Sapienza Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di Laurea in Ingegneria Informatica ed Automatica, Ingegneria dei Sistemi Informatici

Esercitazioni di Progettazione del Software
A.A. 2012/2013

Prova al calcolatore del 2 luglio 2013

Requisiti

Si vuole realizzare un'applicazione per la gestione degli ordini di una pizzeria. Ogni ordine è caratterizzato da un codice identificativo (una stringa), dalla data in cui avviene e dall'indicazione se la consegna deve avvenire a domicilio o se il ritiro viene effettuato dal cliente. Per ciascun ordine è di interesse registrare il cliente che lo effettua (con nome, cognome e indirizzo) e i prodotti ordinati, con relativa quantità. I prodotti della pizzeria, ciascuno con nome e prezzo, si dividono in *pizze* e *bibite*. Di ogni pizza interessano gli ingredienti (per semplicità, una stringa); di ogni bibita interessa la quantità in centilitri. Ogni ordine deve contenere almeno due prodotti, in particolare almeno una pizza e una bibita. Per ogni ordine è di interesse calcolare l'importo totale da pagare, come somma dei prezzi dei singoli prodotti ordinati, per la relativa quantità. Per ogni ordine viene emessa una fattura corrispondente, identificata da un codice.

In Figura 1 è mostrato il diagramma delle classi corrispondente al dominio.

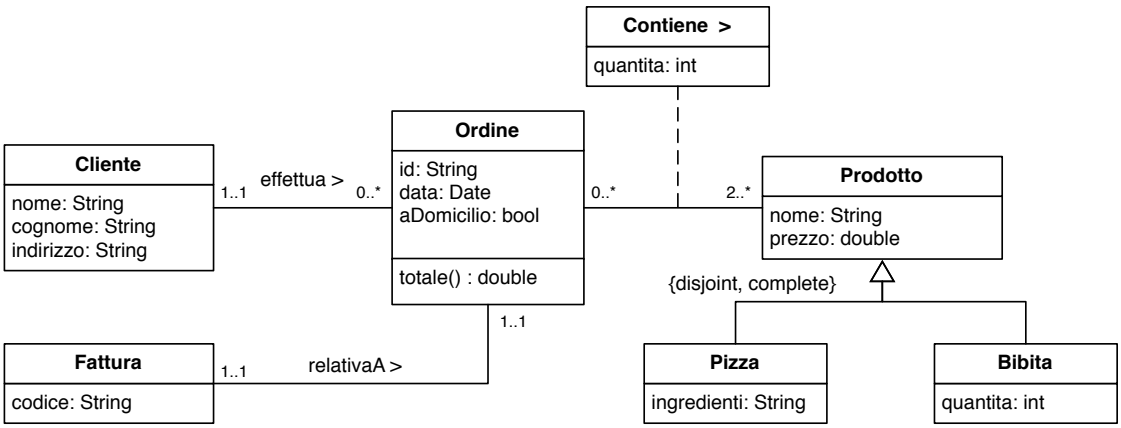

Figura 1: Diagramma UML delle classi

Quando l'operatore della pizzeria riceve la richiesta per un ordine, una sessione di interazione con l'applicazione si svolge come segue:

- l'utente inserisce i dati del cliente, specificando nome, cognome e indirizzo;
- l'utente inserisce i dati dell'ordine, specificandone se la consegna deve avvenire a domicilio; l'identificativo e la data dell'ordine vengono generati dal sistema;
- iterativamente vengono selezionati i prodotti da ordinare; in particolare:
  - viene visualizzata la lista dei prodotti ordinabili e l'utente seleziona il prodotto da ordinare, specificandone la quantità;
  - il prodotto selezionato viene aggiunto all'ordine;
  - si procede poi con l'eventuale prodotto successivo, sulla base della scelta dell'utente;
- dopo aver definito i prodotti da ordinare, si verifica che l'ordine contenga almeno una pizza e una bibita:
  - se l'ordine è valido, si procede come specificato nel seguito;
  - altrimenti, viene mostrato un messaggio di errore e si procede poi con l'eventuale ordine successivo, sulla base della scelta dell'utente;
- l'ordine effettuato dal cliente viene registrato e si procede eseguendo concorrentemente le seguenti sottoattività:
  1. viene emessa la fattura relativa all'ordine e i dati della fattura vengono mostrati;
  2. vengono mostrati i dettagli dell'ordine affinché l'ordine possa essere evaso;
- si procede poi con l'eventuale ordine successivo, sulla base della scelta dell'utente.

