

ALTRI ESERCIZI SECONDA PARTE

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);
```

Soluzioni

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.accellera(30.7f);
        m.accellera(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();
    }
}

```

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.restituisciSaldo());
        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 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'istruzine **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