**Esercizi**

**1**.

**Soluzione**: Si veda il codice dei file sorgente OraNonContemplataException.java, OraOccupataException.java e Scheduler.java

**2**.

**Soluzione**: Si veda il codice del file sorgente RestoDivisone.java

**3**.

**Soluzione**: Si veda il codice del file sorgente FormatoOraNonValidoException.java

**4**.

**Soluzione**: Si veda il codice dei file sorgente OraNonValidaException.java, MinutiNonValidiException.java e FormatoInvalidoException.java

**5**.

**Soluzione**: Si veda il codice del file sorgente OraDelGiorno.java

**6**.

**Soluzione**: Si veda il codice del file sorgente ProgrammaTempo.java

**7**.

**Soluzione**: Si veda il codice dei file sorgente CartaBrano.java, CartaNonAttivataException.java e CartaEsauritaException.java

**8**.

**Soluzione**: Si veda il codice del file sorgente Razionale.java

**9**.

**Soluzione**: Si veda il codice del file sorgente RazionaleConAsserzioni.java

**10**.

**Soluzione**: Si veda il codice dei file sorgente ContatoreStanza.java e ContatoreNegativoException.java

**11**.

**Soluzione**: Si veda il codice del file sorgente ContatoreStanzaConAsserzioni.java

**12**.

**Soluzione**: Modificare il metodo valuta del file sorgente Calcolatrice.java inserendo il seguente caso

case '^':

risposta = Math.pow(n1, n2);

break;

**13**.

**Soluzione**: Si veda il codice dei file sorgente CronometroGiri.java e CronometroException.java

\*\*\*\*\*\*

**Programmi**

**1.**

Questo programma ha la stessa organizzazione di PrendiLatteConEccezioni, Listato 13.2.

**Riferimenti:** Listato 13.2

**Soluzione**: Si veda il codice dei file sorgente MessaggioTroppoLungoException.java e MessaggioTroppoLungoExceptionDemo.java

**2**.

La soluzione proposta lancia eccezioni di tipo SommaNegativaException e SommaInsufficienteException.

**Soluzione**: Si veda il codice dei file sorgente SommaNegativaException.java, SommaInsufficienteException.java e ContoCorrente.java

\*\*\*\*\*\*

**Progetti**

**1.**

È utile far funzionare prima il caso normale e poi aggiungere la gestione delle eccezioni. Ci sono una serie di problemi da risolvere per effettuare la trasformazione da 24 ore a 12 ore. Il primo problema è decidere come analizzare l'input con il simbolo due punti (:) che separa i numeri interi di ore e minuti. Tutti i metodi introdotti fino ad ora analizzano il testo che è delimitato da spazi. La soluzione qui utilizzata è quella di leggere un carattere alla volta e di utilizzare uno switch per convertire le ore e i minuti in numeri interi. Ma questo porta ad un altro problema, come affrontare le variazioni del formato di input. Ad esempio, il numero di ore potrebbe avere nessuna cifra, una cifra o due cifre, ad esempio le seguenti scritture rappresentano tutte l’orario pari a 10 minuti dopo la mezzanotte :10, 0:10 o 00:10. Per facilitare la soluzione, viene imposto un requisito all'input: deve essere in formato xx:xx, cioè deve avere due cifre, due punti e poi altre due cifre. Qualsiasi altro formato è contrassegnato come errore di formattazione. Quindi gli zeri iniziali sono richiesti per orari inferiori a 10. (ad esempio, le ore 9 vanno specificate come 09). Un altro problema è ottenere e salvare l'input in modo che possa essere ristampato nel messaggio di errore. La soluzione prevede di salvare l’input letto in cinque caratteri individualmente (si noti che il primo è con readNonwhiteChar) e tutti i caratteri rimanenti come una stringa.

Viene quindi effettuato un controllo per verificare che ogni carattere immesso è valido: sono ammessi solo 0, 1 o 2 per il primo carattere, ecc. Occorre quindi sottrarre 12 ore se l’orario è superiore alle 12, stampare "mezzogiorno" se l’orario è 12:00, cambiare "AM" in "PM" per ore superiori alle 11, e stampare uno zero iniziale per i minuti inferiori ai dieci.

Dopo aver terminato il codice per analizzare ed elaborare l'input, il modo più semplice per aggiungere la gestione delle eccezioni è modificare il codice nei file DividiPerZeroException.java e DividiPerZeroDemo.java (Listati 13.5 e 13.6). Basta cambiare i nomi nel file di definizione delle eccezioni, sostituire il corpo del file demo con il codice sviluppato sopra, e aggiungere il codice nel catch per ottenere la stampa corretta.

Questo è un ottimo esempio per lavorare sullo sviluppo di test case, poiché ci sono un certo numero di situazioni che richiedono una particolare attenzione nel codice. Ecco alcuni esempi:

* un numero di caratteri insufficienti nel campo hh dovrebbe sollevare un’eccezione
* 00:00 – dovrebbe stampare “0:00 AM”
* 12:00 – dovrebbe stampare “12:00 mezzogiorno”
* 12:01 – dovrebbe stampare “12:01 PM”
* 11:59 – dovrebbe stampare “11:59 AM”
* 23:59 – dovrebbe stampare “11:59 PM”
* 24:00 – dovrebbe sollevare un’eccezione
* 11:60 – dovrebbe sollevare un’eccezione
* Un orario con un carattere errato in una posizione dovrebbe sollevare un’eccezione (ad esempio: a1:15, 1a:15, 11a15, 11:a5, and 11:1a)
* Un input completamente errato dovrebbe sollevare un’eccezione (ad esempio: f\*!bc%)
* Un input corretto ma con caratteri addizionali dovrebbe sollevare un’eccezione (ad esempio: 11:15%xyz)

