15th Edition

Understanding Computers

Today and Tomorrow

Comprehensive

Chapter 13:

Program Development and Programming Languages

Deborah Morley
Charles S. Parker

Copyright 2015 Cengage Learning

Learning Objectives

1. Understand the differences between structured programming, object-oriented programming (OOP), aspect-oriented programming (AOP), and adaptive software development.
2. Identify and describe the activities involved in the program development life cycle (PDLC).
3. Understand what constitutes good program design and list several tools that can be used by computer professionals when designing a program.
4. Explain the three basic control structures and how they can be used to control program flow during execution.

Learning Objectives

4. Discuss some of the activities involved with debugging a program and otherwise ensuring it is designed and written properly.
5. List some tools that can be used to speed up or otherwise facilitate program development.
6. Describe several programming languages in use today and explain their key features.

Overview

- This chapter covers:
  - The most common approaches to program design and development
  - The phases of the program development life cycle (PDLC)
  - Tools that can be used to design and develop a program
  - Good program design techniques and types of program errors
  - Popular programming languages

Approaches to Program Design and Development

- Procedural Programming
  - An approach to program design in which a program is separated into small modules that are called by the main program or another module when needed
    - Procedure call—locating specific tasks in procedures (modules or subprograms) that are called by the main program when needed
    - Allows each procedure to be performed as many times as needed; multiple copies of code not needed
    - Prior to procedural programming, programs were one large set of instructions (used GOTO statements)

Approaches to Program Design and Development

- Structured Programming
  - Goes even further, breaking the program into small modules (Top-down design)
- Variables
  - Named memory locations that are defined for a program
  - Used to store the current value of data items used in the program

Approaches to Program Design and Development

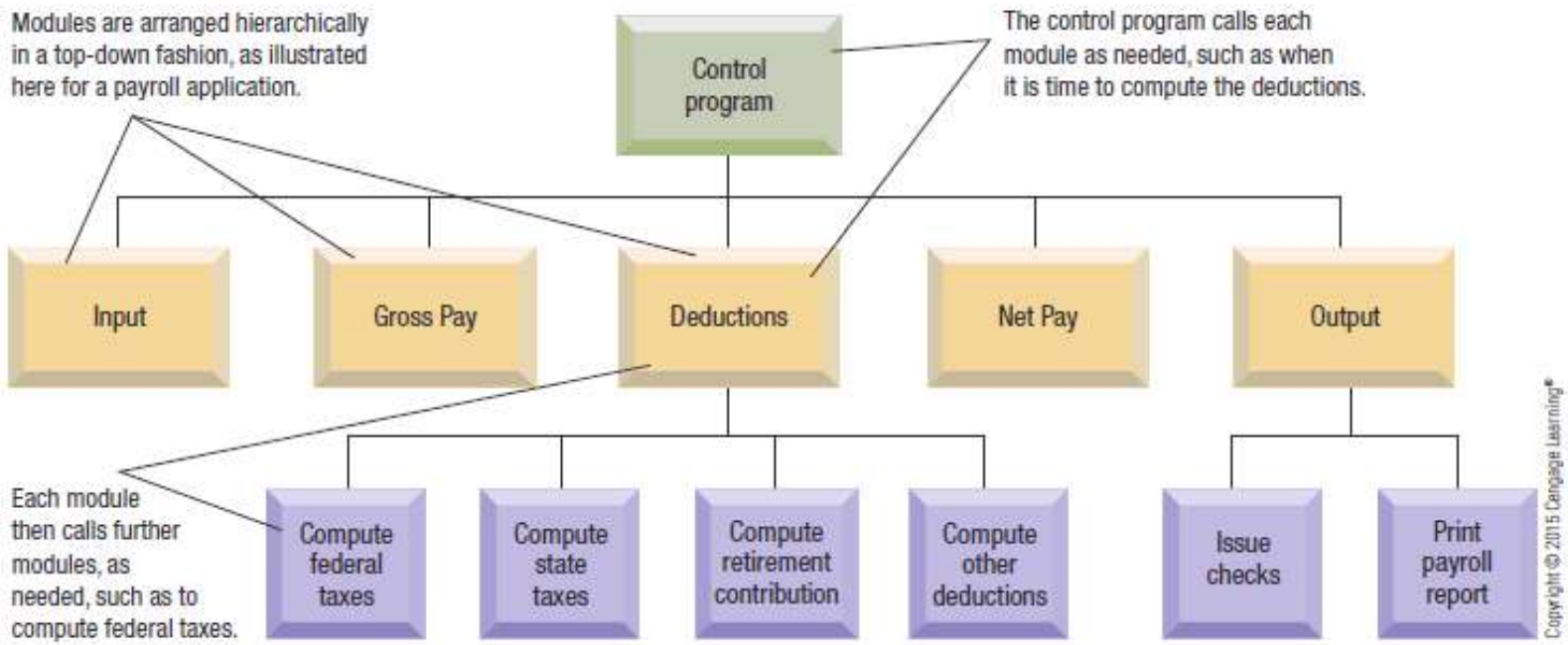


FIGURE 13-1
Structured programming.
A structured program is divided into individual modules; each module represents a very specific processing task.

Approaches to Program Design and Development

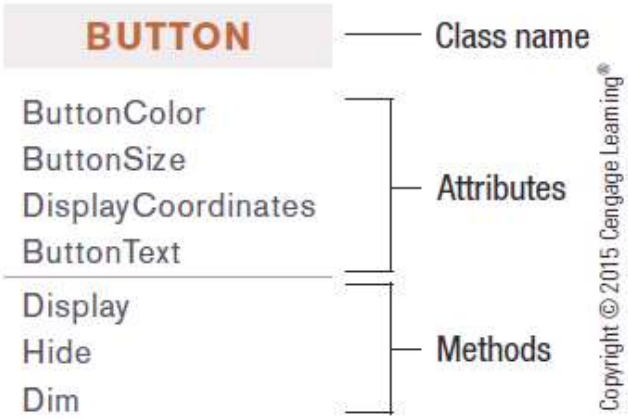
- Object-Oriented Programming (OOP)
  - Programs consist of a collection of objects that contain data and methods to be used with that data
    - Class
      - Group of objects that share some common properties
    - Instance
      - An individual object in a class
      - Inherits the attributes and methods of the class

Approaches to Program Design and Development

- Attributes
  - Data that describes the object
  - Can be in a variety of formats
- Methods
  - Perform actions on an object
  - Can be used with different types of objects

– Objects can be accessed by multiple programs
  - Class libraries

FIGURE 13-2
Button class.
This class diagram illustrates that each object (instance) in the Button class has four attributes to hold data about the current state of the button and three methods to perform actions when messages are received.



Approaches to Program Design and Development

- Aspect-Oriented Programming (AOP)
  - Separates functions so program components can be developed and modified individually from one another
  - The components can be easily reused with separate nonrelated objects
- Adaptive Software Development
  - Designed to make program development faster and more efficient and focuses on adapting the program as it is being written
    - Features iterative and/or incremental development

Approaches to Program Design and Development

- Agile Software Development
  - Goal is to create software quickly
  - Focuses on building small functional program pieces as the project progresses
  - Emphasizes teams of people working closely together (programmers, managers, business experts, customers, and so forth)
  - Some mobile developers are using continuous mobile innovation

The Program Development Life Cycle (PDLC)

- Program Development (application software development)
  - The process of creating application programs
- Program Development Life Cycle (PDLC)
  - The five phases of program development

The Program Development Life Cycle (PDLC)

