

Esercizi di preparazione alla Seconda Parte

Nel compito gli esercizi sulla seconda parte sono del tipo degli esercizi da A a H. Gli altri esercizi sono di ausilio alla comprensione di alcuni argomenti che possono essere trattati nelle domande.

Esercizio A

1. Scrivere la classe **Pacco** avente i seguenti attributi **base**, **altezza** e **profondità**: tre numeri con la virgola rispettivamente la base, l'altezza e la profondità del pacco, e l'attributo **numero** di tipo numero intero che identifica il pacco. Il costruttore ha come parametri tre numeri con la virgola (coerente con la scelta fatta per gli attributi base altezza e profondità) ed assegna il valore dei parametri agli attributi base, altezza e profondità. Scrivere il metodo **setNumero** che ha come parametro una variabile di tipo numero intero, tale metodo assegna il valore del parametro all'attributo numero. Scrivere il metodo **volume** che restituisce il valore del volume del pacco (prodotto dei tre attributi base, altezza e profondità).
2. Scrivere la sottoclasse **PaccoAColori** della classe **Pacco**, avente in più l'attributo **colore** di tipo stringa. La sottoclasse ha i seguenti metodi:
 - il metodo **setColore** che ha un parametro di tipo stringa, il metodo assegna all'attributo colore il valore del parametro
 - il metodo **stampa** che stampa a video i valori degli attributi.
 - Il metodo **stampaVolume** che stampa a video il valore del volume utilizzando il metodo **volume**
3. Scrivere la classe **Prova** che ha il metodo **main** in cui occorre istanziare un oggetto di tipo **Pacco** con base 3,5 altezza 4,5 e profondità 6,0. Richiamare il metodo **setNumero** con parametro pari a 3 e stampare il valore del volume del pacco richiamando opportunamente il metodo **volume**. Istanziare un altro oggetto di tipo **PaccoAColori** con base 2,4 altezza 5,3 profondità 5,5 e richiamare il metodo **setColore** con colore *rosso*. Infine richiamare il metodo **stampaVolume** per stampare il volume del pacco a colori.

Esercizio B

1. Scrivere la classe **Poligono** avente come attributi **numeroLati**: un intero indicante il numero dei lati del poligono; e **lato**: un numero con la virgola indicante la lunghezza dei lati (che sono tutti uguali). Scrivere il metodo costruttore che ha 2 parametri di tipo numerico, il primo è un intero indicante il numero dei lati ed il secondo è dello stesso tipo di lato ed indica la lunghezza del lato, il metodo assegna i valori dei parametri ai rispettivi attributi. Scrivere inoltre:
 - il metodo **perimetro** che restituisce il valore del perimetro del poligono
 - il metodo **stampa** che stampa il numero dei lati del poligono e il valore del lato
 - il metodo **augmenta** che ha un parametro di tipo numero con la virgola, il metodo modifica il valore dell'attributo lato aggiungendogli il valore del parametroIl metodo **volume** che ha come parametro un numero intero indicante l'altezza, il metodo se numeroLati è uguale a 4 restituisce il risultato del prodotto di lato per lato per il valore del parametro altrimenti restituisce il valore zero.
2. Scrivere la classe **Cubo** avente come attributo **peso** un numero con la virgola indicante il peso. Il costruttore della classe ha un solo parametro **x** indicante la lunghezza del lato (quindi dello stesso tipo dell'attributo lato) e richiama il metodo costruttore della superclasse passando come parametri 4 e x. Scrivere il metodo **setPeso** che ha come parametro un

numero con la virgola ed assegna il valore del parametro all'attributo peso. Sovraccaricare il metodo **volume** scrivendo un metodo volume senza parametri: il metodo restituisce il valore del prodotto di lato*lato*lato.

3. Scrivere la classe **UsoPoligono.java** che istanzia un oggetto di tipo Poligono con parametri 4 e 6,8 . Quindi richiamare il metodo **perimetro** e stamparne a video il valore, utilizzare il metodo **stampa** per visualizzare a video il numero dei lati e il loro valore. Infine richiamare il metodo **aumenta** con parametro 1,2 ed il metodo **volume** con parametro 5 stampando a video il valore del volume.

Esercizio C

1. Scrivere la classe **SalaCinema** che ha i seguenti attributi **film** indicante il nome del film proiettato, **postiOcc** il numero di posti occupati, **n_file** indicante il numero delle file di posti a sedere contenuti nella sala e **maxFila** indicante il numero di posti a sedere per fila il cui valore è 21. Il costruttore ha come parametri una stringa per il nome del film e un intero che indica il numero delle file ed assegna opportunamente i valori dei parametri agli attributi film ed n_file ed assegna 0 a postiOcc. Scrivere il metodo **getPosti** che restituisce il numero dei posti ancora liberi (per calcolare il numero di posti liberi occorre calcolare il numero di posti di cui dispone la sala che è uguale al prodotto del numero di file per maxFila e poi a questo sottrarre il numero di posti occupati), scrivere inoltre il metodo **vendiBiglietti** che ha come parametro un numero intero indicante il numero di posti da occupare, il metodo aggiunge all'attributo posti occupati il valore del parametro
2. Scrivere la sottoclasse **SalaCinemaP** che ha in più l'attributo **fileOccupate** che indica il numero di file per le quali non ci sono posti disponibili. Aggiungere il metodo **calcolaFile** che calcola il numero di file per le quali non ci sono posti disponibili assegnandolo all'attributo fileOccupate (occorre dividere il numero di posti occupati per maxFila). Aggiungere il metodo **stampaSituazione** che stampa tutti gli attributi della classe.
3. Scrivere al classe Esonero2a che istanzia un oggetto di tipo SalaCinema con nome del film pari a Van Helsing e numero di file pari a 15 e richiama i metodi vendiBiglietti con parametro pari a 5 e stampare i posti disponibili richiamando il metodo getPosti. Istanziare un oggetto di tipo SalaCinemaP con nome del film pari a Scooby-Doo 2: mostri scatenati e numero di file pari a 18, richiamare il metodo stampaSituazione poi il metodo vendiBiglietti con parametro pari a 24 quindi stampare i posti disponibili richiamando il metodo getPosti, richiamare il metodo calcolaFile e poi stampaSituazione.

Esercizio D

1. Scrivere la classe **Teatro** che ha i seguenti attributi **spettacolo** indicante il nome dello spettacolo che avrà luogo al teatro, **postiPlatea** il numero di posti in platea, **postiGalleria** indicante il numero di posti in galleria. Il costruttore ha come parametri una stringa per il nome dello spettacolo e un intero per il numero dei posti in platea ed un intero per il numero dei posti galleria ed assegna opportunamente i valori dei parametri agli attributi spettacolo, postiPlatea e postiGalleria. Scrivere il metodo **assegnaPostiPlatea** che ha un parametro intero indicante il numero di posti da occupare in platea, il metodo aggiorna l'attributo postiPlatea sottraendogli il valore del parametro. Scrivere il metodo **assegnaPostiGalleria** che ha un parametro intero indicante il numero di posti da occupare in galleria, il metodo aggiorna l'attributo postiGalleria sottraendogli il valore del parametro.

