PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

(6 Crediti – docente Claudio Mirolo)

Obiettivi del corso

L'obiettivo principale del corso è quello di discutere il paradigma della programmazione orientata agli oggetti come evoluzione delle forme di astrazione (procedurale, sui dati, sullo stato) introdotte nel corso di Programmazione. Verranno inoltre presentate le caratteristiche di base di un linguaggio di programmazione orientato agli oggetti. I concetti saranno introdotti soprattutto attraverso la discussione di esempi. Al termine del corso lo studente dovrebbe aver acquisito le competenze necessarie per analizzare programmi orientati agli oggetti, per intervenire sul codice, nonché per progettare semplici soluzioni basate sulla definizione di un insieme di componenti interagenti.

Prerequisiti: Corso di Programmazione; alcuni dei contenuti dei corsi di Matematica Discreta e Analisi Matematica.

Contenuti del corso - Gli argomenti principali trattati durante il corso sono indicati qui di seguito. Gli esempi sono sviluppati utilizzando il linguaggio di programmazione *Java*.

Concetti e metodologie

Nozioni di base: classi e oggetti; metodi e messaggi; responsabilità e protocollo; incapsulamento delle informazioni. Progettazione orientata agli oggetti: analisi funzionale; identificazione delle componenti e definizione delle responsabilità; specifica del protocollo (interfaccia); definizione delle interazioni fra componenti; definizione del comportamento e concetto di stato; eccezioni e relativa gestione. Forme di astrazione nel paradigma orientato agli oggetti: eredità e gerarchie di classi; *overriding* di metodi.
Ereditarietà: sottoclassi, tipi e compatibilità per sostituzione. Specializzazione e altre forme di ereditarietà. Classi astratte e interfacce. Ereditarietà del codice ed ereditarietà del comportamento. Selezione dinamica dei metodi. Confronto fra ereditarietà e aggregazione. Polimorfismo e relativo ruolo. *Overloading*. *Overriding*: sostituzione e raffinamento. Modelli di progetto (*design patterns*): *composite*; *decorator*; *factory method*; *flyweight*; *observer*; *prototype*; *proxy*; *strategy*. Schema *model-view-controller*.

Linguaggio di programmazione Java

Caratteristiche e costrutti principali del linguaggio *Java*. Realizzazioni di classi in *Java*: struttura di un programma; campi per rappresentare le variabili di istanza; costruttori; attributi delle variabili di istanza e dei metodi; istanziazione di oggetti; ereditarietà e interfaccie. Classi nidificate, classi interne e organizzazione del codice. Modello degli eventi in *Java*. Gestione delle eccezioni. *Thread* e sincronizzazione. Clonazione. *Garbage collection*.
Alcune caratteristiche avanzate dell'ambiente di programmazione basato su *Java*. Progetto di interfacce utente (*GUI*) attraverso i package *AWT* e *Swing*: componenti e *layout manager*. *Input* e *output* in *Java*. Altre classi di utilità: *Math*, *Random*, *System*, *Vector*, *String*, *StringTokenizer*. Grafica elementare in *Java*. *Applet* e programmazione per la rete.
Discussione di esempi in *Java*.

Concetti ricorrenti: Livelli di astrazione; astrazione sui dati; astrazione sullo stato; incapsulamento dell'informazione; riutilizzo del codice.

Ulteriori informazioni sul corso e sulle lezioni, in particolare gli esempi discussi in classe e i temi d'esame, sono resi disponibili attraverso le pagine del corso all'indirizzo:

`http://www.dimi.uniud.it/claudio/teaching/prog_oo/`

Materiale di studio consigliato

Timothy Budd
Understanding Object-Oriented Programming with Java
Addison-Wesley, 2000 (ISBN: 0-201-61273-9)

Appunti tratti dalle lezioni.

Testi sul linguaggio Java

David Arnow and Gerald Weiss
Introduction to Programming Using Java: An Object-Oriented Approach
Addison-Wesley, 1998 (ISBN: 0-201-31184-4)

oppure

John Lewis, William Loftus
Java - Fondamenti di progettazione software
Addison-Wesley, 2000 (Edizione italiana 2001; ISBN: 88-7192-092-9)

Testi per consultazione

B. Liskov, J. Guttag
Program Development in Java
Addison-Wesley, 2000;

X. Jia
Object Oriented Software Development Using Java
Addison-Wesley, 2002;

B. Eckel
Thinking in Java
Prentice Hall, 2000;

K.A. Lambert, M. Osborne
A Framework for Program Design and Data Structures
Brooks/Cole, 2000;