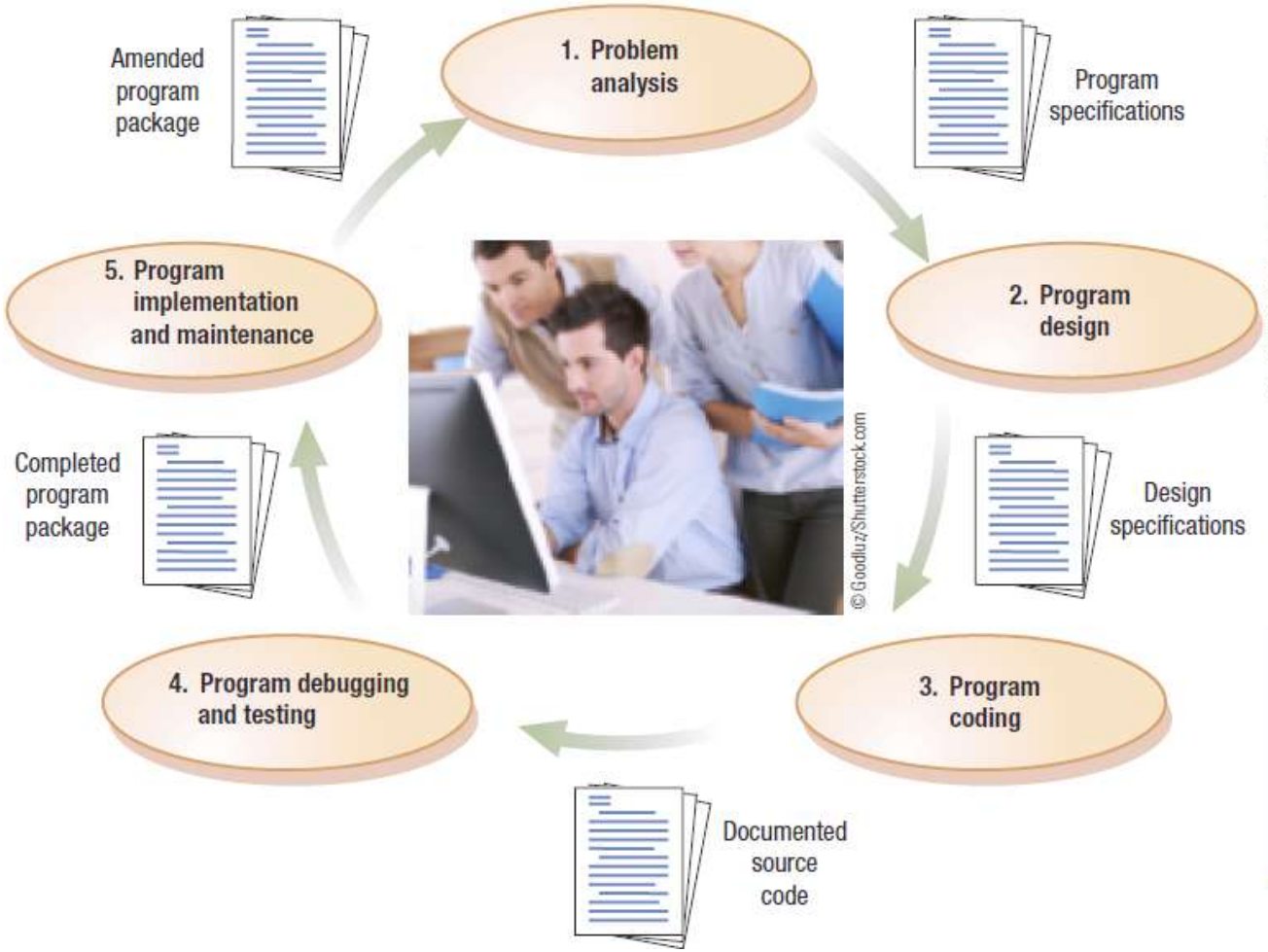


FIGURE 13-3
The program development life cycle (PDLC). Each phase of the program development life cycle produces some type of documentation to pass on to the next phase.

The Program Development Life Cycle (PDLC)

- Problem Analysis
  - The problem is considered and the program specifications are developed
    - Specifications developed during the PDLC are reviewed by the systems analyst and the programmer (the person who will code the program)
    - Goal is to understand the functions the software must perform
  - Documentation: Program Specifications
    - Result of the first phase of the PDLC outlining what the program must do

The Program Development Life Cycle (PDLC)

- Program Design
  - The program specifications are expanded into a complete design of the new program
    - Algorithm for the program is developed
    - Careful planning and design of a computer program are extremely important
  - Program Design Tools
    - Planning tools that include diagrams, charts, tables, and models
    - Structure Charts (hierarchy charts)
      - Depict the overall organization of a program

The Program Development Life Cycle (PDLC)

- Flowcharts
  - Show graphically, step-by-step, the actions a computer program will take
  - Use special symbols and relational operators
  - Can be drawn by hand or with flowcharting software

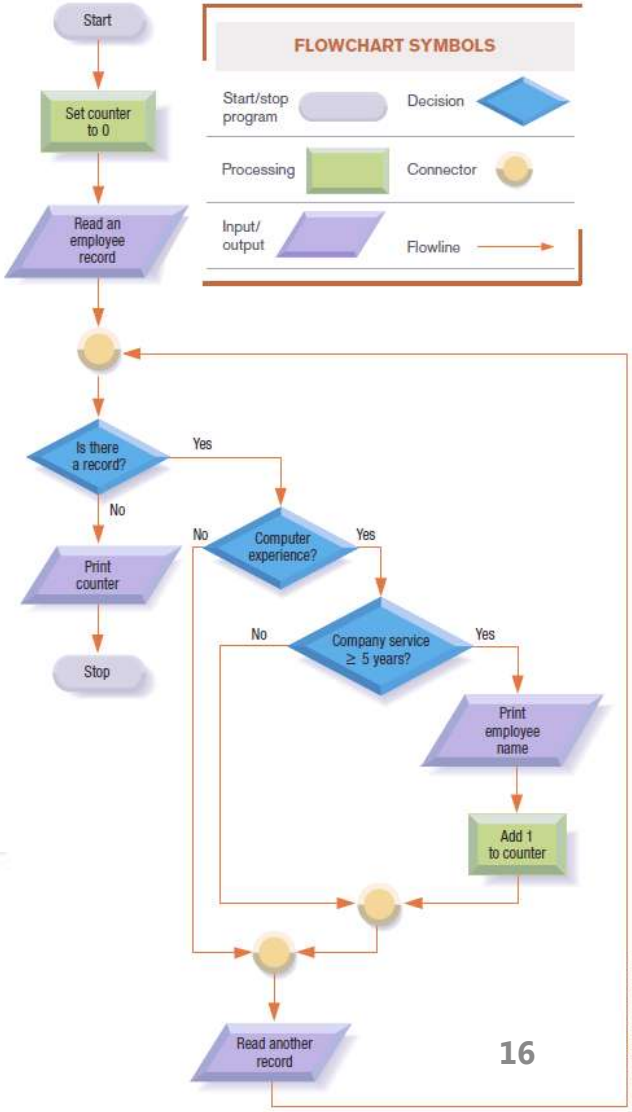


FIGURE 13-4
A flowchart example.

Copyright © 2015 Cengage Learning®

The Program Development Life Cycle (PDLC)

- Wireframes
  - Visual representation of the overall design and logic of an app or Web site

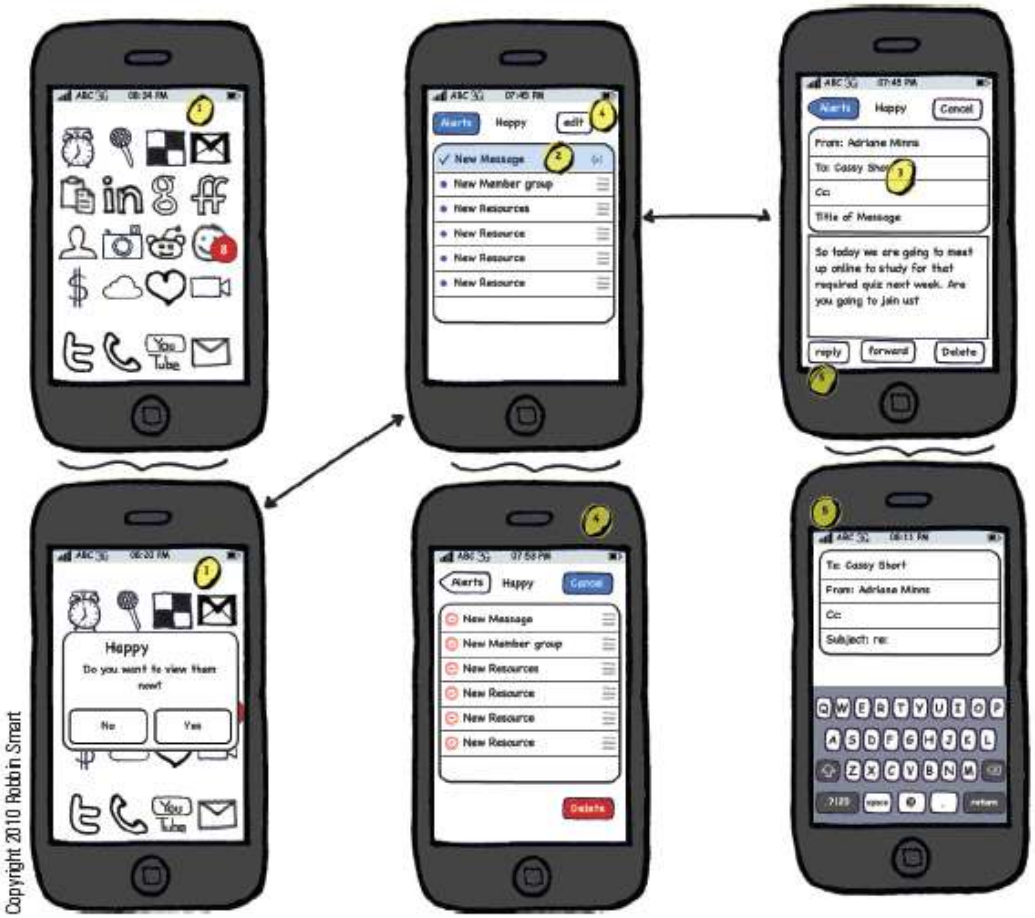


FIGURE 13-5
Wireframes.

