UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Università di degli Studi di Salerno
Dipartimento di Informatica

Programmazione ad Oggetti

a.a. 2023-2024

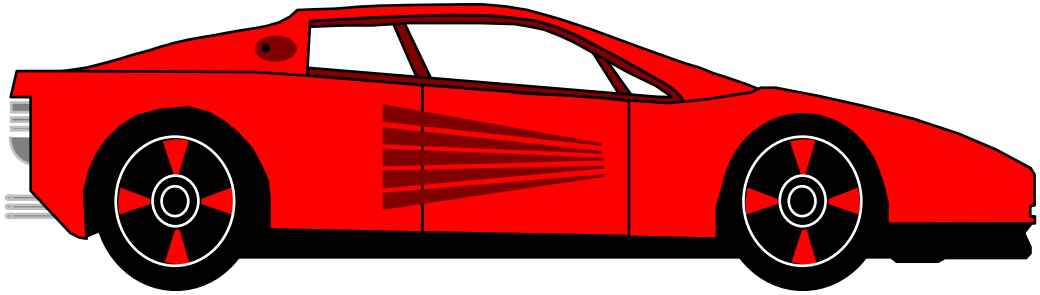
Classi Composte

Docente: Massimo Ficco

E-mail: *mficco@unisa.it*

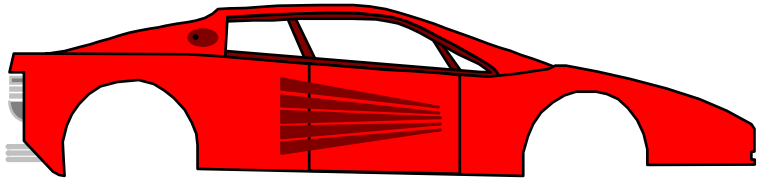
Classi Composte

Una classe essere costituita da altri oggetti

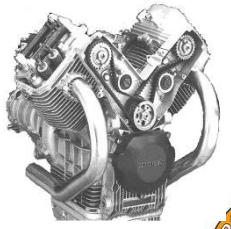


automobile

ruota



carrozzeria



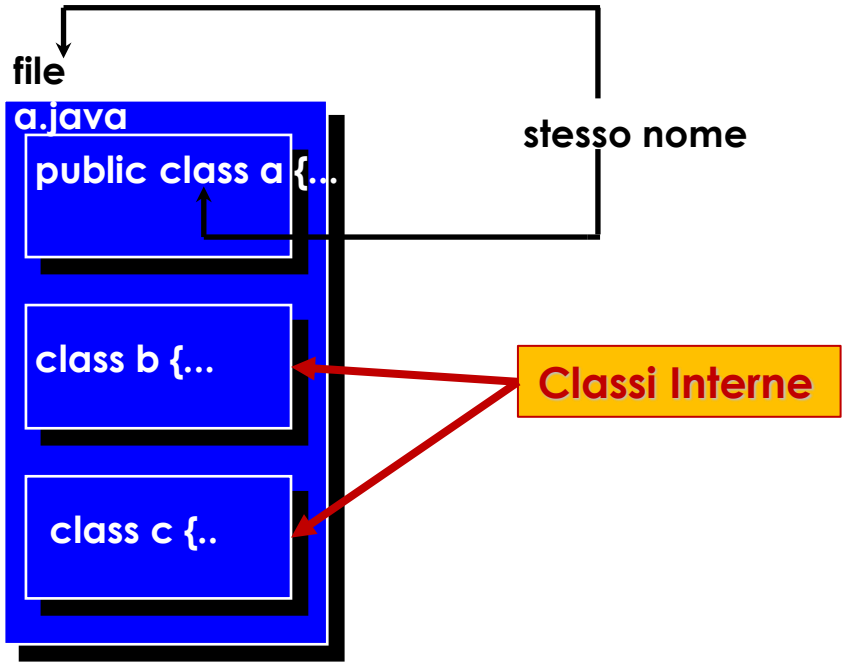
motore

Struttura di una applicazione V:

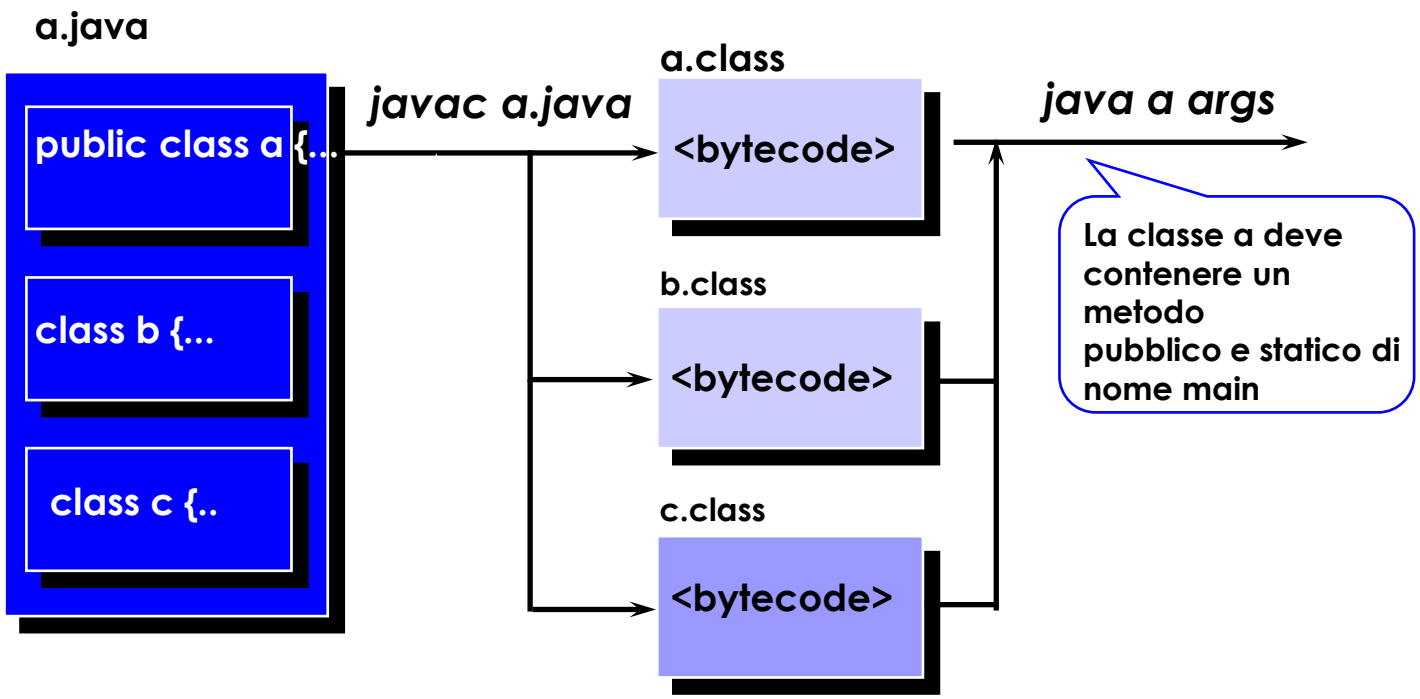
Il sorgente di **un'applicazione** consiste di uno o più file ("unità di compilazione")