2. Scrivere la sottoclasse **VenditeT**. Aggiungere il metodo **vendiBiglietti** che ha come parametro un numero intero **x** indicante il numero di posti da occupare, ed un boolean per capire l'operazione da eseguire, il metodo se il parametro booleano è true utilizza il metodo **assegnaPostiGalleria** passandogli **x**, altrimenti se il parametro booleano è false utilizza il metodo **assegnaPostiPlatea** passandogli **x**. Scrivere inoltre il metodo **riepilogo** che ha come parametro un boolean e se il valore del parametro è true restituisce il valore dell'attributo **postiGalleria**, altrimenti restituisce il valore dell'attributo **postiPlatea**.
3. Scrivere la classe **Esonero2b** che istanzia un oggetto di tipo **Teatro** con titolo dello spettacolo **Aggiungi un posto a tavola**, numero di posti in platea pari a 50 e numero di posti in galleria pari a 20. Stampare il valore degli attributi dell'oggetto, richiamare il metodo **assegnaPostiPlatea** con parametro pari a 15 e **assegnaPostiGalleria** con parametro pari a 5. Istanziare un oggetto di tipo **VenditeT** con titolo dello spettacolo **Rugantino**, numero di posti in platea pari a 40 e numero di posti in galleria pari a 15. Richiamare il metodo **vendiBiglietti** con primo parametro pari a 15 e secondo pari a false, stampare i posti di platea disponibili richiamando il metodo **riepilogo** con parametro posto a false

Esercizio E

1. Scrivere la classe **ContoCorrente** che ha i seguenti attributi: **titolare** una stringa indicante il cognome e nome dell'intestatario del conto, **nMovimenti** un intero indicante il numero dei movimenti effettuati, **maxMovimenti** una costante intera indicante il numero massimo dei movimenti gratuiti pari a 50, **saldo** un numero con la virgola indicante i soldi depositati sul conto. Il costruttore ha 2 parametri: 1 stringa per inizializzare l'attributo **titolare** e 1 parametro numerico coerente con la scelta fatta per l'attributo **saldo** per inizializzare l'attributo **saldo**, infine il costruttore pone a zero l'attributo **nMovimenti**. Scrivere il metodo **preleva** che ha un parametro **x** (di tipo coerente con la scelta fatta per l'attributo **saldo**) che indica quanti euro si vogliono prelevare, il metodo se il valore di **nMovimenti** è minore di **maxMovimenti** assegna a **saldo** il valore di **saldo** meno il parametro **x** altrimenti assegna a **saldo** il valore di **saldo** meno il parametro **x** e meno 0,50, infine il metodo incrementa di uno il numero di movimenti ossia l'attributo **nMovimenti**. Scrivere il metodo **restituisceSaldo** che restituisce il valore di **saldo**.
2. Scrivere la sottoclasse **ContoOnLine** della classe **ContoCorrente** che ha in più l'attributo **maxPrelievo**: un numero con la virgola (coerente con la scelta fatta per l'attributo **saldo**) indicante la cifra che al massimo si può prelevare, tale attributo sarà inizializzato nel costruttore. Scrivere il metodo **stampaSaldo** che stampa il valore degli attributi **titolare** e **saldo**. Ridefinire il metodo **preleva** in modo che se il parametro indicante la cifra da prelevare è minore dell'attributo **maxPrelievo** richiama il metodo **preleva** della superclasse altrimenti non fa nulla.
3. Scrivere la classe **Banca** che ha il metodo **main** in cui istanziare un oggetto di tipo **ContoCorrente** con titolare *Grossi Mario* e saldo 20000,00 €, richiamare i metodi **preleva** con parametro pari a 1750,5 e stampare il valore del saldo richiamando il metodo **restituisceSaldo**. Quindi istanziare un oggetto di tipo **ContoOnLine** con titolare *Rossi Luigi* e saldo 50350,00 € e **maxPrelievo** 1500, stampare il valore del saldo richiamando il metodo **stampaSaldo** ed infine richiamare il metodo **preleva** con parametro pari a 2000.

EsercizioF

1. Scrivere la classe **Motorino** che ha i seguenti attributi **colore**: una stringa indicante il colore dell'automobile, **velocità**: un numero con la virgola indicante la velocità in Km/h che possiede il motorino, **tipo**: una stringa indicante la marca e il modello del motorino es. "Piaggio scarabeo", l'attributo **antifurto** un boolean che indica se è stato inserito l'antifurto (ha un valore iniziale pari a false). Il costruttore ha come parametri una stringa per il colore, una stringa per il tipo, un numero con la virgola per la velocità ed assegna opportunamente i valori dei parametri agli attributi. Scrivere il metodo **getVelocità** che restituisce la **velocità** del motorino, scrivere inoltre il metodo **accelera** che ha come parametro un numero con la virgola indicante i Km/h che si vogliono aggiungere alla velocità, il metodo verifica il valore dell'attributo **antifurto** se è false aggiunge il valore del parametro all'attributo velocità, altrimenti non fa nulla. Scrivere il metodo **inserisciAntifurto** che assegna un valore true all'attributo **antifurto**.
2. Scrivere la classe **MotorinoImmatricolato** sottoclasse della classe Motorino che ha in più 2 attributi: **maxVelocità** un numero con la virgola (coerente con la scelta fatta per l'attributo velocità) indicante la velocità massima in Km/h che il motorino può raggiungere; **targa** una stringa indicante la targa del motorino (a entrambi gli attributi viene assegnato un valore nel costruttore). Aggiungere il metodo **getMax** il metodo stampa il valore dell'attributo maxVelocità. Ridefinire il metodo **accelera** in modo che prima di modificare la velocità effettui un controllo sulla velocità massima raggiunta. Il metodo definisce una variabile s (dello stesso tipo di velocità) ed assegna ad s la somma tra il valore del parametro del metodo ed il valore dell'attributo velocità; se s è minore del valore dell'attributo maxVelocità assegna il valore di s all'attributo velocità altrimenti assegna all'attributo velocità il valore dell'attributo maxVelocità.
3. Scrivere la classe **UsoMotorino** che ha il metodo main in cui si istanzia un oggetto di tipo Motorino di colore *grigiometallizzato* con velocità *40,5* e tipo *Piaggio Liberty*, richiamare il metodo getVelocità memorizzandone il valore in una variabile. Istanziare un altro oggetto di tipo MotorinoImmatricolato con colore *rosso*, velocità *30,5* tipo *Aprilia Scarabeo*, maxvelocità *60* targa *CV1234*, e richiamare il metodo getMax. Infine richiamare il metodo accelera con parametro pari a *30,7* per entrambi gli oggetti e quindi richiamare il metodo getVelocità per entrambi gli oggetti stampando i due valori.

