COMPITO DI LABORATORIO DI PROGRAMMAZIONE 2001-2002

18 Settembre 2002

Soluzione degli Esercizi

ESERCIZIO 1. Si consideri il seguente metodo in linguaggio Java.

```
public static boolean metodo (int[] a) {
  boolean condizione = true;
  for (int i = 0; i < a.length-1; i++) {
    if (a[i] != 2*a[i+1]) {
      condizione = false;
    }
  }
  return condizione;
}
```

1. Determinare quale caratteristica deve soddisfare l'array `a` perché il metodo restituisca il valore `true`.
2. Riscrivere il metodo usando il ciclo `while`. Spiegare quale delle due soluzioni sia la piú efficiente e il perché.

Soluzione
1. Affinché il metodo restituisca il valore `true`, gli elementi dell'array `a` devono essere ciascuno il doppio dell'elemento successivo in `a`.
2. Soluzione con il `while`:

```
public static boolean metodo (int[] a) {
  boolean condizione = true;
  int i = 0;
  while (i < a.length-1 && condizione) {
    if (a[i] != 2*a[i+1]) {
      condizione = false;
    }
    i++;
  }
  return condizione;
}
```

La soluzione piú efficiente è quella con il `while`, in quanto nel caso in cui l'array non soddisfi la caratteristica detta al punto 1., l'iterazione termina non appena si trova un elemento dell'array che non è il doppio del successivo, senza scandire tutto l'array. Tale scansione di tutto l'array viene invece sempre eseguita dalla soluzione con il `for` indipendentemente dal valore finale di `condizione`.

ESERCIZIO 2. Scrivere un metodo in linguaggio Java che, data una stringa `s` ed un array di caratteri `c`, restituisca un array di interi `conta` tale che `conta[i]` sia il numero di occorrenze del carattere `c[i]` nella stringa `s`. (Esempio: se `s = "ciao pippo"` e `c = {'a', 'b', 'c', 'p'}`, il metodo restituisce `conta = {1,0,1,3}`).

Soluzione

```
public static int[] ContaChar (String s, char[] c) {
  int[] conta = new int[c.length];
  for (int i = 0; i < c.length; i++) {
    for (int j = 0; j < s.length(); j++) {
      if (s.charAt[j] == c[i])
         conta[i]++;
    }
  }
  return conta;
}
```

ESERCIZIO 3. Scrivere un metodo in linguaggio Java che, dato un array di interi `a`, restituisca `true` se i suoi elementi sono in ordine non decrescente (`a[0]`$\leq$ `a[1]` $\leq \ldots$), e `false` altrimenti.

Soluzione

```
public static boolean nondecr (int[] a) {
  boolean cond = true;
  int i = 0;
  while (i < a.length-1 && cond) {
    if (a[i] > a[i+1]) {
      cond = false;
    }
    i++;
  }
  return cond;
}
```

ESERCIZIO 4. Risolvere l'esercizio 3 usando la ricorsione.

Soluzione

```
public static boolean nondecrRic (int[] a, int k) {
  if (k == a.length-1)
    return true;
  else
   if (a[k} > a[k+1])
     return false;
   else return nondecrRic(a,k+1);
}

public static boolean nondecrR (int[] a) {
  return nondecrRic(a,0);
}
```

ESERCIZIO 5. Si consideri la seguente classe `polizza_base` per assicurazioni auto comprendenti la sola responsabilità civile (RC). Le variabili istanza sono il veicolo assicurato,

identificato dalla targa (ad esempio, "CA 075 DS"), ed il valore assicurato RC. I metodi sono quelli che restituiscono i valori delle variabili istanza ed il premio annuale (pari ad 1/100 del valore assicurato).

```
public class polizza_base {
  public polizza_base (String targa, int valore_RC) {
    this.targa = targa;
    this.valore_RC = valore_RC;
  }

  public String get_veicolo() {
    return targa;
  }

  public int get_valore_RC() {
    return valore_RC;
  }

  public int premio() {
    return valore_RC/100;
  }

  private String targa;
  private int valore_RC;
}
```

Una polizza auto incendio e furto si differenzia da una polizza base sopra descritta in quanto:

1. prevede, in aggiunta all'assicurazione RC, l'assicurazione per incendio e furto del veicolo;
2. il premio annuale è pari ad un centesimo della somma dei valori assicurati RC ed incendio e furto. (Esempio: valore assicurato RC=5000; valore assicurato incendio e furto=1000; premio=(5000+1000)/100=60).

Scrivere la nuova classe `polizza_incendio_e_furto` motivando le scelte effettuate.

Soluzione

```
public class polizza_incendio_e_furto extends polizza_base {

  private int valore_IF;

  public polizza_incendio_e_furto (String t, int vRC, int vIF) {
    super(t,vRC);
    valore_IF = vIF;
  }

  public int get_valore_IF() {
    return valore_IF;
  }

  public int premio() {
```

```
    return (get_valore_RC() + valore_IF)/100;
  }
}
```

ESERCIZIO 6. La banca dati di un'agenzia assicurativa per auto è caratterizzata da un insieme di polizze di assicurazioni per auto. Usando le classi del precedente esercizio, scrivere una semplice classe `Agenzia_Assicurativa`, dove ciascun oggetto della classe è caratterizzato dal nome dell'agenzia e da un elenco di polizze auto (il cui numero massimo deve essere impostato dal costruttore della classe), che abbia i seguenti metodi:

1. un metodo che aggiunge una polizza;
2. un metodo che restituisce la somma dei premi annui di tutte le polizze dell'agenzia;
3. un metodo che restituisce l'elenco dei valori assicurati RC di tutte le polizze dell'agenzia.

Soluzione

```
public class Agenzia_Assicurativa {

  private String nome;
  private polizza_base[] elenco;
  private int cont;

  public Agenzia_Assicurativa (String s, int max) {
    nome = s;
    elenco = new polizza_base[max];
  }

  public void add_polizza (polizza_base p) {
    if (cont < elenco.length) {
      elenco[cont] = p;
      cont++;
    }
  }

  public int somma_premi() {
    int somma = 0;
    for (int i = 0; i < cont; i++) {
      somma += elenco[i].premio();
    }
    return somma;
  }

  public int[] elenco_valori_RC() {
    int[] a = new int[cont];
    for (int i = 0; i < cont; i++) {
      a[i] = elenco[i].get_valore_RC();
    }
    return a;
  }
}
```