Ogni file contiene una o più dichiarazioni di classi (o di interfacce), di cui al più una dichiarata ***public***

Il nome del file deve essere uguale a quello della sua classe ***public*** , con estensione ***.java***:



COMPILAZIONE ED ESECUZIONE:



UNITA' DI COMPILAZIONE V:

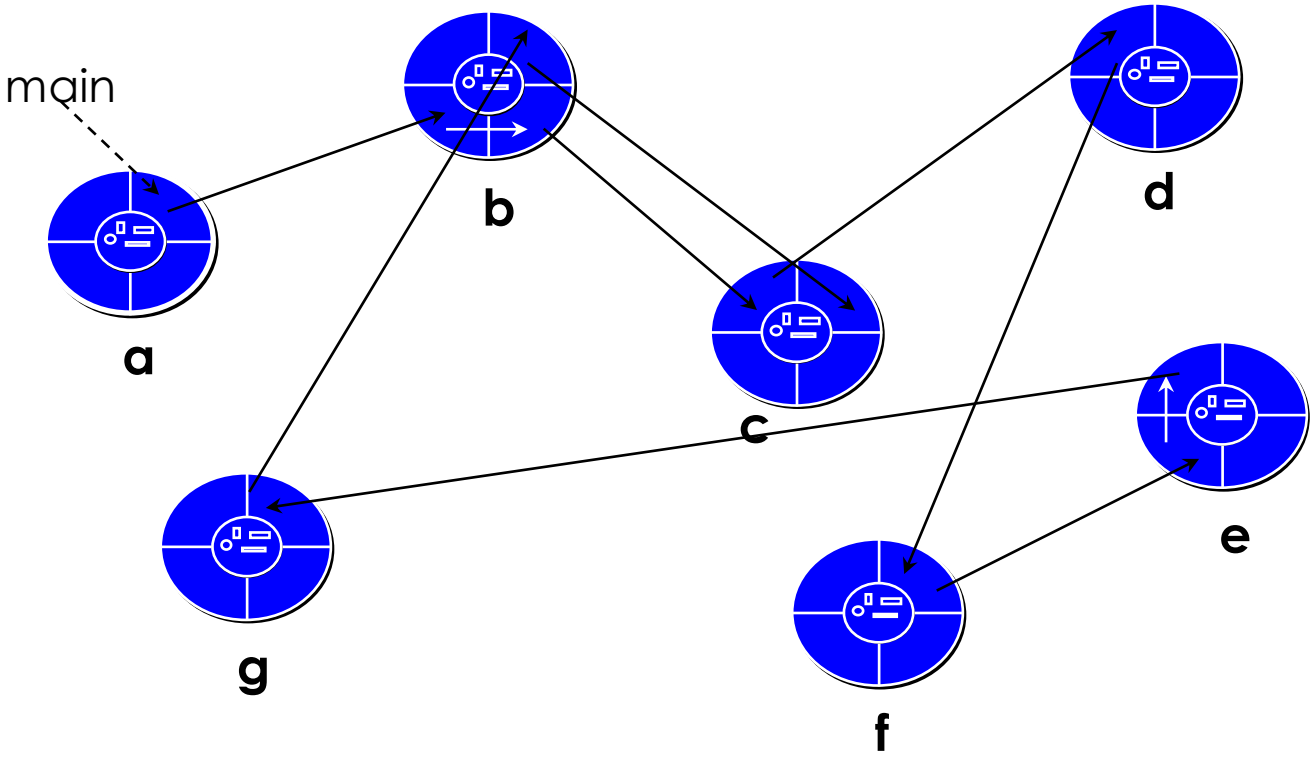
a.java

```
public class a {
    public static void main (String args []) {
            ---
}

class OtherClass {    /* opzionale */
            …
}
```

