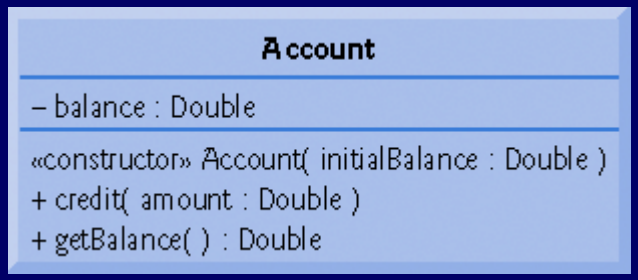
Another example class from Deitel

- Account.java (Fig. 3.13, p. 92)
  - Instance variable, `balance`, is type `double`
  - Method `credit` adds `amount` (also a `double`)
  - Also `getBalance`, but no setBalance
  - UML:



Static class diagram

- AccountTest.java uses 2 `Account` objects

Summary: Objects & classes

- An object represents a thing (e.g., window, spaceship) or a concept (e.g., power, trajectory)
  - A software entity with data and methods
    - i.e., more complex than just a number
  - Technically, an *instance* of a *class*
- A class defines a type of object
  - e.g., `class String` defines what a `String` knows (data), and what a `String` can do (methods)
  - Definition includes a *public* interface (what we use), and a *private* implementation (unimportant to user)

Wednesday:

1st Midterm Exam

Syntax for invoking methods

- Essentially: `methodName(`*list of arguments*`);`
  - Effect – transfers control to the method named; may "pass" data via the list of arguments
  - When method completes – control returns to the point in the program where the method was called
    - Also returns a result if not a `void` method
- Need more if method defined in a different class
  - Full syntax is `objectReference.methodName(…)`
  - Or just `ClassName.name()` if method is `static`

Aside – using dialog boxes

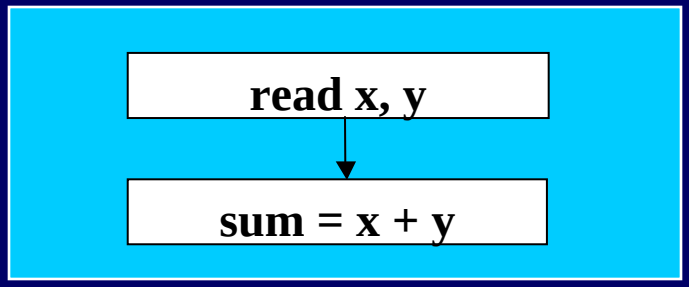
- Simplest type of GUI (Graphical User Interface)
  - Import `javax.swing.JOptionPane`
- Message dialogs – show user something
  - e.g., a `String` (and other types of objects)
- Input dialogs – get a `String` from the user
  - Must parse string to convert to numbers/other
- See NameDialog.java (Fig 3.18, p. 97)

  - and try GUI/Graphics Case Study Exercise 3.1 (p. 98)

Planning before programming

- Algorithms – easier to change than programs
- Pseudocode or flowcharts:

```
get x and y from user
calculate sum
```