**Riferimenti**: Listati 13.5 e 13.6

**Soluzione**: Si veda il codice dei file sorgente FormatoOrarioException.java. e FormatoOrarioExceptionDemo.java

**2.**

Creare una nuova classe che estende Calcolatrice, aggiungere una variabile di istanza per la memoria e aggiungere il codice per le funzionalità aggiuntive. Si noti che occorre utilizzare i metodi setRisultato() e getRisultato() per accedere alla variabile di istanza risultato definita nella classe padre. Ci deve essere uno spazio tra l’operatore e il numero. Il programma non cattura le eccezioni se viene inserito un valore non numerico al posto di un numero.

**Riferimenti**: Listato 13.5, Listato 13.10 e Listato 13.12.

**Soluzione**: Si veda il codice del file sorgente Calcolatrice.java, CalcolatriceMigliorata.java, OpSconosciutaException.java e DivisionePerZeroException.java

**3.**

Il grande problema che influenza la progettazione di questo progetto è che i numeri inseriti sono stringhe ASCII e non interi. L'elaborazione che deve essere effettuata (controllare i numeri di mese validi, controllare i numeri di giorno validi, e tradurre da numeri di mese a nomi di mese) è molto più facile se il mese e il giorno sono numeri interi, per cui questo è l'approccio adottato nella soluzione qui mostrata.

Il primo problema è quello di analizzare l'input per ottenere una o due cifre di caratteri ASCII per il mese e il giorno. Il programma usa i metodi della classe String per trovare la posizione della barra, /, che separa mese e giorno.

Successivamente converte le cifre in formato testo in numeri interi decimali corrispondenti. Una soluzione relativamente "pulita" è quella di scrivere un metodo di supporto che converte una cifra in formato carattere ASCII nel suo valore intero, quindi utilizzare il metodo di supporto per convertire tutte le cifre. Il codice per convertire un carattere intero ASCII in un valore intero decimale è scritto sfruttando uno switch. Il mese deve essere convertito prima del giorno perché viene utilizzato per determinare se il valore del giorno è valido. Una volta che ili mese è convertito in un numero è facile fare l'elaborazione restante. Il mese può essere facilmente controllato (solo da 1 a 12 sono valori validi) e può essere usato come indice in un array di stringhe per ottenere il nome del mese. Come detto sopra, il mese può essere usato per controllare la validità del numero del giorno (ad esempio, se il numero del mese è 1, 3, 5, 7, 8, 10, o 12, il numero del giorno deve essere nell'intervallo 1 - 31). La struttura switch è anche un'ottima scelta per la verifica del giorno poiché è molto compatto e leggibile.

Mantenere l'elaborazione del mese separata dall'elaborazione del giorno consente di separare in modo chiaro e pulito i blocchi di try/catch per le due eccezioni. Quindi la soluzione è organizzata come segue:

* Analizza la stringa di input per ottenere la parte relativa al mese e quella relativa al giorno.
* Converte la parte relativa al mese in un intero, se possibile.
* Controlla la validità del mese: convertire il numero del giorno solo se il mese è valido. È in questo blocco che viene lanciata una MeseException per ogni inserimento non valido per il mese o se manca il carattere slash che separa mese e giorno. Si noti che qualsiasi numero di 3 caratteri per giorno o mese è considerato non valido. Quindi, mentre 01/01 è accettato e convertito al Gennaio 1, 001/01 e 01/001 sono contrassegnati come non validi.
* Se il mese è un numero valido, converte il giorno nel suo valore intero e controlla la validità. È qui che viene lanciata una GiornoException per ogni inserimento di un giorno non valido.

**Soluzione**: Si veda il codice del file sorgente MeseException.java, GiornoException.java e ConvertitoreDate.java

**5.**

**Riferimenti**: Progetto 3, capitolo 11.

**Soluzione**: Si veda il codice dei file sorgente DimensioneException.java e DimensioneExceptionDemo.java.

**6.**

Può essere utile rivedere array di oggetti prima di svolgere questo progetto. Per esempio, anche dopo aver creato l'array di oggetti di tipo Dipendente, è necessario creare ogni elemento con una nuova istruzione all'interno del ciclo che legge le informazioni del dipendente. Si noti che è utile usare la stessa variabile per accedere l’array e per identificare il numero del dipendente opportunamente incrementata di una unità. Quindi, ogni volta che il numero del dipendente deve essere visualizzato, è sufficiente usare l'espressione (variabile + 1).

**Riferimenti**: Programma 1, capitolo 10; Listato 10.1

**Soluzione**: Si veda il codice dei file sorgente DipendenteDemo.java, LunghezzaCFException.java e FormatoCFException,.java

**7.**

Questo progetto è abbastanza semplice. RichiestaServizio è realmente implementata come una coda. L'unico metodo complicato nella classe è rimuoviNome. Un approccio iterativo che lo lascia per ultimo sarebbe una buona idea. Il programma è un ciclo while di base con casi multipli per ciascuna delle operazioni della classe RichiestaServizio.

**Soluzione**: Si veda il codice dei file sorgente RichiestaServizio.java, RichiestaServizoException.java e RichiestaServizioDemo.java