DOMANDE TEORIA ESAMI PASSATI

- **ESAME 13-01-2023**

Domanda 1 (5 punti): Commenti e Javadoc Si spieghi come può essere commentato il codice Java, e qual è la differenza sintattica (ovvero su come vengono scritti all'interno del codice) e semantica (ovvero per il ruolo che ricoprono) tra commenti normali (su linea singola o multipla) e commenti che invece seguono lo standard Javadoc.

Risposta Completa:

• 1 punto: commenti normali con // oppure /*..*/

• 1 punto: commenti Javadoc con /** .. */

• 1 punto: i commenti normali "spariscono" dopo la compilazione

• 1 punto: i commenti Javadoc sono parte integrante della documentazione del codice

• 1 punto: i commenti Javadoc sono strutturati e possono contenere diversi tag

Domanda 2 (5 punti): Override di metodi e final. Si spieghi che cosa significa fare l'override di un metodo. Si spieghi poi come può essere utilizzato il modificatore final in modo da limitare ciò che una sottoclasse può ridefinire della sua sovraclasse.

Risposta Completa:

• 1 punto: override quando un metodo di una sottoclasse ridefinisce un metodo ereditato

• 1 punto: l'override si ottiene implementando un metodo con la stessa firma di un metodo

ereditato

• 1 punto: in questo contesto, final può essere applicato a classi e metodi (sui campi non ha a che vedere con l'ereditarietà)

• 1 punto: final su una classe semplicemente impedisce che ci sia una sottoclasse

• 1 punto: final su un metodo impedisce l'override di un metodo

Domanda 3 (5 punti): Invocazione di un metodo tramite reflection. Si spieghi come si può ottenere un oggetto di tipo Method e come questo possa essere invocato utilizzando i meccanismi di reflection di Java. Si discutano poi le tipologie di errori che si possono ottenere in seguito all'invocazione di tale metodo.

Risposta Completa:

• 1 punto: tramite un oggetto Class possiamo accedere a tutti metodi della classe

• 1 punto: tramite un oggetto Class possiamo chiedere un metodo particolare specificandone

la firma

• 1 punto: tramite il metodo invoke possiamo invocare un metodo

• 1 punto: tramite l'invoke dobbiamo passare tutti i parametri del metodo invocato

• 1 punto: se non abbiamo accesso al metodo o il metodo è astratto/non esiste, otteniamo delle eccezioni ad hoc.

- ## ESAME 14-01-2023

Domanda 1 (5 punti): Modificatori di accesso (access modifier) Si elenchino tutti i modificatori di accesso (access modifier in inglese) supportati da Java, spiegando quali restrizioni sono imposte da ciascuno.

Risposta Completa:

• Elenco completo di tutti e 4 gli access modifiers (public, private, protected, e default/package) e nessun altro modifier di altro tipo

• Descrizione corretta modificatore public (accessibile da ovunque)

• Descrizione corretta modificatore private (accessibile solo all'interno della classe)

• Descrizione corretta modificatore protected (accessibile dal package e dalle sottoclassi)

• Descrizione corretta modificatore default/package (accessibile dal package)

Domanda 2 (5 punti): Annotazioni Si illustri che cos'è un'annotazione in Java, come può essere definita, come si possono restringere le componenti del programma ad oggetti a cui applicare una annotazione, e come si può definire se l'annotazione deve essere visibile solo sul codice sorgente, anche all'interno del bytecode, oppure anche durante l'esecuzione.

Risposta Completa:

• Spiegazione corretta ad alto livello di cos'è un'annotazione (tipo slide 2 lezione 19)

• Una annotazione può essere definita con @interface seguito dal nome dell'annotazione

• La definizione può contenere campi/attributi definiti con <tipo> <nome_campo>()

• @Target restringe le componenti del programma ad oggetti a cui applicare una annotazione

• @Retention definisce a che livello l'annotazione è visibile (codice sorgente, bytecode, esecuzione)

Domanda 3 (5 punti): Definizione e firma di un metodo. Si illustri in che cosa consiste la definizione di un metodo, la firma di un metodo, e quali sono le differenze. Si spieghi poi in che cosa consiste l'override di un metodo e che rapporto c'è con dichiarazione e firma di un metodo.