EsercizioG

1. Scrivere la classe **Dipendente** che ha i seguenti attributi **matricola**: una stringa indicante il numero di matricola del dipendente, **stipendio**: un numero con la virgola indicante lo stipendio base che possiede il dipendente, **straordinario**: un numero con la virgola indicante l'importo dovuto per ciascuna ora di straordinario effettuata dal dipendente. Il costruttore ha come parametri una stringa per la matricola, un numero con la virgola per lo stipendio ed un numero con la virgola per lo straordinario ed assegna opportunamente i valori dei parametri agli attributi. Scrivere il metodo **getStipendio** che restituisce il valore dell'attributo stipendio, scrivere inoltre il metodo **paga** che ha come parametro un numero intero indicante il numero di ore di straordinario effettuate dal dipendente, il metodo restituisce il valore ottenuto sommando all'attributo stipendio il risultato del prodotto tra il parametro del metodo e l'attributo straordinario. Scrivere il metodo **stampa** che stampa il valore degli attributi della classe.
2. Scrivere la sottoclasse **DipendenteA** della classe Dipendente che ha in più l'attributo **malattia**: un numero intero indicante i giorni di malattia presi che viene inizializzato a zero. Aggiungere il metodo **prendiMalattia** che ha come parametro un numero intero indicante i giorni di malattia presi, il metodo modifica il valore dell'attributo malattia aggiungendogli il

valore del parametro. Ridefinire il metodo **paga** in modo che prima definisce una variabile **p** a cui assegna il valore ottenuto richiamando il metodo **paga** della superclasse, poi se l'attributo **malattia** è uguale a zero il metodo restituisce il valore di **p** altrimenti restituisce il valore ottenuto sottraendo a **p** il valore del prodotto tra **malattia** e 15,0. Definire un metodo **stampaMalattia** che stampa il valore dell'attributo **malattia**

3. Scrivere la classe **CalcolaStipendi** che ha il metodo **main** in cui istanziare un oggetto di tipo **Dipendente** con matricola *00309* stipendio *1000,00* e straordinario *7,50*. Richiamare il metodo **paga** con parametro pari a *10*, stampare il valore dello stipendio di tale oggetto richiamando **getStipendio**. Istanziare un altro oggetto di tipo **DipendenteA** con matricola *00201* stipendio *1500,0* e straordinario *8,50*. Richiamare il metodo **prendiMalattia** con parametro pari a *5*, richiamare il metodo **paga** con parametro pari a *3* ed il metodo **stampaMalattia**.

EsercizioH

1. Scrivere la classe **TipoCapoAbbigliamento** che ha i seguenti attributi **marca**:una stringa indicante la marca, **modello**: una stringa indicante il modello, **costo**:un numero con la virgola indicante il costo in euro. Il costruttore ha come parametri una stringa per la marca, una stringa per il modello, un numero con la virgola per il costo ed assegna opportunamente i valori dei parametri agli attributi. Scrivere il metodo **getCosto** che stampa il valore dell'attributo **costo**. Scrivere inoltre il metodo **applicaSconto** che ha come il parametro un numero con la virgola indicante la percentuale di sconto che si vuole applicare, il metodo restituisce il costo del capo di abbigliamento scontato ossia il risultato della sottrazione tra **costo** e **costo** moltiplicato il valore del parametro diviso 100. Scrivere il metodo **modificaCosto** che ha come parametro un numero con la virgola (dello stesso tipo di **costo**) indicante il nuovo costo ed assegna all'attributo **costo** il valore del parametro.
2. Scrivere la classe **CapoAbbigliamento** sottoclasse della classe **TipoCapoAbbigliamento** che ha in più 3 attributi: **taglia** un numero intero che indica la taglia; **quantitàAcquistata** un numero intero indicante il numero di pezzi acquistati, **quantitàDisponibile** un numero intero indicante il numero di pezzi disponibili (agli attributi viene assegnato un valore del costruttore). Scrivere il metodo **venduto** avente come parametro un numero intero indicante il numero di capi venduti, il metodo se **quantitàDisponibile** è maggiore del risultato della sottrazione tra **quantitàDisponibile** ed il parametro, modifica il valore dell'attributo **quantitàDisponibile** sottraendogli il valore del parametro. Scrivere il metodo **getDettagli** che stampa tutti gli attributi della classe.
3. Scrivere la classe **Negozi** che ha il metodo **main** in cui istanziare un oggetto di tipo **TipoCapoAbbigliamento** di marca *Colmar* modello *Giacca E45* con prezzo *350,5* e richiamare il metodo **getCosto** ed il metodo **applicaSconto** con parametro *30* stampando il valore restituito dal metodo. Istanziare un oggetto di tipo **CapoAbbigliamento** con marca *Levis* modello *jeans 570SC* costo *79,8* taglia *40*, quantità acquistata *30* quantità disponibile *30* e richiamare il metodo **venduto** con parametro pari a *2*. Infine richiamare il metodo **getDettagli**.

Esercizi sulle Eccezioni

Modificare l'esercizio E in modo da prevedere il lancio delle eccezioni

Esercizio1

Modificare il metodo preleva della classe ContoCorrente dell'esercizio E in modo che se **saldo** dopo la sottrazione è minore di zero il metodo lancia una eccezione di tipo Exception istanziando l'eccezione con la stringa "il conto è in rosso". Modificare il metodo preleva della sottoclasse ContoOnLine in modo che se $x > \text{maxPrelievo}$ viene lanciata una eccezione di tipo Exception istanziando l'eccezione con la stringa "il prelievo non è disponibile".

Modificare la classe banca in modo da catturare l'eccezione e stampare l'errore ogni volta che si richiama il metodo preleva

Esercizio2

Creare una sottoclasse BancaException della classe Exception aggiungendo l'attributo message di tipo stringa. Scrivere il costruttore con un parametro di tipo stringa, dopo aver richiamato il costruttore della superclasse passando il parametro impostare il valore dell'attributo message uguale al parametro. Ridefinire il metodo toString che restituisce un stringa, il metodo restituisce il valore di message.

Modificare le classi Banca ContoCorrente e ContoOnLine in modo che siano lanciate e catturate eccezioni di tipo BancaException

Esercizi sul passaggio dei parametri.