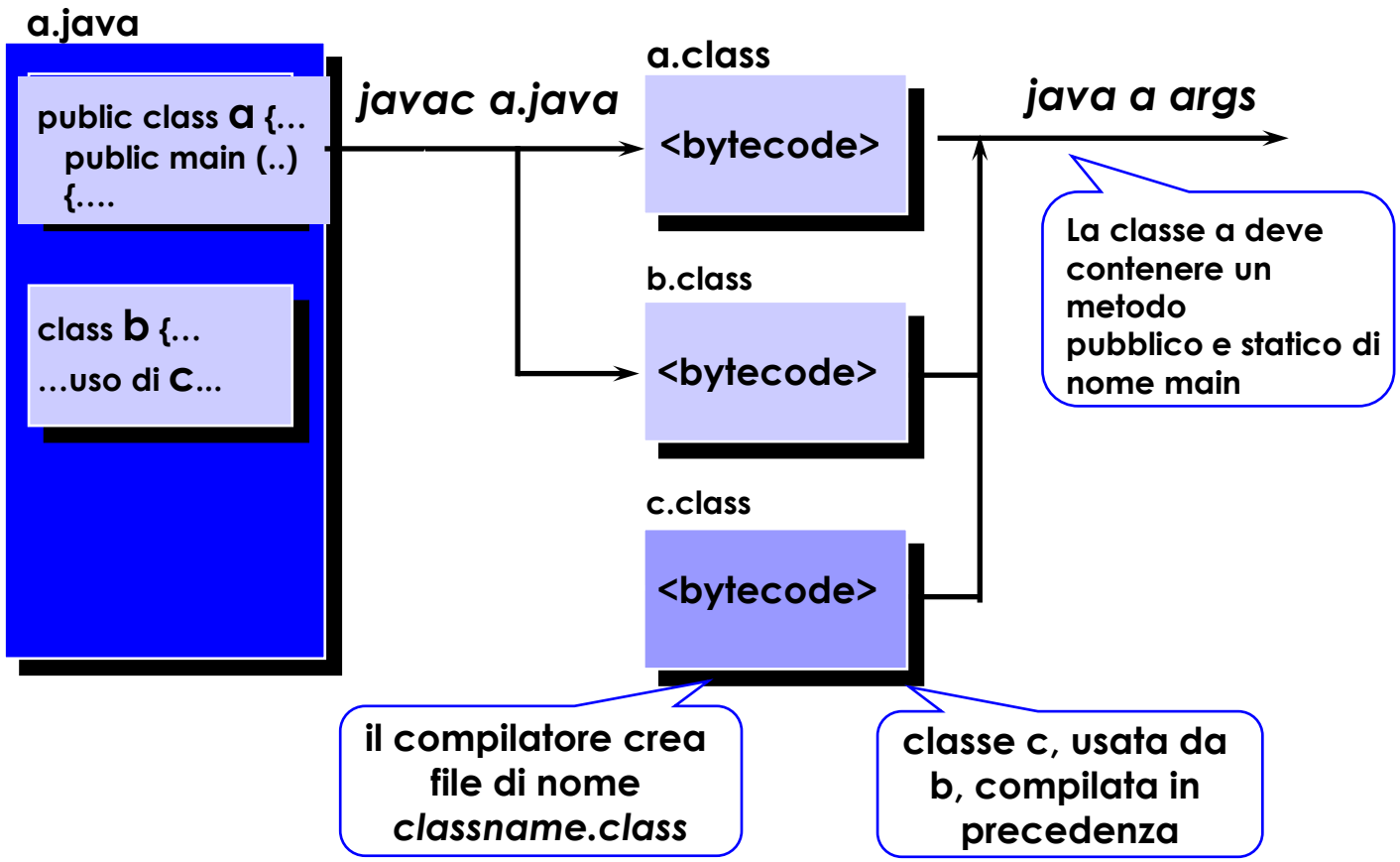
Esempio: Applicazione con 2 classi

```
public class Test
{
   public static void main(String args[]) {
         Auto myCar;
         myCar=new Auto();
         myCar.setSpeed(10);
         System.out.println("speed="+myCar.getSpeed());
         System.out.println("Cilindri"+myCar.getCilindri());
   }
}

 class Auto{
            int  cilindri=4;
            int  speed=0;
            public int getSpeed(){return speed;};
            public void setSpeed(int s){speed= s;};
            public int getCilindri(){return cilindri;};

 }
```

*STRUTTURA DI UN'APPLICAZIONE:



COMPILAZIONE ED ESECUZIONE:



DICHIARAZIONE DI OGGETTI V:

File Shirt.java

```
public class Shirt {

  public int shirtID = 0;
  public String description = "description required-";
  public char colorCode = 'U';
  public double price = 0.0;

  public int quantityInStock = 0;

    public void displayShirtInformation() {

    System.out.println("Shirt ID: " + shirtID);
    System.out.println("Shirt description:" + description);
    System.out.println("Color Code: " + colorCode);
    System.out.println("Shirt price: " + price);
    System.out.println("Quantity in stock: " + quantityInStock);
  }
}
```

DICHIARAZIONE DI OGGETTI V:

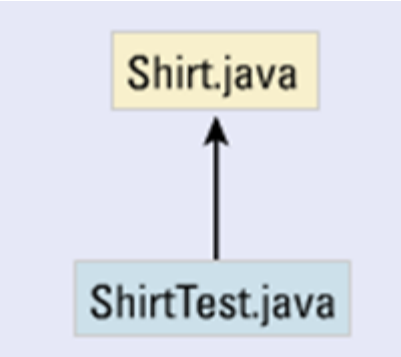
File ShirtTest.java

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```

Allocazione in memoria dei file .class

COMPILAZIONE

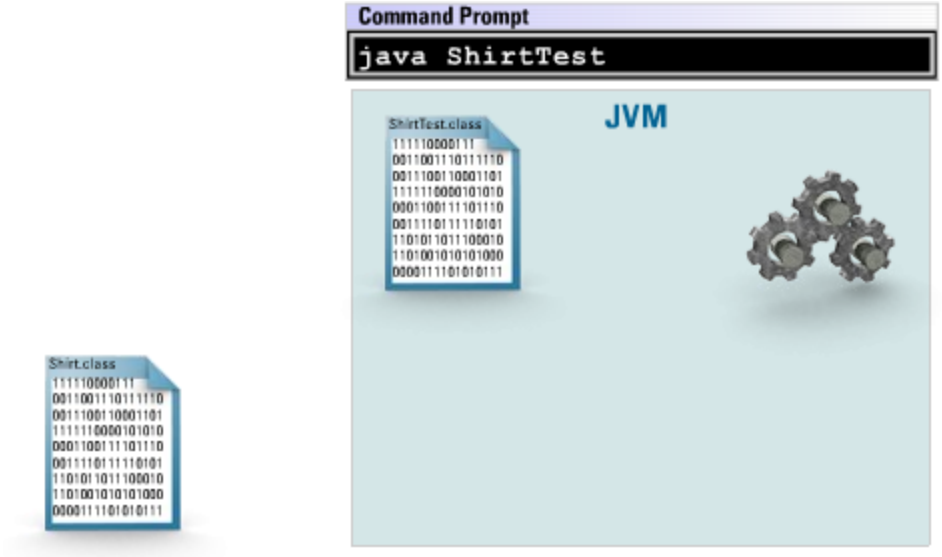
Dipendenza tra Classi



Le classi devono essere compilate in base all'ordine di dipendenza.
La classe che dipende dalle altre va compilate per ultima.

Esecuzione

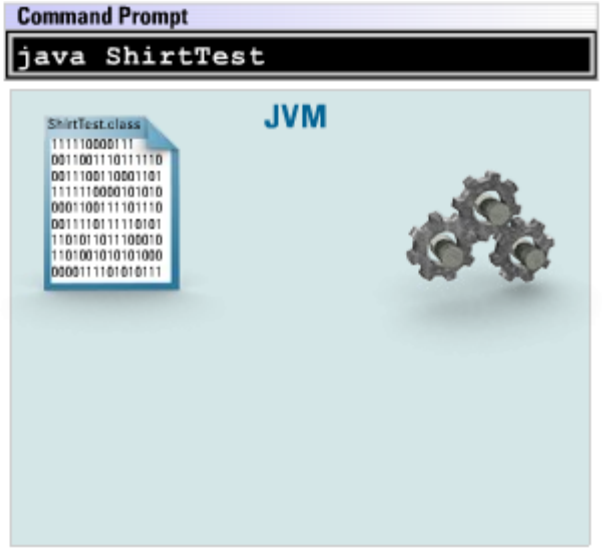
1) Esecuzione dell'applicazione ShirtTest
2) Viene caricata la classe ShirtTest in memoria



Esecuzione

3) Esecuzione della classe ShirtTest

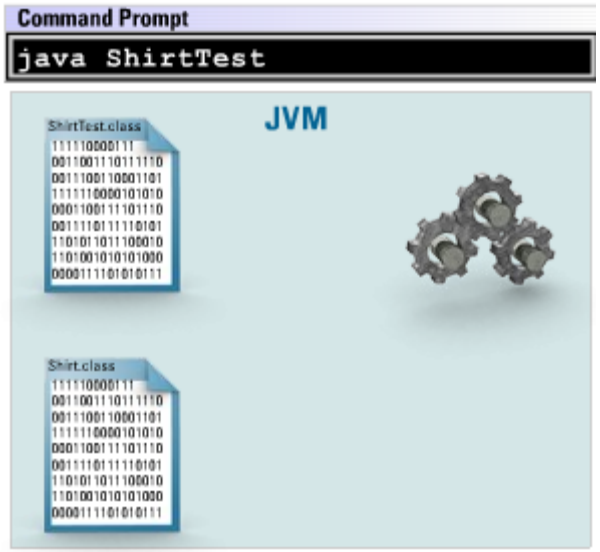
```
public class ShirtTest {

    public static void main (String args[]) {

        Shirt myShirt;
        myShirt = new Shirt();

        myShirt.colorCode = 'G';


    }
}
```



Esecuzione

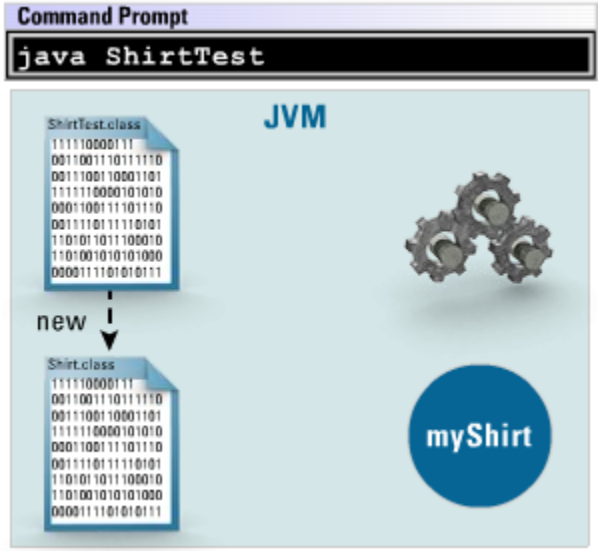
4) Viene caricata la classe ShirtTest in memoria