The Program Development Life Cycle (PDLC)

- Pseudocode
  - Uses English-like statements to outline the logic of a program rather than the flowchart's graphical symbols

```
Start
counter = 0
Read a record
DO WHILE there are records to process
   IF computer_experience
      IF company_service ≥ 5 years
         Print employee_name
         Increment counter
      ELSE
         Next statement
      END IF
   ELSE
      Next statement
   END IF
   Read another record
END DO
Print counter
Stop
```

Copyright © 2015 Cengage Learning®

FIGURE 13-6

Pseudocode. For the flowchart logic shown in Figure 13-4.

The Program Development Life Cycle (PDLC)

- Unified Modeling Language (UML) Models
  - Set of standard notations for creating business models
  - Widely used in object-oriented programs
  - Includes class diagrams and case diagrams

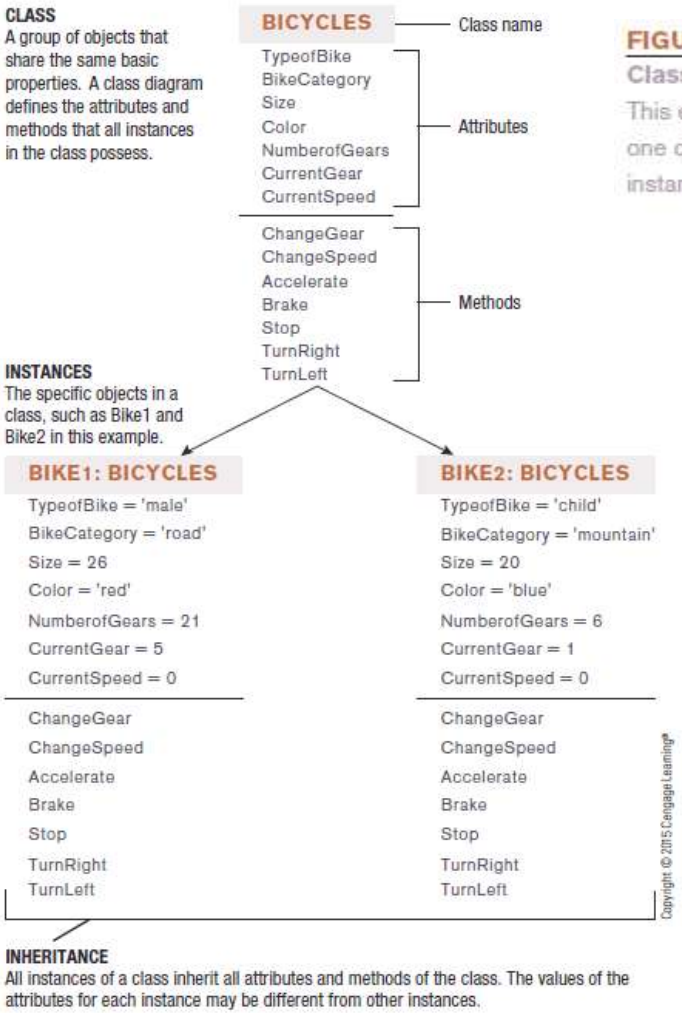


FIGURE 13-7
Class diagrams. This example shows one class and two instances of that class.

The Program Development Life Cycle (PDLC)

- Control Structures
  - A pattern for controlling the flow of logic in a computer program, module, or method
  - The Sequence Control Structure
    - Series of statements that follow one another
  - The Selection Control Structure
    - Multiple paths, direction depends on result of a certain condition
      - » If-then-else
      - » Case control structure

The Program Development Life Cycle (PDLC)

- Repetition Control Structure (iteration control structure)
  - Series of statements in a loop that are repeated until a particular condition is met
  - Two forms
    - » Do while structure
    - » Do until structure

The Program Development Life Cycle (PDLC)

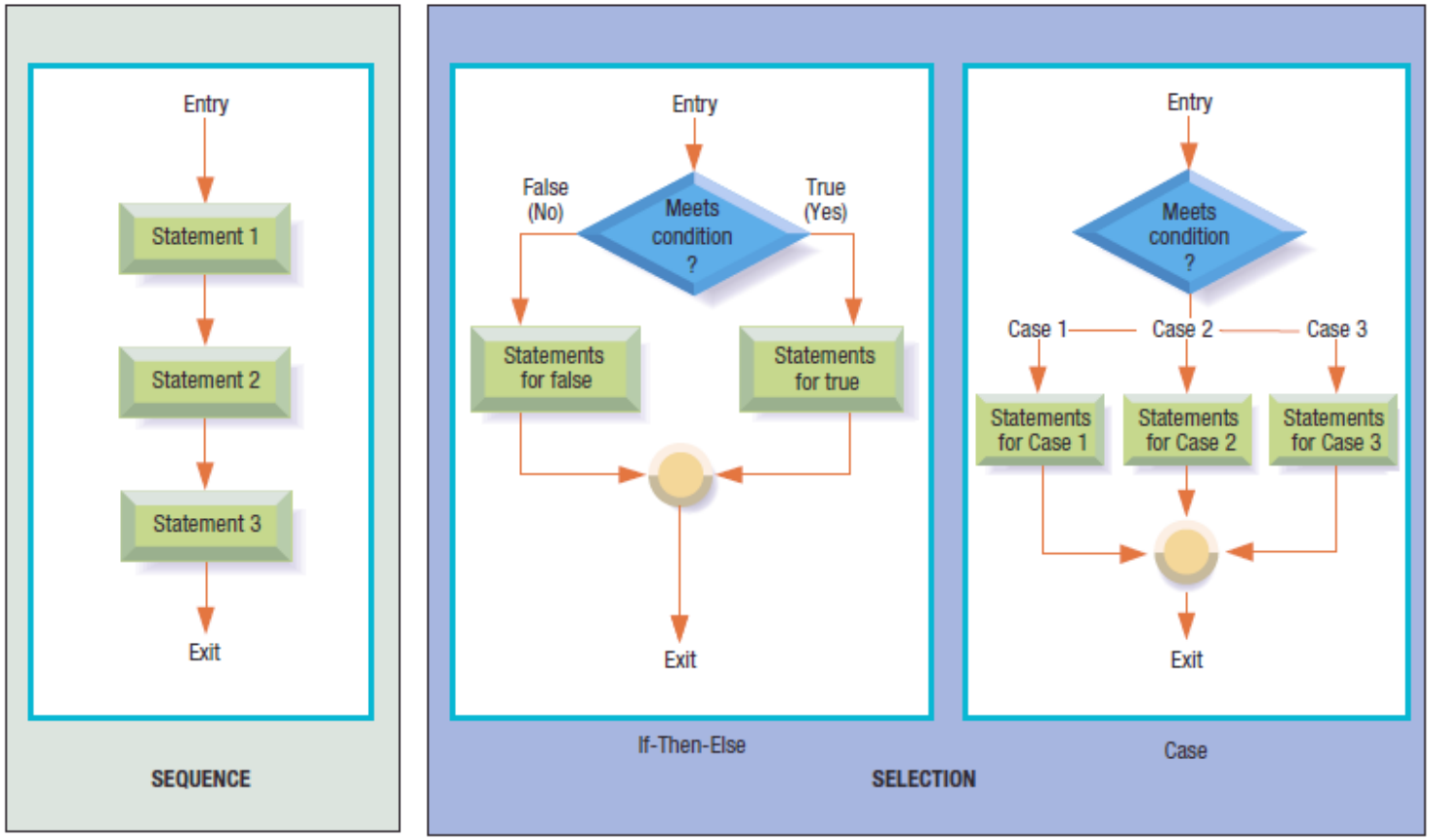


FIGURE 13-8
The three fundamental control structures. Note that each structure has only one entry point and only one exit point.

The Program Development Life Cycle (PDLC)