Risposta Completa:

• La firma è composta da nome metodo, numero di parametri e tipo (eventualmente classe che contiene il metodo)

• La dichiarazione comprende anche altro (return type, visibilità, lista delle eccezioni lanciate, …)

• L'override ridefinisce il comportamento di un metodo ereditato

• L'override significa definire un metodo con la stessa firma di un metodo ereditato

• Il resto della dichiarazione di un metodo di cui si fa override può variare ma deve essere meno "restrittivo".

- **ESAME 1-06-2023**

Domanda 1 (5 punti): Getter e setter. Si spieghi cosa sono i metodi getter e setter. Si discuta poi quali sono i vantaggi e svantaggi di usare tali metodi rispetto ad avere campi con pari visibilità.

Risposta Completa:

• 1 punto: getter ritorna il valore di un campo

• 1 punto: setter modifica il valore di un campo ricevendo tale valore come parametro

• 1 punto: potrebbero non lavorare su un campo ma sullo stato dell'oggetto

• 1 punto: il setter può controllare che il valore rispetti determinate condizioni

• 1 punto: come svantaggio richiedono l'invocazione di metodi, mentre la lettura e scrittura diretta è computazionalmente meno costosa.

Domanda 2 (5 punti): Tipaggio forte e statico, sottotipaggio. Si spieghi cosa significa che un linguaggio di programmazione come Java applica un tipaggio forte (strongly) e statico (static). Si spieghi poi in cosa consiste la relazione di sottotipaggio (subtyping) e come Java permette di definire che una classe è sottotipo di un altro tipo.

Risposta Completa:

• 1 punto: statico significa che il tipo di ciascuna espressione/variabile è noto a tempo di compilazione (tramite tipi dichiarati e regole di inference)

• 1 punto: strongly significa che un'espressione puo' essere assegnata a un'altra solo se e' "compatibile" (stesso tipo o sottotipo)

• 1 punto: la relazione di sottotipaggio significa che un sottotipo può "sostituire" un altro

• 1 punto: una classe che estende un'altra ne è sottotipo

• 1 punto: una classe che implementa un'interfaccia ne è sottotipo

Domanda 3 (5 punti): Boxing e unboxing. Si spieghi quali classi sono fornite da Java per poter effettuare operazioni di boxing e unboxing su valori numerici. Si spieghi poi in cosa consiste la conversione implicita di valori numerici, e quale comportamento si ha invece con valori boxed. Si individui poi un contesto in cui tale operazioni risultano essenziali.

Risposta Completa:

• 1 punto: una classe wrapper/mirror per ogni tipo primitivo

• 1 punto: una classe Number supertipo di tutti i numeri

• 1 punto: con la conversione implicita si puo' assegnare il valore di un tipo primitivo a una variabile di un altro tipo

• 1 punto: non è però possibile assegnare un tipo con "maggiore precisione" ad uno con minore (tipo un double a un float) a meno di non fare un casting/conversione esplicita

• 1 punto: con le classi wrapper/mirror non c'è conversione implicita da una all'altra.

- ## ESAME 30-06-2023

Domanda 1 (5 punti): Commenti di documentazione. Si spieghi quali sono le differenze tra commenti interni al codice e invece i commenti che servono da documentazione del codice. Si spieghi poi come Java supporta entrambe le tipologie di commenti, e come è possibile estrarre documentazione dai commenti.

Risposta Completa:

• 1 punto: commenti interni spariscono dopo la compilazione

• 1 punto: commenti di documentazione fanno parte dell'interfaccia esterna della libreria

• 1 punto: commenti interni con // e /*

• 1 punto: commenti esterni con /**

• 1 punto: javadoc costruisce la documentazione partendo dai commenti di documentazione

Domanda 2 (5 punti): abstract. Si spieghi su quali componenti di un programma orientato agli oggetti in Java si può usare la keyword abstract e con quali effetti.

Risposta Completa:

• 1 punto: su metodo e classe

• 1 punto: classe abstract non puo' essere istanziata

• 1 punto: metodo abstract non contiene il codice

• 1 punto: metodo abstract deve essere all'interno di una classe abstract

• 1 punto: non è necessario che una classe abstract contenga metodi abstract

Domanda 3 (5 punti): Annotazioni Si spieghi cosa sono le annotazioni in Java, come possono essere implementate, e quali caratteristiche di un'annotazione si possono definire e come.