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



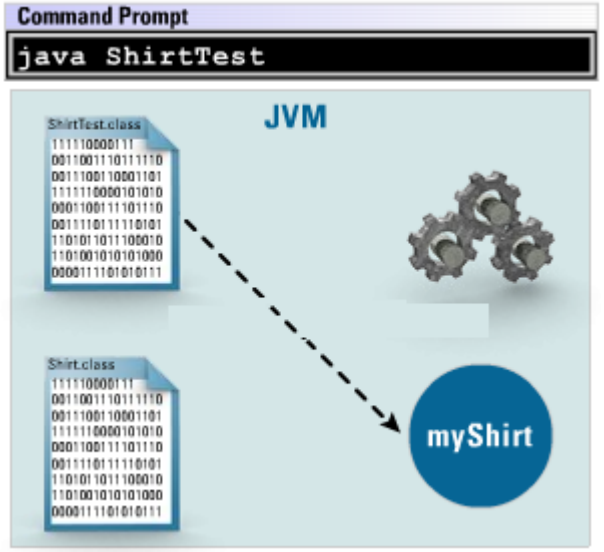
5) Creazione di un oggetto Shirt

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



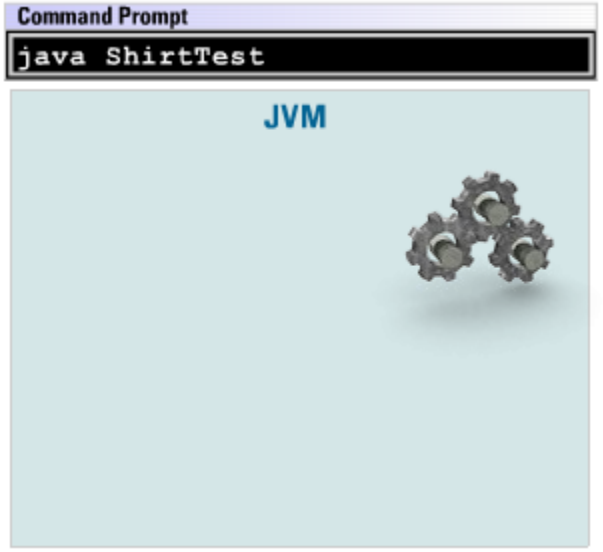
6) Invocare oggetto Shirt per cambiarne il valore dell'attributo colorCode