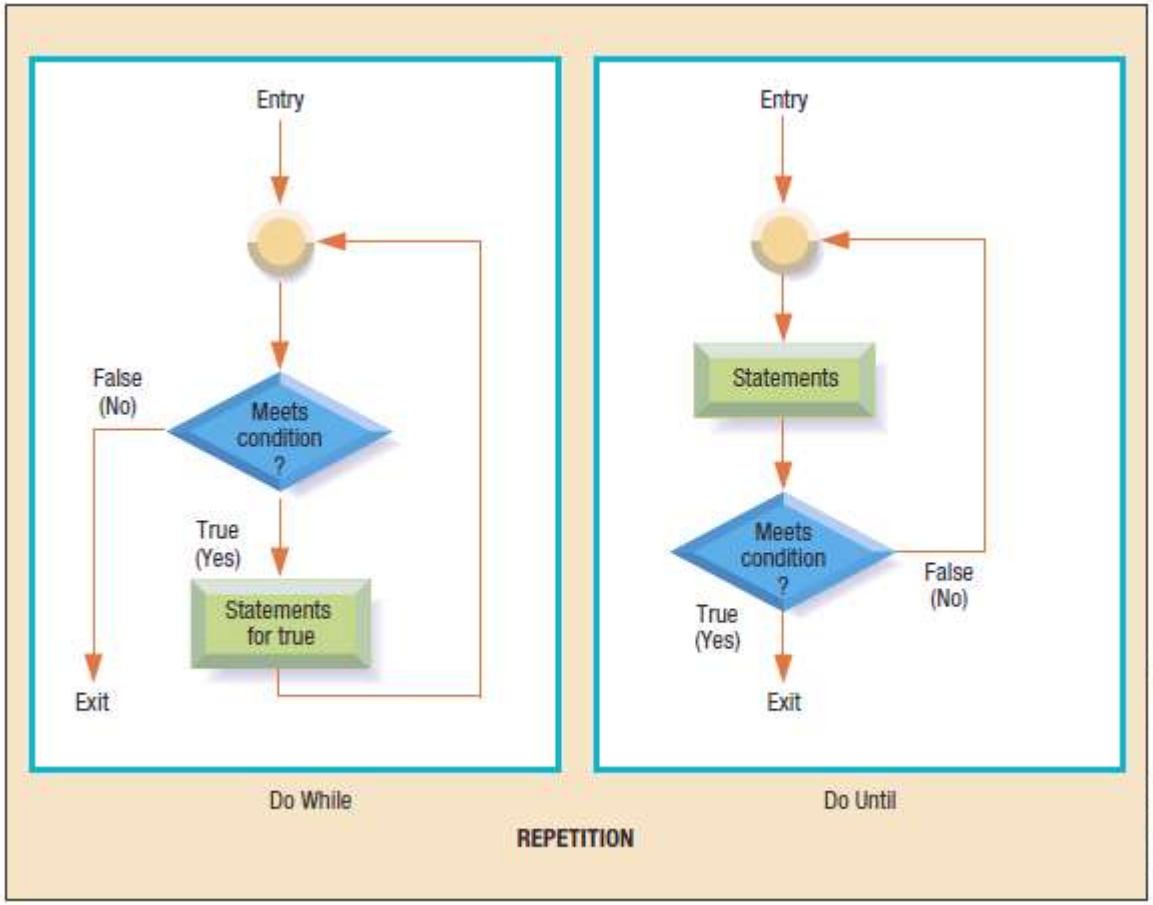


FIGURE 13-8
The three fundamental control structures. Note that each structure has only one entry point and only one exit point.

Copyright © 2015 Cengage Learning®

The Program Development Life Cycle (PDLC)

- Good Program Design
  - Is essential
  - Saves time
  - Good Program Design Principles
    - Be Specific
      - » All things the program must do or consider must be specified
    - Follow the One-Entry-Point/One-Exit-Point Rule
    - No Infinite Loops or Logic Errors
      - » Infinite loop is a series of steps that repeat forever

The Program Development Life Cycle (PDLC)

DIRECTIONS FOR PERSON	DIRECTIONS FOR COMPUTER
1. Please make a piece of toast with margarine for me.	1. Take one slice of bread out of the bag of bread. 2. Put the bread into the slot on the toaster, narrow edge first with the widest part of the bread fitting into the widest part of the slot. 3. Push the start lever on the toaster down. 4. When the toast pops up, remove the toast from the toaster. 5. Place the toast on a plate. 6. Open the silverware drawer, take out a knife, then close the silverware drawer. 7. Open the refrigerator, remove the margarine, then shut the refrigerator. 8. Take the lid off the margarine. 9. Scoop out one teaspoon of margarine with the knife. 10. Spread the margarine on the top side of the toast, evenly covering that surface of the toast. 11. Place the lid back on the margarine. 12. Open the refrigerator, replace the margarine, then shut the refrigerator.

© Les Scholz/Shutterstock.com; Copyright © 2015 Cengage Learning®

FIGURE 13-9
Writing instructions for a computer versus a person.
A computer requires step-by-step instructions.

The Program Development Life Cycle (PDLC)

- Program Design Testing
  - Design should be tested to ensure logic is correct
    - Desk check
    - Tracing tables
- Documentation: Design Specifications
  - Illustrates the program needed to fulfill the program requirements
  - Expressed using structure charts, flowcharts, wireframes, pseudocode, and UML models
  - Include any test data and results from desk checking

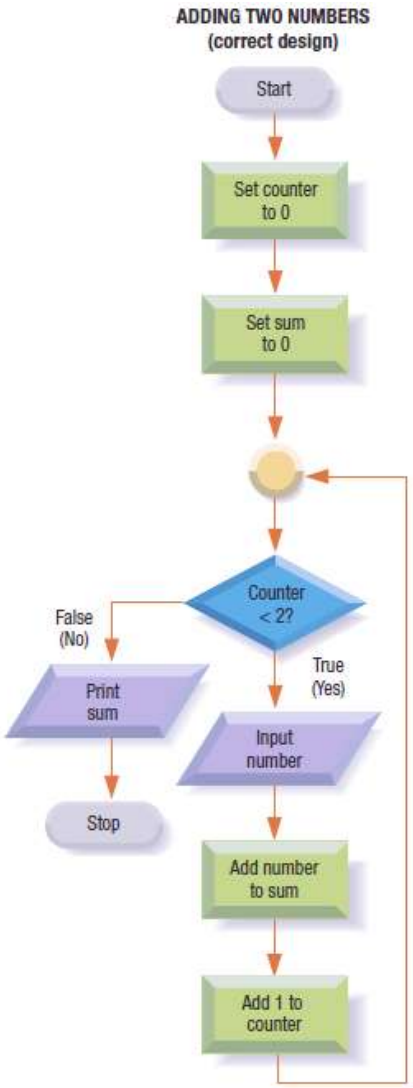
The Program Development Life Cycle (PDLC)

DESK CHECK RESULTS FOR CORRECT FLOWCHART

Flowchart Stage	Counter	Decision Test Results (Counter < 2)	Number	Sum
Initialization	0	–	–	0
First decision test	0	T (enters loop)	–	0
After first loop	1	–	6	6
Second decision test	1	T (enters loop)	6	6
After second loop	2	–	3	9
Third decision test	2	F (exits loop)	3	9

Test data: 6, 3; Expected results: Sum = 9; Actual results: Sum = 9

FIGURE 13-10
Desk checking a flowchart.



The Program Development Life Cycle (PDLC)

DESK CHECK RESULTS FOR INCORRECT FLOWCHART

Flowchart Stage	Counter	Decision Test Results (Counter<2)	Number	Sum
Initialization	1	–	–	0
First decision test	1	T (enters loop)	–	0
After first loop	2	–	6	6
Second decision test	2	F (exits loop)	6	6

Test data: 6, 3; Expected results: Sum = 9; Actual results: Sum = 6

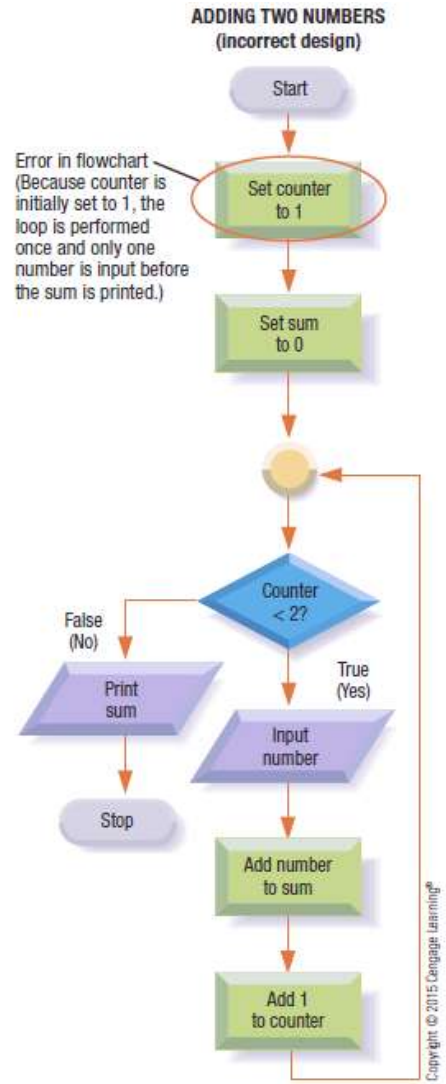


FIGURE 13-10
Desk checking a flowchart.

The Program Development Life Cycle (PDLC)

- Program Coding
  - The program code is written using a programming language
  - Choosing a Programming Language
    - Suitability to the application
    - Integration with other programs
    - Standards for the company
    - Programmer availability
    - Portability if being run on multiple platforms
    - Development speed

The Program Development Life Cycle (PDLC)

- The Coding Process
  - The source code is the computer program before it is compiled
- Coding Standards
  - Rules designed to standardize programming
  - Makes programs more readable and easier to maintain
  - Includes the proper use of comments to:
    - » Identify the programmer and last modification date
    - » Explain variables used in the program
    - » Identify the main parts of the program

The Program Development Life Cycle (PDLC)

COMMENTS
Comments are usually preceded by a specific symbol (such as *, C, ', #, or //); the symbol used depends on the programming language being used. Anything else in a comment line is ignored by the computer.

Comments at the top of a program should identify the name and author of the program, date written and last modified, purpose of the program, and variables used in the program.

