




Fondamenti di Informatica L-B
Esercitazione n°7
Esercizi tipo esame



A.A. 2007/08
Tutor: Barbara Pettazzoni
barbara.pettazzoni@studio.unibo.it



Esame 15-09-2005

Esercizio 1

Il ristorante "I 4 palmenti" ha deciso di informatizzare la gestione delle ordinazioni. A tal scopo, le informazioni sui piatti serviti dal ristorante vengono memorizzate all'interno di un calcolatore. Ogni piatto appartiene ad una certa categoria ("primo", "secondo", o "contorno") e presenta un nome (univoco) ed un prezzo in euro.

2



Esame 15-09-2005

Si scriva una classe **Piatto** per il ristorante "I 4 palmenti" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti metodi che permettano di accedere alle variabili di istanza della classe.
3. Presenti il metodo toString che fornisca una descrizione del piatto (esclusa la categoria).
4. Implementi l'interfaccia Comparable, definendo il metodo compareTo per stabilire la precedenza con un altro piatto, verificata unicamente sulla categoria ("primo" precede "secondo" che precede "contorno").
5. Possieda un metodo equals per stabilire l'uguaglianza con un altro piatto (l'uguaglianza va verificata unicamente sul nome).

3



Esame 15-09-2005

```
class Piatto implements Comparable {  
    private String nome, tipo;  
    private float prezzo;  
    //costruttore  
    public Piatto(String nome, String tipo, float prezzo) {  
        this.nome=nome; this.tipo=tipo; this.prezzo=prezzo;  
    }  
  
    public String getNome() { return nome; }  
  
    public String getTipo() { return tipo; }  
  
    public float getPrezzo() { return prezzo; }  
  
    ...  
}
```

4



Esame 15-09-2005

```
...
public int compareTo(Piatto p) {
    if(this.tipo.equals("primo"))
        if(p.tipo.equals("primo")) return 0;
        else return -1;
    if(p.tipo.equals("primo")) return 1;
    if(this.tipo.equals("secondo"))
        if(p.tipo.equals("secondo")) return 0;
        else return -1;
    if(p.tipo.equals("secondo")) return 1;
    return 0;
}

...
```

5



Esame 15-09-2005

```
...

public boolean equals(Piatto p) {
    return (nome.equals(p.nome));
}

public boolean equals(Object o) {
    return equals((Piatto) o);
}

public String toString() { return nome+"\t"+prezzo; }

public int compareTo(Object o) {
    return compareTo((Piatto) o);
}

}
```

6



Esame 15-09-2005

Esercizio 2

Si scriva una classe **Menu** che memorizzi le informazioni relative al menù del ristorante, che contiene una lista dei piatti serviti dal ristorante (che deve essere mantenuta ordinata per categoria: prima i primi, quindi i secondi ed infine i contorni).

La classe **Menu** deve inoltre:

1. Possedere un opportuno costruttore (la lista dei piatti deve inizialmente essere vuota).
2. Fornire un metodo `trovaPiatto` che, dato il nome di un piatto, restituisca il corrispondente oggetto `Piatto`, se questo è contenuto all'interno del menù.

7



Esame 15-09-2005

3. Provvedere un metodo `aggiungi` che, dato un piatto, lo inserisca all'interno della lista, mantenendo l'ordinamento sulla categoria. (Suggerimento: si utilizzi il metodo `add(int i, Object o)` della classe `List`).
4. Fornire il metodo `numeroPiatti` che, dato il nome di una categoria, restituisca il numero totale di piatti presenti in tale categoria.
5. Presentare il metodo `toString` che restituisca una stringa che fornisca una descrizione del menù, comprese le descrizioni di tutti i piatti separati per tipologia (primi, secondi, ecc.).