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



7) Fine dell'applicazione

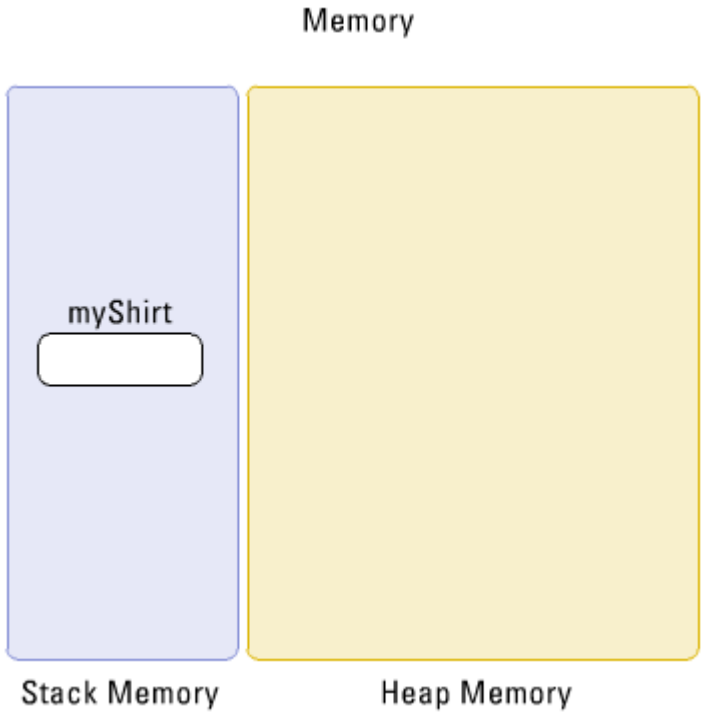
```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



Stack e Heap Memory

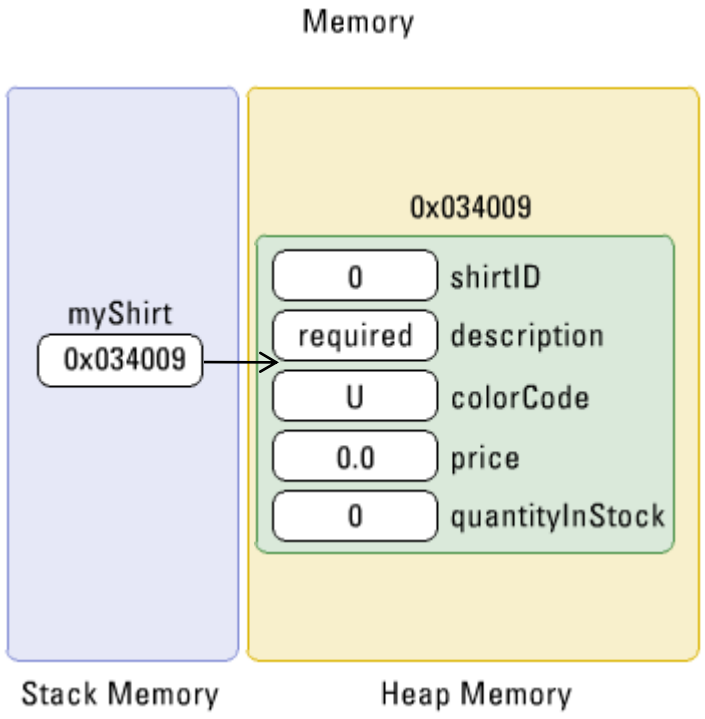
ALLOCAZIONE IN MEMORIA V:

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



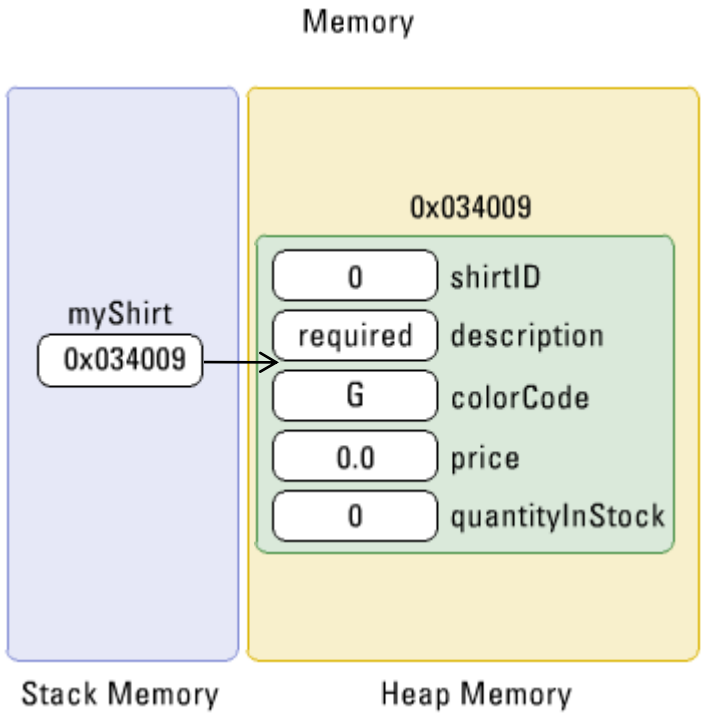
ALLOCAZIONE IN MEMORIA V:

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



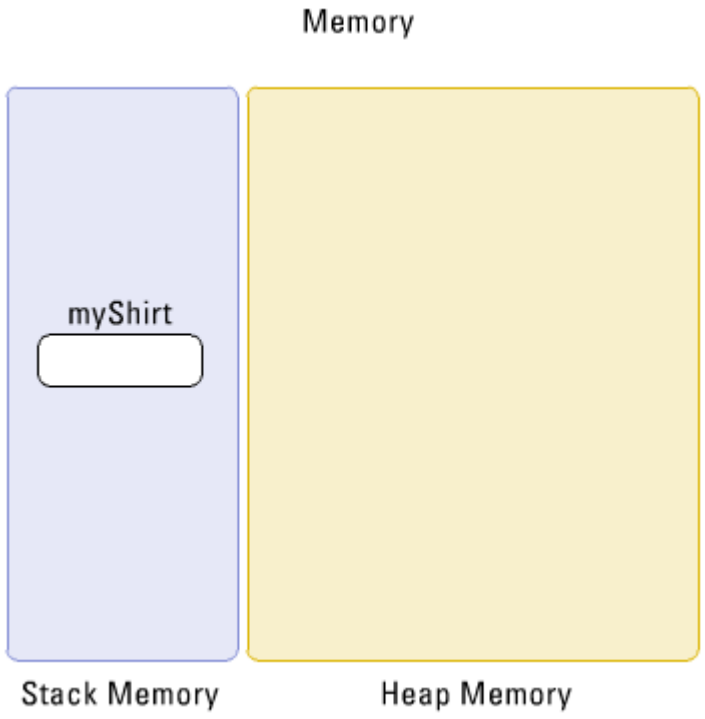
ALLOCAZIONE IN MEMORIA V:

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



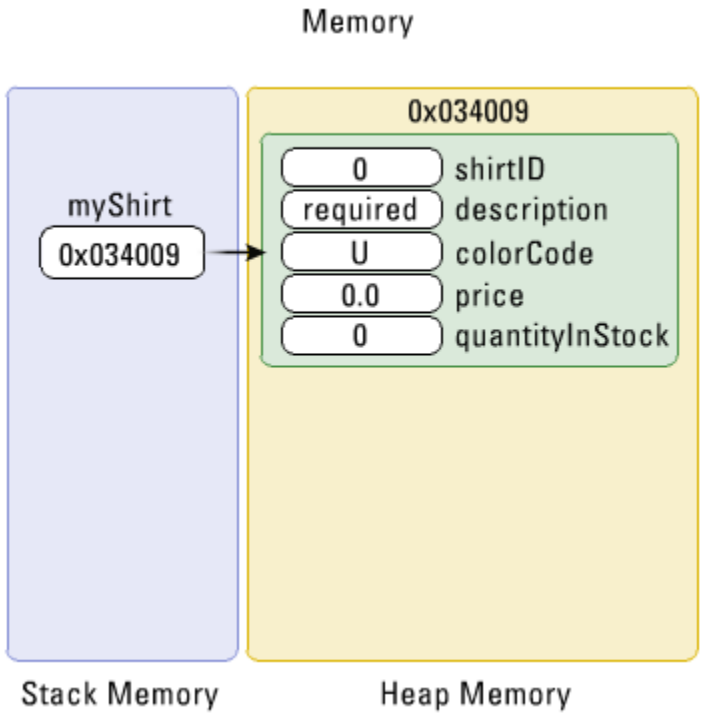
ALLOCAZIONE IN MEMORIA V:

```
public class ShirtTest {

    public static void main (String args[]) {

    Shirt myShirt;
    myShirt = new Shirt();

    myShirt.colorCode = 'G';