Comments in the main part of a program should indicate what each section of the program is doing. Blank comment lines can also be used to space out the lines of code, as needed for readability.

```
*****************************************************************
* This program inputs two numbers, computes their sum,          *
* and displays the sum.                                         *
*                                                               *
* Written by: Deborah Morley  3/12/14                           *
*****************************************************************
* Variable list                                                 *
* SUM: Running sum                                              *
* CNTR: Counter                                                 *
* NUM: Number inputted                                          *
*
      REAL SUM, CNTR, NUM
*****************************************************************
*
* INITIALIZE VARIABLES
      SUM = 0
      CNTR= 0
*
* INPUT NUMBER, ADD IT TO THE SUM, INCREMENT COUNTER, AND THEN
* REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED
      DO 10 CNTR = 1, 2
```

Copyright © 2015 Cengage Learning®

FIGURE 13-11
Program comments.

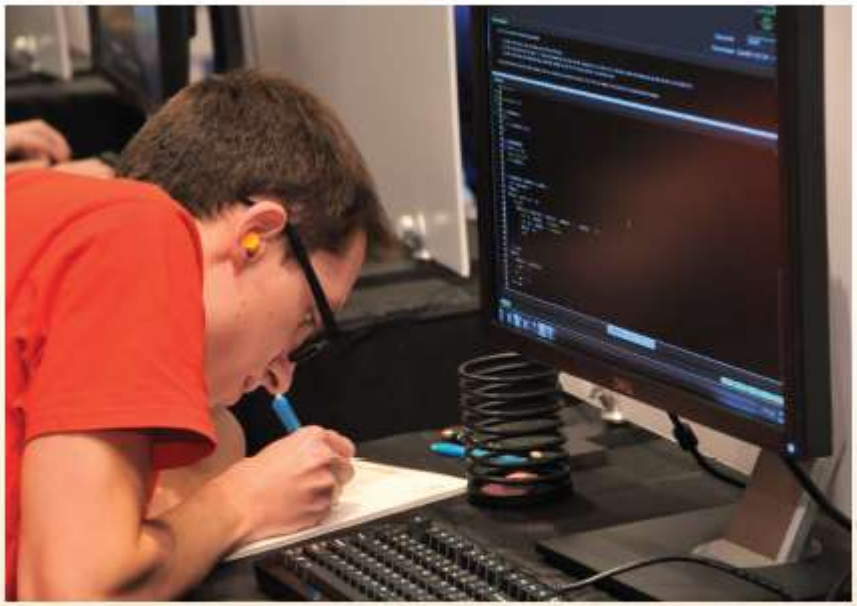
The Program Development Life Cycle (PDLC)

- Reusable code
  - Pretested, error-free code segments that can be used over and over again with minor modifications
  - Can greatly reduce development time
- Documentation: Documented Source Code
  - Program coding phase results in the program written in the desired programming language
  - Should include enough comments (internal documentation) so that the source code is easy to understand and update

Technology and You Box

Programming Contests

- One example is the TopCoder Open
  - Six competitions
  - Initial qualifying rounds are online
  - 48 semifinalists compete on site
  - $300,000 in prizes
  - Other competitions are available online



A semifinalist competing in the TopCoder Open Algorithm contest.

The Program Development Life Cycle (PDLC)

- Program Debugging and Testing
  - The process of ensuring a program is free of errors (bugs) and works as it is supposed to
  - Translating Coded Programs into Executable Code
    - Coded programs need to be translated from source code written by the programmer to object code the computer can execute
    - Converted using a language translator
      - Program that converts source code to object code

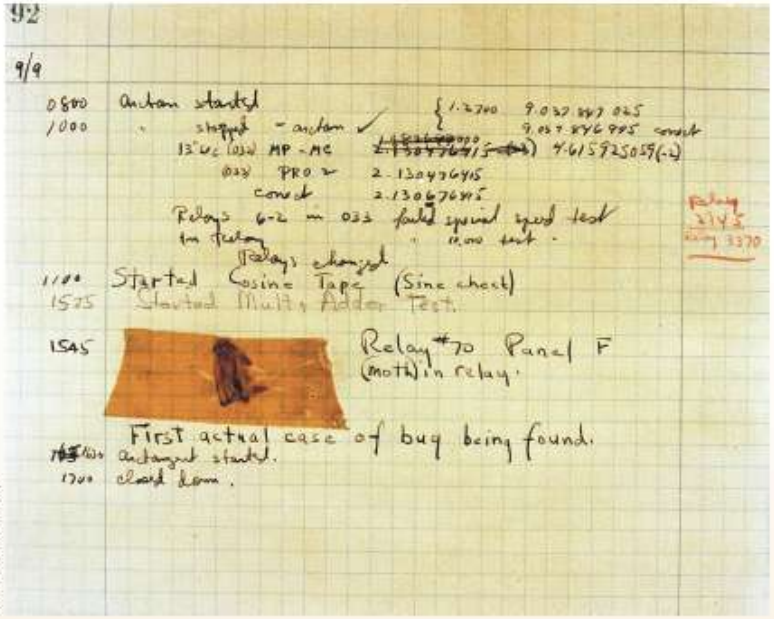
The Program Development Life Cycle (PDLC)

- Compilers
  - Language translator that converts an entire program into machine language before executing it
  - Designed for specific programming languages such as Java or Python
- Interpreters
  - Translates one line of code at one time
- Assemblers
  - Convert assembly language programs into machine language

Inside the Industry Box

The Original Program "Bug"

- A bug is an error that causes a program to malfunction
- First recorded instance of the term "bug" occurred in 1945
- Short circuit caused by a moth caught between two contacts in one of the computer's relays



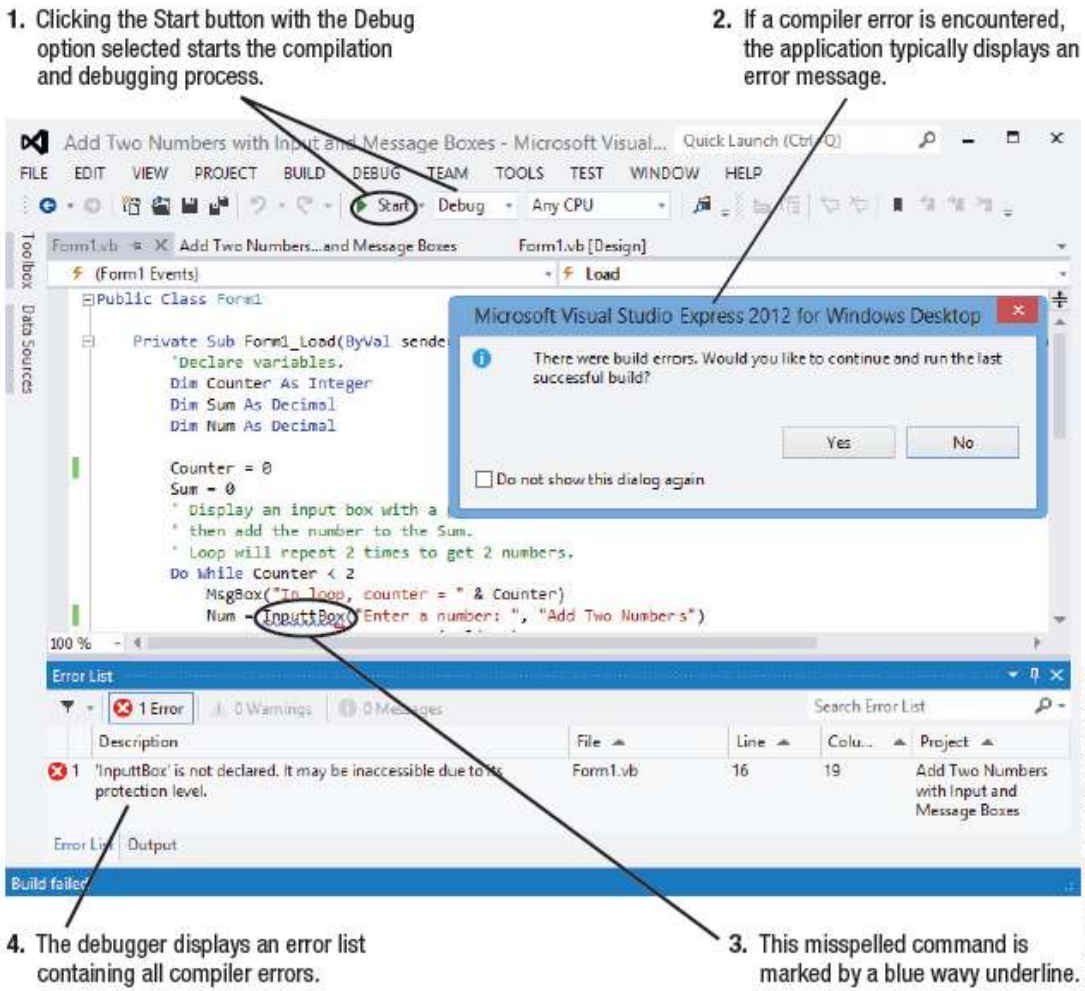
Courtesy U.S. Navy

The dead moth that caused the temporary failure of the Mark II computer in 1945, thought to be the origin for the computer term *bug*, was taped into the actual log book for that computer.

