L'istruzione `if`

- L'istruzione `if` consente ad un programma di effettuare diverse azioni in base ad una condizione

```
if (amount <= balance)
   balance = balance - amount;
```

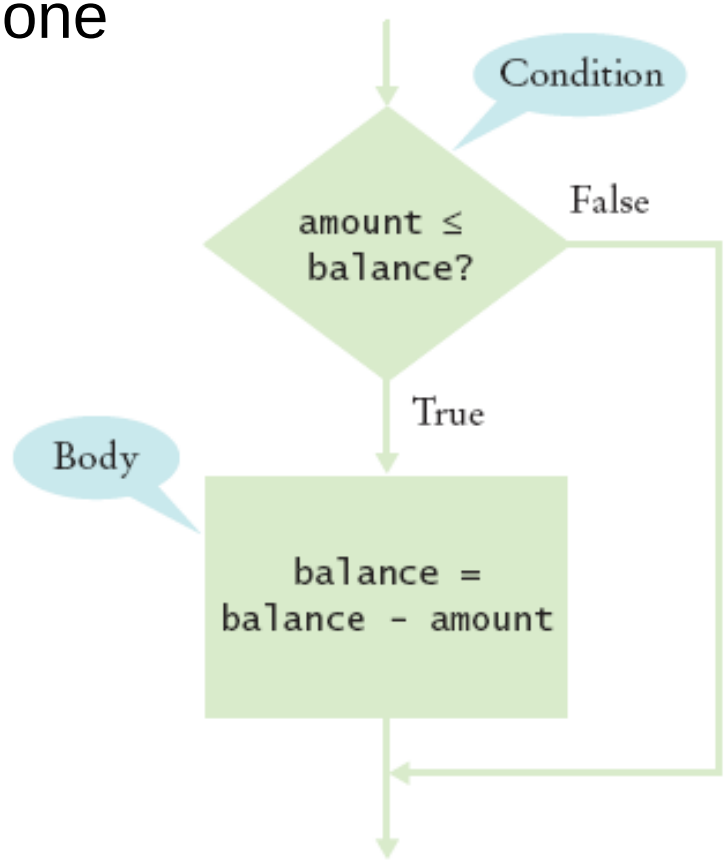

Figure 1
Flowchart for an `if` Statement

L'istruzione `if/else`

```
if (amount <= balance)
      balance = balance - amount;
else
      balance = balance - OVERDRAFT_PENALTY
```

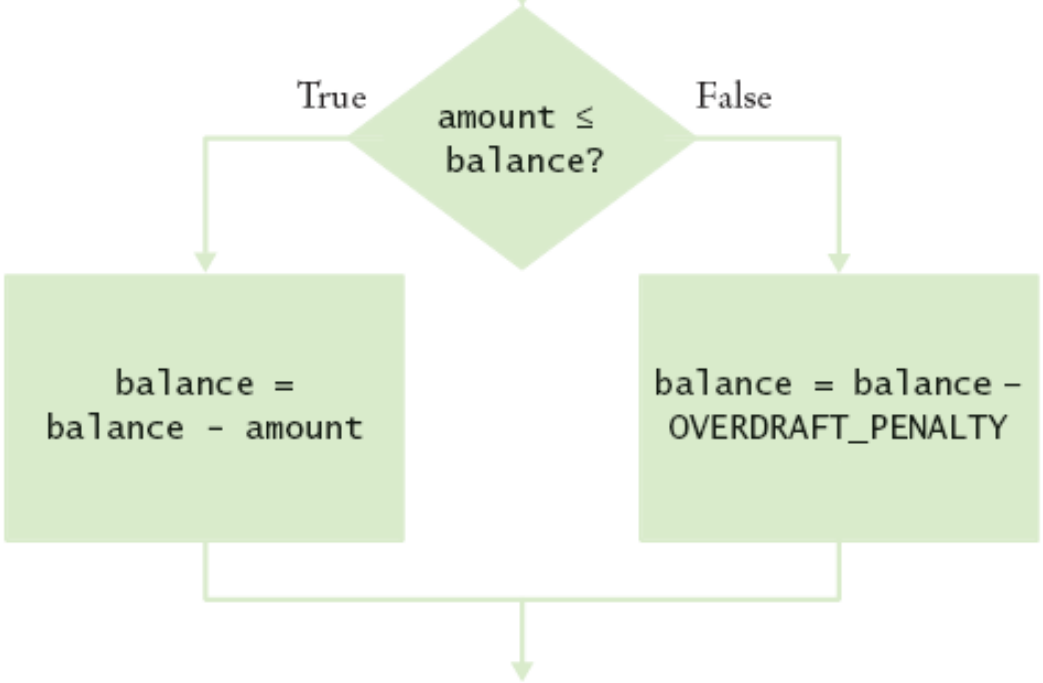

Figure 2
Flowchart for an `if/else` Statement

Tipi di istruzioni

- Istruzione semplice

```
balance = balance - amount;
```

- Istruzione composta

```
if (balance >= amount) balance = balance - amount;
```

ed anche

`while`, `for`, ecc. (istruzioni di ciclo)

