Programmazione Object Oriented

Iterazioni ed Array

Dr. Dario Di Nucci - 14/10/2022

Cosa impariamo oggi?

	Martedì (ore: 9-11 Aula: F/1)	Giovedì (ore: 9-11 Aula: F/1)	Venerdì (ore: 9-13 aula: P/13)
Settimana 1 (20-23 Settembre)	Introduzione al corso Programmazione OO	Utilizzare gli Oggetti	**Lab: Introduzione Lab: Scrittura primi programmi in Java**
Settimana 2 (27-30 Settembre)	Utilizzare le Classi	Realizzare Classi	**Lab: Creare classi ed oggetti**
Settimana 3 (4-6 Ottobre)	Tipi di Dati Fondamentali	Decisioni/Iterazioni	
Settimana 4 (11-14 Ottobre)	Vettori e Array	Progettazione di Classi	**Lab: Iterazioni ed Array**
Settimana 5 (18-21 Ottobre)	Progettazione di Classi	Interfacce e Polimorfismo	**Lab: Packages e variabili statiche**
Settimana 6 (25-27 Ottobre)	Interfacce e Polimorfismo	Ereditarietà	**Lab: Interfacce**
Settimana 7 (3-4 Novembre)		Ereditarietà	**Lab: Ereditarietà**
Settimana 8 (8-11 Novembre)	Testing e Debugging	Collezioni Programmazione Generica	**Lab: GregorianCalendar**
Settimana 9 (15-18 Novembre)	Gestione delle Eccezioni	IO e Flussi	**Lab: Eccezioni**
Settimana 10 (22-25 Novembre)	Programmazione Grafica	Gestione degli Eventi	**Lab: Esercizi su file**
Settimana 11 (29 Nov - 2 Dic)	Interfacce Grafiche Utente	Lambda Expressions	**Lab: Esercizi su eventi e interfacce grafiche**
Settimana 12 (6-9 Dicembre)	Reflections		**Lab: Esercitazione finale**
Settimana 13 (13-16 Dicembre)	Reflections	Simulazione prova	**Lab: Esercitazione finale**

Esercizio - Dado

Scrivere un programma che si comporti come segue:

1. Crea due conti correnti uno (conto giocatore) con importo iniziale pari a 1.000 euro e uno (conto casinò) con importo iniziale pari a 100.000 euro.
2. Prende in input da tastiera un intero **n** e un double **bet** e simula il lancio di un dado con sei facce. Se il risultato del lancio è **n** allora il programma trasferisce un importo pari a cinque volte il valore di **bet** dal conto del casinò al conto del giocatore; se il risultato è diverso da **n** il programma trasferisce un importo pari a **bet** dal conto del giocatore a quello del casinò. Il valore di **bet** deve essere inferiore al saldo del conto del giocatore e cinque volte più piccolo di quello del conto del casinò.
3. Se dopo la scommessa il giocatore dispone ancora di soldi sul conto, il programma chiede se si vuole continuare a scommettere. Se si digita "si" il programma continua dal punto due. Se si digita qualcosa di diverso da "si" oppure il saldo è zero il programma si interrompe stampando il saldo del conto del giocatore.

Esercizio - Borsa

Implementare e testare la classe **`Purse`** come collezione di **`Coin`**:

- una moneta è un'istanza della classe **`Coin`**;
- gli oggetti di **`Coin`** sono immutabili e hanno un nome (nome della moneta) e un valore (valore della moneta);
- deve essere possibile recuperare l'informazione contenuta in ogni oggetto **`Coin`**;
- oltre ai metodi visti a lezione (**`find`**, **`count`**, **`getMinimum`**, **`getMaximum`**, **`getTotal`**), implementare anche:
  - **`remove(Coin coin)`**: rimuove una moneta di valore pari a **`coin`** se presente nella borsa;
  - **`hasCoin(Coin coin)`**: vale **`true`** se e solo se una moneta di valore pari a **`coin`** è presente nella borsa.

Esercizio - Borsa

Aggiungere alla classe **`Purse`**

- un metodo **`toString`** che restituisca una stringa:

  "Purse[Dollar = $, Quarter = q, Dime = d, Nickel = n, Cent = c]"

  dove **`$, q, d, n, c`** sono rispettivamente il numero di dollari, quarti, dime, nickel e cent contenuti nella borsa.
- un metodo **equals** che rispecchi la relazione di uguaglianza per la quale due borse sono uguali se e solo se contengono lo stesso assortimento di monete

Esercizio - Bank

Implementare e testare la classe **`Bank`** che contenga un vettore di oggetti di tipo **`BankAccount`** e abbia i metodi:

- **`addAccount(initialBalance, customerName)`**
- **`deposit(accountNumber, amount)`**
- **`withdraw(accountNumber, amount)`**
- **`getBalance(accountNumber)`**
- **`transfer(fromAccountNumber, toAccountNumber, amount)`**

Si ricordi che in **`withdraw`** è permesso prelevare in negativo.