Esercizio3

Supponendo di avere la seguente classe

```
class Prova{
    int x=0,y=0;
    Prova(int a, int b){
        x=x+a;
        y=y+b;
    }
    int funz2(Prova b, int a){
        b= new Prova(1,1);
        b.x= this.x +a;
        return b.x;
    }
}
```

e che nel main della classe UsoProva ci siano le seguenti istruzioni, che valori stampa l'ultima istruzione, perchè?

```
Prova a= new Prova(4,7);
Prova b= new Prova(2,3);
int x=5;
x=b.funz2(a,x);
System.out.print(a.x+" "+a.y+" "+b.x+" "+b.y+" "+x);
```

Esercizio4

Supponendo di avere la seguente classe

```
class Prova{
    int x=0,y=0;
    Prova(int a,int b){
        x=a;
        y=b;
    }
}
```

```

void scambio(Prova b){
    b = this;
}
int modifica (int b){
    this.x= b;
    return b;
}
}

```

e che nel main della classe UsoProva ci siano le seguenti istruzioni, che valori stampa l'ultima istruzione, perchè?

```

Prova a= new Prova(2,3);
Prova b= new Prova(7,8);
b.scambio(a);
int x=2;
int y=b.modifica(x);
System.out.print(a.x+" "+a.y+" "+b.x+" "+b.y+" "+x+" "+y);

```

Esercizi sull'uso di classi (istanziare e richiamare metodi)

1. Sapendo che la classe Bottiglia ha il seguente costruttore
Bottiglia (double capacità) ed il metodo
double getLitri() che restituisce il numero di litri che la bottiglia contiene
Scrivere le istruzioni per istanziare un oggetto di tipo Bottiglia di 2,5 litri e poi richiamare il metodo getLitri per stampare il numero di litri che contiene la bottiglia.
2. Sapendo che la classe Letto ha il seguente costruttore
Letto(int lunghezza, int larghezza) ed il metodo
int getPiazza() che restituisce l'area del letto
Scrivere le istruzioni per istanziare un oggetto di tipo Letto di lunghezza 2 e larghezza 2 e poi richiamare il metodo getPiazza per stampare l'area del letto.
3. Sapendo che la classe Bacinella ha il seguente costruttore
Bacinella() ed il metodo void setDiametro() che modifica il diametro della bacinella
Scrivere le istruzioni per istanziare un oggetto di tipo Bacinella e poi richiamare il metodo setDiametro per dare un valore al diametro
4. Sapendo che la classe Televisore ha il seguente costruttore
Televisore(int pollici) ed il metodo String getMarca() che restituisce la marca del televisore
Scrivere le istruzioni per istanziare un oggetto di tipo Televisore e poi richiamare il metodo getMarca per stampare la marca.
5. Sapendo che la classe Penna ha il seguente costruttore
Penna (String marca, int modello) ed il metodo
void scrivi(String s) che stampa la stringa s
Scrivere le istruzioni per istanziare un oggetto di tipo Penna di marca *fila* e modello 32 e poi richiamare il metodo scrivi per scrivere la stringa *esonero*
6. Sapendo che la classe Figura ha il seguente costruttore
Figura(int altezza, int larghezza) ed il metodo
int area() che restituisce il valore dell'area della figura

Scrivere le istruzioni per istanziare un oggetto di tipo Figura di altezza 4 e larghezza 7 e poi richiamare il metodo area e stampare l'area della figura

7. Sapendo che la classe LettoreDvd ha il seguente costruttore
LettoreDvd (String marca) ed i metodi
void insertDvd(String nome_film) che inizializza l'attributo nome_dvd
void play() che visualizza il film.
Scrivere le istruzioni per istanziare un oggetto di tipo LettoreDvd di marca Sony e inserire il dvd del film "Matrix" usando il metodo insertDvd e poi usare il metodo play.
8. Sapendo che la classe Cappotto ha il seguente costruttore
Cappotto(String colore, int numero_bottoni) ed il metodo
String getColore() che restituisce la stringa indicante il colore del cappotto
Scrivere le istruzioni per istanziare un oggetto di tipo Cappotto di colore rosso e 3 bottoni e poi richiamare il metodo getColore per stampare il colore.

Soluzioni esercizi di preparazione alla Seconda Parte

Soluzione A1

Scrivere la classe **Pacco**

```
class Pacco {
```

avente i seguenti attributi **base, altezza e profondità**: tre numeri con la virgola

```
double base, altezza, profondità;
```

rispettivamente la base, l'altezza e la profondità del pacco,

e l'attributo **numero** di tipo numero intero che identifica il pacco.

```
int numero
```

Il costruttore ha come parametri tre numeri con la virgola (coerente con la scelta fatta per gli attributi base altezza e profondità) ed assegna il valore dei parametri agli attributi base, altezza e profondità.

```
Pacco( double x, double y, double z) {  
    base=x;  
    altezza=y;  
    profondità=z;  
}
```

Scrivere il metodo **setNumero** che ha come parametro una variabile di tipo numero intero, tale metodo assegna il valore del parametro all'attributo numero.

```
void setNumero( int x) {  
    numero=x;  
}
```

Scrivere il metodo **volume** che restituisce il valore del volume del pacco (prodotto dei tre attributi base, altezza e profondità).

```
double volume() {  
    return (altezza*larghezza*profondità);  
}
```

Si poteva anche scrivere

```
double volume() {  
    double v;  
    v= altezza*larghezza*profondità;  
    return v;  
}
```

Quindi ricomponendo tutti gli elementi la classe Pacco è la seguente:

```
class Pacco {  
    double base, altezza, profondità;  
    int numero;  
    Pacco(double x, double y, double z) {  
        base=x;  
        altezza=y;  
        profondità=z;  
    }  
    void setNumero(int x) {  
        numero=x;  
    }  
    double volume() {  
        return (base*altezza*profondità);  
    }  
}
```

```
}  
}
```

Soluzione A2

Scrivere la sottoclasse **PaccoAColori** della classe **Pacco**:

```
class PaccoAColori extends Pacco {  
    }  
avente in più l'attributo colore di tipo stringa.
```

```
    String colore;
```

Anche se non detto occorre scrivere il costruttore che non viene ereditato

```
    PaccoAColori( double x, double j, double r) {  
        super(x,j,r);  
    }
```

Il costruttore si limita in tal caso a richiamare il costruttore della superclasse.

il metodo **setColore** che ha un parametro di tipo stringa, il metodo assegna all'attributo colore il valore del parametro

```
    void setColore(String c) {  
        colore=c;  
    }
```

il metodo **stampa** che stampa a video i valori degli attributi.

```
    void stampa(){  
        System.out.println("altezza"+altezza+"larghezza"+larghezza+"profondità"+profondità+"colore"+  
colore);  
    }
```

Il metodo **stampaVolume** che stampa a video il valore del volume utilizzando il metodo volume

```
    void stampaVolume() {  
        double x=volume();  
        System.out.println("il volume"+x);  
    }
```

Quindi alla fine la classe è

```
class PaccoAColori extends Pacco {  
    String colore;  
  
    PaccoAColori( double x, double j, double r) {  
        super(x,j,r);  
    }  
  
    void setColore(String c) {  
        colore=c;  
    }
```

```

void stampa(){
    System.out.println("altezza"+altezza+"larghezza"+larghezza+"profondità"+profondità+"colore"+
colore);
}

void stampaVolume() {
    double x=volume();
    System.out.println("il volume"+x);
}

}

```

Soluzione A3

Scrivere la classe Prova che ha il metodo main vuol dire scrivere:

```

class Prova {
    public static void main(String args[]){

