E-commerce

Realizzare un sistema per gestire l'attività di un e-commerce. Tutte le classi, ad eccezione della classe `Esempio`, si devono trovare nel package `ecommerce`. Si mettano in pratica i vari princìpi della programmazione a oggetti (*encapsulation*, *information hiding*, *delega*, ecc.) gestendo in maniera opportuna la *visibilità* dei membri delle classi.

R1. E-commerce

Il sistema, rappresentato da un oggetto di classe `Ecommerce`, permette di vendere e consegnare dei prodotti; il numero massimo dei prodotti gestibili dallo specifico sistema di ecommerce viene passato come parametro al costruttore della classe `Ecommerce`. Tale numero rappresenta il numero massimo di unità che l'ecommerce è in grado di gestire indipendentemente dalla tipologia di prodotto. Ogni prodotto è identificato da una stringa riportante un codice alfanumerico univoco, una descrizione, il prezzo per unità e il numero di unità disponibili. Per definire un prodotto si utilizza il metodo **nuovoProdotto()** della classe `Ecommerce`, che riceve come parametri l'identificativo, la descrizione del prodotto, il prezzo unitario e il numero di unità di quel prodotto da aggiungere nel sistema. Il metodo crea dunque il nuovo prodotto e restituisce il riferimento all'oggetto appena creato. Cercando di aggiungere un prodotto già esistente nel sistema, il metodo incrementa le unità per il prodotto in considerazione (fino a esaurimento spazio). Nel caso di incremento di almeno un'unità il metodo restituisce il riferimento all'oggetto aggiornato; diversamente, in caso di spazio terminato, il metodo non sortisce invece alcun effetto (e il valore di ritorno è `null`).

I prodotti sono rappresentati da oggetti di classe `Prodotto`, dotati di opportuni metodi getter per accedere alle suddette informazioni. Attraverso il metodo **cercaProdotto()** è possibile cercare il prodotto il cui codice è passato come parametro. Il metodo restituisce il riferimento all'oggetto trovato (`null` nel caso il prodotto non sia definito).

Attraverso il metodo **cercaProdotti()** è invece possibile cercare tutti i prodotti che contengono la stringa passata come parametro – o parte di essa – nella descrizione del prodotto o nel codice alfanumerico (ignorando le differenze tra caratteri maiuscoli e minuscoli). Il metodo restituisce un array, della dimensione pari al numero di prodotti trovati, contenente i riferimenti agli oggetti che soddisfano il criterio di ricerca (riportati nell'ordine in cui i prodotti sono stati definiti).

R2. Utenti e Acquisti

Ogni sistema di e-commerce può gestire più utenti, rappresentati da oggetti di classe `Utente`. Si assuma che in un sistema possano essere gestiti al più 100 utenti. Per definire un nuovo utente si utilizza il metodo **nuovoUtente()** della classe `Ecommerce`, che riceve come parametri il codice fiscale (di cui deve essere controllata l'univocità), il cognome, il nome e la disponibilità economica dell'utente. Nel caso in cui esista già un utente con quel codice fiscale, vengono semplicemente aggiornate le informazioni e ne viene restituito il riferimento.

La classe `Utente` mette a disposizione opportuni metodi getter per accedere alle suddette informazioni. Il metodo **cercaUtente()** riceve come parametro un codice fiscale di un utente e restituisce l'oggetto corrispondente (oppure `null`).

Per acquistare uno o più prodotti si utilizza il metodo **acquisto()** della classe `Ecommerce`. Il metodo riceve come parametri il codice fiscale dell'utente ed un array di stringhe identificanti una serie di acquisti. Ogni stringa riporta il codice di un prodotto e il numero di unità di quel prodotto che l'utente desidera acquistare separati dal carattere `';'`. Nel caso in cui il codice fiscale dell'utente esista il metodo effettua un controllo sulla possibilità di portare a termine l'acquisto. L'acquisto di ciascuno dei prodotti passati come parametro è infatti possibile solo se il codice prodotto corrisponde ad un prodotto esistente e se il numero di unità richieste è minore o uguale al numero di unità disponibili (si assuma che all'interno dell'array, lo stesso prodotto non possa comparire in più acquisti). Infine, il metodo verifica che la spesa complessiva (per i prodotti e le unità che hanno superato la prima verifica) non superi la disponibilità economica dell'utente. In caso di acquisto avvenuto, il metodo restituisce un array di valori booleani della stessa lunghezza dell'array di stringhe passato come parametro; in corrispondenza della posizione di una coppia prodotto-unità il cui acquisto sia andato a buon fine l'array risultante riporterà il valore `true`, in caso di acquisto non riuscito il valore `false`. Inoltre, il metodo aggiorna le quantità dei prodotti e la disponibilità economica dell'utente, ed assegna a ciascuno degli acquisti un codice numerico incrementale (a partire da `1`); tale codice identifica in modo univoco un determinato acquisto nel sistema di e-commerce. Nel caso in cui la spesa complessiva superi la disponibilità economica dell'utente, il valore di ritorno è un array della stessa lunghezza riportante il valore `false` per ogni coppia prodotto-unità (nessuno degli acquisti va a buon fine).

