Esercizio 4. Un *medico* può essere caratterizzato tramite nome, cognome, luogo di nascita, anno di nascita, anno in cui ha conseguito la laurea e la sede di lavoro. Scrivere una classe `Medico` con un opportuno costruttore ed i metodi che restituiscono i valori delle variabili istanza. Definire inoltre un metodo che modifica la sede di lavoro di un medico, un metodo che verifica se, dato un intero k, un medico si è laureato prima dell'anno k, un metodo `equals` che verifica se due oggetti della classe `Medico` sono uguali, un metodo che, dati due medici, restituisce quello più giovane ed infine un metodo che restituisce una stringa che descrive un oggetto della classe `Medico`.

```
public class Medico {

 private String nome, cognome, luogoN, sedeL;
 private int annoN, annoL;

 public Medico (String n, String c, String ln, int an, int al, String s) {
  this.nome = n;
  this.cognome = c;
  this.luogoN = ln;
  this.annoN = an;
  this.annoL = al;
  this.sedeL = s;
 }

 public String leggiNome() { return this.nome; }
 public String leggiCognome() { return this.cognome; }
 public String leggiLuogoN() { return this.luogoN; }
 public int leggiAnnoN() { return this.annoN; }
 public int leggiAnnoL() { return this.annoL; }
 public String leggiSedeL() { return this.sedeL; }

 public void modSedeL(String s) { this.sedeL = s; }

 public boolean laureaAnno(int k) {
  return this.annoL < k;
 }

 public boolean equals(Medico d) {
  return this.nome.equals(d.nome) && this.cognome.equals(d.cognome) &&
         this.luogoN.equals(d.luogoN) && this.annoN == d.annoN &&
         this.annoL == d.annoL && this.sedeL.equals(d.sedeL);
 }

 public Medico giovane(Medico d) {
  if (this.annoN > d.annoN)
    return this;
  else
    return d;
 }

 public String descr() {
```

```
  return this.nome + " " + this.cognome + " nato a " + this.luogoN + " nel "
         + this.annoN + " ha conseguito la laurea nel " + this.annoL +
         " ed ora lavora presso " + this.sedeL;
 }
}
```

Osservazione
Il metodo `equals` definito nella classe `Medico` non sovrascrive il metodo `equals` della superclasse `Object`. La sovrascrittura del metodo si ha con la seguente definizione:

```
public boolean equals(Object obj) {
 if (!(obj instanceof Medico)) return false;
 Medico d = (Medico)obj;
 return this.nome.equals(d.nome) && this.cognome.equals(d.cognome) &&
        this.luogoN.equals(d.luogoN) && this.annoN == d.annoN &&
        this.annoL == d.annoL && this.sedeL.equals(d.sedeL);
}
```

Esercizio 5. Un *medico specializzato* è un medico caratterizzato anche da una specializzazione, dall'anno in cui l'ha conseguita e dalla durata del corso di specializzazione. Scrivere una classe `MedicoS` con un opportuno costruttore ed i metodi che restituiscono i valori delle variabili istanza. Definire inoltre un metodo che verifica se due medici specializzati hanno conseguito la laurea nello stesso anno e hanno la stessa specializzazione, un metodo che, dati due medici specializzati, restituisce quello che ha conseguito una specializzazione più recentemente ed un metodo che restituisce una stringa che descrive un oggetto della classe `MedicoS`.

```
public class MedicoS extends Medico {

 private String spec;
 private int annoS, durataS;

 public MedicoS(String n, String c, String ln, int an, int al, String s,
 String sp, int aS, int dS) {
  super(n,c,ln,an,al,s);
  this.spec = sp;
  this.annoS = aS;
  this.durataS = dS;
 }

 public String leggiSpec() { return this.spec; }
 public int leggiAnnoS() { return this.annoS; }
 public int leggiDurataS() { return this.durataS; }

 public boolean verifica(MedicoS d) {
  return this.leggiAnnoL() == d.leggiAnnoL() && this.spec.equals(d.spec);
 }

 public MedicoS specRecente(MedicoS d) {
  if (this.annoS > d.annoS)
    return this;
```

```
  else
    return d;
 }

 public String descr() {
  return super.descr() + "Ha conseguito una specializzazione in " +
         this.spec + " nel " + this.annoS + " dopo un corso di " +
          this.durataS + "anni.";
 }
}
```

Esercizio 6. Un *ambulatorio* può essere caratterizzato tramite nome, nome del responsabile, indirizzo e l'elenco dei medici che vi lavorano. Scrivere una classe `Ambulatorio`, il cui costruttore imposta il numero massimo di medici in un ambulatorio. Oltre ai metodi che restituiscono i valori delle variabili istanza, definire i seguenti metodi:
- un metodo che, dato un medico d, aggiunge d nell'elenco dei medici di un ambulatorio a patto che la sede di lavoro di d coincida con il nome dell'ambulatorio (il metodo deve sollevare opportune eccezioni *distinte* in caso di elenco pieno e di nomi diversi);
- un metodo che, dato un medico d, verifica se d occorre nell'elenco dei medici di un ambulatorio (N.B. utilizzare il metodo `equals` definito nella classe `Medico`);
- un metodo che, data una stringa s, restituisce l'elenco dei medici specializzati di un ambulatorio aventi s come specializzazione.

```
class ArrayPienoExc extends Exception {}
class NomiDiversiExc extends Exception {}

public class Ambulatorio {

 private String nome, resp, indirizzo;
 private Medico[] elenco;
 private int cont;

 public Ambulatorio(String n, String r, String i, int max) {
  this.nome = n;
  this.resp = r;
  this.indirizzo = i;
  this.elenco = new Medico[max];
 }

 public String leggiNome() { return this.nome; }
 public String leggiResp() { return this.resp; }
 public String leggiInd() { return this.indirizzo; }
 public Medico[] leggiElenco() { return this.elenco; }

 public void aggMedico(Medico d) throws ArrayPienoExc, NomiDiversiExc {
  if (cont == elenco.length) throw new ArrayPienoExc();
  if (!this.nome.equals(d.leggiSedeL())) throw new NomiDiversiExc();
  elenco[cont] = d;
  cont++;
 }
```

```
 public boolean esisteM(Medico d) {
  for (int i=0; i<cont; i++) {
    if (elenco[i].equals(d)) return true;
  }
  return false;
 }

 public MedicoS[] elencoMS(String s) {
  int c = 0;
  for (int i=0; i<cont; i++) {
    if (elenco[i] instanceof MedicoS &&
        ((MedicoS)elenco[i]).leggiSpec().equals(s))
      c++;
  }
  MedicoS[] a = new MedicoS[c];
  int j = 0;
  for (int i=0; i<cont; i++) {
    if (elenco[i] instanceof MedicoS &&
        ((MedicoS)elenco[i]).leggiSpec().equals(s)) {
      a[j] = (MedicoS)elenco[i];
      j++;
    }
  }
  return a;
 }

}
```