Fondamenti di informatica T-1 (A – K)
Esercitazione 3: assegnamento e variabili

A.A. 2018/2019

Tutor

Lorenzo Rosa

lorenzo.rosa@unibo.it

Esercitazione 3

- Introduzione al calcolatore e Java
- **Linguaggio Java, basi e controllo del flusso**
- Eclipse ed esercizi di consolidamento
- Stringhe ed array
- Metodi, classi, oggetti
- Ereditarietà e polimorfismo
- Collezioni Java e interfacce
- Esercizi d'esame

Assegnamento

- Per poter memorizzare un valore, è necessario salvarlo in memoria. I linguaggi di programmazione effettuano questa azione attraverso l'assegnamento: **int a = 2;**
- **int** indica il tipo del valore che stiamo salvando;
- **a** è una variabile, è il "contenitore" dentro cui verrà memorizzato il valore "2";
- Dunque l'operatore "=" non ha il significato di uguaglianza matematica. Possiamo leggere l'espressione come "inserisco il valore 2 nella variabile a".

La macchina cambia-monete

- Si vuole realizzare una macchina in grado di calcolare la minima combinazione di monete corrispondente a un importo arbitrariamente scelto.
- Requisiti
  - L'utente inserisce un importo compreso tra 0 e 99 centesimi.
  - Il programma determina la combinazione di monete corrispondenti.
- Esempio: a 55 centesimi corrispondono una moneta da 50 centesimi e una da 5.

La macchina cambia-monete

- Esempio di output

```
Inserisci un intero compreso tra 1 e 99
identifichero' una combinazione di monete
che corrisponde a tale cifra.

87

87 centesimi in moneta corrispondono a:
1 monete da cinquanta centesimi
1 monete da venti centesimi
1 monete da dieci centesimi
1 monete da cinque centesimi
1 monete da due centesimi
0 monete da un centesimo
```

Algoritmo – prima versione

1. Leggi l'importo.
2. Trova il massimo numero di monete da 50 cent nell'importo.
3. Sottrai il valore di tali monete dall'importo.
4. Ripeti gli ultimi due passi per le monete da venti, dieci, cinque, due, uno.
5. Stampa l'importo iniziale e la quantità per ogni tipo di moneta.

Quali variabili?

```
int quantita
```

Valore letto da input

```
int cinquantaCent
int ventiCent
int dieciCent
int cinqueCent
int dueCent
int unCent
```

Quantità di monete, una per ogni tipo

Algoritmo – prima versione

- Con le variabili che abbiamo individuato, l'algoritmo non funziona bene
  - L'importo iniziale è cambiato ad ogni passo.
  - Il valore iniziale di **`quantita`** viene perso.
- Aggiungiamo una variabile

```
int quantitaIniziale
```

- Aggiorniamo l'algoritmo

Algoritmo – seconda versione

1. Leggi l'importo.
2. Crea una copia dell'importo.
3. Trova il massimo numero di monete da 50 cent nell'importo.
4. Sottrai il valore di tali monete dall'importo.
5. Ripeti gli ultimi due passi per le monete da venti, dieci, cinque, due, uno.
6. Stampa l'importo iniziale e la quantità per ogni tipo di moneta.

Qualche calcolo (1/2)

- Come facciamo a determinare il numero di monete (es. da 20 cent) in un importo?
- Ci sono 2 monete da 20 in 45 cent, ma anche 2 monete da 20 in 55 cent.
- Ci riferiamo a divisioni intere:

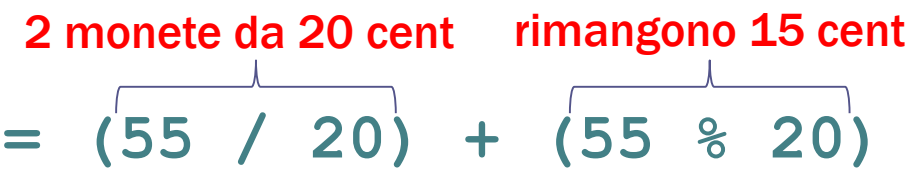
```
45 / 20 = 2 e anche 55 / 20 = 2.
```

Qualche calcolo (2/2)

- Come determinare il resto?
- Il resto della divisione intera si può determinare usando l'operatore di modulo (`%`):

```
45 % 20 = 5 e 55 % 20 = 15
```

- Sul risultato si applicherà nuovamente la divisione, considerando un taglio inferiore di moneta.