Risposta Completa:

• 1 punto: meccanismo per aggiungere informazioni alle componenti di una classe accessibili a runtime tramite reflection

• 1 punto: @interface

• 1 punto: come una classe con i propri campi/attributi

• 1 punto: target

• 1 punto: retention

- **ESAME 6-09-2022**

Domanda 1 (5 punti): static. Si spieghi l'utilizzo del modificatore static, e in particolare su quali elementi di un programma Java può essere applicato e con quali effetti.

Risposta Completa (Nota: manca file con master solution, di seguito risposta da ChatGPT)

Il modificatore static in Java viene utilizzato per dichiarare un membro di classe che appartiene alla classe stessa anziché a un'istanza specifica di quella classe. Può essere applicato a vari elementi di un programma Java con effetti diversi. Ecco alcuni contesti in cui può essere utilizzato:

Variabili statiche (campi statici):

Quando viene applicato a una variabile di classe (campo statico), tale variabile è condivisa da tutte le istanze della classe. Ogni istanza condivide lo stesso valore di questa variabile statica. Viene allocata una sola copia della variabile statica per l'intera classe.

Metodi statici:

Quando applicato a un metodo, il metodo appartiene alla classe e non a un'istanza specifica di quella classe. I metodi statici possono essere chiamati senza creare un'istanza della classe e possono accedere solo a variabili statiche o altri membri statici della classe.

Blocchi di inizializzazione statici:

I blocchi di inizializzazione statici sono utilizzati per inizializzare variabili statiche o eseguire altre operazioni una sola volta quando la classe viene caricata in memoria. Sono eseguiti prima che venga creata un'istanza della classe o chiamato qualsiasi metodo statico.

Classi nidificate statiche:

Una classe può essere dichiarata statica se è una classe nidificata all'interno di un'altra classe. Una classe statica può accedere solo a membri statici della classe esterna.

L'utilizzo del modificatore static contribuisce alla gestione della memoria, all'efficienza e alla progettazione del codice, ma va utilizzato con attenzione poiché può portare a problemi di concorrenza in applicazioni multithreading.

Domanda 2 (5 punti): override ed overload. Si spieghi cosa si intende per override e per overload di un metodo, in quali casi si verifica il primo e in quali il secondo.

Risposta Completa:

Override:

L'override in Java si riferisce alla capacità di una classe derivata (sottoclasse o classe figlia) di fornire una propria implementazione di un metodo che è già definito nella classe base (superclasse o classe genitore). L'override avviene quando si dichiara un metodo nella classe derivata con la stessa firma (nome del metodo, tipo e numero dei parametri) di un metodo nella classe base. L'override è comunemente utilizzato per fornire un comportamento specifico per una classe derivata, garantendo che il metodo chiamato tramite un riferimento alla classe base eseguirà l'implementazione della classe derivata.

Overload:

L'overload si verifica quando una classe ha più di un metodo con lo stesso nome, ma con firme diverse (parametri diversi). Gli overload consentono a una classe di fornire diverse implementazioni di un metodo, o di fornire metodi con funzionalità simili ma con parametri diversi.

In breve, l'override si verifica quando una classe figlia fornisce un'implementazione specifica di un metodo della classe genitore, mentre l'overload si verifica quando una classe ha più di un metodo con lo stesso nome ma firme diverse. L'override è legato all'ereditarietà delle classi, mentre l'overload è legato alla firma dei metodi.

Domanda 3 (5 punti): implements ed extends. Si spieghi l'utilizzo delle keyword implements ed extends nel linguaggio di programmazione Java, ed in particolare in quali contesti possono essere utilizzate e che cosa specificano nei vari casi.

Risposta Completa:

Le keyword implements ed extends in Java sono utilizzate per definire relazioni tra classi e interfacce. Vediamo come vengono utilizzate e in quali contesti:

extends:

Utilizzato per la creazione di una relazione di ereditarietà tra classi. Una classe può estendere (inherits) da un'altra classe, ereditando così i suoi campi e metodi. La sottoclasse (classe figlia) può quindi estendere o sovrascrivere il comportamento della classe genitore (classe base).

implements:

Utilizzato per dichiarare che una classe implementa un'interfaccia. Un'interfaccia in Java è un contratto che specifica un insieme di metodi che una classe che implementa quell'interfaccia deve fornire. Quando una classe implementa un'interfaccia, deve fornire implementazioni concrete per tutti i metodi dichiarati nell'interfaccia.