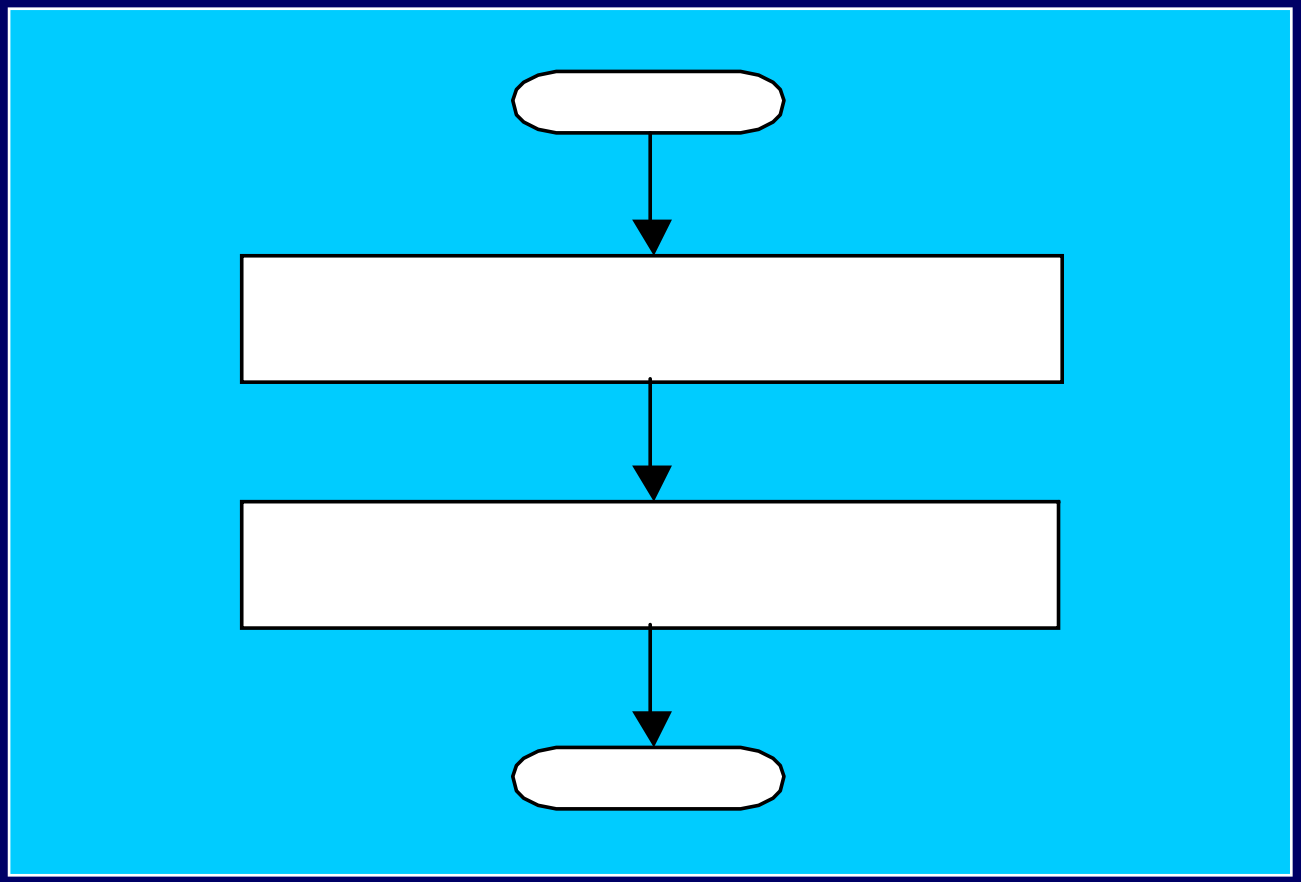


- Plan in terms of control structures
  - i.e., no "`go to`" logic
  - A fact: *any algorithm* can be written as a combination of sequence, selection, &/or iteration structures