- Blocco di istruzioni

```
{
   double newBalance = balance - amount;
   balance = newBalance;
}
```

Sintassi 5.1 L'istruzione `if`

```
if(condition)
   statement
if (condition)
   statement1
else
```

Esempio:

```
if (amount <= balance)
   balance = balance - amount;
if (amount <= balance)
   balance = balance - amount;
else
```

Obiettivo:

Eseguire un'istruzione quando una condizione è vera o falsa. Per eseguire più istruzioni queste vanno raggruppate in un blocco.

Sintassi 5.2 Blocco di istruzioni

```
{
   statement1
   statement2
   . . .
}
```

Esempio:

```
{
   double newBalance = balance - amount;
   balance = newBalance;
}
```

Scopo:

Raggruppare più istruzioni assieme per formare un’unica istruzione.

Confronto di valori: operatori relazionali

- Operatori relazionali confrontano valori

Java	Notazione matematica	Descrizione
>	>	Maggiore di
>=	≥	Maggiore od uguale a
<	<	Minore di
<=	≤	Minore od uguale a
==	=	Uguale
!=	≠	Non uguale

- Il simbolo == denota confronto di uguaglianza

```
a = 5; // Assign 5 to a
if (a == 5) . . . // Test whether a equals 5
```

Confronto di oggetti

- `==` verifica l'identità, `equals` l'identico contenuto

- ```
  Rectangle box1 = new Rectangle(5, 10, 20, 30);
  Rectangle box2 = box1;
  ```

- ```
  Rectangle box3 = new Rectangle(5, 10, 20, 30);
  ```
  `box1 == box3` rende `false`

- ma `box1.equals(box3)` rende `true`
  `box1 == box2` rende `true`

- Attenzione: `equals` deve essere un metodo definito per la classe, altrimenti non si può usare. Nel caso sopra il metodo equals è definito per la classe Rectangle (vedi documentazione API)

