

Soluzioni esercizi di preparazione al Secondo Esonero

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+" altezza "+altezza+" 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+" altezza "+altezza+" 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 extends Poligono{
    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);
    }
}
```