

Programmazione ad Oggetti

Informazioni generali

- ▶ **Docente**

- ▶ Giacomo Cabri

- ▶ **Come contattarmi**

- ▶ Via email (consigliato) giacomo.cabri@unimore.it

- ▶ Telefono 059/2058320

- ▶ **Ricevimento**

- ▶ Lunedì dalle 15 alle 17 presso Matematica, secondo piano

- ▶ **Sito web dell'insegnamento**

- ▶ http://didattica.agentgroup.unimore.it/wiki/index.php/Programmazione_ad_Oggetti

Testo

- ▶ **G. Cabri, F. Zambonelli, "Programmazione a oggetti in Java: dai fondamenti a Internet", Pitagora editrice, 2003**
- ▶ **Altri testi utili**
 - ▶ B. Eckel, "Thinking in Java", 4 edizione italiana, Pearson, (scaricabile gratuitamente da Internet la versione inglese <http://www.BruceEckel.com>).
 - ▶ Arnold, Gosling, Holmes, "Il linguaggio Java. Manuale ufficiale", Pearson.
 - ▶ C.S. Horstmann, G. Cornell, "Java 2: i Fondamenti", Mc Graw Hill, The Sun Microsystems Press.
- ▶ **Per chi vuole approfondire:**
 - ▶ Mazzanti, Milanese, "Programmazione di applicazioni grafiche in Java", Apogeo.
 - ▶ Gamma, Helm, Johnson, Vlissides, "Design Patterns", Addison-Wesley.
 - ▶ J.R. Hubbard, "Strutture dati in Java", McGraw-Hill.
 - ▶ M.A. Weiss, "Data Structures And Problem Solving Using Java", 2nd Edition, Addison Wesley.

Finalità dell'insegnamento

- ▶ **Programmazione ad Oggetti**
 - ▶ concetti generali, quali incapsulamento, ereditarietà e polimorfismo
 - ▶ concetti di riusabilità e di composizione dei componenti software
 - ▶ passaggio dalla programmazione modulare alla programmazione ad oggetti
- ▶ **Il linguaggio Java**
 - ▶ esempio di linguaggio ad oggetti
 - ▶ concetti generali implementati in Java
 - ▶ interfacce grafiche

Programma

- ▶ Concetti generali
- ▶ Il Linguaggio Java
- ▶ Classi ed ereditarietà in Java
- ▶ I/O in Java
- ▶ Interfacce grafiche in Java
- ▶ Programmazione basata sugli eventi
- ▶ Strutture dati in Java
- ▶ Programmazione concorrente in Java

Esame

- ▶ L'esame permette di acquisire 9 CFU
- ▶ Si compone di:
 - ▶ Una prova scritta
 - ▶ Una serie di esercizi in cui si chiede di implementare una semplice entità in Java
 - ▶ La discussione di un progetto
 - ▶ Un programma più complesso che sfrutta diverse caratteristiche di Java

Prova scritta

- ▶ Serve a verificare che lo studente abbia acquisito le nozioni di astrazione, classificazione, ereditarietà e polimorfismo
- ▶ Consiste in alcuni (di solito 2-3) esercizi in cui si chiede di scrivere del codice Java che implementa una semplice entità

Prova scritta – informazioni utili

- ▶ Per l'iscrizione all'esame scritto è necessario utilizzare **ESSE3** e iscriversi entro **3 giorni prima** dell'appello scritto stesso
- ▶ Il voto della prova scritta vale circa **1 anno** (ad es., lo scritto di gennaio è valido fino alla sessione invernale dell'anno successivo)
- ▶ La consegna di uno scritto **cancella** il voto precedente; la partecipazione a uno scritto senza consegnare non cancella il voto precedente
- ▶ È necessario aver **superato** lo scritto per presentare il progetto
- ▶ La presentazione del progetto può avvenire anche in un appello **diverso** da quello dello scritto

Progetto

- ▶ Consiste nello sviluppo di un **programma** in Java
- ▶ Il progetto deve avere le seguenti caratteristiche:
 - ▶ sfruttare i meccanismi della programmazione ad oggetti:
 - ▶ **incapsulamento**
 - ▶ **ereditarietà** e, se necessario, le classi astratte e le interfacce (si considerano **escluse** le relazioni di ereditarietà diretta da classi di libreria Java)
 - ▶ **polimorfismo**
 - ▶ essere dotato di **interfaccia grafica** tramite cui interagire con il programma stesso
 - ▶ sfruttare le classi di sistema Java per la gestione dell'**input/output** (Reader, Writer, InputStream, OutputStream)
 - ▶ sfruttare i **generics** Java

Progetto – altre caratteristiche

- ▶ Il software deve essere accompagnato da pagine di **documentazione** HTML (tipicamente le pagine generate tramite Javadoc) che descrivano le scelte di progetto effettuate e la struttura del sistema software
- ▶ Il programma deve essere una **applicazione** Java e **NON** una applet
- ▶ Il programma deve essere eseguito da **linea di comando**
- ▶ Non devono essere usate librerie proprietarie (ad es. quelle degli ambienti integrati) a meno di averlo concordato con il docente
- ▶ È opportuno **provare** il proprio programma in laboratorio prima di presentarlo in sede d'esame
- ▶ Portare: **sorgenti, compilati, documentazione**

Progetto – che cosa fare

- ▶ Il docente proporrà una **tesina**
- ▶ È possibile implementare un progetto diverso dalla tesina, MA è necessario farlo approvare dal docente
 - ▶ Anche via email
 - ▶ L'importante è che rispetti le caratteristiche richieste

Progetto – problemi da evitare

- ▶ **Problemi DA EVITARE** in sede di esame:
 - ▶ Il programma **non** funziona
 - ▶ Si deve venire all'esame con il programma funzionante
 - ▶ **Manca** qualcosa (di solito, la documentazione)
 - ▶ Controllare nell'elenco delle caratteristiche se c'è tutto
 - ▶ Il progetto è **diviso** su più programmi
 - ▶ Il programma deve essere unico (a meno di casi concordati)
 - ▶ L'ereditarietà è usata **male**
 - ▶ Si capirà più avanti come va usata
 - ▶ “Non mi ricordo perché ho fatto il programma tempo fa”
 - ▶ L'esame consiste nella presentazione e discussione del progetto, che vanno preparate