IN490 Linguaggi di Programmazione

A.A. 2017/2018

Prof. Marco Pedicini

1. Macchine astratte, compilatori e interpreti

- Macchine astratte: concetto di astrazione, linguaggio formale, linguaggio macchina. Macchina astratta e macchina hardware. Formalizzazione dei concetti di interprete e compilatore. Modalità di implementazione: hardware, firmware, software. Macchina RAM.
- Implementazioni di un linguaggio formale: schema interpretativo puro e schema compilativo puro. Macchina intermedia. Implementazione di tipo interpretativo e di tipo compilativo della macchina intermedia. Gerarchie di macchine astratte.

2. Linguaggi di programmazione

- Linguaggi procedurali o imperativi, linguaggi funzionali, linguaggi dichiarativi o logici, linguaggi strutturati, cenni ad altri tipi di linguaggi.
- Definizione di linguaggio formale. Grammatiche libere da contesto (Chomsky), derivazione di stringhe. Definizione di linguaggio generato. Alberi di derivazione. Grammatiche ambigue. Gestione delle ambiguità, metodi di disambiguazione. Vincoli sintattici contestuali. Descrizione delle fasi di compilazione: analisi lessicale, sintattica, semantica. Grammatiche regolari. Esempio di compilazione.
- Ambiente, associazione di un oggetto a un nome, blocchi di codice: visibilità, operazioni sull'ambiente, operazioni sugli oggetti denotabili. Tempo di vita di associazioni e oggetti. Regole di visibilità: definizione di scope statico. Scope statico dei nomi: proprietà di indipendenza dalla posizione e indipendenza dai nomi locali. Definizione di scope dinamico. Confronto fra scope statico e scope dinamico.

3. Programmazione ad oggetti

- Principi fondamentali, concetto di classe e di oggetto. Richiami e definizioni formali di tipo di dato, sistema di tipi di un linguaggio di programmazione, sicurezza di un sistema di tipi.
- Controllo statico e dinamico dei tipi. Astrazione sui dati: interfaccia, implementazione, occultamento dell'informazione, controllo dell'accessibilità.

- Introduzione al paradigma orientato agli oggetti: caratteri essenziali (incapsulamento, occultamento dell'informazione, ereditarietà, sottotipi, selezione dinamica delle operazioni). Classi e oggetti, principi di organizzazione, estendibilità, riuso del codice. Terminologia. Definizione di classe, costruttori, membri di classe e d'istanza. Riferimento implicito "this" e allocazione.
- Overloading. Polimorfismo universale parametrico implicito ed esplicito. Polimorfismo di sottotipo e relazione con i generici. Utilizzo di un linguaggio di diagrammazione di classe: cenni ad UML.
- Fragilità nella gerarchia delle classi. Implementazione dell'ereditarietà multipla mediante vtable. Modelli di implementazione dell'ereditarietà multipla: implementazione mediante vtable con replicazione e con condivisione. Polimorfismo di sottotipi. Equivalenza per nome ed equivalenza strutturale di tipi.
- Ridefinizione di metodi (overriding) e mascheramento di campi (shadowing) nelle sottoclassi, astrazione ed ereditarietà. Relazione con i sottotipi, supertipo immediato, erditarietà singola e multipla. Problemi legati all'ereditarietà di classi: ereditarietà a diamante, conflitto di nomi (name clash), rimedi implementativi, implicazioni sulla gerarchia di sottotipi. Ereditarietà e astrazione: clausola "abstract" e derivazione di classi astratte, clausola "implements", implementazione ed estensione di interfacce.
- Selezione dinamica dei metodi: differenza fra tipo del riferimento e tipo dell'oggetto. Late binding. Implementazione dell'ereditarietà singola con tipi statici usando i record e della copia dei riferimenti agli oggetti.
- Gestione del controllo mediante eccezioni. Definizione di eccezione. Esempi. Propagazione delle eccezioni e catena dinamica. Cenno al record di attivazione per le procedure. Cenno all'implementazione delle eccezioni.

4. OOP in Java

- Variabili di classe, d'istanza, locali e loro allocazione in memoria. I costruttori. Incapsulamento e modificatori di accessibilità. Relazione di sottotipo e supertipo, clausola "extends", derivazione di classi, concetto di ereditarietà, gerarchia di tipi, compatibilità e polimorfismo. Compatibilità verso il basso (downcasting), clonazione e metodo 'clone' di Java.
- Convenzioni sull'uso dei nomi. Tecniche di occultamento dell'informazione. Il metodo speciale "main" e punti d'ingesso della JVM. Caricamento di package e appartenenza di una classe ad un package. Modificatori di accesso. Sintassi ed esempi di implementazione delle eccezioni in Java. Metodi anonimi: le lambda-espressioni in Java.
- Documentazione del codice in Java: caratteristiche di javadoc, tag, generazione della documentazione.

Testi consigliati

[1] Maurizio Gabbrielli, Simone Martini, *Linguaggi di programmazione - Principi e paradigmi, 2/ed*. McGraw-Hill, (2011).
[2] Dean Wampler, Alex Payne, *Programming Scala: Scalability = Functional Programming + Objects, 2 edizione*. O'Reilly Media, (2014).
[3] David Parsons, *Foundational Java Key Elements and Practical Programming*. Springer-Verlag, (2012).

Bibliografia supplementare

[4] Kip R Irvine, *Assembly Language for X86 Processors*. Pearson, (2015).
[5] Bruce Tate, *Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages Pragmatic Bookshelf*. (2010).
[6] Daniel P. Friedman, Mitchell Wand, *Essentials of Programming Languages*. MIT Press, (2008).

Modalità d'esame

- valutazione in itinere ("esoneri")		■ SI	☐ NO
- esame finale	scritto	■ SI	☐ NO
	orale	■ SI	☐ NO
- altre prove di valutazione del profitto (meglio descritte sotto)		■ SI	☐ NO

L'esame consiste di due parti: un esame scritto e della discussione di una progetto di programmazione object oriented in Java.

La prova orale è prevista per riparare le insufficienze lievi o per migliorare la valutazione delle prove scritte.

L'argomento del progetto di programmazione deve essere concordato con il docente e va presentata solo dopo il superamento dell'esame scritto.

Le due prove di esonero sostituiscono la prova scritta. Il voto finale si compone al 65% del voto del secondo esonero e al 35% dalla media aritmetica tra il voto del primo esonero e il voto del progetto di programmazione.

Nel caso di studenti che non sono esonerati il voto finale si compone al 65% dal voto dello scritto e al 35% dal voto del progetto di programmazione.