In sintesi:

extends è utilizzato per stabilire un rapporto di ereditarietà tra classi.

implements è utilizzato per dichiarare che una classe implementa un'interfaccia.

Entrambe le keyword sono utilizzate principalmente per facilitare la progettazione del codice, favorire la riutilizzabilità e permettere una migliore organizzazione delle classi in un programma Java. La relazione extends è legata all'ereditarietà tra classi, mentre implements è legato all'implementazione di interfacce. Un'importante distinzione è che una classe può estendere una sola classe genitore, ma può implementare più di un'interfaccia.

- **ESAME 1-07-2022**

Domanda 1 (5 punti): Tipi valore e tipi referenza. Si spieghi cosa sono i tipi valore (value types) e si elenchino tutte le possibilità. Si spieghi poi cosa sono i tipi referenza (reference type) e che differenze ci sono tra tipi valore e tipi referenza.

Risposta Completa:

• 1 punto: tipi valore sono int double long float … (almeno 4)

• 1 punto: i tipi valore vengono passati per valore

• 1 punto: tipi referenza sono tutti quelli definiti da class o interface

• 1 punto: i tipi referenza sono tutti sottotipo di Object

• 1 punto: i tipi referenza vengono passati per referenza/indirizzo e creano quindi

problematiche di aliasing

Domanda 2 (5 punti): Eccezioni Si spieghi cosa sono le eccezioni in Java, come si puo' definire una propria eccezione, e quali costrutti Java supporta per lanciare, catturare e gestire le eccezioni.

Risposta Completa:

• 1 punto: quando lanciata, un'eccezione termina il flusso sequenziale di esecuzione ed entra

in una sorta di stato di errore

• 1 punto: classe che estende Exception/Throwable

• 1 punto: eccezioni lanciate con throw

• 1 punto: try {…} catch (E e) {…} cattura eccezioni di tipo E o sottotipo lanciate dal blocco nel

try

• 1 punto: finally viene sempre eseguito, se eccezione poi si riprende in stato "eccezionale"

Domanda 3 (5 punti): Reflection. Si spieghi in che cosa consiste la reflection in Java, e si elenchino le classi e metodi principali delle librerie Java per utilizzare la reflection.

Risposta Completa:

• 1 punto: le reflection permette di vedere la struttura del programma (classi, metodi, campi, …)

• 1 punto: class Class, Method, Field

• 1 punto: getField/Method/Constructor

• 1 punto: setField per assegnare un campo, invoke method

• 1 punto: vedere come il codice è annotato

- **ESAME 3-06-2022**

Domanda 1 (5 punti): Modificatori final. Si elenchino tutti i possibili usi del modificatore final in Java.

Risposta Completa:

• 1 punto: si puo' usare su classi, metodi e campi

• 1 punto: un campo final non puo' essere assegnato nei metodi

• 1 punto: un campo final puo' essere assegnato solo nel costruttore o nella dichiarazione del

campo stesso

• 1 punto: non è possibile fare l'override di un metodo final

• 1 punto: non è possibile estendere una classe final

Domanda 2 (5 punti): Principio di sostituzione. Si enunci il principio di sostituzione (substitution principle). Si spieghi poi come Java assicura il principio di sostituzione in presenza di ereditarietà (inheritance).

Risposta Completa:

• 1 punto: un oggetto di una classe puo' sostituire un oggetto di un'altra classe (in linea

generale)

• 1 punto: l'interfaccia dell'oggetto sostituito deve essere più piccola dell'interfaccia

dell'oggetto che lo sostituisce

• 1 punto: una classe che estende un'altra classe ha un'interfaccia più ampia

• 1 punto: una classe che estende un'altra classe è suo sottotipo

• 1 punto: una classe che estende un'altra classe la puo' quindi sostituire.

Domanda 3 (5 punti): Versionamento e compatibilità. Si illustri che cosa si intende per compatibilità "all'indietro" (backward compatibility) delle librerie software. Si spieghi poi come vengono identificate le diverse versioni di tali librerie, e quali sono le differenze principali all'interno del sistema di numerazione delle versioni.