Confronto di oggetti

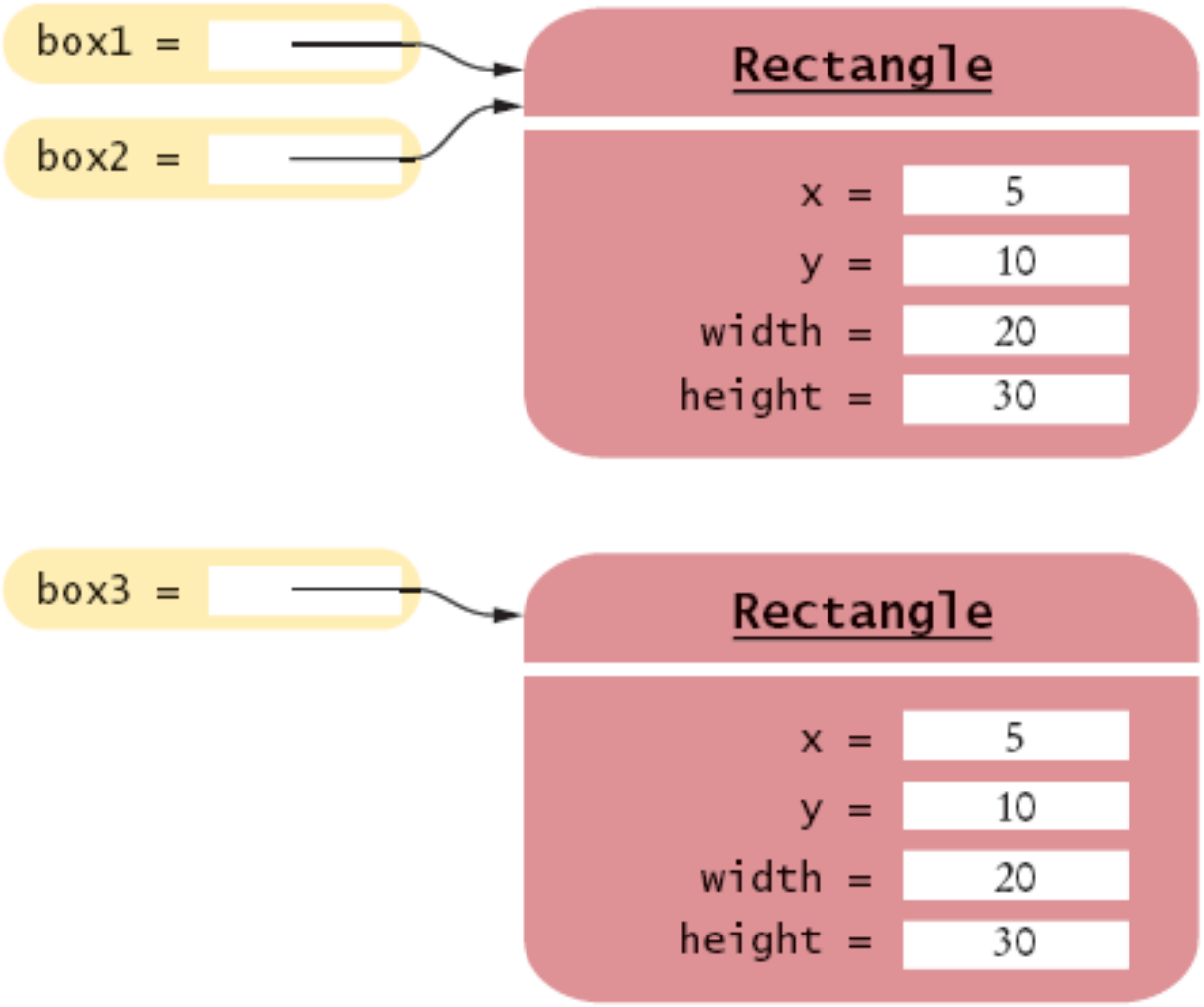

Figure 4 Comparing Object References

Confrontare con null

- `null` indica la mancanza di riferimento ad alcun oggetto

```
String middleInitial = null; // Not set
if ( . . . )
   middleInitial = middleName.substring(0, 1);
```

- Può essere impiegato in test:

```
if (middleInitial == null)
   System.out.println(firstName + " " + lastName);
else
   System.out.println(firstName + " " + middleInitial +
      ". " + lastName);
```

- Si usi `==`, non `equals`, per confronti con `null`
- `null` non è la stessa cosa della stringa vuota `""`

Alternative multiple: sequenze di confronti

```
if (condition1)
     statement1;
  else if (condition2)
     statement2;
     . . .
  else
     statement4;
```

- Viene eseguita l'istruzione associata alla prima condizione soddisfatta
- L'ordine ha importanza

```
if (richter >= 0) // always passes
   r = "Generally not felt by people";
else if (richter >= 3.5) // not tested
   r = "Felt by many people, no destruction";
. . .
```

Alternative multiple: if nidificati

- Più if uno interno all'altro

```
if (condition1)
{
   if (condition1a)
      statement1a;
   else
      statement1b;
}
else
   statement2;
```

Usare espressioni booleane: metodi predicativi

- Un metodo predicativo restituisce un valore boolean

```
public boolean isOverdrawn()
{
   return balance < 0;
}
```

- Si usa nelle condizioni, ad esempio

```
if (harrysChecking.isOverdrawn())
```

- Ci sono utili metodi predicativi nella classe `Character`:

```
isDigit
isLetter
isUpperCase
isLowerCase
```

Continua

Usare espressioni booleane: metodi predicativi (continua)

- `if (Character.isUpperCase(ch)) ...`
- Ci sono utili metodi predicativi nella classe `Scanner`:
  `hasNextInt() and hasNextDouble()`

```
if (in.hasNextInt()) n = in.nextInt();
```

Usare espressioni booleane: gli operatori booleani

- `&&` and
- `||` or
- `!` not
- `if (0 < amount && amount < 1000) . . .`
- `if (input.equals("S") || input.equals("M"))...`

Usare variabili booleane

- `private boolean married;`
- Assegnare valori di verità:
  `married = input.equals("M");`
- Da usare in condizioni:
  `if (married) . . . else . . . if (!married) . . .`
- Nelle condizioni sono anche chiamati *flag*
- E' considerato goffo scrivere un test nella forma
  `if (married == true) . . . // Don't`
- Si usi invece il test più semplice
  `if (married) . . .`