## PROVA PRATICA

**Contenuti:**

- *Uso del JDK e di un ambiente integrato per lo sviluppo*
- *Semplici programmi che utilizzino:*
  - *Classi*
  - *Classi astratte*
  - *(Interfacce)*
  - *Ereditarietà (protezione, costruttori)*
- *Tempo a disposizione: 30 min*

**Lo scritto (se superato) vale anche se la PP non è superata (e viceversa)**

## ESERCIZIO

**Problema**

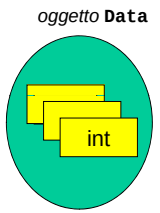
- *Si realizzi un componente software Data che incapsula al proprio interno una data nel formato giorno, mese e anno*
- *Il componente deve esportare i seguenti metodi:*
  - *un costruttore a tre argomenti (gg, mm, aa);*
  - *un metodo displayData che visualizzi la data (gg, mm, aa);*
  - *un metodo trasformatore setData che modifichi la data (acquisendo in ingresso una nuova tripla gg, mm, aa).*





## LA CLASSE `Data`

**Definiamo la classe `Data` che *incapsuli giorno, mese e anno (tutti `int`)***




- **Costruttore:** costruisce un oggetto **`Data`** a partire da tre **`int`**
- **Metodo `displayData`:** visualizza i tre **`int`** da un oggetto **`Data`**
- **Metodo `setData`:** cambia il valore dei tre campi **`int`** contenuti in un oggetto **`Data`**

## LA CLASSE `Data`

```
public class Data {
 private int giorno,mese,anno;
 public Data(int gg, int mm, int aa)
    {giorno=gg; mese=mm; anno=aa; }

 public void  displayData()
    {System.out.printl("GG/MM/AA: " +
      giorno + "/" + mese + "/" + anno);}

 public void setData(int gg,int mm,int aa)
       {giorno=gg; mese=mm; anno=aa; }
                    }
```

## ESERCIZIO (cont.)

- *Si realizzi poi una funzione main in una classe Prova che:*
  - *dichiara due oggetti oggi e domani , istanze della classe Data ,*
  - *assegna loro rispettivamente la data odierna e quella di domani e*
  - *visualizza il loro valore.*

## LA CLASSE Prova

```
public class Prova {
 public static void main(String args[]){
  Data oggi, domani;
  oggi = new Data(20,3,2001);
  domani = new Data(21,3,2001);

  oggi.displayData();
  domani.displayData();
  }
}
```

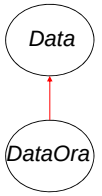
```
GG/MM/AA: 20/3/2001
GG/MM/AA: 21/3/2001
```





## ESERCIZIO (cont.)

- *Si realizzi poi un componente DataOra che estenda il componente Data aggiungendovi l'orario in termini di ore e minuti*




## ESERCIZIO (cont.)

- *Del nuovo componente è possibile invocare i seguenti metodi:*
  - *un costruttore a cinque argomenti (gg, mm, aa, ore, min);*
  - *displayData che visualizza la data (gg, mm, aa);*
  - *displayOra che visualizza l'orario (ore, minuti);*
  - *un metodo trasformatore setDataOra che modifica la data e l'orario (acquisendo in ingresso una nuova quintupla gg, mm, aa, ore, minuti).*

## LA CLASSE DataOra

```
public class DataOra extends Data {
 private int ore,minuti;
 public DataOra(int gg, int mm, int aa,
                int o, int m)
   {super.Data(gg,mm,aa); ore=o;  min=m;}

 public void  displayOra()
  {System.out.printl("Ore:Min " + ore +
  ":" + min);}

 public void setDataOra(int gg,int mm,int
  aa, int o, int m)
    {super.setData(gg,mm,aa);
     ore=o;  min=m; }
}
```

## LA CLASSE DataOra

```
public class DataOra extends Data {
 private int ore,min;
 public DataOra(int gg, int mm, int aa,
                int o, int m)
   {super(gg,mm,aa); ore=o;  min=m;}

 public void  displayOra()
  {System.out.printl("Ore:Min " + ore +
  ":" + min);}

 public void setDataOra(int gg,int mm,int
  aa, int o, int m)
    {super.setData(gg,mm,aa);
     ore=o;  min=m; }
}
```





## ESERCIZIO (cont.)

- *Si modifichi poi la funzione main della classe Prova in modo che in aggiunta:*
  - *dichiari anche un oggetto giorno della classe DataOra,*
  - *assegni a questo la data odierna e l'orario 13:30*
  - *ne visualizzi il contenuto utilizzando i metodi displayData e displayOra.*

## LA NUOVA CLASSE Prova

```
public class Prova {
 public static void main(String args[]){
  Data oggi, domani;
  oggi = new Data(20,3,2001);
  domani = new Data(21,3,2001);

  oggi.displayData();
  domani.displayData();   // continua
```

## LA NUOVA CLASSE Prova

```
DataOra giorno;
  giorno= new DataOra(20,3,2001,13,30);

  giorno.displayData(); // ereditato
  giorno.displayOra();
  }
}
```

```
C:\temp> java Prova

GG/MM/AA: 20/3/2001
GG/MM/AA: 21/3/2001
GG/MM/AA: 20/3/2001
Ore:Min 13:30
```

