UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO	Matematica e Informatica
ANNO ACCADEMICO OFFERTA	2022/2023
ANNO ACCADEMICO EROGAZIONE	2023/2024
CORSO DILAUREA	INFORMATICA
INSEGNAMENTO	LINGUAGGI DI PROGRAMMAZIONE
TIPO DI ATTIVITA'	B
AMBITO	50166-Discipline Informatiche
CODICE INSEGNAMENTO	04758
SETTORI SCIENTIFICO-DISCIPLINARI	INF/01
DOCENTE RESPONSABILE	FICI GABRIELE Professore Associato Univ. di PALERMO
ALTRI DOCENTI	
CFU	9
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	153
NUMERO DI ORE RISERVATE ALLA DIDATTICA ASSISTITA	72
PROPEDEUTICITA'	05880 - PROGRAMMAZIONE E LABORATORIO C.I.
MUTUAZIONI	
ANNO DI CORSO	2
PERIODO DELLE LEZIONI	1° semestre
MODALITA' DI FREQUENZA	Facoltativa
TIPO DI VALUTAZIONE	Voto in trentesimi
ORARIO DI RICEVIMENTO DEGLI STUDENTI	**FICI GABRIELE** Martedì 15:00 16:00 Studio 202, DMI. Giovedì 15:00 16:00 Studio 202, DMI.

DOCENTE: Prof. GABRIELE FICI

PREREQUISITI	Conoscenze di programmazione, matematica discreta (relazioni, funzioni, calcolo combinatorio), conoscenze di base di logica (logica proposizionale, tecniche di dimostrazione, principio di induzione matematica).
RISULTATI DI APPRENDIMENTO ATTESI	Conoscenza e capacità di comprensione: Conoscenza panoramica dei vari paradigmi di programmazione. Conoscenza approfondita del paradigma orientato agli oggetti e del paradigma funzionale. Conoscenza approfondita del linguaggio Java Standard Edition nelle sue ultime versioni. ---------- Capacità di applicare conoscenza e comprensione: Capacità di valutare le funzionalità dei diversi paradigmi di programmazione. Capacità di scrivere codice anche avanzato nel linguaggio di programmazione Java. ---------- Autonomia di giudizio: Capacità di valutare e comparare autonomamente le soluzioni di un problema di limitata complessità. ---------- Abilità comunicative: Capacità di organizzarsi in gruppi di lavoro. Capacità di comunicare efficacemente in forma orale, anche utilizzando termini in inglese, concetti propri dei linguaggi di programmazione. ---------- Capacità di apprendimento: Capacità di catalogare, schematizzare e rielaborare le nozioni acquisite. ---------- Competenze acquisite: Saper formalizzare un problema in termini risolvibili da un programma informatico. Saper scegliere il linguaggio di programmazione adeguato per scrivere un programma, in base alle caratteristiche del problema e dei paradigmi di programmazione disponibili. Saper organizzare il codice sorgente di un programma, anche mediante l'uso di strumenti di sviluppo integrati. Saper scrivere codice sorgente corretto, efficace e sicuro. Saper valutare la correttezza del codice. Saper individuare e correggere gli errori di programmazione. Saper valutare la qualità del programma prodotto.
VALUTAZIONE DELL'APPRENDIMENTO	L'esame finale consiste in una prova pratica e una prova orale. La prova pratica è costituita dall'assegnazione di un piccolo progetto individuale da sviluppare in Java. Verrà richiesto di sviluppare un programma stand alone in Java SE, diviso in classi, a partire da una descrizione e una lista di requisiti funzionali e non funzionali. La valutazione della prova pratica terrà conto dei seguenti criteri di valutazione: i) correttezza dell'organizzazione complessiva del progetto in base ai requisiti, ii) correttezza, chiarezza e qualità del codice sorgente e della documentazione, iii) coerenza con i requisiti e usabilità del programma. La prova orale, alla quale si accede solo dopo avere superato la prova pratica, consiste in un colloquio che verte ad approfondire il livello di profondità dell'acquisizione dei concetti teorici introdotti durante il corso. Concorrono a determinare il voto: 1) Il risultato della prova pratica; 2) La chiarezza e correttezza delle risposte al colloquio orale; Fasce di valutazione: Il voto va da 18/30 (prova pratica sufficiente, conoscenze minime sul programma, valutate in base alle risposte fornite alle domande rivolte al colloquio orale) a 30/30, con eventuale lode (prova pratica eccellente, ottime conoscenze sul programma, valutate in base alle risposte fornite alle domande rivolte al colloquio orale).
