Sapienza - Università di Roma
Facoltà di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Esercitazioni di Progettazione del Software - A.A. 2009/2010

Prova al calcolatore del 17 settembre 2010

Requisiti

Si vuole realizzare un'applicazione per il commercio di souvenir turistici. Ciascun souvenir è caratterizzato dal prezzo (reale), dal nome della località che ricorda (stringa), e da una *descrizione* (stringa) il cui formato dipende dal particolare souvenir (v. metodo *toString()* delle classi *Berretto* e *Magnete*). Esistono solo due tipi di souvenir: *magneti* e *berretti*. Dei primi, che sono rettangolari, interessano le dimensioni di base ed altezza (due interi positivi), mentre dei secondi interessano il colore (una stringa) e la taglia (un intero positivo).

Durante ciascun acquisto (v. sotto), l'applicazione mantiene uno scontrino aggiornato, che include i souvenir acquistati, con le relative quantità, ed è caratterizzato da un numero progressivo, dalla data di emissione e dal totale della spesa. Dato un souvenir, è d'interesse conoscere gli scontrini che lo includono.

Il processo d'acquisto si svolge come segue. All'inizio della sessione, viene chiesto all'utente di selezionare la categoria di prodotti che intende acquistare: magneti, berretti o entrambi. Dopo la selezione, viene dapprima creato uno scontrino vuoto con opportuno numero (progressivo) e data di emissione, quindi viene mostrata un'interfaccia per ciascuna categoria di souvenir selezionata. Tali interfacce permettono di specificare i souvenir e le relative quantità che si desidera acquistare.

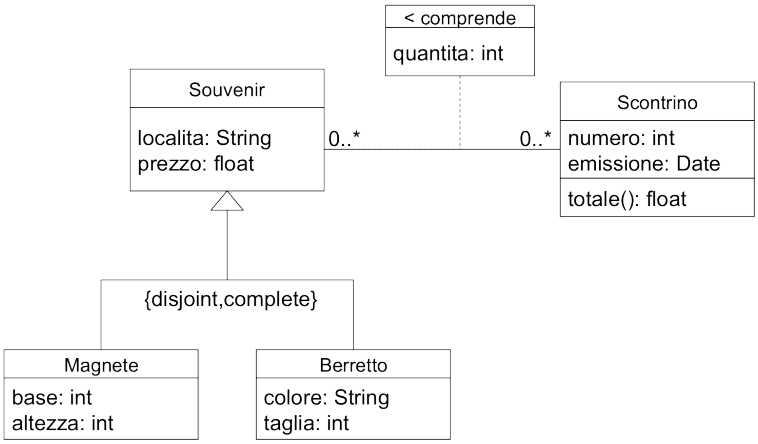
Ogni volta che l'utente specifica un acquisto, selezionando un tipo di souvenir (della categoria selezionata precedentemente) e la relativa quantità, lo scontrino viene aggiornato. Dopo ciascuna selezione, l'applicazione chiede all'utente se abbia intenzione di comprare altri souvenir della stessa categoria. In caso affermativo, ha luogo una nuova iterazione; in caso contrario, non sarà più possibile acquistare altri souvenir della stessa categoria (nella sessione d'acquisto corrente).

Quando l'utente non ha più intenzione di acquistare souvenir, l'applicazione mostra i dettagli dello scontrino con la spesa totale, e la sessione d'acquisto termina.

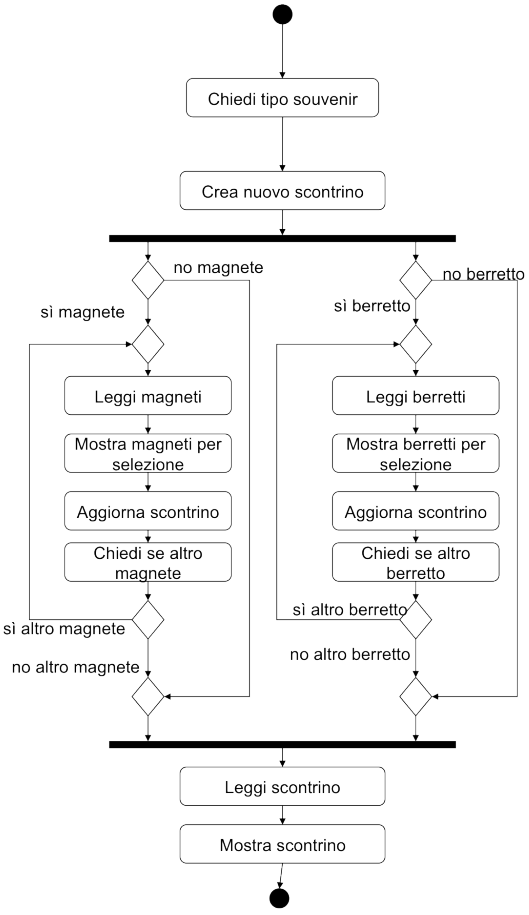
In Figura 1(a) ed in Figura 1(b) sono mostrati, rispettivamente, il diagramma delle classi di dominio ed il diagramma delle attività del processo sopra descritti.