L'alternativa consiste nel lanciare un'eccezione (ne parleremo in seguito).

Esercizio - Tris

Scrivere un metodo **`getWinner`** per la classe **`TicTacToe`** che restituisca **`“x”`**, **`“o”`** o **`“Nessun vincitore”`** a seconda sa vince il giocatore **`“x”`**, il giocatore **`“o”`** o non ci sia nessun vincitore.

Scrivere un metodo **`main`** che esegua il gioco. Il programma disegna il tavolo da gioco, cambia il giocatore dopo ogni mossa corretta, e visualizza il vincitore.

Esercizio - Quadrato Magico

Una matrice **n×n** riempita con i numeri **1, 2, 3, ..., n^2** è un quadrato magico se la somma degli elementi di ogni riga, di ogni colonna e delle due diagonali ha lo stesso valore. Per esempio, questo è un quadrato magico:

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Scrivere un programma che legga **n^2** valori dalla tastiera e verifichi se, disposti in una matrice, formino un quadrato magico.

Quando viene inserito un valore nel quadrato dovete verificare se questo non sia già stato inserito in precedenza. Nel caso questo sia avvenuto si deve far inserire un nuovo valore.

Realizzare una classe **`Square`** con i metodi seguenti:

```
public void add(int i)    public boolean isMagic()
```

Esercizio - Compagnia Telefonica

Una compagnia telefonica memorizza i consumi trimestrali dei propri utenti in un file organizzato come segue:

- Codice dell'utente;
- Nome dell'utente;
- Cognome dell'utente;
- Numero di minuti consumati;
- Numero di sms inviati;
- Numero di MB consumati.

Progettare e realizzare un programma per la fatturazione trimestrale, supponendo che i costi per minuti, sms e GB siano stabiliti trimestre per trimestre.

Esercizio - Concessionaria Auto

Una concessionaria di auto usate ha deciso di organizzare le proprie giacenze in un archivio elettronico. L'archivio è realizzato mediante un file che associa ad ogni auto disponibile quattro righe successive con le seguenti informazioni:

- la prima riga contiene la marca dell'auto (es. Tesla)
- la seconda il modello di auto (es. Model Y)
- la terza l'anno di immatricolazione (es. 2020)
- la quarta il costo (es. 35.000)

La concessionaria vuole realizzare un sistema in grado di fornire risposte alle seguenti interrogazioni:

- conoscere se è disponibile un'auto di una certa marca e modello;
- conoscere se è disponibile un'auto di una certa marca e modello immatricolata non prima di un certo anno;
- conoscere se è disponibile un'auto di una certa marca e modello in una determinata fascia di prezzo.

Cosa impariamo la prossima volta?

	Martedì (ore: 9-11 Aula: F/1)	Giovedì (ore: 9-11 Aula: F/1)	Venerdì (ore: 9-13 aula: P/13)
Settimana 1 (20-23 Settembre)	Introduzione al corso Programmazione OO	Utilizzare gli Oggetti	**Lab: Introduzione Lab: Scrittura primi programmi in Java**
Settimana 2 (27-30 Settembre)	Utilizzare le Classi	Realizzare Classi	**Lab: Creare classi ed oggetti**
Settimana 3 (4-6 Ottobre)	Tipi di Dati Fondamentali	Decisioni/Iterazioni	
Settimana 4 (11-14 Ottobre)	Vettori e Array	Progettazione di Classi	**Lab: Iterazioni ed Array**
Settimana 5 (18-21 Ottobre)	Progettazione di Classi	Interfacce e Polimorfismo	**Lab: Packages e variabili statiche**
Settimana 6 (25-27 Ottobre)	Interfacce e Polimorfismo	Ereditarietà	**Lab: Interfacce**
Settimana 7 (3-4 Novembre)		Ereditarietà	**Lab: Ereditarietà**
Settimana 8 (8-11 Novembre)	Testing e Debugging	Collezioni Programmazione Generica	**Lab: GregorianCalendar**
Settimana 9 (15-18 Novembre)	Gestione delle Eccezioni	IO e Flussi	**Lab: Eccezioni**
Settimana 10 (22-25 Novembre)	Programmazione Grafica	Gestione degli Eventi	**Lab: Esercizi su file**
Settimana 11 (29 Nov - 2 Dic)	Interfacce Grafiche Utente	Lambda Expressions	**Lab: Esercizi su eventi e interfacce grafiche**
Settimana 12 (6-9 Dicembre)	Reflections		**Lab: Esercitazione finale**
Settimana 13 (13-16 Dicembre)	Reflections	Simulazione prova	**Lab: Esercitazione finale**