8



Esame 15-09-2005

```
import java.util.*;

class Menu {
    private List portate;

    //costruttore
    public Menu() {
        this.portate=new LinkedList();
    }

    public String toString() {
        String s="";
        for(int i=0; i<portate.size(); i++)
            s+=portate.get(i)+"\n";
        return s;
    }
}
```

9

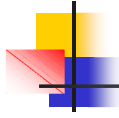


Esame 15-09-2005

```
...
public Piatto trovaPiatto(String nome) {
    for(int i=0; i<portate.size(); i++) {
        Piatto p=(Piatto) portate.get(i);
        if(p.getNome().equals(nome)) return p;
    }
    return null;
}

public void aggiungi(Piatto p) {
    int i=0;
    while((i<portate.size())&&
        (p.compareTo(portate.get(i))>=0)) i++;
    portate.add(i, p);
}
...
```

10



Esame 15-09-2005

```
...
public int numeroPiatti(String tipo) {
    int i=0, N=0;
    while((i<portate.size()) &&
        (!(Piatto)portate.get(i)).getTipo().equals(tipo)) i++;
    while((i<portate.size()) &&
        ((Piatto)portate.get(i)).getTipo().equals(tipo)) {
        i++;
        N++;
    }
    return N;
}
}
```

11



Esame 15-09-2005

Esercizio 3

Si scriva un'applicazione per il ristorante "I 4 palmenti" che:

1. Crei un oggetto di tipo Menu.
2. Crei un insieme ordinazione di oggetti Piatto.
3. Letti da tastiera il nome di alcuni piatti, provveda a ricercare tali piatti all'interno del menù e li inserisca all'interno dell'insieme ordinazione.
4. Stampi a video lo scontrino relativo ai piatti contenuti nell'insieme ordinazione, compresa la spesa totale del pasto.

12



Esame 15-09-2005

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `boolean leggiBoolean()` → Legge un boolean (delimitato da spazi).
- `char leggiChar()` → Legge un singolo carattere.
- `double leggiDouble()` → Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` → Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` → Legge un intero (delimitato da spazi).
- `String leggiLinea()` → Legge una linea di testo.
- `String leggiString()` → Legge una parola senza spazi al suo interno.

13



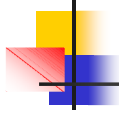
Esame 15-09-2005

```
import java.util.*;
import fiji.io.*;

class Applicazione {

    public static void main(String[] args) {
        Menu m=new Menu(); // domanda 1
        Set ord=new TreeSet(); // domanda 2
        int N=Lettore.in.leggiInt();
        float totale=0;
        // domanda 3
        for(int j=0; j<N; j++)
            ord.add(m.trovaPiatto(Lettore.in.leggiLinea()));
        ...
    }
}
```

14



Esame 15-09-2005

```
...
Iterator i=ord.iterator();
while(i.hasNext()) {
    Piatto p=(Piatto) i.next();
    totale+=p.getPrezzo();
    System.out.println(p);
}
System.out.println("Totale\t"+totale); // domanda 4
}
```

15



Esame 12-12-2005

Esercizio1

In previsione del Santo Natale, la cooperativa "Babbi Natale" sta approntando le slitte che caricheranno i regali da distribuire ai bambini di tutto il mondo. Visto il numero sempre crescente di letterine, la cooperativa ha deciso di informatizzare la gestione delle slitte e dei relativi carichi di regali. Le informazioni sui bambini da memorizzare comprendono il nome, l'indirizzo, la data di nascita ed il regalo richiesto dal bimbo nella letterina spedita a Babbo Natale.

16



Esame 12-12-2005

Si scriva una classe **Bambino** per la cooperativa "Babbi Natale" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Presenti il metodo toString che fornisca una descrizione testuale del bambino.
4. Possieda un metodo eta che calcoli l'età del bambino al giorno di Natale.
5. Possieda un metodo equals per stabilire l'uguaglianza con un altro bambino (l'uguaglianza va verificata sul nome e l'indirizzo).