• 1 punto: un sistema nuovo è compatibile con uno vecchio/legacy

• 1 punto: sulle librerie significa che forniscono una interfaccia più ampia della precedente

• 1 punto: ciascuna versione è identificato da un numero univoco

• 1 punto: un numero di versione è composto da più numeri distinti

• 1 punto: la major cambia l'interfaccia e non è backward compatible.

- **ESAME 14-01-2022**

Domanda 1 (5 punti): Modificatori di accesso (access modifier) Si elenchino tutti i modificatori di accesso (access modifier in inglese) supportati da Java, spiegando quali restrizioni sono imposte da ciascuno.

Risposta Completa:

Elenco completo di tutti e 4 gli access modifiers (public, private, protected, e default/package) e nessun altro modifier di altro tipo

• Descrizione corretta modificatore public (accessibile da ovunque)

• Descrizione corretta modificatore private (accessibile solo all'interno della classe)

• Descrizione corretta modificatore protected (accessibile dal package e dalle sottoclassi)

• Descrizione corretta modificatore default/package (accessibile dal package)

Domanda 2 (5 punti): Annotazioni. Si illustri che cos'è un'annotazione in Java, come puo' essere definita, come si possono restringere le componenti del programma ad oggetti a cui applicare una annotazione, e come si puo' definire se l'annotazione deve essere visibile solo sul codice sorgente, anche all'interno del bytecode, oppure anche durante l'esecuzione.

Risposta Completa:

• Spiegazione corretta ad alto livello di cos'è un'annotazione (tipo slide 2 lezione 19)

• Una annotazione puo' essere definita con @interface seguito dal nome dell'annotazione

• La definizione puo' contenere campi/attributi definiti con <tipo> <nome_campo>()

• @Target restringe le componenti del programma ad oggetti a cui applicare una annotazione

• @Retention definisce a che livello l'annotazione è visibile (codice sorgente, bytecode, esecuzione)

Domanda 3 (5 punti): Definizione e firma di un metodo. Si illustri in che cosa consiste la definizione di un metodo, la firma di un metodo, e quali sono le differenze. Si spieghi poi in che cosa consiste l'override di un metodo e che rapporto c'è con dichiarazione e firma di un metodo.

Risposta Completa:

• La firma è composta da nome metodo, numero di parametri e tipo (eventualmente classe che contiene il metodo)

• La dichiarazione comprende anche altro (return type, visibilità, lista delle eccezioni lanciate, …)

• L'override ridefinisce il comportamento di un metodo ereditato

• L'override significa definire un metodo con la stessa firma di un metodo ereditato

• Il resto della dichiarazione di un metodo di cui si fa override puo' variare ma deve essere meno "restrittivo".

- ## ESAME PROVA 2022

Domanda 1 (5 punti): A quali componenti di un programma ad oggetti puo' essere applicato il modificatore final? Cosa comporta ciò nei vari casi?

Risposta Completa:

• 1 punto: a campi, metodi e classi

• 1 punto: una classe final non puo' essere estesa

• 1 punto: non si puo' fare l'override di un metodo final

• 1 punto: non si puo' assegnare più di una volta un campo final

• 1 punto: un campo final deve essere inizializzato nel costruttore (o nella sua definizione).

Domanda 2 (5 punti): È possibile implementare un metodo all'interno di un'interfaccia? Se sì, come? Se no, perché'? Ed è possibile definire un campo? Che restrizioni ci sono rispetto a quello che si puo' implementare e definire all'interno di una classe?

Risposta Completa:

• 1 punto: sì si puo' implementare un metodo

• 1 punto: è necessario definirlo di default

• 1 punto: sì si possono definire campi

• 1 punto: i campi sono static e final

• 1 punto: l'implementazione dei metodi non ha restrizioni.

Domanda 3 (5 punti): Quali sono le differenze tra una variabile di tipo int e una di tipi Integer? È possibile assegnare una variabile di tipo int a una variabile di tipo Integer? Ed una variabile di tipo Integer a una variabile di tipo int? Perché?

Risposta Completa:

• 1 punto: uno è un tipo valore e l'altra è un tipo reference (numero/oggetto)

• 1 punto: Integer eredita i metodi di Object

• 1 punto: Integer è un wrapper di int

• 1 punto: sì è possibile

• 1 punto: autoboxing.