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:

- `Main` (package `applicazione`)
- `Scontrino` (package `souvenir`)
- `Souvenir` (package `souvenir`)



(a) Diagramma UML delle classi



(b) Diagramma UML delle attività

- `ManagerComprende` (package `souvenir`)
- `TipoLinkComprende` (package `souvenir`)
- `AggiornaScontrino` (package `attivita_atomiche`)
- `AttivitaPrincipale` (package `attivita_composte`)
- `AttivitaSottoramoBerretti` (package `attivita_composte`)
- `AttivitaSottoramoMagneti` (package `attivita_composte`)

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.

Attività di I/O

`InizioSpecificaAttivitàAtomica ChiediTipoSouvenir`
`ChiediTipoSouvenir ():(RecordTipoAcquisti)`
`pre: --`
`post:` Mostra all'utente un menù dal quale selezionare *Magneti*, *Berretti* o *Magneti e Berretti*
result è il *RecordTipoAcquisti* (v.sotto) corrispondente alla selezione.
`FineSpecifica`

RecordTipoAcquisti è un record contente due campi booleani, *magneti* e *berretti*, impostati al valore *true* dall'attività di I/O *ChiediTipoSouvenir* se e solo se l'utente ha intenzione di acquistare, rispettivamente, magneti o berretti. I valori di tali campi sono restituiti dai metodi *isMagneti()* ed *isBerretti()*.

`InizioSpecificaAttivitàAtomica MostraSouvenirPerSelezione`
`MostraSouvenirPerSelezione (Set<String> insiemeSouvenir):(RecordAcquisto)`
`pre: --`
`post:` Mostra all'utente un menù per selezionare un souvenir da acquistare, specificandone la quantità desiderata, tra quelli corrispondenti alle descrizioni presenti in *insiemeSouvenir*. *result* è il *RecordAcquisto* (v.sotto) corrispondente alla selezione.
`FineSpecifica`

RecordAcquisto è un record contente due campi: la descrizione (campo *descrizione*, come restituita dal metodo *toString()* della classe *souvenir*), del souvenir che l'utente intende acquistare, e la relativa quantità (campo *quantità* selezionata dall'utente).

`InizioSpecificaAttivitàAtomica AltroMagnete`
`AltroMagnete ():(Boolean)`
`pre: --`
`post:` Mostra all'utente una finestra per specificare se abbia intenzione di acquistare un altro magnete. In caso affermativo *return* è *true*, altrimenti è *false*.

`InizioSpecificaAttivitàAtomica AltroBerretto`
`AltroBerretto ():(Boolean)`
`pre: --`
`post:` Mostra all'utente una finestra per specificare se abbia intenzione di acquistare un altro berretto. In caso affermativo *return* è *true*, altrimenti è *false*.
`FineSpecifica`

`MostraScontrino`
`MostraScontrino (RecordScontrino rsc):()`
`pre: --`
`post:` Mostra una finestra contenente i dati associati al *RecordScontrino rsc* fornito in input.
`FineSpecifica`

Attività Atomiche

InizioSpecificaAttivitàAtomica CreaNuovoScontrino
    CreaNuovoScontrino ():()
    pre: --
    post: Crea un nuovo oggetto di classe *Scontrino*, con data di emissione e numero progressivo opportuni, e lo restituisce.
FineSpecifica

InizioSpecificaAttivitàAtomica LeggiMagneti
    LeggiMagneti ():(Set<String>)
    pre: --
    post: Legge l'insieme dei magneti in vendita e restituisce un insieme di stringhe (come restituite dal metodo *toString()* della classe *Souvenir*) che li identificano.
FineSpecifica