```

il testo dell'esercizio continua dicendo: "in cui occorre istanziare un oggetto di tipo Pacco con base 3,5 altezza 4,5 e profondità 6,0" ricordando che i numero con la virgola vanno indicati con il punto si ha:

```

Pacco p1= new Pacco(3.5,4.5,6.0);

```

"Richiamare il metodo setNumero con parametro pari a 3":

```

p1.setNumero(3);

```

"Stampare il valore del volume del pacco richiamando opportunamente il metodo volume". Per eseguire tale richiesta si può scrivere:

```

System.out.println("il volume del pacco e' "+p1.volume());

```

o alternativamente si poteva scrivere:

```

double v = p1.volume();

```

```

System.out.println("il volume del pacco e' "+v);

```

"Istanziare un altro oggetto di tipo PaccoAColori con base 2,4 altezza 5,3 profondità 5,5" quindi:

```

PaccoAColori p2 = new PaccoAColori(2.4,5.3,5.5);

```

"richiamare il metodo setColore con colore *rosso*" occorre ricordare che le stringhe vanno messe tra doppi apici:

```

p2.setColore("rosso");

```

"Infine richiamare il metodo stampaVolume per stampare il volume del pacco a colori."

```

p2.stampaVolume();

```

Quindi alla fine la classe è :

```

class Prova {
    public static void main(String args[]){
        Pacco p1= new Pacco(3.5,4.5,6.0);
        p1.setNumero(3);
        System.out.println("il volume del pacco e' "+p1.volume());
        PaccoAColori p2 = new PaccoAColori(2.4,5.3,5.5);
        p2.setColore("rosso");
        p2.stampaVolume();
    }
}

```

Soluzione B1

```
class Poligono{
    int numeroLati;
    double lato;
    Poligono(int x,double y) {
        numeroLati=x;
        lato=y;
    }
    double perimetro() {
        return lato*numeroLati;
    }
    void stampa() {
        System.out.println("numero lati= "+numeroLati+ "lato= "+lato);
    }
    void aumenta(double x){
        lato=lato+x;
    }
    double volume(int x){
        if (numeroLati==4)return lato*lato*x;
        else return 0;
    }
}
```

Soluzione B2

```
class Cubo{
    double peso;

    Cubo(double x) {
        super(4,x);
    }
    void setPeso(double x) {
        peso=x;
    }
    double volume(){
        return lato*lato*lato;
    }
}
```

Soluzione B3

```
class UsoPoligono{
    public static void main(String args[]){
        Poligono p=new Poligono(4,6.8);
        double x=p.perimetro();
        System.out.println("perimetro= "+x);
        p.stampa();
        p.aumenta(1.2);
        System.out.println("volume= "+p.volume(5));
    }
}
```

```
}
```

Soluzione C1

```
/* Scrivere la classe SalaCinema */
```

```
class SalaCinema{
```

```
/*che ha i seguenti attributi
```

```
film indicante il nome del film proiettato,
```

```
postiOcc il numero di posti occupati,
```

```
n_file indicante il numero delle file di posti a sedere contenuti nella sala e
```

```
maxFila indicante il numero di posti a sedere per fila il cui valore è 21.*/
```

```
String film;
```

```
int postiOcc,n_file,maxFila=21;
```

```
/*Il costruttore ha come parametri
```

```
una stringa per il nome del film e
```

```
un intero che indica il numero delle file
```

```
ed assegna opportunamente i valori dei parametri agli attributi film ed n_file
```

```
ed assegna 0 a postiOcc*/
```

```
SalaCinema(String s, int a){
```

```
film=s;
```

```
n_file =a;
```

```
postiOcc=0;
```

```
}
```

```
/* Scrivere il metodo getPosti che
```

```
restituisce il numero dei posti ancora liberi
```

```
(per calcolare il numero di posti liberi occorre calcolare il numero di posti di cui dispone la sala che  
è uguale al prodotto del numero di file per maxFila e poi a questo sottrarre il numero di posti  
occupati)*/
```

```
int getPosti(){
```

```
return (n_file*maxFila)-postiOcc;
```

```
}
```

```
/* Scrivere inoltre il metodo vendiBiglietti
```

```
che ha come parametro un numero intero indicante il numero di posti da occupare,
```

```
il metodo aggiunge all'attributo posti occupati il valore del parametro.*/
```

```
void vendiBiglietti(int p){
```

```
postiOcc = postiOcc +p;
```

```
}
```

```
}
```

Soluzione C2

```
/* Scrivere la sottoclasse SalaCinemaP della classe SalaCinema */

class SalaCinemaP extends SalaCinema{

/*che ha in più l'attributo fileOccupate
che indica il numero di file per le quali non ci sono posti disponibili */

    int fileOccupate;

/*Il costruttore non è ereditato quindi va definito all'interno della sottoclasse
*/

    SalaCinemaP(String s, int a){
        super(s,a);
        fileOccupate=0;
    }

/* Aggiungere il metodo calcolaFile
che calcola il numero di file per le quali non ci sono posti disponibili
assegnandolo all'attributo fileOccupate ( occorre dividere il numero di posti occupati per
maxFila)*/

    void calcolaFile(){
        fileOccupate =postiOcc/maxFila;
    }

/* Aggiungere il metodo stampaSituazione
che stampa tutti gli attributi della classe*/

    void stampaSituazione(){
        System.out.println("nella sala è proiettato il film "+film+" la sala ha "+n_file+" file di
        "+maxFila+" posti, attualmente sono occupate "+fileOccupate+" file per un totale di "+postiOcc+"
        posti occupati");
    }

}
```

Soluzione C3

```
class Esonero2a {
    public static void main (String args[]){
        SalaCinema s1= new SalaCinema("Van Helsing",15);
        s1.vendiBiglietti(5);
        System.out.println("posti disponibili in S1="+s1.getPosti());
        SalaCinemaP s2= new SalaCinemaP(" Scooby-Doo 2: mostri scatenati",18);
        s2.stampaSituazione();
        s2.vendiBiglietti(24);
    }
}
```

```

    System.out.println("posti disponibili in S2="+s2.getPosti());
    s2.calcolaFile();
    s2.stampaSituazione();
}
}

```

Soluzione D1

/* Scrivere la classe Teatro */

```
class Teatro{
```

/*che ha i seguenti attributi:

spettacolo indicante il nome dello spettacolo che avrà luogo al teatro,

postiPlatea il numero di posti in platea,

postiGalleria indicante il numero di posti in galleria */

```
String spettacolo;
```

```
int postiPlatea,postiGalleria;
```

/*Il costruttore ha come parametri:

una stringa per il nome dello spettacolo

un intero per il numero dei posti in platea

un intero per il numero dei posti galleria

ed assegna opportunamente i valori dei parametri agli attributi spettacolo, postiPlatea e postiGalleria*/

```
Teatro(String s, int a,int b){
```

```
    spettacolo=s;
```

```
    postiPlatea=a;
```

```
    postiGalleria=b;
```

```
}
```