The Program Development Life Cycle (PDLC)

- Preliminary Debugging
  - Compiler and Syntax Errors
    - As programs are compiled or interpreted, errors occur which prevent the program from running properly
    - Syntax errors occur when the programmer has not followed the rules of the programming language
  - Run Time and Logic Errors
    - Run time errors occur when the program is running
    - Logic errors are errors in the logic of the program
      - » Program will run but produces incorrect results

The Program Development Life Cycle (PDLC)



Used with permission from Microsoft Corporation

FIGURE 13-13

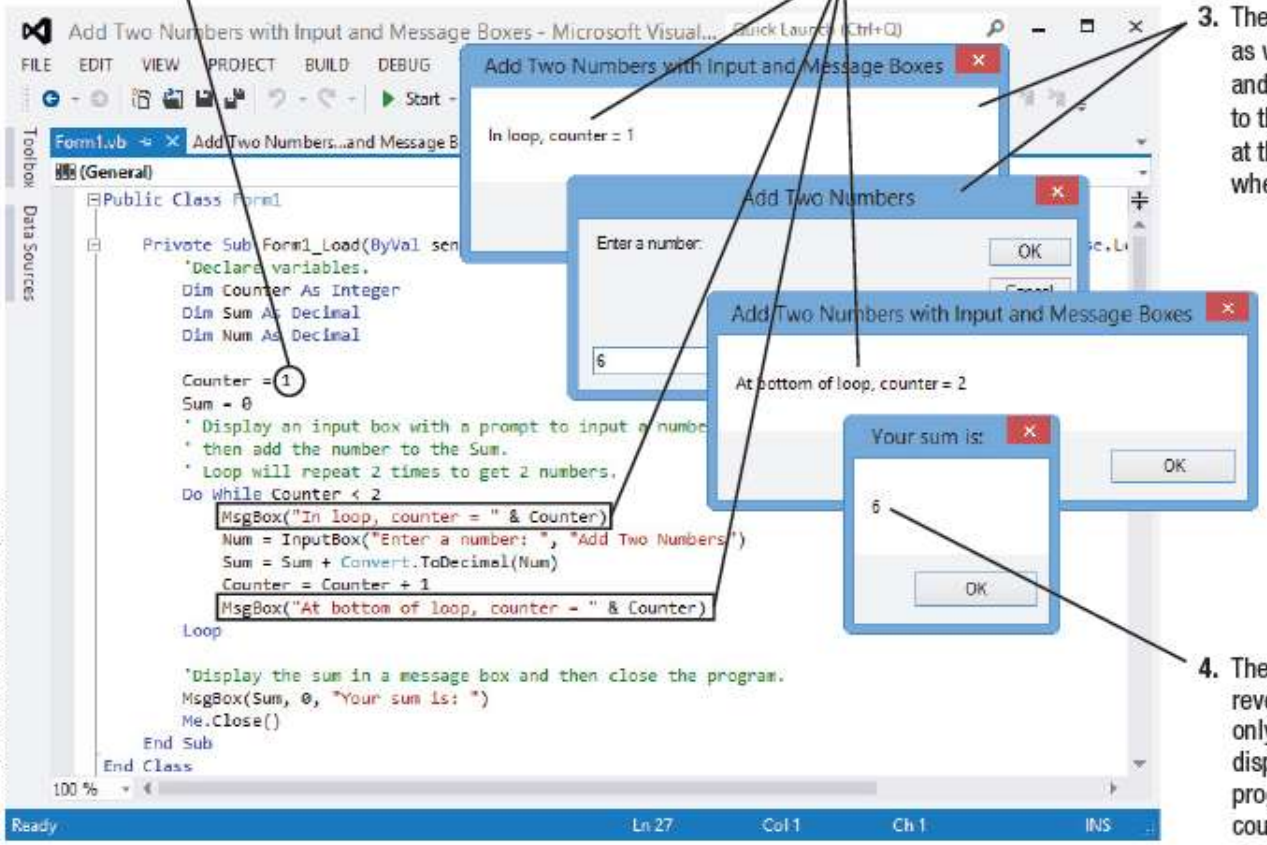
Syntax errors. Occur when the syntax (grammar rules) for a program is not followed precisely; they become obvious when compiling a program.

The Program Development Life Cycle (PDLC)

1. With logic errors, such as initializing a counter to the wrong number as shown here, the program will run but the output will be wrong.

2. Adding dummy print statements to display the values of key variables and key locations in the program can help to determine the error.

3. The dummy print statements, as well as the regular input and output messages belonging to the program, are displayed at the appropriate times when the program is executed.



4. The dummy print statements reveal that the loop is performed only once before the sum is displayed and help the programmer locate the counter initialization error.

FIGURE 13-14

Logic errors. Are more difficult to identify; dummy print statements can help determine the error.

Used with permission from Microsoft Corporation

The Program Development Life Cycle (PDLC)

- Testing
  - Occurs after the preliminary debugging process to find additional errors
  - Uses good test data—data that is very similar to the actual data that will be used in the finished program
  - Tests conditions that will occur when the program is implemented
  - Checks for nonstandard situations or possible input errors

The Program Development Life Cycle (PDLC)

- Two stages
  - Alpha test—internal on-site test
  - Beta test—outside test

– Documentation: Completed Program Package

- Copy of the test data, test results, finished program code, and other documentation generated during the testing phase should be added to the program package
  - Developer documentation
  - User documentation

The Program Development Life Cycle (PDLC)

- Program Implementation and Maintenance
  - Once the system containing the program is up and running, the implementation process is complete
  - Program maintenance
    - Process of updating software so it continues to be useful
    - Very costly
  - Documentation: Amended program package
    - Program package should be updated to reflect new problems or issues that occur and what changes to the program were necessary

Quick Quiz

1. Which approach to programming uses the concept of inheritance?
   a. Procedural
   b. Object-oriented
   c. Aspect-oriented
2. True or False: An infinite loop is an example of a logic error.
3. A(n) _____ is a program design tool that shows graphically step-by-step the actions a computer program will take.

Answers:

1) b; 2) True; 3) flowchart

Tools for Facilitating Program Development

- Application Lifecycle Management (ALM) Tools
  - Creating and managing an application during its entire lifecycle, from design through retirement
  - Tools include:
    - Requirements management
      - Keeping track of and managing the program requirements as they are defined and then modified
    - Configuration management
      - Keeping track of the progress of a program development project

Tools for Facilitating Program Development

- Issue tracking
  - Recording issues such as bugs or other problems that arise during development or after the system is in place

- Application Generators
  - Software program that helps programmers develop software
  - Macros
    - Sequence of saved actions that can be replayed when needed
    - Programmers write them in a macro programming language such as Visual Basic for Applications

Tools for Facilitating Program Development

- Report Generators and User Interface (UI) Builders
  - Report generator
    - Tool that prepares reports to be used with a software program quickly and easily
  - User interface (UI) builders
    - Create the menus, forms, and input screens used with a program or database
  - Integrated development environment (IDE)
    - A set of programming tools for writing software applications

Tools for Facilitating Program Development

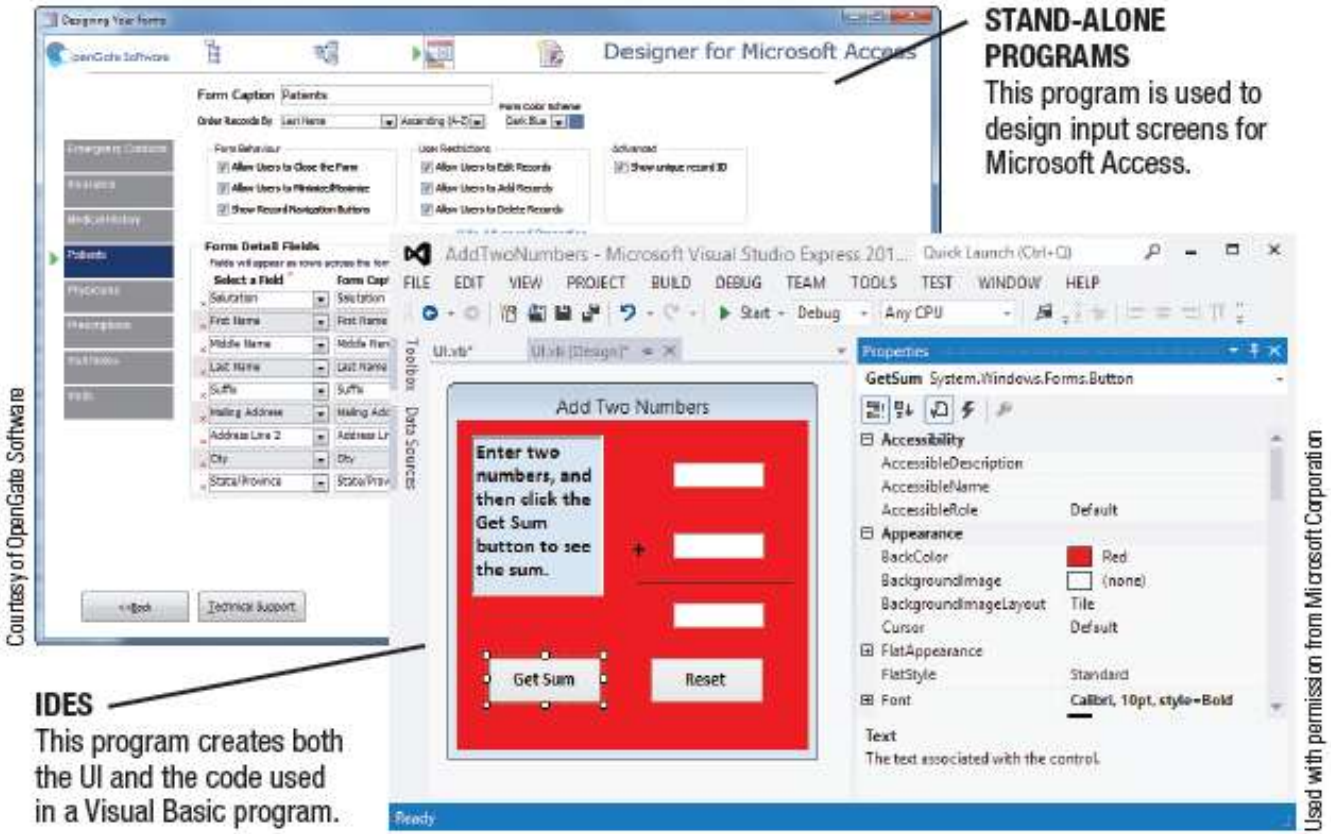


FIGURE 13-16
User interface (UI) builders.

