Java

Modificatori di classe

Parola chiave	Significato
abstract	Deve essere estesa da un'implementazione.
final	Non può essere estesa da un'implementazione.
strictfp	Il codice della classe usa semantica FP

Modificatori di metodo

Parola chiave	Significato
abstract	Deve essere implementato da una classe di estensione.
static	Legato alla classe e non ad una istanza.
final	Non può essere reimplementato da una classe di estensione.
native	Implementato da una libreria nativa.
strictfp	Il codice del metodo usa semantica FP restrittiva.
synchronized	Il metodo può essere usato da un solo thread per volta.

Visibilità

Modificatore	Classe	Package	Sottoclasse	Universo
Public	V	V	V	V
Protected	V	X	V	X
default	V	V	X	X
Private	V	X	X	X

Eccezioni

Tutte le eccezioni derivano dalla classe **Throwable**.

- **Exception**: gli errori nonostante i quali il programma dovrebbe essere in grado di proseguire.
- **Error**: gli errori dai quali il programma non è in grado di proseguire.
- **RuntimeException**: rappresenta ogni errore che può avvenire durante la normale valutazione di espressioni. Sono dette unchecked exceptions
- Tutte le altre sono dette checked exception e devono essere dichiarate nella definizione di un metodo.

Static nested classes

Una classe static non ha un accesso privilegiato ai membri (statici o meno) della classe ospite.

Inner classes

Ha lo stesso ciclo di vita, ed ha un riferimento privilegiato all'oggetto ospitante, non può dichiarare membri static ma solo membri di istanza.

Inizializzazione

- Inizializzatori statici
- Supercostruttore
- Inizializzatori di istanza

Interfaccia

Alle interfacce viene permesso di avere dei default method, cioè dei metodi implementati che si comportano in modo simile a quello delle superclassi.

Per mezzo dei default method una interfaccia può essere modificata con nuovi metodi senza che le implementazioni debbano essere toccate; se il metodo nuovo non è gestito dalla classe, viene usato quanto dichiarato nell'interfaccia.

Il Diamond Problem viene rilevato al momento della compilazione: se la gerarchia di ereditarietà ed implementazioni di una classe porta ad una ambiguità nella selezione di un metodo, il compilatore segnala un errore, che può essere risolto solo modificando il codice.

Annotazioni

Il loro scopo è arricchire di metadati la struttura a cui sono applicate, in modo da consentirne la rilevazione e l'uso durante la compilazione o l'esecuzione.

Le annotazioni possono avere parametri (solo di determinati tipi) e anche metodi (che possono ritornare lo stesso limitato insieme di tipi). Il compilatore può eseguire degli Annotation Processors che, durante la compilazione, possono produrre nuovi file (nuovi sorgenti, nuove classi, o qualsiasi altro tipo di file) a partire da annotazioni presenti nel codice.

Lambda expression

```
(<lista-parametri>) -> istruzione
```

Espressione Switch

Non c'è fall-through. L'elenco dei casi deve essere esaustivo.

Try with resources

Permette di dichiarare delle variabili. queste devono implementare l'interfaccia AutoClosable, e verranno automaticamente , e con certezza, "chiuse". In questa forma le clausole catch e finally sono entrambe opzionali.

Streams

A molto interfacce è stato aggiunto il metodo stream() che permette di trattare le collezioni con questa metafora.

Le operazioni sugli Stream vengono composte in sequenza, in una pipeline, fino ad arrivare ad una operazione detta terminale che produce il risultato.

Le operazioni intermedie sugli stream di dividono in stateful e stateless. Il loro uso influenza la costruzione e l'efficienza della pipeline che le contiene.

Il codice che implementa la pipeline ha ampie libertà su come riordinare e disporre l'esecuzione delle operazioni intermedie. Queste ultime devono:

- non interferire, cioè non modificare gli elementi dello stream
- (nella maggior parte dei casi) non avere uno stato interno