17



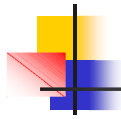
Esame 12-12-2005

```
import java.util.*;

class Bambino {
    private String nome, indirizzo, regalo;
    private int g, m, a;
    //costruttore
    public Bambino(String nome, String indirizzo, String regalo,
                   int g, int m, int a) {

        this.nome=nome;
        this.indirizzo=indirizzo;
        this.regalo=regalo;
        this.g=g;
        this.m=m;
        this.a=a;
    }
    ...
}
```

18



Esame 12-12-2005

```
...
public String getNome() { return nome; }

public String getIndirizzo() { return indirizzo; }

public String getRegalo() { return regalo; }

public String getData() { return ""+g+"/"+m+"/"+a; }

public int eta() {
    if((m<12)|| (g<=25)) return 2005-a;
    return 2005-a-1;
}
...
```

19



Esame 12-12-2005

```
...
public String toString() {
    return nome + ", " + indirizzo + ", nato il " + getData() +
        ": "+regalo;
}

public boolean equals(Bambino b) {
    return(nome.equals(b.nome) && indirizzo.equals(b.indirizzo));
}

public boolean equals(Object o) {
    return equals((Bambino) o);
}
}
```

20



Esame 12-12-2005

Esercizio 2

Si scriva una classe **Slitta** che memorizzi le informazioni relative alle slitte della cooperativa "Babbi Natale". In particolare ogni slitta, oltre alla propria targa, deve memorizzare le informazioni sui bambini che riceveranno i regali caricati su tale slitta all'interno di un insieme. Le slitte hanno capienze diverse, quindi il numero massimo di regali che possono essere caricati su ogni slitta va specificato all'atto della creazione della slitta.

La classe **Slitta** deve inoltre:

1. Possedere un opportuno costruttore con parametri e metodi che accedano ai dati della slitta.

21



Esame 12-12-2005

2. Presentare un metodo `cercaBambino` che, dato il nome e l'indirizzo di un bambino, indichi se la slitta contiene o meno il regalo destinato a tale bimbo.
3. Possedere un metodo `piena` che indichi se la slitta ha raggiunto o meno la sua capienza massima in termini di numero di regali.
4. Presentare un metodo `aggiungi` che, dato un oggetto `Bambino`, provveda ad inserirlo nell'insieme dei regali, a patto che la capienza massima della slitta non sia già stata raggiunta; il metodo deve inoltre indicare se il bambino è stato effettivamente inserito o meno.
5. Presentare il metodo `toString` che restituisca una stringa che fornisca una descrizione della slitta, comprese le descrizioni dei bambini che riceveranno i regali caricati sulla slitta stessa.

22



Esame 12-12-2005

```
import java.util.*;

class Slitta {
    private String targa;
    private int capienza;
    private Set bambini;
    //costruttore
    public Slitta(String targa, int capienza) {
        this.targa=targa;
        this.capienza=capienza;
        this.bambini=new HashSet();
    }

    public String getTarga() { return targa; }

    public boolean piena() { return(bambini.size()==capienza); }
}
```



Esame 12-12-2005

```
public boolean cercaBambino(String nome, String indirizzo) {
    Bambino b=new Bambino(nome, indirizzo, "", 1, 1, 2000);
    return(bambini.contains(b));
}

public boolean aggiungi(Bambino b) {
    if(piena()) return false;
    bambini.add(b);
    return true;
}

public String toString() {
    return targa + ", capienza: "+capienza+": "+bambini;
}
}
```



Esame 12-12-2005

Esercizio 3

Si scriva un'applicazione per la cooperativa "Babbi Natale" che:

1. Crei una lista di oggetti Slitta.
2. Lette da tastiera le informazioni relative ad una nuova slitta, provveda ad inserire un nuovo oggetto in coda alla lista.
3. Crei un oggetto di tipo Bambino, lette da tastiera le informazioni necessarie.
4. Cerchi se esiste una slitta, tra quelle in lista, che contiene i dati del bambino di cui al punto 3.
5. In caso che la ricerca del punto 4. dia risultato negativo, inserisca le informazioni del bambino all'interno della prima slitta in lista che non sia già piena.

25



Esame 12-12-2005

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `char leggiChar()` → Legge un singolo carattere.
- `double leggiDouble()` → Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` → Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` → Legge un intero (delimitato da spazi).
- `String leggiLinea()` → Legge una linea di testo.
- `String leggiString()` → Legge una parola senza spazi al suo interno.