Tools for Facilitating Program Development

- Device Development Tools
  - Assist with developing embedded software to be used on devices, such as cars, ATM machines, and consumer devices
- Integrated Development Environments (IDEs)
  - Collection of tools used with a particular programming language to develop and test software
- Software Development Kits (SDKs)
  - Programming package designed for a particular platform
  - Enables programmers to develop applications for that platform more quickly and easily

Tools for Facilitating Program Development

- Application Program Interfaces (APIs)
  - Help applications interface with a particular operating system
  - Often used in conjunction with Web sites
  - Google's Maps API and Google's OpenSocial API allow developers to add Google Maps or social networking applications easily to Web sites, respectively

Trend Box

Mobile App Builders

- Many tools are available to help develop mobile apps and deploy them on various platforms
- One example is appsbar
- After the app is created, appsbar tests it and then submits it to major app markets for publication

Quick Quiz

1. Which of the following is not an Application Lifecycle Management (ALM) tool?
   a. Requirements definition software
   b. Code generator
   c. Application program interface (API)
2. True or False: A software development kit (SDK) is designed for a particular platform and allows programmers to develop applications quickly for that platform.
3. A(n) _____ is a sequence of saved actions (such as keystrokes, mouse clicks, and menu selections) that can be replayed whenever needed within the application program in which it was created.

Answers:
1) c; 2) True; 3) macro

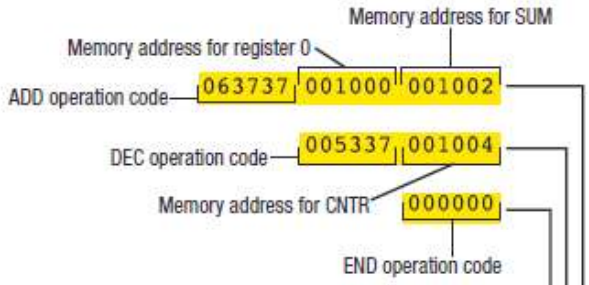
Programming Languages

- What Is a Programming Language?
  - A set of rules, words, symbols, and codes used to write computer programs
  - To write a program, you need the appropriate software for the programming language being used
- Categories of Programming Languages
  - Classified by the types of programs they are designed to create: procedural or object-oriented languages
  - Often categorized by their level or generation

Programming Languages

- Low-Level Languages (earliest programming languages)
  - Machine language
    - Written at a very low level, just using 1s and 0s
    - First generation of programming languages
  - Assembly language
    - Uses names and other symbols to replace some of the 1s and 0s in machine language
    - Second generation of programming languages
    - Programs take longer to write and maintain

Programming Languages



MACHINE LANGUAGE
Machine language instructions are typically in binary form, and the memory address locations, as well as the instructions themselves, need to be specified. The highlighted machine language instructions shown to the left correspond to the highlighted assembly language statements below.

ASSEMBLY LANGUAGE
Assembly language instructions use mnemonic operating codes to make the instructions much easier to understand. Note that data must still be moved in and out through the registers (register 0 in this example).

Labels | Operation codes | Operands | Comments

```
        .TITLE   SUM TWO NUMBERS
        .ENABL   AMA                      ; Enable absolute memory addressing
        .GLOBL   RNUM, PNUM               ; Subroutines to be used
        .MCALL   .TTYIN, .TTYOUT, .EXIT   ; System library macros to be used

START:  MOV      #2,CNTR                  ; Initialize counter to 2
        MOV      #0,SUM                   ; Initialize sum to 0

LOOP:   JSR      PC,RNUM                  ; Jump to subroutine RNUM to input number
        ADD      %0,SUM                   ; Add inputted number (in register 0) to sum
        DEC      CNTR                     ; Decrement counter
        BNE      LOOP                     ; Repeat loop if counter is not equal to 0

        MOV      SUM,%0                   ; Move sum to register 0
        JSR      PC,PNUM                  ; Jump to subroutine PNUM to print sum
        .EXIT                               (in register 0)

CNTR:   .BLKW    1                        ; Reserve 1 word of memory space for CNTR
SUM:    .BLKW    1                        ; Reserve 1 word of memory space for SUM
        .END     START                    ; End of program
```

FIGURE 13-18
Assembly and machine language

Copyright © 2015 Cengage Learning®

Programming Languages

- High-Level Languages
  - Closer to natural languages
  - Machine independent
  - Includes 3GLs (FORTRAN, BASIC, COBOL, C, etc.) and object-oriented languages (Visual Basic, C#, Python, Java, etc.)
  - Visual programming environments (VPEs)
    - Use graphical interface to create programs
    - Some are designed for educational purposes
      - Scratch

Programming Languages

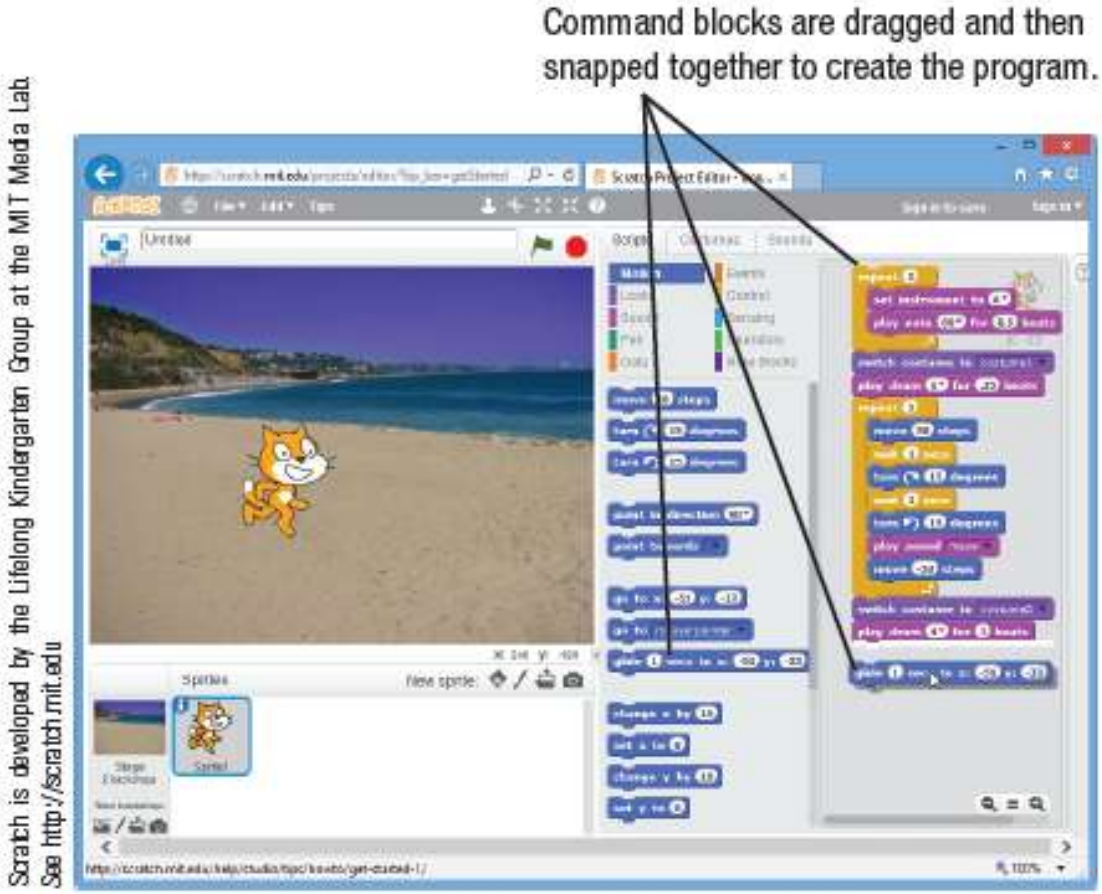


FIGURE 13-19
The Scratch graphical programming language.

