metodi variadici

Un metodo variadico è un particolare tipo di metodo in cui è possibile inserire un numero arbitrario di parametri dello stesso tipo, essi verranno memorizzati in un array.
Per un corretto funzionamento, il parametro variadico deve essere l'ultimo nella lista dei parametri formali, può essere preceduto da parametri non variadici e può corrispondere a $\emptyset$ o più parametri.

metodi sovraccaricati

I metodi sovraccaricati sono metodi, aventi lo stesso nome ma firma differente, con cui è possibile effettuare la stessa operazione o ottenere lo stesso risultato in modi differenti.

metodi generici

I metodi generici sono particolari metodi che prendono dei tipi come parametri, in questo modo possiamo utilizzare uno stesso metodo per più tipi di dato

ereditarietà

L'ereditarietà permette lo sviluppo incrementale riutilizzando e specializzando le funzioni esistenti e aggiungendone di nuove. In Java l'ereditarietà è singola: una sorta di ereditarietà multipla è ottenibile implementando le interfacce. Una sottoclasse di una classe eredita tutti i membri di quest'ultima più quelli delle superclassi, non eredita costruttori e inizializzazioni. In questo modo la sottoclasse estende la classe con nuovi membri.

binding dinamico

Il binding dinamico consiste nell'esecuzione versioni differenti a seconda della sottoclasse che lo invoca, questo è possibile grazie alla sovrascrittura dei metodi nelle sottoclassi

NOTA: la sovrascrittura deve mantenere la firma, il tipo del risultato (o uno compatibile) e la visibilità (uguale o maggiore) del metodo da sovrascrivere.

Accedere alla superclasse da una sottoclasse

È possibile accedere campi/metodi di una superclasse attraverso super, essa funziona in modo analogo a this, denotando l'istanza corrente ma vista come istanza di una superclasse. A differenza di this però, super è utilizzabile solo con la dot notation e non può essere iterato. Super viene solitamente utilizzato per accedere a metodi sovrascritti altrimenti non accessibili.
Esiste anche l'invocazione di un costruttore di una superclasse che funziona in modo analogo a this(), ovvero super().

NOTA: super() e this() non possono essere presenti all'interno dello stesso costruttore.
Ogni costruttore senza this() o super() invoca implicitamente il costruttore senza argomenti della superclasse.

NOTA: Se la superclasse è Object, super() non fa nulla, altrimenti se la superclasse non ha un costruttore senza argomenti ci sarà un errore di compilazione.

classi astratte

Le classi astratte sono classi non istanziabili e sono utili per raggruppare classi concrete in un unico tipo. Una classe astratta, per essere tale, deve contenere almeno un metodo astratto, ossia un metodo senza implementazione.
Le classi che estendono classi astratte devono implementare tutti i metodi astratti.

Programmazione a oggetti

le classi

Le classi sono collezioni di variabili e metodi che servono per manipolare gli oggetti (le relative istanze), essa è identificata da un nome e può estendere altre classi o implementare interfacce. Una classe è detta generica quando sono presenti i cosiddetti parametri di tipo, verranno poi rimpiazzati da un tipo specifico al momento della dichiarazione. Una classe ammette i seguenti modificatori:

- public: visibile in tutto il programma;
- default (niente): visibile solo nel package in cui è stata;
- abstract, final: legati all'ereditarietà;
- protected, private e static: utilizzate nelle classi membro.

Nel corpo della classe sono presenti:

- variabili/metodi di classe: legate alla classe e possono interagire solamente con variabili/metodi statici;
- variabili/metodi d'istanza: sono legati alla singola istanza della classe e quindi è possibile modificarne lo stato.
- costruttori: sono metodi speciali, richiamabili con new, che permettono d'istanziare un oggetto e inizializzarlo.

variabili e metodi nella classe

una variabile è un contenitore in cui è possibile inserire un valore o un riferimento a un oggetto.
Un metodo è una sequenza d'istruzioni volte a risolvere un determinato compito restituendo o meno un valore al chiamante

variabili e metodi possono essere:

- di classe: precedute da static e riferite alla classe stessa, non possono modificare lo stato degli oggetti
- d'istanza: riferiscono ai singoli oggetti

In più possono esserci altri modificatori:

- final per renderla costante;
- public, private e protected determinano l'accessibilità;
- transient: il campo non è parte dello stato dell'oggetto;
- volatile: un campo è accessibile da più thread.

Per accedere ai singoli campi si utilizza la dot notation.

i costruttori

i costruttori sono metodi speciali, richiamabili con new, che completano l'inizializzazione di un oggetto. Un costruttore ammette come modificatori solo public, private e protected.
Nel caso non ci siano costruttori nella classe, Java fornisce un costruttore di default con zero argomenti. Dato che in una classe possono esserci più costruttori o più classi, si applica il cosiddetto overloading, cioè l'esecuzione di un determinato metodo/costruttore in base alla firma (composta dal nome più i parametri).
È possibile invocare un costruttore all'interno di un altro costruttore con this(): a seconda dei parametri dati, invoca il costruttore adeguato, alla fine ritorna a finire quello precedente.
NOTA: this() deve essere la prima istruzione del costruttore per essere utilizzato.

incapsulamento

l'incapsulamento consiste nel definire un interfaccia con cui colloquiare con le classi e le sue istanze.