Modalità d'esame

L'esame di Programmazione Orientata agli Oggetti prevede due prove di accertamento, che si svolgono al termine dei due periodi didattici in cui è articolato il corso, e una prova orale. All'orale sono ammessi gli studenti che hanno conseguito una valutazione complessiva di almeno 15 punti su 30 nelle due prove di accertamento. La discussione orale non è obbligatoria per valutazioni delle prove di accertamento comprese fra 20 e 28 punti. Opzionalmente, ai fini della discussione orale il candidato può concordare con il docente la realizzazione di un progetto di programmazione che applichi concetti e metodologie introdotti durante il corso. In tal caso, la discussione del progetto costituirà la parte prevalente della prova orale. Nel caso venga sostenuta, la prova orale concorre alla valutazione finale nei termini di una media pesata.

La valutazione di una singola prova di accertamento viene espressa nei seguenti livelli: ottimo, buono, discreto, sufficiente, quasi sufficiente, insufficiente. La valutazione complessiva delle prove di accertamento o dei recuperi che vertono sull'intero programma viene espressa con un punteggio da 18 a 30, se sufficiente; da 15 a 17, se consente di sostenere la prova orale; insufficiente altrimenti.

Il primo appello scritto di Programmazione Orientata agli Oggetti consiste normalmente nello svolgimento della seconda prova di accertamento. A partire dal secondo appello sono invece previste (unicamente) prove scritte di recupero, che vertono sull'intero programma del corso, rivolte a coloro che non abbiano potuto partecipare alle prove di accertamento o abbiano conseguito una valutazione insufficiente per l'ammissione all'orale. Indipendentemente dalla valutazione, le prove scritte di recupero non consentono l'esonero dalla discussione orale.

OBJECT ORIENTED PROGRAMMING

(6 Credits – teacher: Claudio Mirolo)

Aims

The main objectives of the course are to discuss the object-oriented programming paradigm, as it has evolved from structured programming and abstract data types, and to develop basic skills to design object-oriented solutions. The fundamental features of an object-oriented programming language will be presented. The concepts will be introduced by showing suitable examples.

Prerequisites: Basic knowledge of Programming and Mathematics.

Course Syllabus

Concepts and methodologies.

Basic concepts: interacting modules, messages and methods, responsibility, classes and instances, class hierarchies and inheritance, method overriding, exceptions. Object-oriented design: identification of the components and their responsibilities; behavior; interface; concept of state. Benefits of object-oriented programming: information hiding, reliability, code reuse, code sharing, polymorphism, fast prototyping.
Inheritance: subclasses, types and substitutability. Specialization and other forms of inheritance. Code inheritance and behavior inheritance. Dynamic binding. Class inheritance and composition; dynamic composition. Inner classes. Clones. Polymorphism and polymorphic variables. Overloading. Overriding: substitution and refinement. Abstract methods. Design patterns: *composite*; *decorator*; *factory method*; *flyweight*; *observer*; *prototype*; *proxy*; *strategy*. M*odel-view-controller* paradigm.

Java programming language.

Features of the programming language *Java*. Classes in *Java*: program structure; data fields; constructors; modifiers; instances; inheritance and interfaces. Inner classes and control frameworks. Event model in *Java*. Exception handling. Multiple threads of execution and synchronization. Garbage collection.
More advanced features. Design of the user interface (*GUI*). *Application Windowing Toolkit* (*AWT*): components and layout managers. Input e output in *Java*. Other utility classes: *Math*, *Random*, *System*, *String*. Elementary graphics in *Java*. Collections: *Vector* vs. *array*; Dictionary. *Applet* and network programming.
Examples.

Textbooks

Timothy Budd
Understanding Object-Oriented Programming with Java
Addison-Wesley, 2000 (ISBN: 0-201-61273-9)

Java

David Arnow and Gerald Weiss
Introduction to Programming Using Java: An Object-Oriented Approach
Addison-Wesley, 1998 (ISBN: 0-201-31184-4)

or

John Lewis, William Loftus
Java Software Solutions - Foundations of Program Design
Addison-Wesley, 2002 (Edizione italiana 2001; ISBN: 0-201-78129-8)

Further References

B. Liskov, J. Guttag
Program Development in Java
Addison-Wesley, 2000;

X. Jia
Object Oriented Software Development Using Java
Addison-Wesley, 2002;

B. Eckel
Thinking in Java
Prentice Hall, 2000;

K.A. Lambert, M. Osborne
A Framework for Program Design and Data Structures
Brooks/Cole, 2000;

Exams

Organization of the exams:

- Two (written) tests, scheduled at the end of the two terms;
- Oral discussion (optional laboratory assignment).