OBIETTIVI FORMATIVI	Fornire competenze di base e avanzate, sia metodologiche che tecniche, sui paradigmi dei linguaggi di programmazione, in particolar modo sul paradigma orientato agli oggetti e su quello funzionale.
ORGANIZZAZIONE DELLA DIDATTICA	Lezioni frontali in laboratorio. Gli studenti avranno a disposizione dei computer per seguire le lezioni in modo interattivo. La frequenza delle lezioni è fortemente raccomandata.
TESTI CONSIGLIATI	1) C. De Sio Cesari, Il nuovo Java. Guida completa alla programmazione moderna, Hoepli, ISBN: 9788820399306 (per la parte su Java - for the part on Java); 2) C. Horstmann, Concetti di informatica e fondamenti di Java, VII edizione, Apogeo, ISBN: 9788891639431; (per la parte su Java - for the part on Java); 3) Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft, Modern Java in Action: Lambdas, Streams, Functional and Reactive Programming, I edizione, Manning, ISBN: 9781617293566 (per la parte su programmazione funzionale - for the part on functional programming)

PROGRAMMA

ORE	Lezioni
2	Introduzione e Primi Passi in Java: Sintassi di base - Primo programma - Commenti al codice - Letterali - Dichiarazione di variabili - Identificatori - Tipi primitivi: byte, short, int, long, float, double, boolean, char - Operatori unari, binari, condizionali e di concatenazione
3	Le classi Java I: Classi e oggetti - Attributi e metodi - Convenzioni su maiuscole e minuscole - Il metodo main - Costruttori - L'operatore new - Il costruttore di default - Riferimenti - Confronto di riferimenti - Variabili di istanza e variabili locali - Accesso ad attributi e metodi delle classi con dot notation - La Javadoc.
3	Le classi Java II: I metodi print, println e printf - Incapsulamento dei dati - Interfaccia pubblica e interfaccia privata - Modificatori di visibilita' public e private - Il riferimento implicito this - Deallocazione automatica della memoria e Garbage Collector - Firma di un metodo - Overloading di metodi - Overloading di costruttori - Invocazione di altri costruttori mediante "this()" - Variabili final
3	Le classi Java III: Attributi static - Metodi static - Organizzazione e compilazione di un package - Visibilita' package-private - Creazione di file jar eseguibili
2	Controllo del flusso di esecuzione: Costrutti di selezione, iterazione e salto: if, switch, while, do-while, for, break, continue, return - Il for esteso
6	Stringhe e Wrapper: Il tipo String - Istanziazione di stringhe - Il pool di stringhe - La codifica Unicode per i caratteri - L'operatore di concatenazione - Il metodo equals per le stringhe - Il metodo compareTo per le stringhe - I metodi per le stringhe: startsWith e endsWith, length, substring, contains - toLowerCase e toUpperCase, trim, charAt, indexOf, lastIndexOf, replaceFirst, replaceAll, valueOf - Espressioni regolari - I metodi che usano le espressioni regolari: matches, replaceFirst, replaceAll, split - La classe StringBuilder - I metodi append, insert, delete, deleteCharAt, replace, reverse - La classe Scanner e i suoi metodi: next, nextLine, nextInt, nextDouble - Le classi wrapper - Boxing e unboxing - I metodi isDigit e isLetter della classe Character - I metodi parseInt, parseShort, parseFloat - I metodi toBinaryString, toOctalString e toHexString
4	Array e ArrayList: Creazione e inizializzazione di un array - Array di riferimenti a oggetti - L'attributo length - Il for esteso sugli array - I metodi della classe Arrays - La classe ArrayList - Cenni su Generics e classi parametriche - I metodi per gli ArrayList: add, size, get, set, remove, indexOf, lastIndexOf, contains, toArray - I metodi della classe Collections
3	Metodi Universali: I metodi presenti in tutte le classi - I metodi equals, toString e hashCode - Overriding del metodo toString - Overriding del metodo equals - Funzioni hash - Collisioni e hashing perfetto - Bucket e tabelle hash - Overriding del metodo hashCode - Mutua compatibilita' di questi overriding.
