SAPIENZA - Università di Roma
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Informatica
Esercitazioni di Progettazione del Software - A.A. 2008/2009
Prova al calcolatore del 17 settembre 2009

Requisiti

L'applicazione da progettare riguarda un sistema per la distribuzione di quotidiani. Ciascun quotidiano è caratterizzato dal titolo, dal nome del direttore, dall'anno di fondazione e dal prezzo per copia. Ogni utente, caratterizzato da nome, cognome, indirizzo e password per il pagamento, può effettuare degli ordini, ciascuno dei quali relativo ad un dato numero di copie di *uno stesso quotidiano*. Un ordine è caratterizzato dalla data di creazione, dalla data in cui è stato pagato dall'utente (inizialmente nulla), dalla data di invio (inizialmente nulla), cioè quando è stato preso in carico dal corriere per la consegna all'utente, dal prezzo totale, ottenuto moltiplicando il prezzo unitario del quotidiano per il numero di copie ordinate, dal codice IBAN per l'addebito del pagamento e dal numero d'ordine (un intero progressivo). In Figura 1(a) è riportato il diagramma UML delle classi, corrispondente al dominio descritto.

Il processo di gestione degli ordini è schematizzato in Figura 1(b). Inizialmente il sistema offre la possibiltà di inserire i dati relativi al primo ordine; a partire da tali dati, viene creato il primo ordine ed aggiunto all'archivio.

Fatto ciò vengono avviate due attività contemporanee:

1. Da un lato viene gestito l'ordine appena creato (si veda sotto)
2. Dall'altro lato viene chiesto all'utente se occorre creare un ulteriore ordine. In caso affermativo, vengono richiesti i dati relativi al nuovo ordine, che viene successivamente creato. A questo punto da un lato viene gestito il nuovo ordine e dall'altro viene nuovamente chiesto se un ulteriore ordine si rende necessario. In sostanza, l'attività di richiesta di ulteriori ordini viene ripetuta continuamente fino a quando nuovi ordini si rendono necessari.

La gestione di ordine creato consiste dei passi seguenti

1. Il primo task consiste nel chiedere le informazioni sul pagamento, che qui sono semplificate con il solo codice IBAN della banca. Per questioni di sicurezza, il sistema richiede che l'utente inserisca la propria password. Se la password è errata, viene chiesto all'utente di inserire nuovamente il codice IBAN e la password (corretta)
2. Se la password è corretta, vengono aggiornate le informazioni sul pagamento (la data del pagamento e il codice IBAN dell'ordine).
3. Infine l'ordine viene inviato (viene aggiornata la data di invio). Non è d'interesse la gestione dell'ordine da parte del corriere.

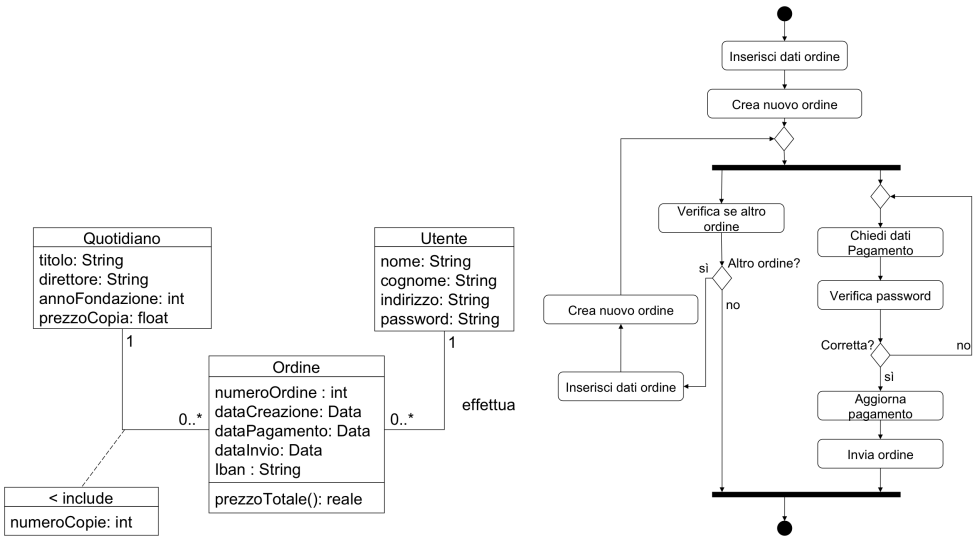
L'applicazione, inoltre, deve offrire una funzionalità che prenda in input il titolo di un quotidiano e ne calcoli e visualizzi il numero di copie vendute.