In Figura 2 è riportato il diagramma delle attività corrispondente.

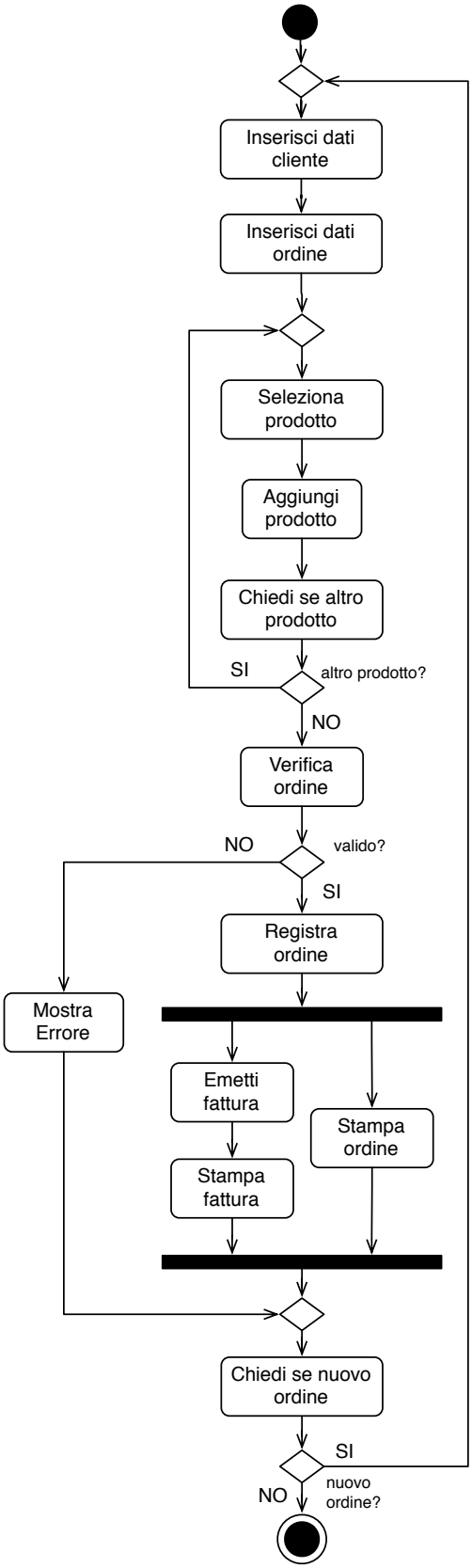

Figura 2: Diagramma delle attività

La prova consiste nel completare o modificare il codice fornito insieme al testo, in modo da soddisfare i requisiti sopra riportati. Seguendo le indicazioni riportate nei commenti al codice[1], si chiede di intervenire sulle seguenti classi:

- `FinestraPrincipale` (package `app.gui`) (si vedano le considerazioni in fondo al documento di specifica)
- `Ordine` (package `app.dominio`)
- `Fattura` (package `app.dominio`)
- `AggiungiProdotto` (package `app.attivita.atomiche`)
- `AttivitaSottoramo1` (package `app.attivita.complesse`)
- `AttivitaPrincipale` (package `app.attivita.complesse`)

Tempo a disposizione: **3 ore.**

Gli elaborati non accettati dal compilatore saranno considerati insufficienti.

