Dipartimento di Informatica e Sistemistica
Antonio Ruberti

“Sapienza” Università di Roma

Esercitazione 7

Corso di Tecniche di programmazione

Laurea in Ingegneria Informatica

(Canale di Ingegneria delle Reti e dei Sistemi Informatici - Polo di Rieti)

Anno Accademico 2007/2008

Tutor: Ing. Diego Rughetti

Esercizio 1 (1)

Si vuole gestire il noleggio di imbarcazioni da parte di un'agenzia di noleggio. Dell'agenzia di noleggio sono di interesse il nome ed il suo parco imbarcazioni. Per ciascuna imbarcazione sono di interesse: il tipo (motoscafo o gommone), un codice alfanumerico univoco, la capienza massima di persone, e il suo stato: se l'imbarcazione è noleggiata oppure no, e nel caso lo sia, il nominativo della persona che la ha noleggiata. Gli oggetti AgenziaNoleggio supportano le seguenti funzionalità :

- crea: che, data una stringa n che rappresenta il nome di un'agenzia di noleggio, crea un oggetto AgenziaNoleggio con nome n che inizialmente non ha alcuna imbarcazione a noleggio;
- agenzia: che restituisce il nome dell'agenzia di noleggio;
- nuovaImbarcazione: che, dato un codice b, un tipo t e un valore n, aggiunge una imbarcazione di tipo t, con codice b e capienza di persone n al parco dell'agenzia; se tale codice è già utilizzato lancia un'eccezione; si noti che un'imbarcazione appena acquisita non è noleggiata;
- dismetti: che, dato il codice b di una imbarcazione, la elimina dal parco dell'agenzia; se il codice b non è utilizzato non fa nulla; se il codice b è utilizzato ma la imbarcazione corrispondente è noleggiata lancia un'eccezione;
- verificaCodice: che, dato il codice b di una imbarcazione, restituisce true se il codice b è quello di una imbarcazione a noleggio nell'agenzia, false altrimenti;
- noleggiata: che, dato il codice b di una imbarcazione, restituisce il nominativo della persona che ha noleggiato l'imbarcazione b; se l'imbarcazione non è noleggiata restituisce null; infine, se il codice non è quello di una imbarcazione del noleggio lancia un'eccezione;
- noleggia: che, dati il codice b di una imbarcazione e il nominativo p di una persona, noleggia l'imbarcazione b alla persona p; se il codice b non è utilizzato oppure l'imbarcazione b è già noleggiata lancia un'eccezione;
- codiceImbarcazioneLibera: che, dato il tipo di imbarcazione t e un numero di persone n, restituisce il codice di una imbarcazione di tipo t con capienza pari a n, se non esiste una imbarcazione libera con tali caratteristiche, ne cerca una con capienza superiore a n, altrimenti restituisce null;
- barcheNoleggiate: che restituisce il numero di imbarcazioni noleggiate;
- personeBarcheNoleggiate: che restituisce un array di stringhe contenenti i nomi delle persona che hanno noleggiato una imbarcazione

Esercizio 1 (2)

Domanda 1. Scrivere una classe Java AgenziaNoleggio per rappresentare oggetti con le funzionalità descritte sopra. Specificare il costo asintotico in tempo e in spazio di ciascun metodo, indicando esplicitamente la dimensione dell'input, l'istruzione dominante e descrivendo il caso peggiore quando necessario.

Domanda 2. Realizzare un metodo statico verificaNoleggia cliente della classe AgenziaNoleggio che, dati un oggetto a della classe AgenziaNoleggio, una lista di stringhe l rappresentanti nomi di persone, verifichi che tutte le persone nella lista l abbiano noleggiato una imbarcazione presso l'agenzia a. La lista delle persone è memorizzata in un oggetto della classe Lista con la seguente interfaccia (sostituire a <TYPE> il tipo opportuno)

```
class Lista {
   public Lista()
   public boolean empty()                 // restituisce true se la lista e' vuota
   public <TYPE> firstElement()           // restituisce il primo elemento della lista
   public Lista insertFirstElement(<TYPE> x)// restituisce una lista ottenuta
                                             // inserendo x come primo elemento
   public Lista removeFirstElement()     // restituisce una lista ottenuta eliminando
                                         // il primo elemento
}
```

N.B. Supporre che le precedenti operazioni facciano side-effect sull'oggetto di invocazione.