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, si chiede di intervenire sulle seguenti classi:

- `Ordine`
- `Quotidiano`
- `OperazioneUtente`
- `AggiornaPagamento`
- `ManagerInclude`
- `PrincipaleListener`
- `AttivitaGestioneOrdine`
- `AttivitaVerificaAltroOrdine`

Tempo a disposizione: **3 ore.**

Gli elaborati non accettati dal compilatore saranno considerati insufficienti.



(a) Diagramma UML delle classi.

(b) Diagramma UML delle attività.

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

Analisi

Operazioni di Classe e Utente

```
InizioSpecificaOperazioniClasse Ordine
  context Ordine::prezzoTotale(): real
    pre:--
    post: result=self.include.Quotidiano.prezzoCopia * self.include.numeroCopie
FineSpecifica
```

```
InizioSpecificaOperazioniClasse OperazioniUtente

  context OperazioniUtente::quanteCopie(q:Quotidiano): int

    pre:--

    post: result è pari alla somma  dell'attributo numCopie di tutti i link di tipo include

            che coinvolgono q e tali che la data di pagamento dell'ordine sia diversa da null

FineSpecifica
```

Segnatura delle Attività di I/O

```
InizioSpecificaOperazioniIO AttivitaPrincipale


  operazione: inserisciDatiOrdine
    -- Legge da input il quotidiano ed il numero di copie che l'utente vuole ordinare
    in: --
    out: recordOrdine : RecordOrdine


  operazione: verificaSeAltroOrdine
    -- Verifica se l'utente vuole inserire un nuovo ordine
    in: --
    out: altroOrdine : boolean


  operazione: chiediDatiPagamento(ordine:Ordine)
    -- Legge da input iban e password per il pagamento dell'ordine
    in: --
    out: recordPagamento : RecordPagamento


FineSpecifica
```

NOTE:
`RecordOrdine` è un record con due campi, `numeroCopie` (di tipo `int`) e `nomeQuotidiano` (di tipo `String`), usati per memorizzare il valore dei campi dato di un ordine.

`RecordPagamento` è un record con due campi, `iban` (di tipo `String`) e `password` (di tipo `String`), usati per memorizzare i dati necessari al pagamento di un ordine.

Definizione delle Variabili dell'Attività Principale

```
InizioSpecificaVariabiliAttività AttivitaPrincipale

  recordOrdine: RecordOrdine -- dati dell'ordine forniti in input

  ordine: Ordine -- ordine corrente

  utente : Utente -- utente che sta effettuando l'ordine

  recordPagamento : RecordPagamento -- dati per il pagamento dell'ordine

FineSpecifica
```

Definizione delle Attività Atomiche

```
InizioSpecificaAttività CreaNuovoOrdine
    pre: --
    post: -- Crea un nuovo ordine a partire dai dati inseriti in input e memorizzati in recordOrdine ed
      -- assegna un numero progressivo al campo dati numOrdine
FineSpecifica

InizioSpecificaAttività VerificaPassword
    pre: --
    post: -- verifica se la password memorizzata in recordPagamento
      -- corrisponde a quella dell'utente che sta effettuando l'ordine
FineSpecifica

InizioSpecificaAttività AggiornaPagamento
    pre: --
    post: -- assegna il codice iban memorizzato in recordPagamento all'Ordine ordine ed
      aggiorna il campo dati dataPagamento dell'Ordine ordine alla data corrente
FineSpecifica


InizioSpecificaAttività InviaOrdine
```

```
    pre: --
    post: -- aggiorna il campo dati dataInvio dell'Ordine ordine alla data corrente
FineSpecifica
```

NOTE:
Altre eventuali attività presenti nell'implementazione possono essere trascurate.

Progetto

Responsabilità sulle Associazioni

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

Associazione	Classe	Ha Responsabilità
effettua	Utente	NO
	Ordine	SÌ (M)
include	Quotidiano	SÌ (O)
	Ordine	SÌ (O,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 e la tabella delle assunzioni sulla nascita.

Classe UML	Proprietà Immutabile
Quotidiano	intero annoFondazione
Ordine	dataCreazione numeroOrdine
Utente	nome cognome

	Proprietà	
Classe UML	**Nota alla nascita**	**Non nota alla nascita**
-	-	-

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.