    }
}
```



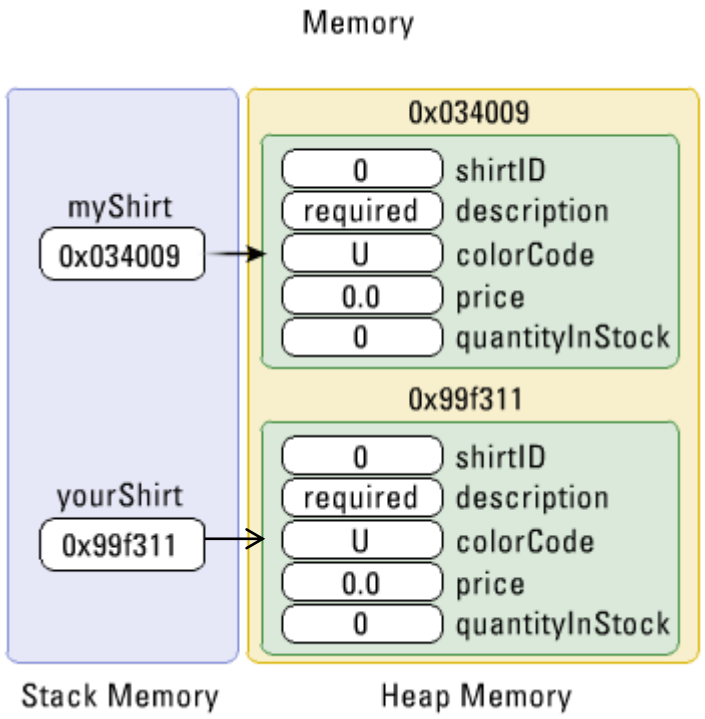
ISTANZIARE OGGETTI

```
public class ShirtTestTwo {
    public static void main (String args[]) {
    Shirt myShirt = new Shirt();
    Shirt yourShirt = new Shirt();
    myShirt.displayShirtInformation();
    yourShirt.displayShirtInformation();
    myShirt.colorCode = 'R';
    yourShirt.colorCode = 'G';
    myShirt.displayShirtInformation();
    yourShirt.displayShirtInformation();
    }
}
```



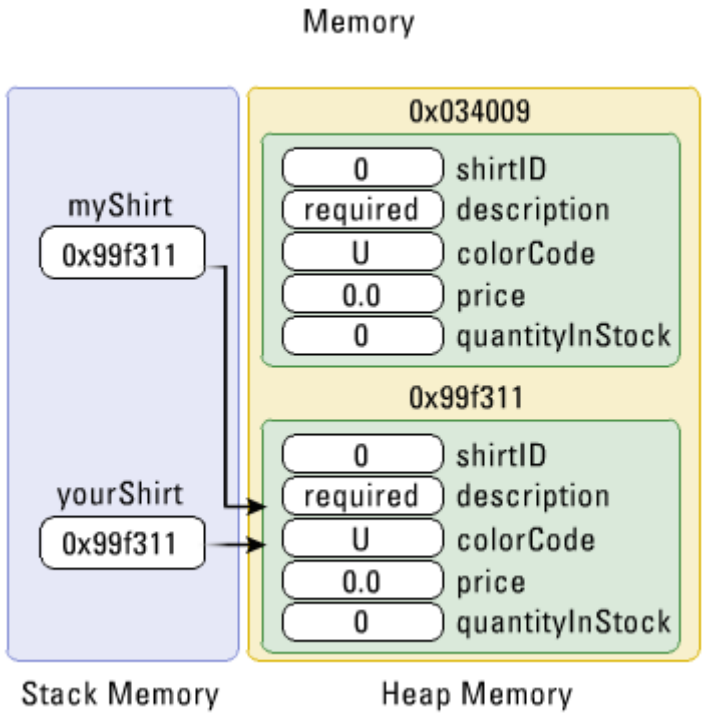
ISTANZIARE OGGETTI

```
public class ShirtTestTwo {
    public static void main (String args[]) {
    Shirt myShirt = new Shirt();
    Shirt yourShirt = new Shirt();
    myShirt.displayShirtInformation();
    yourShirt.displayShirtInformation();
    myShirt.colorCode = 'R';
    yourShirt.colorCode = 'G';
    myShirt.displayShirtInformation();
    yourShirt.displayShirtInformation();
    }
}
```



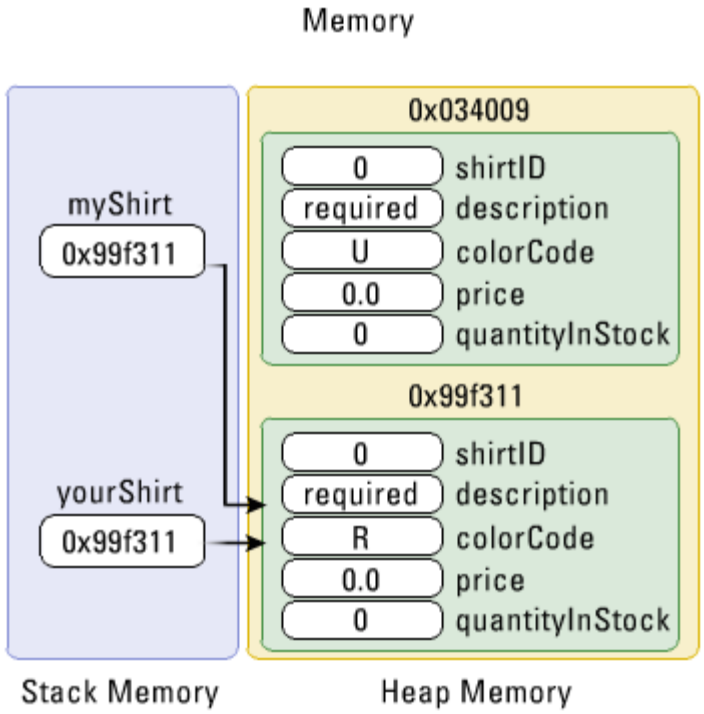
ISTANZIARE OGGETTI

