

Testo Esonero A

Domanda 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

Domanda2

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.

Domanda 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.

Testo Esonero B

Domanda1

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.

Domanda2

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.

Domanda3

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

Testo Esonero C

Domanda1

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.

Domanda2

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.

Domanda3

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.

ESERCIZI AGGIUNTIVI

sulle Eccezioni

Esercizio1

Modificare il metodo preleva della classe ContoCorrente 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

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);

```

Soluzioni Esonero

ESONERO A

Domanda 1

/* 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;
```

```
}
```

```
}
```

Domanda 2

/* 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");  
    }  
  
}
```

Domanda3

```
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();
}
}
```

ESONERO B

Domanda 1

/* 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;
```

```
}
```

```
}
```

Domanda 2

/* 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;  
}
```

```
}
```

Domanda3

```
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));
    }
}
```

Esonero C

```
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;
    }

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

}
```

```

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);
    }
}

```

Soluzioni Esercizi Aggiuntivi

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;
        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;

    MyException(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

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 parametri 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 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 **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 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 scambio assume lo stesso valore della variabile a (parametro attuale) del metodo main quindi b contiene il riferimento all'oggetto Prova avente 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 paramentro 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**