- Le monete da un centesimo saranno semplicemente `quantita % 2`.

Algoritmo – implementazione (1/6)

1. **Leggi l'importo.**
2. **Crea una copia dell'importo.**

- Stampare a video un messaggio di accoglienza e la richiesta dell'importo.
- Leggere da input l'importo.
- Salvare l'importo nella variabile `quantita` e una sua copia nella variabile `quantitaIniziale`

Algoritmo – implementazione (2/6)

1. **Leggi l'importo.**
2. **Crea una copia dell'importo.**

```
int quantita, quantitaIniziale;
int cinquantaCent, ventiCent, dieciCent, cinqueCent,
dueCent, unCent;

System.out.println("Inserisci un intero, maggiore di
zero");
System.out.println("identifichero' una combinazione di
monete");
System.out.println("che corrisponde a tale cifra.");

Scanner tastiera = new Scanner(System.in);
quantita = tastiera.nextInt();
quantitaIniziale = quantita;
```

Algoritmo – implementazione (3/6)

3. Trova il massimo numero di monete da 50 cent nell'importo.

```
cinquantaCent = quantita/50;
```

4. Sottrai il valore di tali monete dall'importo.

```
quantita = quantita % 50;
```

Algoritmo – implementazione (4/6)

5. Ripeti gli ultimi due passi per le monete da venti, dieci, cinque, due, uno.

```
ventiCent = quantita/20;
quantita = quantita % 20;

dieciCent = quantita/10;
quantita = quantita % 10;

cinqueCent = quantita/5;
quantita = quantita % 5;

dueCent = quantita / 2;
quantita = quantita % 2;

unCent = quantita;
```

Algoritmo – implementazione (5/6)

6. Stampa l'importo iniziale e la quantità per ogni tipo di moneta.

```
System.out.println(quantitaIniziale +
   " centesimi in moneta corrispondono a:");

System.out.println(cinquantaCent + " monete
da cinquanta centesimi");
```

Algoritmo – implementazione (6/6)

6. Stampa l'importo iniziale e la quantità per ogni tipo di moneta.

```
System.out.println(ventiCent + " monete da venti
                                  centesimi");
System.out.println(dieciCent + " monete da
                                  dieci centesimi");
System.out.println(cinqueCent + " monete da
                                  cinque centesimi");
System.out.println(dueCent + " monete da due
                                  centesimi");
System.out.println(unCent + " monete da un
                                  centesimo");
```

Algoritmo – testing

- Il programma deve essere testato con diversi importi per vedere se funziona correttamente.
- Test con importi come
  - valori estremi, come 0, 1, 98 e 99
  - valori delle monete, come 10, 20 o 50.

Algoritmo – testing

```
import java.util.Scanner;

public class CambiaMonete {

    public static void main(String[] args) {

        int quantita, quantitaIniziale;
        int cinquantaCent, ventiCent, dieciCent, cinqueCent, dueCent, unCent;

        System.out.println("Inserisci un intero compreso tra 1 e 99.");
        System.out.println("Identifichero' una combinazione di monete");
        System.out.println("che corrisponde a tale cifra.");
        Scanner tastiera = new Scanner(System.in);
        quantita = tastiera.nextInt();
        quantitaIniziale = quantita;

        cinquantaCent = quantita / 50;
        quantita = quantita % 50;
        ventiCent = quantita / 20;
        quantita = quantita % 20;
        dieciCent = quantita / 10;
        quantita = quantita % 10;
        cinqueCent = quantita / 5;
        quantita = quantita % 5;
        dueCent = quantita / 2;
        quantita = quantita % 2;
        unCent = quantita;
```

50 centesimi stanno in 97 una volta con resto di 47.

