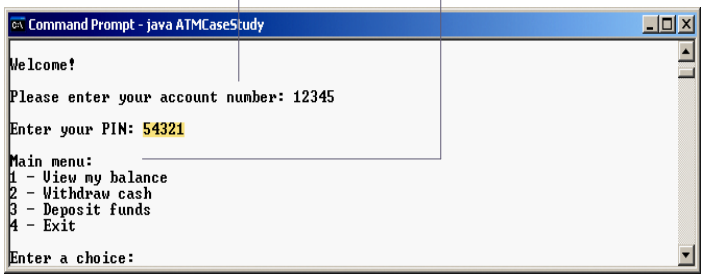
ATM welcome message

Enter account number prompt



# ATM application (4)

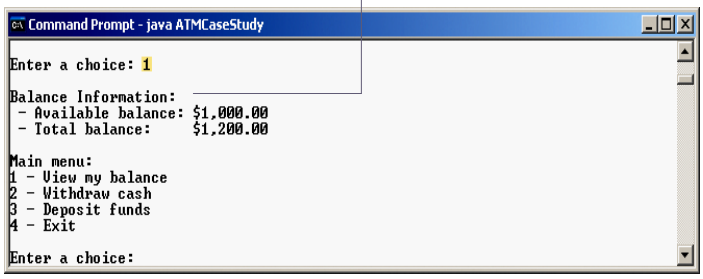
Enter valid PIN  ATM main menu





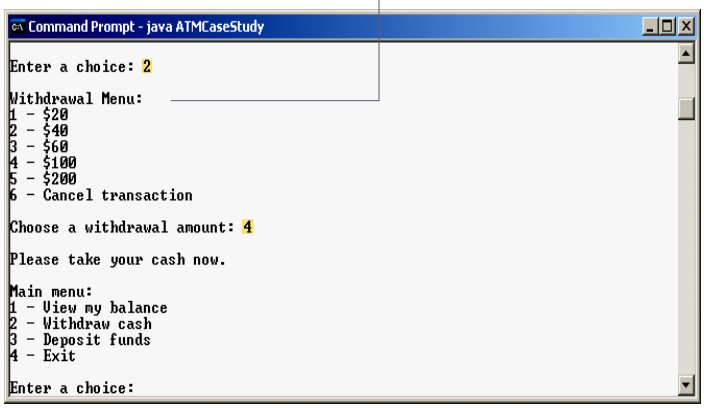

# ATM application (5)

Account balance information



# ATM application (6)

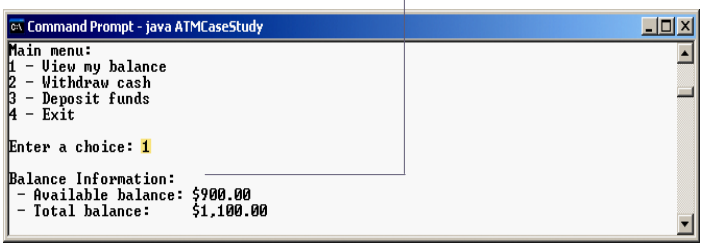
ATM withdrawal menu



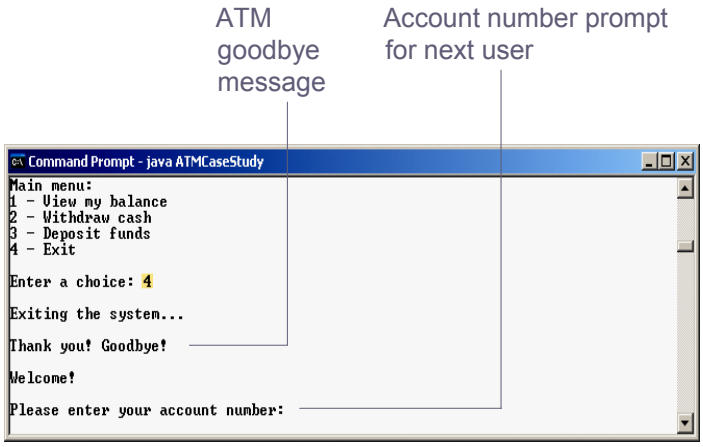



# ATM application (7)

Confirming updated account balance information after withdrawal transaction



# ATM application (8)






# Deitel: ulteriori applicazioni Java

Assumendo di aver copiato il codice nella cartella `C:\examples ...`

| Application Name | Chapter Location | Commands to Run |
|---|---|---|
| Tic-Tac-Toe | Chapters 8 and 24 | `cd C:\examples\ch01\Tic-Tac-Toe`<br>`java TicTacToeTest` |
| Guessing Game | Chapter 11 | `cd C:\examples\ch01\GuessGame`<br>`java GuessGame` |
| Logo Animator | Chapter 21 | `cd C:\examples\ch01\LogoAnimator`<br>`java LogoAnimator` |
| Bouncing Ball | Chapter 23 | `cd C:\examples\ch01\BouncingBall`<br>`java BouncingBall` |

# Prossima lezione

- tipi di dato in Java
- uso di classi di libreria (GUI)
  - esempio
- la memoria
- aritmetica
- decisioni
  - esempio