/* Scrivere il metodo assegnaPostiPlatea che ha

un parametro intero indicante il numero di posti da occupare in platea,

il metodo aggiorna l'attributo postiPlatea sottraendogli il valore del parametro */

```
void assegnaPostiPlatea(int x){
```

```
    postiPlatea=postiPlatea-x;
```

```
}
```

/* Scrivere il metodo assegnaPostiGalleria che ha

un parametro intero indicante il numero di posti da occupare in galleria,

il metodo aggiorna l'attributo postiGalleria sottraendogli il valore del parametro.*/

```
void assegnaPostiGalleria(int p){
```

```
    postiGalleria = postiGalleria-p;
```

```
}
```

```
}
```

Soluzione D2

/ Scrivere la sottoclasse VenditeT della classe Teatro*/*

```
class VenditeT extends Teatro{
```

```
/*Non eredita il costruttore  
quindi occorre scrivere il costruttore della sottoclasse  
richiamando il costruttore della superclasse  
*/
```

```
VenditeT(String s, int a,int b){  
    super(s,a,b);  
}
```

```
/* Aggiungere il metodo vendiBiglietti che ha come parametro  
un numero intero x indicante il numero di posti da occupare,  
ed un boolean per capire l'operazione da eseguire,  
il metodo se il parametro booleano è true utilizza il metodo assegnaPostiGalleria passandogli x,  
altrimenti se il parametro booleano è false utilizza il metodo assegnaPostiPlatea passandogli x */
```

```
void vendiBiglietti(int x,boolean b){  
    if(b==true) assegnaPostiGalleria(x);  
    else assegnaPostiPlatea(x);  
}
```

```
/* Scrivere inoltre il metodo riepilogo che ha  
come parametro un boolean e  
se il valore del parametro è true restituisce il valore dell'attributo postiGalleria  
altrimenti restituisce il valore dell'attributo postiPlatea */
```

```
int riepilogo(boolean x){  
    if (x==true) return postiGalleria;  
    else return postiPlatea;  
}
```

```
}
```

Soluzione D3

```
class Esonero2b {  
    public static void main (String args[]){  
        Teatro t= new Teatro("Aggiungi un posto a tavola",50,20);  
        System.out.println("Al teatro c'e' lo spettacolo "+t.spettacolo+" con posti disponibili:  
"+t.postiPlatea+" in platea e "+t.postiGalleria+" in galleria");  
        t.assegnaPostiPlatea(15);
```

```

t.assegnaPostiGalleria(5);
VenditeT v= new VenditeT ("Rugantino",40,15);
v.vendiBiglietti(15,false);
System.out.println("sono rimasti i seguenti posti in platea "+v.riepilogo(false));

}
}

```

Soluzione E1

```

class ContoCorrente{

    String titolare;
    int nMovimenti;
    final int maxMovimenti=50;
    double saldo;

    ContoCorrente(String titolare, double saldo){
        this.titolare=titolare;
        this.saldo=saldo;
        nMovimenti=0;
    }

    void preleva(double x){
        if (nMovimenti<maxMovimenti) saldo=saldo-x;
        else saldo=saldo-x-0.50;
        nMovimenti++;
    }

    double restituisciSaldo(){
        return saldo;
    }

}

```

Soluzione E2

```

class ContoOnLine extends ContoCorrente{

    double maxPrelievo;

    ContoOnLine(String titolare, double saldo,double maxP){
        super(titolare,saldo);
        maxPrelievo=maxP;
    }

    void stampaSaldo(){
        System.out.println("il conto e' del titolare="+titolare+" il cui saldo e' "+saldo);
    }

    void preleva(double x){

```

```

    if (x<maxPrelievo) super.preleva(x);
}

}

```

Soluzione E3

```

class Banca {

    public static void main(String args[]){

        ContoCorrente conto1=new ContoCorrente("Grossi Mario",20000.0);
        conto1.preleva(1750.5);
        System.out.println(conto1.restituisceSaldo());
        ContoOnLine conto2 = new ContoOnLine("Rossi Luigi", 50350.0, 1500);
        conto2.stampaSaldo();
        conto2.preleva(2000);
    }
}

```

Esercizio F1

```

class Motorino {
    String colore;
    float velocità;
    String tipo;
    boolean antifurto =false;
    Motorino(String c, String t, float x){
        colore=c;
        velocità=x;
        tipo=t;
    }
    float getVelocità(){
        return velocità;
    }
    void accelera(float x){
        if(antifurto == false) velocità = velocità+x;
    }
    void inserisciAntifurto(){
        antifurto=true;
    }
}

```

EsercizioF2

```

class MotorinoImmatricolato extends Motorino {
    float maxVelocità;
    String targa;

    MotorinoImmatricolato(String c, String t, float x, float y, String targa){
        super(c,t,x);
    }
}

```

```

    maxVelocità = y;
    this.targa = targa;
}

void getMax(){
    System.out.println(maxVelocità);
}

void accelera(float x){
    float s=velocità+x;
    if(antifurto == false)
        if(s<=maxVelocità) velocità=s;
        else velocità=maxVelocità;
    }
}

```

EsercizioF3

```

class UsoMotorino {

    public static void main( String args[]){
        Motorino m;
        m= new Motorino("grigiometallizzato","Piagio Liberty",40.5f);
        float x = m.getVelocità();

        MotorinoImmatricolato mI;
        mI= new MotorinoImmatricolato("rosso","Aprilia Scarabeo",30.5f, 60.0f,"CV1234");
        mI.getMax();
        mI.accelera(30.7f);
        m.accelera(30.7f);
        System.out.println(mI.getVelocità());
        System.out.println(m.getVelocità());

    }
}

```

EsercizioG1

```

class Dipendente{
    String matricola;
    double stipendio;
    double straordinario;
    Dipendente(String m, double x, double y){
        matricola=m;
        stipendio=x;
        straordinario=y;
    }
    double getStipendio(){
        return stipendio;
    }
    double paga(int x){

```

```

    return stipendio+x*straordinario;
}
void stampa(){
    System.out.println("matricola =" +matricola+" stipendio ="+" straordinario= "+straordinario);
}
}

```

EsercizioG2

```

class DipendenteA extends Dipendente{
    int malattia;

    DipendenteA(String m, double x, double y){
        super(m,x,y);
    }
    void prendiMalattia(int x){
        malattia+=x;
    }

    double paga(int x){
        double p= super.paga(x);
        if (malattia==0) return p;
        else return p-malattia*15.0;
    }

    void stampaMalattia(){
        System.out.println("malattia =" +malattia);
    }
}

```

EsercizioG3

```

class CalcolaStipendi
{
    public static void main( String args[]){
        Dipendente a;
        a= new Dipendente("00309",1000.00,7.50);
        System.out.println(a.paga(10));
        System.out.println(a.getStipendio());

        DipendenteA a1;
        a1= new DipendenteA("00201",1500.00,8.50);
        a1.prendiMalattia(5);
        System.out.println(a1.paga(3));
        a1.stampaMalattia();

    }
}