26



Esame 12-12-2005

```
import java.util.*;
import fiji.io.*;

class Applicazione {

    public static void main(String[] args) {
        List slitte=new LinkedList(); // domanda 1
        Bambino b;
        int i=0;
        // domanda 2
        slitte.add(new Slitta(Lettore.in.leggiString(),
                               Lettore.in.leggiInt()));

        ...
    }
}
```

27



Esame 12-12-2005

```
        // domanda 3
        b=new Bambino(Lettore.in.leggiString(),
                      Lettore.in.leggiString(),
                      Lettore.in.leggiString(), Lettore.in.leggiInt(),
                      Lettore.in.leggiInt(), Lettore.in.leggiInt());
        // domanda 4
        while(i<slitte.size())
            if(!((Slitta) (slitte.get(i))).cercaBambino(b.getNome(),
                                                         b.getIndirizzo())) i++;
        // domanda 5
        if(i==slitte.size()) {
            i=0;
            while(((Slitta) (slitte.get(i))).piena()) i++;
            ((Slitta) (slitte.get(i))).aggiungi(b);
        }
    }
}
```

28



Esame 18-07-2005

Esercizio 1

La Azienda Trasporti Comunali della città di Collate (Parma) vuole informatizzare il servizio di autobus operativo nelle strade comunali. A tal scopo, le informazioni sulle linee vengono memorizzate all'interno di un calcolatore. Ogni linea corre una sola volta ogni giorno (durante la notte le corse sono sospese) e prevede un certo numero di fermate, di cui vanno memorizzati alcuni dati, come il nome della fermata ed i minuti impiegati per raggiungere tale fermata a partire dal capolinea.

29



Esame 18-07-2005

Si scriva una classe **Fermata** che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti metodi che permettano di accedere alle variabili di istanza della classe.
3. Presenti il metodo toString che fornisca una descrizione della fermata.
4. Implementare l'interfaccia Comparable, definendo il metodo compareTo per stabilire la precedenza con un'altra fermata, verificata sul tempo necessario per raggiungere le fermate.
5. Possieda un metodo equals per stabilire l'uguaglianza con un'altra fermata (l'uguaglianza va verificata unicamente sul nome).

30



Esame 12-12-2005

```
class Fermata implements Comparable {  
    private String nome;  
    private int minuti;  
    //costruttore  
    public Fermata(String nome, int minuti) {  
        this.nome=nome;  
        this.minuti=minuti;  
    }  
  
    public String getNome() { return nome; }  
  
    public int getMinuti() { return minuti; }  
  
    public String toString() { return nome+" (" +minuti+")"; }  
  
    public int compareTo(Fermata f) { return minuti-f.minuti; }  
}
```



Esame 12-12-2005

```
...  
    public boolean equals(Fermata f) {  
        return (nome.equals(f.nome));  
    }  
  
    public int compareTo(Object o) {  
        return compareTo((Fermata)o);  
    }  
  
    public boolean equals(Object o) {  
        return equals((Fermata) o);  
    }  
}
```




Esame 18-07-2005

Esercizio 2

Si scriva una classe `Linea` che memorizzi le informazioni relative ad ogni linea, tra cui il numero della linea, i dati relativi al conducente, l'orario di partenza ed una lista delle fermate effettuate dalla linea (che deve essere mantenuta ordinata).

La classe `Linea` deve inoltre:

1. Possedere un opportuno costruttore (la lista delle fermate deve inizialmente essere vuota).
2. Presentare metodi per accedere alle variabili di istanza (numero e dati del conducente).
3. Fornire un metodo `trovaFermata` che, dato il nome di una fermata, se questa è raggiunta dalla linea, restituisca l'indice corrispondente all'interno della lista, o un valore negativo, altrimenti.