## LA NUOVA CLASSE Prova

```
DataOra giorno;
  giorno= new DataOra(20,3,2001,13,30);

  giorno.displayData(); // ereditato
  giorno.displayOra();
  }
}
```

```
GG/MM/AA: 20/3/2001
GG/MM/AA: 21/3/2001
GG/MM/AA: 20/3/2001
Ore:Min 13:30
```





## SOLUZIONE ALTERNATIVA

- *Se la classe Data viene progettata già pensando a possibili estensioni:*
  - *conviene dichiararne lo stato come protected anziché private*
  - *anche i metodi di DataOra hanno cosi' la visibilità della parte di stato definita in Data*

## SOLUZIONE II PER Data

```
public class Data {
 protected int giorno,mese,anno;
 public Data(int gg, int mm, int aa)
    {giorno=gg; mese=mm; anno=aa; }

 public void  displayData()
    {System.out.printl("GG/MM/AA: " +
      giorno + "/" + mese + "/" + anno);}

 public void setData(int gg,int mm,int aa)
       {giorno=gg; mese=mm; anno=aa; }
                    }
```

## SOLUZIONE II PER `DataOra`:

```
public class DataOra extends Data {
 private int ore,minuti;
 public DataOra(int gg, int mm, int aa,
                int o, int m)
   {super(gg,mm,aa); ore=o;  min=m;}

 public void  displayOra()
  {System.out.printl("Ore:Min " + ore +
  ":" + min);}

 public void setDataOra(int gg,int mm,int
  aa, int o, int m)
    {giorno=gg;  mese=mm;  anno=aa;
     ore=o;  min=m; }
}
```

## Prova Pratica 31 Marzo 2004

- **Da realizzare in un *unico file* .java**
- **Nella soluzione, prediligere il maggior riutilizzo di codice e la maggiore protezione possibile.**





## Classe astratta

- **Si realizzi una classe astratta Animale che rappresenta la classe degli animali, con attributo nome e che definisce i seguenti metodi:**
  - **Un metodo costruttore a un argomento che riceve in ingresso il nome dell'animale (String);**
  - **Un metodo (astratto)**

    **public abstract String who();**

- **Si realizzi poi una classe Cane (derivata dalla precedente) che rappresenta la classe dei cani e che definisce il metodo who in modo tale da fargli restituire il nome**
- **Si realizzi poi una funzione main in una classe Prova che:**
  - **Dichiari un oggetto c, istanza della classe Cane, con nome "boby";**
  - **Chiamando i metodi opportuni, stampi a video il nome dell'oggetto.**

## Soluzione 31 Marzo 2004

```
abstract class Animale
{ protected String nome;
  public Animale(String s)
                 { nome=s; }
  public abstract String who();
}
```

```
class Cane extends Animale {
 public Cane(String s) { super(s); }
 public String who(){return nome; }
}

public class Prova {
 public static void main(String args[])
  {Cane  c = new cane("boby");

  System.out.println("Nome:"+c.who());}
}
```





## Prova Pratica 24 Marzo 2004

- **Da realizzare in un *unico file* .java**
- **Nella soluzione, prediligere il maggior riutilizzo di codice e la maggiore protezione possibile.**

## Interfaccia

- **Si realizzi un componente interfaccia Persona che rappresenta la tipologia della persone.**
- **In tale interfaccia si definisce il metodo:**
  - **public int coetanei(Object X)**

    **che ricevendo il riferimento a un oggetto X, stabilisce se esso è uguale a quello su cui è invocato il metodo stesso.**

- **Si realizzi poi una classe PersonaFisica che implementa l'interfaccia precedente.**
  - **Ciascuna PersonaFisica ha un codice fiscale (stringa) e un anno di nascita.**
  - **Il metodo coetanei restituisce 1 se i due oggetti hanno lo stesso anno di nascita, 0 altrimenti.**

- **Si realizzi poi una funzione main in una classe Prova che:**
  - **Dichiari un oggetto o1, istanza della classe PersonaFisica, con CF "LMMVLN60C46A977T", anno di nascita 1960, e un oggetto o2, istanza della classe PersonaFisica, con CF "GHKNBM60C35A977T", anno di nascita 1960;**
  - **Chiamando il metodo coetanei verifichi se o1 e o2 sono coetanei.**





## Soluzione 24 Marzo 2004

```
interface Persona {
  public int coetanei(Object X); }

class PersonaFisica implements Persona
{ public String cod_fiscale;
  public int anno_nasc;
  PersonaFisica(String cod, int anno)
     {this.anno_nasc=anno;
      this.cod_fiscale=cod;    }
  public int coetanei(Object X)
{if(this.anno_nasc==((PersonaFisica)X).anno_nasc)
   {return 1;}
 else { return 0;}}
}
```

## Soluzione 24 Marzo 2004

```
public class Prova {
 public static void main(String[] args)
  {PersonaFisica o1 = new
      PersonaFisica("LMMVLN60C46A977T",1960);
   PersonaFisica o2 = new
      PersonaFisica("GHKNBM60C35A977T",1960);
   if (o1.coetanei(o2)==1)
    { System.out.println("Le due persone"+
                  "sono coetanee !!"); }
  else
    { System.out.println("Le due persone"+
                  "non sono coetanee !!"); }
  }
 }
```