```

EsercizioH1

```
class TipoCapoAbbigliamento{
    String marca, modello;
    double costo;

    TipoCapoAbbigliamento(String marca, String modello, double costo){
        this.marca=marca;
        this.modello=modello;
        this.costo=costo;
    }
    void getCosto(){
        System.out.println(costo);
    }
    double applicaSconto(double x){
        return costo-(costo*x/100);
    }
    void modificaCosto(double x){
        costo=x;
    }
}
```

EsercizioH2

```
class CapoAbbigliamento extends TipoCapoAbbigliamento{
    int taglia, quantitàAcquistata, quantitàDisponibile;

    CapoAbbigliamento(String marca, String modello, double costo, int t,int a,int d){
        super(marca,modello,costo);
        taglia=t;
        quantitàAcquistata=a;
        quantitàDisponibile=d;
    }

    void venduto(int x){
        if(quantitàDisponibile>quantitàDisponibile-x) quantitàDisponibile=quantitàDisponibile-x;
    }
    void getDettagli(){
        System.out.println("marca="+marca+" modello="+modello+" costo="+costo+" taglia="+taglia+"
quantitàAcquistata="+quantitàAcquistata+" quantitàDisponibile="+quantitàDisponibile);
    }
}
```

EsercizioH3

```
class Negozio {

    public static void main( String args[]){
        TipoCapoAbbigliamento x;
        x= new TipoCapoAbbigliamento("Colmar","Giacca E45",350.5);
        System.out.println(x.applicaSconto(30));

        CapoAbbigliamento y;
        y= new CapoAbbigliamento("Levis","jeans 570SC",79.8,40,30,30);
        y.venduto(2);
    }
}
```

```
y.getDettagli();  
}  
}
```

Soluzioni esercizi sulle Eccezioni

Esercizio1

```
class ContoCorrente{  
  
    String titolare;  
    int nMovimenti;  
    final int maxMovimenti=50;  
    double saldo;  
  
    ContoCorrente(String titolare, double saldo){  
        this.titolare=titolare;  
        this.saldo=saldo;  
        nMovimenti=0;  
  
    }  
  
    void preleva(double x)throws Exception{  
        if (nMovimenti<maxMovimenti) saldo=saldo-x;  
        else saldo=saldo-x-0.50;  
        nMovimenti++;  
        if (saldo<0) throw new Exception("il conto è in rosso");  
    }  
  
    double restituisciSaldo(){  
        return saldo;  
    }  
}  
  
class ContoOnLine extends ContoCorrente{  
  
    double maxPrelievo;  
  
    ContoOnLine(String titolare, double saldo,double maxP){  
        super(titolare,saldo);  
        maxPrelievo=maxP;  
    }  
  
    void stampaSaldo(){  
        System.out.println("il conto e' del titolare="+titolare+" il cui saldo e' "+saldo);  
    }  
  
    void preleva(double x)throws Exception{  
        if (x<maxPrelievo) super.preleva(x);  
        else throw new Exception("il prelievo non e' disponibile");  
    }  
}
```

```
}  
}
```

```
class Banca{  
  
    public static void main(String args[]){  
  
        ContoCorrente conto1=new ContoCorrente("Grossi Mario",20000.0);  
  
        try{  
            conto1.preleva(1750.5);  
        }catch(Exception e){  
            System.out.println(e);  
        }  
  
        System.out.println(conto1.restituisceSaldo());  
        ContoOnLine conto2 = new ContoOnLine("Rossi Luigi", 50350.0, 1500);  
        conto2.stampaSaldo();  
        try{  
            conto2.preleva(2000);  
        }catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

Compilando ed eseguendo il programma si ha:

```
C:\esercizi>javac Banca.java
```

```
C:\esercizi>java Banca
```

```
18249.5
```

```
il conto e' del titolare=Rossi Luigi il cui saldo e' 50350.0
```

```
java.lang.Exception: il prelievo non è disponibile
```

ESERCIZIO2

```
class BancaException extends Exception{
```

```
    String message;
```

```
    BancaException(String s){  
        super(s);  
        message=s;  
    }
```

```
    String toString(){  
        return message;  
    }  
}
```

```
class ContoCorrente{
```

```

String titolare;
int nMovimenti;
final int maxMovimenti=50;
double saldo;

ContoCorrente(String titolare, double saldo){
    this.titolare=titolare;
    this.saldo=saldo;
    nMovimenti=0;
}

void preleva(double x)throws BancaException{
    if (nMovimenti<maxMovimenti) saldo=saldo-x;
    else saldo=saldo-x-0.50;
    if (saldo<0) throw new BancaException("il conto è in rosso");
}

double restituisciSaldo(){
    return saldo;
}
}

class ContoOnLine extends ContoCorrente{

    double maxPrelievo;

    ContoOnLine(String titolare, double saldo,double maxP){
        super(titolare,saldo);
        maxPrelievo=maxP;
    }

    void stampaSaldo(){
        System.out.println("il conto e' del titolare="+titolare+" il cui saldo e' "+saldo);
    }

    void preleva(double x)throws BancaException{
        if (x<maxPrelievo) super.preleva(x);
        else throw new BancaException("il prelievo non e' disponibile");
    }
}

class Banca{

    public static void main(String args[]){

        ContoCorrente conto1=new ContoCorrente("Grossi Mario",20000.0);

        try{
            conto1.preleva(1750.5);
        }catch(BancaException e){

```

```

        System.out.println(e);
    }

    System.out.println(conto1.restituisceSaldo());
    ContoOnLine conto2 = new ContoOnLine("Rossi Luigi", 50350.0, 1500);
    conto2.stampaSaldo();
    try {
        conto2.preleva(2000);
    } catch (BancaException e) {
        System.out.println(e);
    }
}
}
}

```

Compilando ed eseguendo il programma si ha:

C:\esercizi>javac Banca.java

C:\esercizi>java Banca

18249.5

il conto e' del titolare=Rossi Luigi il cui saldo e' 50350.0

il prelievo non e' disponibile

Soluzioni esercizi sul passaggio dei parametri.

Esercizio3

Le istruzioni del main sono:

Prova a= new Prova(4,7); ← istanzia un oggetto **a** di tipo Prova avente gli attributi x=4 e y=7

Prova b= new Prova(2,3); ← istanzia un oggetto **b** di tipo Prova avente gli attributi x=2 e y=3

int x=5; assegna il valore 5 alla variabile intera **x**

x=b.funz2(a,x); ← assegna alla variabile intera **x** il valore restituito dal metodo funz2 dell'oggetto **b** di tipo Prova passando come paramentri attuali

- la variabile di riferimento **a** ad un oggetto di tipo Prova (avente gli attributi x=4 e y=7)
- la variabile **x** avente valore 5

il metodo funz2 è così definito:

```

int funz2(Prova b, int a){
    b= new Prova(1,1);
    b.x= this.x +a;
    return b.x;
}

```

il passaggio dei paramentri avviene per valore quindi il valore contenuto nella variabile indicata nel parametro attuale è copiato nella variabile indicata nel parametro formale

applicando questa regola nell'esercizio in esame:

- la variabile **b** (parametro formale) del metodo funz2 assume lo stesso valore della variabile **a** (parametro attuale) del metodo main quindi **b** contiene il riferimento all'oggetto Prova avente gli attributi x=4 e y=7,

- la variabile a (parametro formale) del metodo funz2 assume lo stesso valore della variabile x (parametro attuale) del metodo main quindi a è uguale a 5.

Le istruzioni del metodo sono

b= new Prova(1,1); ← la variabile b del metodo funz2 assume un nuovo valore e si riferisce ad un nuovo oggetto di tipo Prova avente attributi x=1 e y=1

b.x= this.x +a; ← l'attributo x dell'oggetto riferito dalla variabile b assume un nuovo valore dato dalla somma del valore della variabile a=5 ed il valore dell'attributo x dell'oggetto a cui è stato richiesto di eseguire il metodo (indicato da this) tale oggetto è visibile nel main in cui è presente l'istruzione **b.funz2(a,x)**; quindi l'oggetto a cui è richiesto di eseguire il metodo è l'oggetto riferito dalla variabile b del main che ha attributi x=2 e y=8, quindi b.x=2+5;

return b.x; ← il metodo restituisce il valore dell'attributo x dell'oggetto riferito da b che per quanto detto in precedenza è 7

la variabile x del main assume il valore restituito dal metodo quindi è uguale a 7, al termine dell'esecuzione del metodo gli oggetti riferiti dalle variabili a e b del main non sono stati modificati

System.out.print(a.x+" "+a.y+" "+b.x+" "+b.y+" "+x); ← la seguente istruzione stampa a video **4 7 2 3 7**

ESERCIZIO4

Supponendo di avere la seguente classe

```
class Prova{
    int x=0,y=0;
    Prova(int a,int b){
        x=a;
        y=b;
    }
    void scambio(Prova b){
        b = this;
    }
    int modifica (int b){
        this.x= b;
        return b;
    }
}
```

e che nel main della classe UsoProva ci siano le seguenti istruzioni, che valori stampa l'ultima istruzione, perchè?

```
Prova a= new Prova(2,3);
Prova b= new Prova(7,8);
b.scambio(a);
int x=2;
int y=b.modifica(x);
System.out.print(a.x+" "+a.y+" "+b.x+" "+b.y+" "+x+" "+y);
```

Le istruzioni del main sono:

Prova a= new Prova(2,3); ← istanzia un oggetto a di tipo Prova avente gli attributi x=2 e y=3

Prova b= new Prova(7,8); ← istanzia un oggetto b di tipo Prova avente gli attributi x=7 e y=8

b.scambio(a); ← richiede all'oggetto riferito dalla variabile b di eseguire il metodo scambio passando come parametro attuale la variabile di riferimento a.

il metodo scambio è così definito:

```
void scambio(Prova b){  
    b = this;  
}
```

il passaggio dei parametri avviene per valore quindi il valore contenuto nella variabile indicata nel parametro attuale è copiato nella variabile indicata nel parametro formale

applicando questa regola nell'esercizio in esame:

- la variabile b (parametro formale) del metodo scambio assume lo stesso valore della variabile a (parametro attuale) del metodo main quindi b contiene il riferimento all'oggetto Prova avente gli attributi x=2 e y=3,

La sola istruzione del metodo è

b = this; ← la variabile b del metodo scambio assume un nuovo valore e si riferisce all'oggetto a cui è stato chiesto di eseguire il metodo (indicato con this) quindi è l'oggetto riferito dalla variabile b del main che è un oggetto di tipo Prova avente attributi x=7 e y=8

al termine dell'esecuzione del metodo gli oggetti a e b del main non sono stati modificati

int x=2; assegna il valore 2 alla variabile intera x

int y=b.modifica(x); ← assegna alla variabile intera y il valore restituito dal metodo modifica dell'oggetto b di tipo Prova passando come parametro attuale la variabile intera x avente valore 2

il metodo modifica è così definito:

```
int modifica (int b){  
    this.x= b;  
    return b;  
}
```

il passaggio dei parametri avviene per valore quindi il valore contenuto nella variabile indicata nel parametro attuale è copiato nella variabile indicata nel parametro formale

applicando questa regola nell'esercizio in esame:

- la variabile b (parametro formale) del metodo modifica assume lo stesso valore della variabile x (parametro attuale) del metodo main quindi b è uguale a 2.

Le istruzioni del metodo sono

this.x= b; ← l'attributo x dell'oggetto a cui è stato richiesto di eseguire il metodo (indicato con this) assume un nuovo valore uguale a 2 (tale oggetto è visibile nel main in cui è presente l'istruzione **b.modifica(x)**; quindi l'oggetto a cui è richiesto di eseguire il metodo è l'oggetto riferito dalla variabile b del main che ha attributi x=7 e y=8) l'oggetto riferito dalla variabile b del main è modificato ed assume un nuovo valore nell'attributo x (x=2)

return b; ← il metodo restituisce il valore della variabile b che è uguale a 2

al termine dell'esecuzione del metodo l'oggetto riferito dalla variabile b del main è stato modificato ed ha attributi x=2 ed y=8 e la variabile y del main assume valore 2

System.out.print(a.x+" " +a.y+" " +b.x+" "+b.y+" "+x+" "+y); ← la seguente istruzione stampa a video **2 3 2 8 2 2**

Soluzioni esercizi sull'uso di classi (istanziare e richiamare metodi)

1.
Bottiglia b = new Bottiglia (2.5);
double x= b.getLitri();
System.out.println(x);
2.
Letto unLetto=new Letto(2, 2);
System.out.println("il letto ha un'area di "+unLetto.getPiazza()+"mq");
3.
Bacinella c = new Bacinella();
c.setDiametro();
4.
Televisore tele = new Televisore(24);
String pippo= tele.getMarca();
System.out.println("la marca del televisore e' "+pippo);

altrimenti
Televisore tele = new Televisore(24);
String pippo;
pippo= tele.getMarca();
System.out.println("la marca del televisore e' "+pippo);
5. Penna pen=new Penna ("fila",32);
pen.scrivi("esonero");
6. Figura fig = new Figura (4,7);
System.out.println("la figura ha un'area di "+fig.area());
7. LettoreDvd myDvD = new LettoreDvd ("Sony");
myDvD.insertDvd ("Matrix");
myDvD.play();
8. Cappotto cappottino = new Cappotto ("rosso",3);
String coloreCappotto= cappottino.getColore ();
System.out.println("il colore del cappotto e' "+ coloreCappotto);

altrimenti:

Cappotto cappottino;
cappottino = new Cappotto ("rosso",3);
System.out.println("il colore del cappotto e' "+ cappottino.getColore ());