Per facilitare la comprensione del codice e lo svolgimento della prova, nel seguito sono riportati i documenti di specifica risultanti dalle fasi di analisi e di progetto.

Analisi

Operazioni Classe `Ordine`

`InizioSpecificaOperazioniClasse Ordine`

`InizioSpecificaOperazione Totale`
`Totale():(double)`
`pre: --`
`post:` Calcola l'importo totale dell'ordine `this`.
Sia C l'insieme dei link di tipo *Contiene* che coinvolgono l'ordine `this`;
`result` vale $\sum_{c \in C} c.prodotto.prezzo \times c.quantita$
`FineSpecifica`

`FineSpecifica`

Attività di I/O

`InizioSpecificaAttivitàAtomica InserisciDatiCliente`
`InserisciDatiCliente():(Cliente)`
`pre: --`
`post:` Legge il nome, cognome e indirizzo di un cliente, forniti in input dall'utente.
`result` è il cliente creato a partire dai dati inseriti.
`FineSpecifica`

[1] le porzioni di codice su cui intervenire sono identificate dal commento `/* DA COMPLETARE A CURA DELLO STUDENTE */`

`InizioSpecificaAttivitàAtomica InserisciDatiOrdine`
`InserisciDatiOrdine():(Ordine)`
`pre: --`
`post:` Mostra identificativo e data di un ordine generati dal sistema, e consente all'utente di specificare se la consegna è a domicilio.
`result` è l'ordine creato a partire dai dati specificati.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica SelezionaProdotto`
`SelezionaProdotto():(RecordSelezioneProdotto)`
`pre: --`
`post:` Mostra l'insieme dei prodotti ordinabili e consente di selezionare il prodotto da ordinare, specificandone la quantità.
`result` è l'oggetto *RecordSelezioneProdotto*[2] che contiene il prodotto selezionato e la quantità.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica ChiediSeAltroProdotto`
`ChiediSeAltroProdotto():(Bool)`
`pre: --`
`post:` Chiede all'utente se vuole aggiungere un altro prodotto.
`result` è `true` in caso affermativo, `false` altrimenti.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica StampaFattura`
`StampaFattura(f:Fattura):()`
`pre: --`
`post:` Visualizza le informazioni relative alla fattura `f`, con i dettagli dell'ordine per cui è stata emessa, del cliente che l'ha effettuato, e dei prodotti ordinati.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica StampaOrdine`
`StampaOrdine(o:Ordine):()`
`pre: --`
`post:` Visualizza le informazioni relative all'ordine `o`, con i dettagli del cliente che l'ha effettuato e dei prodotti ordinati.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica ChiediSeNuovoOrdine`
`ChiediSeNuovoOrdine():(Bool)`
`pre: --`
`post:` Chiede all'utente se vuole processare un nuovo ordine.
`result` è `true` in caso affermativo, `false` altrimenti.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica MostraErrore`
`MostraErrore():()`
`pre: --`
`post:` Visualizza un messaggio di errore che informa l'utente che l'ordine non è valido.
`FineSpecifica`

[2] *RecordSelezioneProdotto* è un record contente due campi: il prodotto e la quantità.

Attività Atomiche

`InizioSpecificaAttivitàAtomica AggiungiProdotto`
`AggiungiProdotto(o:Ordine, rp:RecordSelezioneProdotto):()`
`pre: --`
`post:` Crea un link *link* di tipo *Contiene* tale che *link.ordine* = `o`, *link.prodotto* = `rp.prodotto` e *link.quantita* = `rp.quantita`.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica VerificaOrdine`
`VerificaOrdine(o:Ordine):(Bool)`
`pre: --`
`post:` Verifica se l'ordine `o` contiene almeno una pizza e una bibita.
Sia C l'insieme dei link di tipo *Contiene* che coinvolgono l'ordine `o`; `result` è `true` se $\exists c \in C$ tale che *c.prodotto* è una pizza e $\exists c \in C$ tale che *c.prodotto* è una bibita, `false` altrimenti.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica RegistraOrdine`
`RegistraOrdine(o:Ordine, c:Cliente):()`
`pre: --`
`post:` Crea un link *link* di tipo $Effettua$ tale che *link.ordine* = `o` e *link.cliente* = `c`.
`FineSpecifica`