4	Ereditarieta' I: Riutilizzazione e distribuzione dei programmi Java - Sottoclassi e superclassi - La parola chiave extends - Ereditarieta' di attributi e metodi - Il modificatore protected - Overriding di metodi - L'annotazione @Override - Attributi adombrati (shadowing) - Classi e metodi final
4	Ereditarieta' II: Ereditarieta' e costruttori - Invocare costruttori della superclasse con "super()" - Il riferimento super - Ereditarieta' e membri static - Ereditarieta' singola e Diamond Problem - Subtyping
4	Polimorfismo: Type checking statico - Binding dinamico - Definizione di polimorfismo - polimorfismo per metodi e polimorfismo per dati - array polimorfi - metodi virtuali e attributi adombrati - Il costrutto instanceof
6	Classi astratte e interfacce: Il modificatore abstract - Classi astratte e sottoclassi concrete - Classi astratte e membri static - Definizione e dichiarazione di un'interfaccia - Contenuto di un'interfaccia - Implementare un'interfaccia - Conflitto di nomi di variabili tra un'interfaccia e una classe che la implementa - Implementazioni parziali di interfacce in classi astratte - Estensione di interfacce - Ereditarieta' multipla per interfacce - Diamond problem, shadowing, overriding e overloading nell'ereditarieta' multipla per interfacce - Riferimenti a un'interfaccia - Interfacce ed ereditarieta' - Interfacce e polimorfismo - Focus sull'interfaccia Comparable - Comparatori e interfaccia Comparator - Implementazione di comparatori - Classi interne e classi interne anonime - Implementazione di un'interfaccia in una classe interna anonima - Tipi enumerativi
6	Input/output: Flussi (stream) di dati - Dati in ingresso e dati in uscita - Flussi di caratteri e flussi di byte - La classe File - I metodi della classe File: delete, exists, getAbsolutePath, getName, getPath, isDirectory, isFile, length, list, mkdir e renameTo - Usare Scanner per leggere dati da un file testuale - I metodi hasNext e hasNextLine, next e nextLine - Gestione di flussi di byte: le classi astratte InputStream e OutputStream e le loro sottoclassi FileInputStream e FileOutputStream - I metodi read e write per FileInputStream e FileOutputStream - Gestione di flussi di caratteri: le classi astratte Reader e Writer e le loro sottoclassi - I metodi read e write per FileReader e FileWriter - Bufferizzazione: le classi BufferedReader e BufferedWriter - La classe PrintWriter Il metodo printf - Filtraggio di classi - Flussi di oggetti - Le classi ObjectInputStream e ObjectOutputStream - Serializzazione e deserializzazione - L'interfaccia Serializable - I metodi readObject e writeObject - Serializzazione e membri static - Il modificatore transient - Memorizzazione dei dati di un programma su file - Gestione delle eccezioni - Eccezioni controllate e non - Il costrutto try/catch - Il try with resources
6	Programmazione generica - Tipi parametrizzati - Collezioni - Java Collection Framework - Le interfacce Iterable, Collection, Set e List - Le classi HashSet, TreeSet, ArrayList e LinkedList - Le interfacce Map e SortedMap - Le classi HashMap e TreeMap
4	Programmare un'interfaccia grafica per un'applicazione Java - AWT e Swing - Delegation model e gestione degli eventi - Ascoltatori - I componenti di Swing
6	Interfacce funzionali - Espressioni lambda - Method references - Stream - La streaming API di Java - Operazioni intermedie e terminali - Operazioni di aggregazione e di riduzione - La classe Collectors
6	Il paradigma di programmazione funzionale - Cenni storici - Funzioni di ordine superiore - Currying - Chiusure - Trasparenza referenziale (determinismo, assenza di effetti collaterali, assenza di mutazioni di stato) - Valutazione lazy e valutazione eager - Java e la programmazione funzionale - Stile funzionale e programmazione ricorsiva in Java - Programmazione funzionale pura in Java: tecniche e limiti