Creare un eccezione

La creazione di un eccezione è analoga alla creazione di una classe, essa però deve estendere la classe Exception. Ogni oggetto Throwable contiene una rappresentazione della pila degli ambienti al momento della creazione dell'istanza, può anche contenere anche una descrizione e contiene anche una causa: se diversa da null, è riferita a un'altra eccezione ed è responsabile dell'anomalia.

Lanciare un eccezione

Un eccezione può essere lanciata attraverso il comando throw.
Quando viene lanciata un eccezione in un metodo, quest'ultimo termina in modo anomalo e propaga l'eccezione al chiamante. Al termine il programma stampa tipo e descrizione dell'eccezione e tutti i metodi che l'hanno rilanciata.

Catturare un eccezione

Per catturare un eccezione, si utilizza il costrutto try-catch: il blocco try permette permette intercettare l'eccezione che, quando lanciata, viene catturata e viene eseguito il blocco catch corrispondente all'eccezione. In questo costrutto vi è anche un blocco finally, il quale verrà sempre eseguito dopo il termine di try o catch, anche in presenza di return nei blocchi. La cattura delle eccezioni è utile per evitarne la propagazione e quindi l'arresto anomalo del programma.

Tipi d'eccezioni

Le eccezioni possono essere:
- non controllate: eventi fatali o ristorabili con dei controlli (Error, RuntimeException);
- controllate: eventi non evitabili coi controlli.

Clausola throws

La clausola throws serve a informare che un determinato può sollevare eccezioni durante l'esecuzione.

NOTA: per essere sollevata, l'eccezione dev'essere sottoclasse di Throwable.

Classi annidate

Una classe è annidata se è definita all'interno di un ulteriore classe, essa può essere statica oppure no. Una classe membro statica differisce da quella principale solo per la visibilità, infatti può essere protected o private e può essere riferita esternamente coi nomi di entrambe le classi. La classe membro può utilizzare i metodi statici di quella esterna mentre la top-level può vedere i membri di quella interna. Le classi membro non statiche devono avere le istanze associate a ulteriori istanze della classe top-level, può essere associata a membri della top-level e non può contenere metodi statici.

Classi anonime

Una classe è anonima quando viene dichiarata dentro l'espressione new. Un possibile uso di queste è nelle interfacce grafiche in cui ogni evento prende in input oggetti che in genere generano un solo oggetto.
È possibile inserire anche parametri di metodo nelle classi anonime.

le interfacce

le interfacce hanno una struttura simile a quella di una classe astratta con la differenza che i metodi all'interno sono tutti astratti. A differenza delle classi astratte, una classe può implementare più interfacce. Ogni interfaccia introduce un nuovo tipo riferimento le cui istanze sono tutti gli oggetti delle classi che la implementano. Un'interfaccia all'interno di un altra interfaccia è detta interfaccia membro. I modificatori ammessi sono public, protected, private, static, abstract. Ogni interfaccia è implicitamente abstract mentre static, protected e private sono utilizzate nelle interfacce membro. Tutti i membri di un interfaccia sono implicitamente abstract, i campi sono implicitamente static e final mentre tutti i membri sono implicitamente abstract. Come per le classi, le interfacce possono avere parametri di tipo, vengono dette interfacce generiche.

classi generiche

Le classi generiche sono classi in cui è presente un parametro di tipo, in questo modo è possibile utilizzare la classe con più tipi di dato.

costruttori generici

Un costruttore è generico quando vi è un parametro di tipo nel nome, non è necessario che la classe sia generica per esserlo e, all'invocazione, è possibile associare un tipo riferimento al parametro di tipo. Nel caso mancasse, il sistema inferisce il tipo corretto.

possibili utilizzi delle interfacce

Le interfacce possono avere vari ruoli:
- Tipi di dati astratti (TDA): modella entità matematiche insieme a specifiche operazioni;
- contratti (CTR): disaccoppiano l'uso delle classi dalla loro realizzazione;
- meccanismi d'astrazione (MdA): raccolgono funzionalità comuni a più classi
- proprietà (PROP): rappresentano un predicato che una classe può soddisfare o meno.

tipi generici e parametrizzati

Interfacce e classi generiche definiscono tipi generici utili per controllare strutture complesse indipendentemente dagli elementi che contengono. La struttura e le funzionalità vengono implementate una sola volta e vengono instanziate molte volte attraverso i tipi parametrizzati.

eccezioni

l'eccezione è una situazione che modella comportamenti eccezionali e imprevedibili durante l'esecuzione del programma, che richiedono attenzione e possono portare a cambiamenti nel controllo di flusso
Java permette di:
- definire nuove eccezioni;
- crearne istanze;
- lanciarle;
- catturarle;
- propagarle al chiamante.

Tutte le eccezioni sono sottoclassi di Throwable, ma ha due sottoclassi dirette:
- Error: errori fatali difficili da gestire;
- Exception: errori relativi a determinate azioni gestibili.

Exception ha la sottoclasse RuntimeException, che raccoglie le eccezioni risolvibili con normali controlli