Il metodo **ultimoAcquisto()** della classe `Ecommerce` restituisce l'ultimo acquisto per l'utente il cui codice fiscale è passato come parametro (`null` se l'utente non è definito o se non ha ancora fatto acquisti). Il valore di ritorno consiste in una stringa riportante il codice dell'acquisto, il codice fiscale dell'utente, il codice del prodotto, il numero delle unità acquistate e la spesa relativa a quel determinato acquisto separati dal carattere `';'`.

Il metodo **acquistiUtente()** riceve come parametro un codice di un utente e restituisce la lista dei suoi acquisti andati a buon fine uno per riga, in ordine di esecuzione (nello stesso formato del metodo precedente). In modo simile, il metodo **utentiProdotto()** riceve come parametro il codice di un prodotto e restituisce i codici fiscali degli utenti che lo hanno acquistato almeno una volta, uno per riga, in ordine di acquisto ed evitando duplicati. Nel caso in cui utente o prodotto non siano definiti, i suddetti metodi restituiscono `null`.

R3. Consegne Standard e Prime

Su ogni acquisto gestito dal sistema di e-commerce è possibile richiedere una consegna, rappresentata da un oggetto di classe `Consegna`. Sono possibili due tipi di consegne, ovvero standard e prime, rappresentati dalle classi omonime (`Standard` e `Prime`).

Per richiedere una consegna su un determinato acquisto si utilizza il metodo **nuovaConsegna()** della classe `Ecommerce`. Per le consegne standard, il metodo riceve come parametri il codice di un acquisto e un indirizzo di consegna, e restituisce un oggetto di classe `Standard`. Il metodo verifica che il codice dell'acquisto esista e, in caso positivo, crea la nuova consegna e restituisce un riferimento all'oggetto creato. Per le consegne prime, il metodo riceve come parametro aggiuntivo la data di consegna desiderata e restituisce un oggetto di classe `Prime`. Il metodo verifica che il codice acquisto esista e che l'utente possa permettersi il costo della spedizione prime. Infatti, su tali spedizioni viene applicato un costo aggiuntivo pari al 10% del costo complessivo dei prodotti acquistati. Se tutte le suddette verifiche sono soddisfatte, il metodo crea la nuova consegna e sottrae l'importo del servizio dalla disponibilità economica dell'utente. Cercando di attivare una consegna per un acquisto non definito il metodo non sortisce alcun effetto. Allo stesso modo, se l'utente non può coprire il costo della consegna, il metodo non sortisce alcun effetto e il valore di ritorno è `null`. Per entrambi i tipi di consegna, in caso di corretta creazione di una nuova consegna, il metodo assegna all'oggetto creato un codice alfanumerico univoco composto dal carattere che identifica il tipo di consegna (`S` oppure `P`) e da un numero incrementale che rappresenta il numero di consegne finora richieste al sistema di e-commerce (a partire da `1`). Ad esempio, in caso di richiesta di due consegne standard e una consegna prime (in quest'ordine), i codici assegnati a tali consegne saranno `S1`, `S2` e `P3`.

La classe `Ecommerce` mette a disposizione un metodo **descriviConsegna()** che riceve come parametro il codice di una consegna e, in caso di consegna esistente, restituisce una stringa contenente codice fiscale dell'utente, il codice dell'acquisto, il tipo di consegna (`S` oppure `P`) e la spesa totale legata ad acquisto e consegna. Per la classe `Standard`, l'ammontare sarà semplicemente quello relativo all'acquisto; per la classe `Prime` sarà invece comprensivo dei costi aggiuntivi di spedizione e la stringa includerà – dopo l'ammontare totale – anche la data di consegna richiesta dall'utente.

Il metodo **consegne()** della classe `Ecommerce` restituisce un array contenente tutte le consegne finora richieste (in ordine di richiesta). Una variante del metodo riceve come parametro il codice fiscale di un utente e restituisce le sole consegne richieste da quell'utente (nel medesimo ordine). Un'ulteriore variante riceve oltre al codice fiscale di un utente anche una stringa che identifica un tipo di consegna (`S` oppure `P`) e restituisce solo le consegne di quel tipo associate a tale utente.