Principali informazioni sull'insegnamento	
Titolo insegnamento	Programmazione II (B)
Corso di studio	Informatica e Tecnologie per la Produzione del Software
Crediti formativi	9
Denominazione inglese	Computer Programming II (B)
Obbligo di frequenza	
Lingua di erogazione	Italiano

Docente responsabile	Nome Cognome	Indirizzo Mail
	Nicola Fanizzi	nicola.fanizzi@uniba.it
Luogo ed Orario di Ricevimento	Dip. Informatica 5° Piano – stanza 522	Martedì dalle 15:00 alle 17:00 o per appuntamento

Dettaglio crediti formativi	Ambito disciplinare	SSD	Crediti
	INFORMATICO	ING-INF/05	7
	"	ING-INF/05	2

Modalità di erogazione	
Periodo di erogazione	Primo Semestre
Anno di corso	Secondo Anno
Modalità di erogazione	Lezioni frontali Esercitazioni guidate in aula e Laboratorio

Organizzazione della didattica	
Ore totali	225
Ore di corso	86 (56+30)
Ore di studio individuale	139 (119+20)

Calendario	
Inizio attività didattiche	25 Settembre 2017
Fine attività didattiche	12 Gennaio 2018

Syllabus	
Prerequisiti	Competenze di base relative alla programmazione (in C) e ai linguaggi di programmazione
Risultati di apprendimento previsti (declinare rispetto ai Descrittori di Dublino) (si raccomanda che siano coerenti con i risultati di apprendimento del CdS, compreso i risultati di apprendimento trasversali)	• ***Conoscenza e capacità di comprensione*** Concetti e competenze di programmazione. In particolare, comprensione concettuale e consapevolezza del ruolo centrale di algoritmi e strutture dati nel contesto del paradigma di riferimento, conoscenze atte progettare unità strutturali dal punto di vista della programmazione "in the large", consapevolezza dell'importanza del riuso del codice, della generalità delle interfacce di comunicazione, della robustezza del codice prodotto.

	• ***Conoscenza e capacità di comprensione applicate*** Applicazioni pratiche che partono dalle tecniche di base apprese in altri corsi fino ad arrivare a caratteristiche avanzate che rappresentano recenti sviluppi nell'ambito del paradigma acquisiti nello stesso linguaggio (Java) o in altri; esposizione ad una gamma di problemi applicativi che richiedono il collegamento delle competenze a livello teorico con la pratica dello sviluppo in un contesto reale. • ***Autonomia di giudizio*** Capacità di scelta di tecniche algoritmiche e modellazione dei dati atte alla soluzione di problemi; capacità di interpretazione dei dati sulle prestazioni delle soluzioni realizzate attraverso test empirici • ***Abilità comunicative*** Abilità nel trasmettere le nozioni apprese. Abilità di comunicazione su temi specialistici, ad es. in un team di sviluppo, affinate attraverso il lavoro in gruppo nelle esercitazioni pratiche • ***Capacità di apprendere*** Lo studente dovrà mostrare di aver sviluppato capacità di apprendere e di orientarsi agilmente nell'ambito degli sviluppi del paradigma attraverso le novità degli strumenti tecnologici a supporto.
Contenuti di insegnamento	**Riepilogo** sui concetti di programmazione di base: Introduzione ai computer e a JAVA (1/E; 1/S); Concetti e nozioni di base di programmazione (1-2/E; 1-2/S); Flussi di controllo: la selezione e i cicli (3/E; 3-4/S); **I metodi**: concetti di base (4/E; 5/S); Definizione e invocazione di metodi; Linee guida sulla scrittura dei metodi **Gli array** (7/E; 6/S): Concetti di base sugli array; Utilizzare gli array nei metodi; Ordinamento e ricerca con gli array; Array multidimensionali **Ricorsione** (9/E; 7/S): Le basi della ricorsione; Programmare utilizzando la ricorsione **La programmazione a oggetti**: Definizione di classi (5/E; 8/S); Information hiding e incapsulamento (5/E; 8/S); Oggetti e riferimenti (5/E; 8/S); Costruttori (5/E; 9/S); Variabili e metodi statici (4,5/E; 9/S); Overloading (5/E; 9/S); Enumerazioni come classi (2,10/E;. 9/S); La strutturazione in package (4/E; 9/S); Concetti di base sull'ereditarietà (5/E; 10/S); Incapsulamento ed ereditarietà (5/E; 10/S); Polimorfismo (5/E; 11/S); Classi astratte (5/E; 11/S), Interfacce (5/E; 11/S); Rappresentazione in UML delle classi e delle loro relazioni (9,11/S) **La gestione delle eccezioni** (8/E; 13/S): Concetti di base sulle eccezioni; Definizione di nuove eccezioni; Approfondimenti sulle classi di eccezioni

	Stream e I/O da file (11/E; 14/S): Introduzione ai flussi dati e all'I/O su file; I/O con file di testo; Tecniche generiche per la gestione dei file; I/O su file binari **Strutture dati dinamiche** (9-10/E; 15/S): Liste concatenate, pile e code; Tabelle Hash; Insiemi; Alberi **Collezioni, mappe e iteratori**: Le collezioni in JAVA (10/E; 12,16/S); Iteratori (10/E; 16/S) **Possibili approfondimenti**: cenni sulle annotazioni; estensioni funzionali di Java 8 e 9; RTTI Legenda (indice capitoli): #/E --> Eck #/S --> Savitch

Programma	
Testi di riferimento	• D.Eck Introduction to Programming Using Java. 7.0.2/e. 2015-17 • W. Savitch: Java: An Introduction to Problem Solving and Programming. 7/e. Pearson. 2015 (cfr. ed. italiana) • J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley: The Java® Language Specification – Java SE 8 Edition. 2015
Note ai testi di riferimento	Integrabili anche dalla doc. ufficiale Java http://docs.oracle.com/javase/specs/ e da altro materiale fornito dal docente
Metodi didattici	Lezioni frontali ed esercitazioni pratiche con l'utilizzo di IDE per la programmazione a oggetti e il test delle unità realizzate. Uso della piattaforma di E-learning per esercizi da svolgere autonomamente.
Metodi di valutazione (indicare almeno la tipologia scritto, orale, altro)	• prova scritta: risposta a domande di carattere teorico-pratico sul paradigma • prova pratica: sviluppo e test di una soluzione ad un problema data una sua specifica
Criteri di valutazione (per ogni risultato di apprendimento atteso su indicato, descrivere cosa ci si aspetta lo studente conosca o sia in grado di fare e a quale livello al fine di dimostrare che un risultato di apprendimento è stato raggiunto e a quale livello)	• Padronanza a livello teorico/tecnico dei principali aspetti del Paradigma della *programmazione orientata agli oggetti* (in Java) tale da consentire una rapida integrazione in team di sviluppo che si avvalgano delle dette tecnologie nonché di essere in grado di trasferire ad altri le proprie competenze in veste di tutor / docente. • Capacità di soluzione di problemi (non completamente specificati in dettaglio) con la realizzazione di soluzioni attraverso unità di codice debitamente testato.
Altro	**Propedeuticità *obbligatorie***: Programmazione, Architettura

	degli elaboratori e Sistemi Operativi, Laboratorio di Informatica ***consigliate***: Linguaggi di programmazione