33



Esame 18-07-2005

4. Possedere un metodo `orario` che, dato un valore intero, restituisca una stringa che rappresenti l'orario in cui la fermata corrispondente a tale indice viene effettuata.
5. Provvedere un metodo `aggiungi` che, data una fermata, inserisca tale fermata all'interno della lista, mantenendo l'ordinamento sul tempo di visita delle fermate. (Suggerimento: si utilizzi il metodo `add(int i, Object o)` della classe `List`).
6. Presentare il metodo `toString` che restituisca una stringa che fornisca una descrizione della linea, compresi gli orari di arrivo a tutte le fermate.
7. Fornire il metodo `equals` per il confronto tra due linee (tenendo conto del solo numero).

34



Esame 12-12-2005

```
import java.util.*;

class Linea {
    private int numero, ora, minuto;
    private String nome, cognome;
    private List fermate;
    //costruttore
    public Linea(int numero, int h, int m, String nome, String
                                                cognome) {

        this.numero=numero;
        this.ora=h; this.minuto=m;
        this.nome=nome; this.cognome=cognome;
        this.fermate=new LinkedList();
    }
    ...
}
```

35



Esame 12-12-2005

```
...
public int getNumero() { return numero; }

public String getnome() { return nome+" "+cognome; }

public int trovaFermata(String nome) {
    for(int i=0; i<fermate.size(); i++)
        if(((Fermata) fermate.get(i)).getNome().equals(nome))
            return i;
    return -1;
}

public String orario(int n) {
    int h=ora+(minuto+((Fermata) fermate.get(n)).getMinuti())/60;
    int m=(minuto+((Fermata) fermate.get(n)).getMinuti())%60;
    return "" + h + ":" + m;
}
}
```

36



Esame 12-12-2005

```
public void aggiungi(Fermata f) {
    int i=0;
    while((i<fermate.size())&&
        (f.compareTo(fermate.get(i))>=0)) i++;
    fermate.add(i, f);
}

public String toString() {
    String s="" + numero + "\n" + getNome() + "\nPartenza:"
        + ora + ":" + minuto;

    for(int i=0; i<fermate.size(); i++)
        s+="\n" + ((Fermata) fermate.get(i)).getNome()
            + "\t" + orario(i);

    return s;
}

public boolean equals(Linea l) { return numero==l.numero; }
public boolean equals(Object o){ return equals((Linea) o); }
}
```



Esame 18-07-2005

Esercizio 3

Si scriva un'applicazione per l'Azienda Trasporti Comunali di Collate (Parma) che:

1. Crei un vettore di 50 oggetti Linea.
2. Lette da tastiera il numero di una linea e le informazioni relative ad alcune fermate, provveda ad inserire tali fermate nella linea corrispondente.
3. Letti da tastiera i nomi di due fermate, provveda a visualizzare le informazioni sulla linea che comprende entrambe le fermate (la seconda dopo la prima) e visita la seconda fermata prima possibile.



Esame 18-07-2005

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `char leggiChar()` → Legge un singolo carattere.
- `double leggiDouble()` → Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` → Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` → Legge un intero (delimitato da spazi).
- `String leggiLinea()` → Legge una linea di testo.
- `String leggiString()` → Legge una parola senza spazi al suo interno.

39



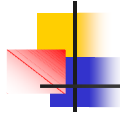
Esame 12-12-2005

```
import fiji.io.*;

class Applicazione {

    public static void main(String[] args) {
        Linea[] v=new Linea[50]; // domanda 1
        int linea=Lettore.in.leggiInt();
        int N=Lettore.in.leggiInt(), l=0, i;
        while(v[l].getNumero()!=linea) l++;
        // domanda 3
        for(i=0; i<N; i++)
            v[l].aggiungi(new Fermata(
                Lettore.in.leggiLinea(),Lettore.in.leggiInt()));
        String n1=Lettore.in.leggiLinea();
        String n2=Lettore.in.leggiLinea();
        String orario="";
```

40



Esame 12-12-2005

```
l=-1;
// domanda 3
for(i=0; i<v.length; i++) {
    int i1=v[i].trovaFermata(n1);
    int i2=v[i].trovaFermata(n2);
    if((i1>=0)&&(i2>=i1)&&((l<0)||
        (v[i].orario(i2).compareTo(orario)<0))) {
        l=i;
        orario=v[i].orario(i2);
    }
}
}
```