**Test Gestione di Errori e Assertion**

1. Quanti e quali tipi di errori sono presenti in java?

Sono 4 le tipologie di errori presenti in Java: errori di input dell’ utente, errori dei dispositivi, restrizioni fisiche, errori di codice

1. In caso di errore, java permette di uscire dal metodo con un percorso che non restituisce nessun valore?

Si, Java permette di uscire dal metodo con un percorso che non restituisce nessun valore, generalmente però si restituisce uno speciale codice di errore che il metodo chiamante analizza

1. Cosa sono le eccezioni?

Le eccezioni sono gli errori che si possono gestire e che sono rappresentati da oggetti derivati dalla classe Exception

1. Quali sono le gerarchie che partano da Throwable? Scrivere in dettaglio.

Da Throwable partono due gerarchie: Error ed Exception. La famiglia di Error descrive errori interni ad esaurimento delle risorse nell’ ambito del sistema di esecuzione Java, di solito non sono gestibili. La famiglia Exception è invece una situazione imprevista che il flusso di un applicazione può incontrare, sono gestibili e si affrontano

1. Qual’ è la differenza tra checked e unchecked Exception e come derivano?

Le Exception si suddividono in UnChecked e Checked. Le UnChecked derivano da RuntimeException, non rientrano quindi nel meccanismo vincolante delle Exception e solitamente non si gestiscono. Le Checked invece non derivano da RuntimeException e quindi vincolano il programmatore a doverle gestire.

1. Possiamo lanciare delle eccezioni personalizzate?

Si, è possibile lanciare eccezioni personalizzate. Devono però estendere Exception oppure una sua sottoclasse

1. Throw e return lavorano insieme per la gestione degli errori?

* Si
* No √

1. Come possiamo intercettare le eccezioni?

Per intercettare e gestire le eccezioni si utilizza il blocco try/catch. La parte del try è dedicata alla porzione di codice dove si potrebbe presentare una situazione di errore, se si genera un eccezione allora si salta il rimanente del blocco try e si esegue il codice del gestore nel blocco catch

1. È possibile intercettare eccezioni multiple?

si, è possibile intercettare eccezioni multiple. In un blocco try possono essere gestite più eccezioni utilizzando più blocchi di istruzioni catch distinte tra loro

1. Scrivere la giusta interpretazione per il meccanismo vincolato dell’eccezione

Gestione dell’eccezione => blocco try / catch

Propagazione dell’eccezione => throws

1. Possiamo estendere il tipo di un’eccezione della classe super?

* Si √
* No

1. A cosa serve finally?

Il finally è un blocco che contiene codice che viene eseguito indipendentemente che si verifichi. O meno un eccezione, viene quindi usato per rilasciare risorse di sistema, chiudere file o connessioni

1. Scrivere il blocco di codice dove lanci e gestisci una eccezione.

valida = true;

Int = n;

System.out.println("Inserisci un numero intero : ");

try {

n = tastiera.nextInt();

}

catch (InputMismatchException e) {

System.out.println("Non hai inserito un valore intero");

valida = false;

}

1. Nel Try catch, quando possiamo utilizzare l’istruzione break, continue e return? A cosa servono?

Se ci sono istruzioni break, continue, return all’ interno del try catch possono essere eseguite solo dopo l’ esecuzione del blocco finally (che viene eseguito sempre). Servono per poter terminare, far continuare oppure ritornare dei valori.

1. Cosa si intende per assertion e come lo possiamo usare in termini di linguaggio java?

Una assertion è un istruzione che permette di testare eventuali comportamenti che deve avere un’ applicazione. In termini di linguaggio Java la sintassi è:

* assert espressione\_booleana;
* assert espressione\_booleana: espressione\_stampabile;

Se l’ espressione è true il programma può continuare altrimenti viene lanciata un AssertionError