Specificare il costo asintotico in tempo e in spazio del metodo, indicando esplicitamente la dimensione dell'input, l'istruzione dominante e descrivendo il caso peggiore quando necessario. Le operazioni della classe Lista hanno costo O(1)

Soluzione (1)

```
public class Imbarcazione{
	String tipo;
	String codice;
	int capienza;
	boolean noleggiata;
	String persona;
	Imbarcazione next;
	public Imbarcazione(String t, String c, int cap) throws Exception{
		if(!t.equals("motoscafo") && !t.equals("gommone")){
			throw new Exception("tipo passato come argomento errato!");
		}
		if(cap < 0){
			throw new Exception("numero posti negativo!");
		}
		this.tipo = t;
		this.codice = c;
		this.capienza = cap;
		this.noleggiata = false;
		this.persona = null;
	}
	public void noleggia(String p){
		this.noleggiata = true;
		this.persona = p;
	}
	public void restituzione(){
		this.noleggiata = false;
		this.persona = null;
	}
}
```

Soluzione (2)

```
public class AgenziaNoleggio{
	private String nome;
	private Imbarcazione init;
	public AgenziaNoleggio(String nome){
		this.nome = nome;
		init = null;
	}
	public String AgenziaNoleggio(){
		return this.nome;
	}
	public void nuovaImbarcazione(String codice, String tipo, int capienza) throws Exception{
		if(this.verificaCodice(codice)){
			throw new Exception("Codice barca già presente");
		}
		Imbarcazione b = new Imbarcazione(tipo, codice, capienza);
		b.next = init;
		init = b;
	}
```

Soluzione (3)

```
public void dismetti(String codice) throws Exception{
	if(!this.verificaCodice(codice)){
		return;
	}
	if(this.noleggiata(codice) != null){
		throw new Exception("barca noleggiata, impossibile dismettere");
	}
	Imbarcazione y = init;
	if(y.codice.equals(b)){
		init = init.next;
	}
	while(y.next != null){
		if(y.next.codice.equals(codice)){
			y.next = y.next.next;
			return;
		}else{
			y = y.next;
		}
	}
}
public boolean verificaCodice(String codice){
	if(this.init == null){
		return false;
	}
	Imbarcazione y = this.init;
	while(y!= null){
		if(y.codice.equals(codice)){
			return true;;
		}else{
			y = y.next;
		}
	}
	return false;	}
```

Soluzione (4)

```
public String noleggiata(String codice) throws Exception{
        if(!this.verificaCodice(codice)){
            throw new Exception("Codice imbarcazione non presente");
        }
        Imbarcazione y = this.init;
        while(y!= null){
            if(y.codice.equals(codice)){
                return y.persona;
            }else{
                y = y.next;
            }
        }
        return null;
    }
    public void noleggia(String codice, String persona) throws Exception{
        if(!this.verificaCodice(codice)){
            throw new Exception("Codice imbarcazione non presente");
        }
        Imbarcazione y = this.init;
        while(y!= null){
            if(y.codice.equals(codice)){
                if(y.noleggiata){
                    throw new Exception("Imbarcazione già noleggiata!");
                }else{
                    y.noleggia(persona);
                }
            }else{
                y = y.next;
            }
        }
    }
```

Soluzione (5)

```
public String codiceImbarcazioneLibera(String tipo, int capienza){
	if(this.init == null) return null;
	Imbarcazione y;
	Imbarcazione temp = null;
	y = this.init;
	while(y!= null){
		if(!y.noleggiata && y.capienza == capienza && y.tipo == tipo){
			return y.codice;
		}else if(!y.noleggiata && y.capienza >= capienza){
			temp = y;
		}
		y = y.next;
	}
	return temp.codice;
}
public int barcheNoleggiate(){
	if(this.init == null) return 0;
	int i = 0;
	Imbarcazione y = this.init;
	while(y!= null){
		if(y.noleggiata){
			i++;
		}
		y = y.next;
	}
	return i;
}
```

Soluzione (6)

```
public String[] personeBarcheNoleggiate(){
        if(this.init == null) return null;
        int i = 0;
        Imbarcazione y = this.init;
        String [] persone = new String[this.barcheNoleggiate()];
        while(y!= null){
            if(y.noleggiata){
                persone[i] = y.persona;
                i++;
            }
            y = y.next;
        }
        return persone;
    }
```

Soluzione (7)

```
public class ClienteAgenziaNoleggio{

	public static boolean verificaNoleggia(AgenziaNoleggio a, Lista l){
		String [] persone = a.personeBarcheNoleggiate();
		return this.verifica(persone, l);
	}

	public static boolean verifica(String [] s, Lista l){
	   if(l.empty()){
	      return true;
	   }
	   String persona = l.firstElement();
	   boolean s = verifica(s, l.removeFirstElement());
	   l.insertFirstElement(persona);
	   return (s && this.presente(persona, s));
	}

	private static boolean presente(String s, String[] pers){
		int i = 0;
		while(i<pers.length){
			if(s.equals(pers[i])){
				return true;
			}
			i++;
		}
		return false;
	}
}
```