97 / 50 è 1
97 % 50 è 47.

97 corrisponde a una moneta da 50 centesimi più 47 centesimi

Algoritmo – testing

```
        System.out.println(quantitaIniziale +
                           " centesimi in moneta corrispondono a:");
        System.out.println(cinquantaCent + " monete da cinquanta centesimi");
        System.out.println(ventiCent + " monete da venti centesimi");
        System.out.println(dieciCent + " monete da dieci centesimi");
        System.out.println(cinqueCent + " monete da cinque centesimi");
        System.out.println(dueCent + " monete da due centesimi e");
        System.out.println(unCent + " monete da un centesimo");
    }
}
```

Esempio di output

```
Inserisci un intero compreso tra 1 e 99.
Identifichero' una combinazione di monete
che corrisponde a tale cifra.
97
97 centesimi in moneta corrispondono a:
1 monete da cinquanta centesimi
2 monete da venti centesimi
0 monete da dieci centesimi
1 monete da cinque centesimi
1 monete da due centesimi e
0 monete da un centesimo
```

Esercizio 1

- Estendere l'esercizio precedente per comprendere anche le monete da 1€ e 2€
  - dato un input in centesimi, il comportamento deve essere lo stesso del caso precedente.
  - ammettiamo quindi che possano essere inseriti importi superiori a 99.

Esercizio 2

- Modificare l'esercizio precedente utilizzando le sotto-unità del dollaro:
  - 1 quarter = 25 cents
  - 1 dime = 10 cents
  - 1 nickel = 5 cents
  - 1 penny = 1 cent

Esercizio 3 (1/5)

- Estendere l'esercizio 1 prendendo in ingresso non più i centesimi, ma un valore espresso in euro.
  - Ad esempio, l'input dovrà essere 1,50 invece di 150
  - Quindi, non potremo più usare solo variabili di tipo **`int`**.
  - Ma dovremo comunque fare in modo che le divisioni siano divisioni tra interi.

Esercizio 3 (2/5)

- Di che tipo dovrà essere la variabile in cui salvo l'input?
  - Non più di tipo **`int`**, ma di tipo **`double`**.
- Le divisioni, però devono rimanere divisioni tra interi
  - Letto l'input in euro, lo dovrò trasformare in centesimi e salvare in una variabile intera;
  - Da quel punto in poi posso proseguire nello stesso modo dell'esercizio 1.

Esercizio 3 (3/5) - Algoritmo

1. **Leggi l'importo in euro.**
2. **Trasforma l'importo in centesimi.**
3. **Trasforma i centesimi in un valore intero.**
4. Crea una copia dell'importo.
3. Trova il massimo numero di monete da 1 euro nell'importo.
4. Sottrai il valore di tali monete dall'importo.
5. Ripeti gli ultimi due passi per le monete da venti, dieci, cinque, due, uno.
6. Stampa l'importo iniziale e la quantità per ogni tipo di moneta.

Esercizio 3 (4/5) - Implementazione

- Per i passi 1 e 2 mi servono quindi due variabili di appoggio:

  1. `double quantitaInEuro` per salvare l'input.

- Quindi, sostituire la riga

```
quantita = tastiera.nextInt();
```

con

```
quantitaInEuro = tastiera.nextDouble();
```

2. `double quantitaInCentesimi` in cui salvo `quantitaInEuro` trasformato in centesimi.
   Il risultato è ancora un `double`!

Esercizio 3 (5/5) - Implementazione

- Al passo 3 devo convertire `quantitaInCentesimi` in un intero.

- L'operatore *cast* mi consente di fare proprio questa conversione: `( <nome_nuovo_tipo> )`

```
int quantita = (int) quantitaInCentesimi;
```

- Adesso posso proseguire esattamente come nell'Esercizio 1.