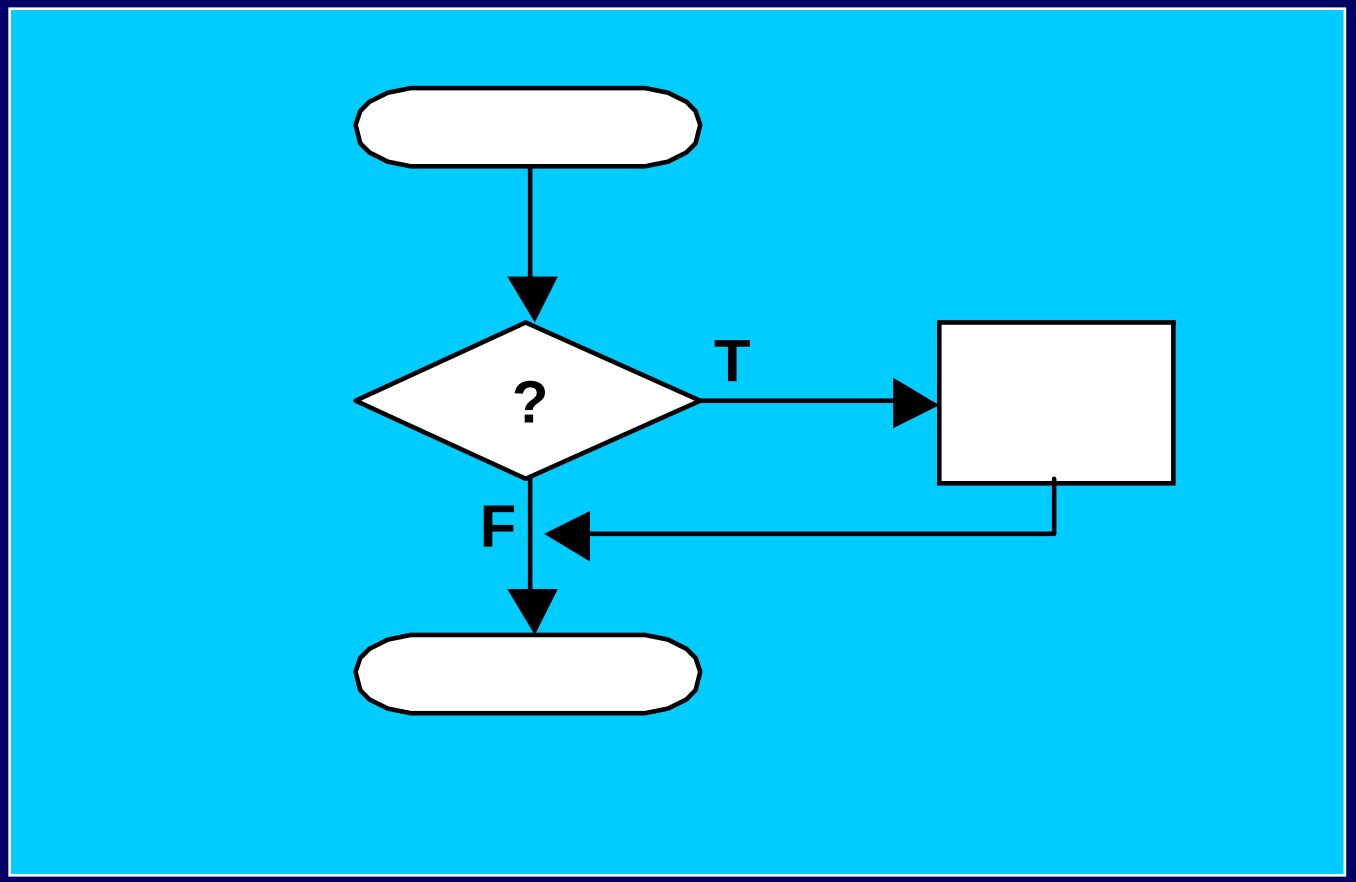
Java has 7 control structures

- 1st is trivial: sequence structure
- 3 choices of selection structures:
  - `if`
  - `if/else`
  - `switch`
- 3 choices of iteration structures ("loops"):
  - `while`
  - `for`
  - `do/while`

Sequence (it really is a structure)



if Selection Structure



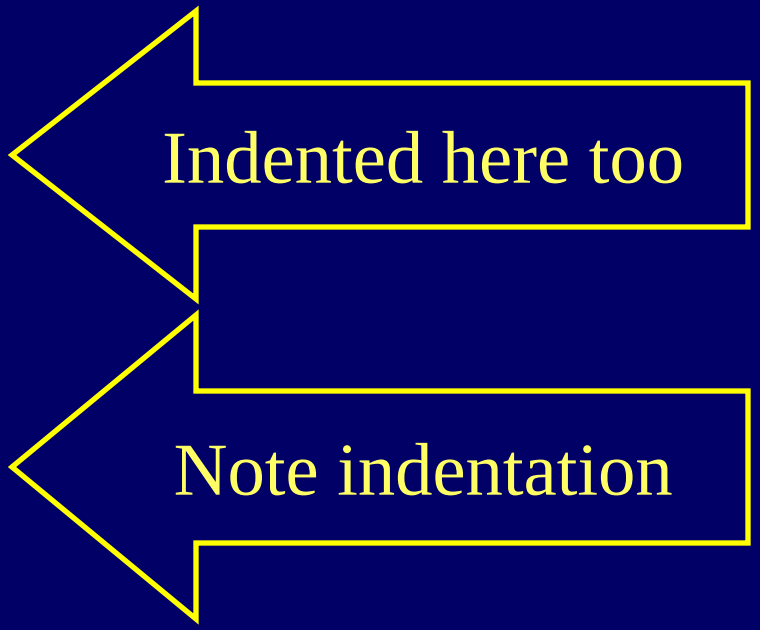
Implementing `if`

Either

```
if (boolean expression)
    one statement;
```

Or

```
if (boolean expression) {
    multiple statements;
     separated by ;
}
```



- `boolean` expressions: evaluate to `true` or `false`