```
public class ShirtTestTwo {
  public static void main (String args[]) {
  Shirt myShirt = new Shirt();
  Shirt yourShirt = new Shirt();
  myShirt = yourShirt;
  myShirt.colorCode = 'R';
  yourShirt.colorCode = 'G';
  }
}
```



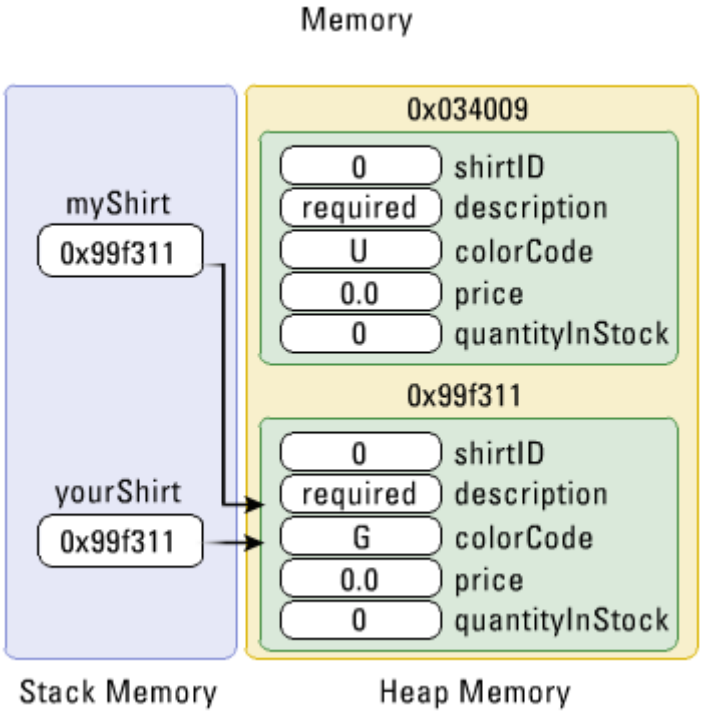
ISTANZIARE OGGETTI

```
public class ShirtTestTwo {
  public static void main (String args[]) {
  Shirt myShirt = new Shirt();
  Shirt yourShirt = new Shirt();
  myShirt = yourShirt;
  myShirt.colorCode = 'R';
  yourShirt.colorCode = 'G';
  }
}
```



ISTANZIARE OGGETTI

```
public class ShirtTestTwo {
  public static void main (String args[]) {
   Shirt myShirt = new Shirt();
   Shirt yourShirt = new Shirt();
   myShirt = yourShirt;
   myShirt.colorCode = 'R';
   yourShirt.colorCode = 'G';
   }
}
```



Questa cosa in c++ non andrebbe mai fatta perché si perde qualunque riferimento all'oggetto

Classi Interne

V:

- In Java è possibile posizionare la definizione di una classe all'interno di un'altra, realizzando quello che prende il nome di classe interna. Ciò consente di raggruppare classi che sono logicamente correlate e di controllare la loro visibilità.

```
public class Document {
     class Destination {
          private String label;
          Destination(String whereTo) {label = whereTo;}
          String readLabel() {return label;}
     }
     public void ship(String dest) {
          Destination d = new Destination(dest);
          System.out.println(d.readLabel());
     }
     public static void main(String[] args) {
          Document doc = new Document();
          p.ship("Tanzania");
     }
}
```

Classi Interne

V:

- In Java è possibile posizionare la definizione di una classe all'interno di un'altra, realizzando quello che prende il nome di classe interna. Ciò consente di raggruppare classi che sono logicamente correlate e di controllare la loro visibilità.

```
public class Document {
    class Destination {
        private String label;
        Destination(String whereTo) {label = whereTo;}
        String readLabel() {return label;}
    }
    public void ship(String dest) {
        Destination d = new Destination(dest);
    }
}
```

Nessuna differenza nell'uso di Destination, bisogna solo ricordarsi che i nomi sono innestati nella classe Document, quindi se si vuole far riferimento alla classe interna al di fuori di Document (o in metodi statici), la sintassi è Document.Destination.

Classi Interne

V:

Ogni classe interna ha completa visibilità degli attributi e dei metodi della classe che la contiene.

```
public class Document {
        private String title = "Titolo";
        class Destination {
                private String label;
                Destination(String whereTo) {label = whereTo;}
                String readLabel() {return label;}
                String returnTitolo() {return Document.this.title;}
        }
        public void ship(String dest) {
                Destination d = new Destination(dest);
                System.out.println(d.readLabel());
        }
        public static void main(String[] args) {
                Document doc = new Document();
                Document.Destination dec = doc.new Destination("Tanzania");
                String str = dec.returnTitolo();
        }}
```

Classi Interne

Ogni classe interna ha completa visibilità degli attributi e dei metodi della classe che la contiene.

```
public class Document {
        private String title = "Titolo";
        class Destination {
                                                                    }
                                                                itle;}
        }
        public void ship(String dest) {
                Destination d = new Destination(dest);
                System.out.println(d.readLabel());
        }
        public static void main(String[] args) {
                Document doc = new Document();
                Document.Destination dec = doc.new Destination("Tanzania");
                String str = dec.returnTitolo();
        }}
```

Un'istanza di una classe interna è ottenibile solo a partire da un'istanza della classe esterna.

Classi Interne

Ogni classe interna ha completa visibilità degli attributi e dei metodi della class

Possiamo disciplinare la visibilità della classe interna con le parole chiave public, private prected.

```
publi
        private String title = "Titolo";
        private class Destination {
                private String label;
                Destination(String whereTo) {label = whereTo;}
                String readLabel() {return label;}
                String returnTitolo() {return Document.this.title;}
        }
        pub
        }
        public static void main(String[] args) {
                Document doc = new Document();
                Document.Destination dec = doc.new Destination("Tanzania");
                String str = dec.returnTitolo();
        }}
```

Così da non rendere possibile istanziare un oggetto della classe interna