InizioSpecificaAttivitàAtomica LeggiBerretti
    LeggiMagneti ():(Set<String>)
    pre: --
    post: Legge l'insieme dei berretti in vendita e restituisce un insieme di stringhe (come restituite dal metodo *toString()* della classe *Souvenir*) che li identificano.
FineSpecifica

InizioSpecificaAttivitàAtomica AggiornaScontrino
    AggiornaScontrino (RecordAcquisto ra, Scontrino s):()
    pre: --
    post: A partire dalla descrizione del souvenir e dalla quantità memorizzate in *ra*, ottiene il souvenir che l'utente desidera acquistare (tramite il metodo *getSouvenirDaDescrizione()* della classe *Articoli*). Quindi inserisce, se non già presente, un nuovo link *comprende* con valore del campo *quantita* pari a quello della quantità specificata in *ra*, tra il souvenir ottenuto e lo scontrino *s* fornito in input; se il link tra il souvenir e lo scontrino è già presente, rimpiazza il link esistente con uno equivalente, la cui quantità è pari alla somma della quantità in *ra* e di quella memorizzata nel link preesistente.
FineSpecifica

InizioSpecificaAttivitàAtomica LeggiScontrino
    LeggiScontrino (Scontrino sc):(RecordScontrino)
    pre: --
    post: *return* è il *RecordScontrino* (v. sotto) contenente le stesse informazioni presenti nello Scontrino *sc* fornito in input.
FineSpecifica

RecordScontrino è un record usato per memorizzare le informazioni relative ad uno Scontrino. Esso contiene: un campo *numero* corrispondente al numero progressivo dello scontrino, un campo *emissione* corrispondente alla data di emissione, un campo *totale* corrispondente alla spesa totale associata allo scontrino, un insieme (campo *insieme*) di *RecordAcquisto* (v. sopra) corrispondente agli acquisti (descrizione e quantità) dei vari souvenir associati allo Scontrino.

Attività Composte

InizioSpecificaAttività AttivitaSottoramoBerretti

```
    AttivitaSottoramoBerretti(Scontrino scontrinoCorrente):()

    Variabili Processo: –
        ancoraBerretti: Boolean
        berrettiLetti: Set<String>
        berrettoSelezionato: RecordAcquisto
    Inizio Processo
        ancoraBerretti := true
        while(ancoraBerretti){
                LeggiBerretti():(berrettiLetti)
                MostraSouvenirPerSelezione(berrettiLetti):(berrettoSelezionato)
                AggiornaScontrino(berrettoSelezionato, scontrinoCorrente):()
        }
```

FineSpecifica

InizioSpecificaAttività AttivitaSottoramoMagneti

```
AttivitaSottoramoMagneti(Scontrino scontrinoCorrente):()
```

Variabili Processo: –

```
ancoraMagneti: Boolean
magnetiLetti: Set<String>
magneteSelezionato: RecordAcquisto
```

Inizio Processo

```
ancoraMagneti := true
while(ancoraMagneti){
        LeggiMagneti():(magnetiLetti)
        MostraSouvenirPerSelezione(magnetiLetti):(magneteSelezionato)
        AggiornaScontrino(magneteSelezionato, scontrinoCorrente):()
}
```

FineSpecifica

InizioSpecificaAttività AttivitaPrincipale

```
AttivitaPrincipale():()
```

Variabili Processo:

```
scontrinoCorrente: Scontrino
tipoAcquisti: RecordTipoAcquisti
recordScontrino: RecordScontrino
```

Inizio Processo

```
ChiediTipoSouvenir():(tipoAcquisti);
CreaNuovoScontrino():(scontrinoCorrente);
fork{
  thread t1:{
            if (tipoAcquisti.isMagneti()){
                      AttivitaSottoramoMagnete(scontrinoCorrente):();
            }
  }
  thread t2:{
            if (tipoAcquisti.isBerretti()){
                      AttivitaSottoramoBerretto(scontrinoCorrente):();
            }
  }
};
join t1, t2;
LeggiScontrino(scontrinoCorrente):(recordScontrino);
MostraScontrino(recordScontrino);
```

FineSpecifica

Progetto

Responsabilità sulle Associazioni

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

Associazione	Classe	Ha Responsabilità
comprende	Souvenir	SÌ (R)
	Scontrino	SÌ (O)

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
Souvenir	localita prezzo
Scontrino	numero emissione
Magnete	base altezza
Berretto	colore taglia

	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.