`InizioSpecificaAttivitàAtomica EmettiFattura`
`EmettiFattura(o:Ordine):(Fattura)`
`pre: --`
`post:` Crea una fattura f con codice generato dal sistema, e crea un link *link* di tipo *RelativaA* tale che *link.ordine* = `o` e $link.fattura = f$.
`result` è la fattura f creata.
`FineSpecifica`

Attività Composte

`InizioSpecificaAttività AttivitaSottoramo1`

`AttivitaSottoramo1(o:Ordine):()`

Variabili Processo:
`ordine:Ordine` -- l'ordine corrente
`fattura:Fattura` -- la fattura per l'ordine corrente

Inizio Processo
`EmettiFattura(ordine):(fattura);`
`StampaFattura(fattura):();`
Fine Processo

`FineSpecifica`

`InizioSpecificaAttività AttivitaSottoramo2`

`AttivitaSottoramo2(o:Ordine):()`

Variabili Processo:
`ordine:Ordine` -- l'ordine corrente

Inizio Processo
`StampaOrdine(ordine):();`
Fine Processo

`FineSpecifica`

InizioSpecificaAttività AttivitaPrincipale

AttivitaPrincipale():()

Variabili Processo:
cliente: Cliente -- cliente corrente
ordine: Ordine -- ordine corrente
recordProdotto: RecordSelezioneProdotto -- record selezione prodotto
altroProdotto: Bool -- inserire altro prodotto?
ordineValido: Bool -- ordine valido?
nuovoOrdine: Bool -- inserire altro ordine?

Inizio Processo:

```
do {
  InserisciDatiCliente():(cliente);
  InserisciDatiOrdine():(ordine);

  do {
    SelezionaProdotto():(recordProdotto);
    AggiungiProdotto():(recordProdotto);
    ChiediSeAltroProdotto():(altroProdotto);
  } while(altroProdotto);

  VerificaOrdine(ordine):(ordineValido);

  if (!ordineValido) {
    MostraErrore():();
  }
  else {
    RegistraOrdine(ordine, cliente):();
    fork {
      thread t1:{AttivitaSottoramo1(ordine):();}
      thread t2:{AttivitaSottoramo2(ordine):();}
    }
    join t1, t2;
  }

  ChiediSeNuovoOrdine():(nuovoOrdine);

} while(nuovoOrdine);
```

Fine Processo

FineSpecifica

Progetto

Responsabilità sulle Associazioni

R: Requisiti; O: Specifica delle Operazioni/Attività; M: Vincoli di Molteplicità

Associazione	Classe	Ha Responsabilità
effettua	Cliente	NO
	Ordine	SI (R,O,M)
contiene	Ordine	SI (R,O,M)
	Prodotto	NO
relativaA	Fattura	SI (O,M)
	Ordine	SI (R,M)

Strutture di Dati

Rappresentiamo le collezioni omogenee di oggetti mediante le classi `Set` ed `HashSet` del Collection Framework di Java.

Tabelle di Gestione delle Proprietà delle Classi UML

Riassumiamo le scelte differenti da quelle di default mediante la tabella delle proprietà immutabili.

Classe UML	Proprietà Immutabile
Cliente	nome cognome
Ordine	id data aDomicilio
Prodotto	nome

Altre Considerazioni

Non dobbiamo assumere una particolare sequenza di nascita degli oggetti.

Non esistono valori di default per qualche proprietà che siano validi per tutti gli oggetti.

La finestra principale dell'applicazione deve essere simile a quella in Figura 3.

Figura 3: La finestra principale