Programming Languages

- Fourth-Generation Languages (4GLs)
  - Even closer to natural languages and easier to work with than high-level languages
  - Declarative rather than procedural
  - Commonly used to access databases

Common Programming Languages

- FORTRAN
  - High-level programming language used for mathematical, scientific, and engineering applications
  - Still used today for high-performance computing tasks (weather forecasting)
  - Fortress
    - Version designed for high-performance computing
    - Takes advantage of multi-core processors and computers with multiple processors
    - Not being updated

Common Programming Languages

Comments are preceded by an asterisk or a C.

```
      REAL SUM, CNTR, NUM
*
* INITIALIZE VARIABLES
      SUM = 0
*
* INPUT NUMBER, ADD IT TO THE SUM, AND THEN
* REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED
      DO 10 CNTR = 1, 2
         WRITE(*,*) 'Enter number'
         READ(*,*) NUM
         SUM = SUM + NUM
10    CONTINUE
*
* PRINT THE SUM
      WRITE(*,*) 'SUM IS ', SUM
*
      END
```

Program statements can be numbered in order to control loops and other types of branching.

Copyright © 2015 Cengage Learning®

FIGURE 13-20
The adding-two-numbers program written in FORTRAN.

Common Programming Languages

- COBOL
  - Designed for business transaction processing
  - Makes extensive use of modules
  - Strength lies in batch processing and its stability
  - Programs are lengthy and take a long time to write
  - Considered to be outdated by some
  - New versions are evolving
    - COBOL.NET

Common Programming Languages

Comments are preceded by an asterisk.

Most COBOL programs use a number of modules to break the program into manageable pieces. These submodules are called from the main control module using these statements.

Three submodules are used in this program.

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01    RESULT          PIC 9(3) VALUE ZERO.
01    CNTR            PIC 9(1) VALUE ZERO.
01    NUM             PIC 9(2) VALUE ZERO.

*******************
PROCEDURE DIVISION.
*******************
      PERFORM InitVariables
      PERFORM GetNumber UNTIL CNTR = 2
      PERFORM PrintSum
      STOP RUN.

*******************
InitVariables.
*******************
*  This module initializes the RESULT and CNTR variables to 0.
      MOVE 0 TO RESULT
      MOVE 0 TO CNTR.
*End of InitVariables

*******************
GetNumber.
*******************
*  This module inputs a number, adds it to the result, and
*  increments the counter.
      DISPLAY "Enter Number: " WITH NO ADVANCING
      ACCEPT NUM
      COMPUTE RESULT = RESULT + NUM
      COMPUTE CNTR = CNTR + 1.

*End of GetNumber module.

*******************
PrintSum.
*******************
*  This module prints the final RESULT.
       DISPLAY "The sum of the numbers you entered is " RESULT.
*End of PrintSum module.
```

Copyright © 2015 Cengage Learning®

FIGURE 13-21
The adding-two-numbers program written in COBOL.

Common Programming Languages

- Pascal
  - Named after mathematician Blaise Pascal
  - Created as a teaching tool to encourage structured programming
  - Contains a variety of control structures used to manipulate modules systematically
- BASIC and Visual Basic
  - Easy-to-learn, high-level programming language that was developed to be used by beginning programmers
  - Visual Basic
    - Object-oriented version of BASIC; uses a visual environment

Common Programming Languages

Comments are enclosed in { } braces.

The symbol := is used instead of the equal sign.

Semicolons mark the end of command statements.

```
program sum_numbers;

        var
          Num, Sum : real;
          Cntr : integer;

        begin

{ Initialize variables }
          Sum := 0;

{ Input a number, add it to the sum, and repeat }
{ until two numbers have been entered           }
          for Cntr := 1 to 2 do
            begin
              write('Enter number: ');
              readln(Num);
              Sum:= Sum + Num;
            end;

{ Print the sum }
          writeln('The sum of the numbers you entered is ',Sum);
        end.
```

Copyright © 2015 Cengage Learning®

FIGURE 13-22

The adding-two-numbers program written in Pascal.

Common Programming Languages

Comments are preceded by a single quotation mark.

Programs typically include input statements that pause the program until the user supplies the appropriate data.

```
'Clear the screen
CLS
'
'Initialize variables
SUM = 0
CNTR = 0
'
'Input number and add it to sum until two numbers have been
'entered.
DO
    INPUT "Enter number: ", NUM
    SUM = SUM + NUM
    CNTR = CNTR + 1
LOOP UNTIL CNTR = 2
'
'When done looping, display Sum on screen
PRINT "The sum of the numbers you entered is "; SUM
END
```

Copyright © 2015 Cengage Learning®

FIGURE 13-23

The adding-two-numbers program written in BASIC.

Common Programming Languages

- C, C++, and C#
  - C : Much closer to assembly language than other high-level languages
  - C++: Object-oriented version of C
    - Very popular for graphical applications
  - C# (C sharp): Hybrid of C and C++
    - Used to create Web applications, XML-based Web services, and Windows apps
  - Objective-C: For iPhone and other Apple applications

Programming Languages

Comments are preceded by two slashes //.

The instructions in a function or loop are enclosed in { } braces.

```
#include <iostream.h>

void main ()
{

// Declare and initialize variables
      float fSum = 0;
      float fNum;
      int iCntr = 0;

// Input a number, add it to the sum, and repeat
// until two numbers have been entered
      do
        {
        cout << "Enter number: "; // Prompt for input
        cin >> fNum;
        fSum = fSum + fNum;
        iCntr = iCntr + 1;
        }
       while(iCntr < 2);

// Print the sum
       cout << "The sum of the numbers you entered is " << fSum;
}
```

Copyright © 2015 Cengage Learning®

FIGURE 13-24
The adding-two-numbers program written in C++.

Common Programming Languages

- Java
  - High-level, object-oriented programming language frequently used for Web-based applications
  - Java programs are compiled into bytecode
  - Can run on any computer that includes Java Virtual Machine (Java VM)
  - Can be used to write Java applets
- Dart
  - High-level, open source, object-oriented programming language developed by Google
  - Designed to replace JavaScript in Web applications

Common Programming Languages

The java.io package will handle the user input; * indicates all classes will be available.

Comments within the code are preceded by two slashes //.

The *out* attribute and *println* method in the System class of the java.io package are used to output the results.

```
import java.io.*;
public class AddTwo {
      public static void main(String[] args) throws IOException {
       BufferedReader stdin =
          new BufferedReader ( new InputStreamReader( System.in ) );
       String inData;
       int iSum = 0;
       int iNum = 0;
       int iCntr = 0;

 // Input a number, add it to the sum, and repeat
 // until two numbers have been entered
     do
        {
        System.out.println("Enter number: ");
        inData = stdin.readLine();              // get number in character form
        iNum = Integer.parseInt( inData );      // convert inData to integer
        iSum = iSum + iNum;
        iCntr = iCntr + 1;
        }
     while (iCntr < 2);

// Print the sum
     System.out.println("The sum of the numbers you entered is " + iSum);
            }
        }
```

Copyright © 2015 Cengage Learning®

FIGURE 13-25
The adding-two-numbers program written in Java.

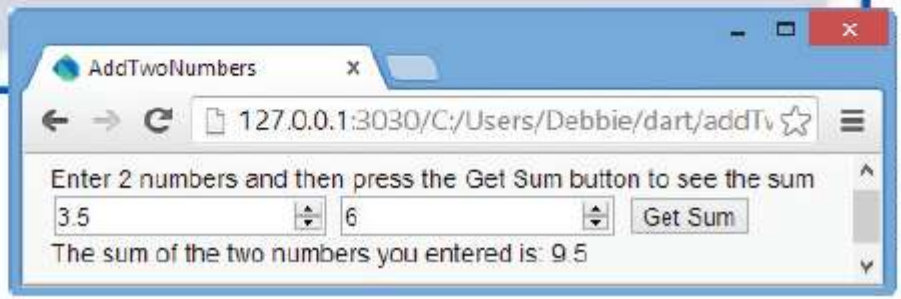
Common Programming Languages

Comments start with /**

Comments end with */.

```
import 'dart:html';

/**Declare function to add 2 numbers and display sum*/
void addTwo(MouseEvent event) {
  num x = (query("#firstnum") as InputElement)
 .valueAsNumber;
  num y = (query("#secondnum") as InputElement)
 .valueAsNumber;
  num sum = x+y;
  query("#sum").text="The sum of your two numbers
  is: $sum";
}
/**Execute function when Get Sum button is clicked*/
void main() {
  query("#GetSum").onClick.listen(addTwo);
}
```



Copyright © 2015 Cengage Learning®. Source: Dart is an open-source project with contributors from Google and elsewhere.

FIGURE 13-26

The adding-two-numbers program written in Dart.

Common Programming Languages

- Ruby
  - High-level, open source, object-oriented programming language that is often used to develop Web applications
- Python
  - Open-source, dynamic, object-oriented language that can be used to develop a variety of applications: e.g., gaming, scientific, and databases
  - Used by large organizations and some colleges, such as MIT

Common Programming Languages

```
# Initialize variable
total = 0.0

# Input a number, add it to the total, and repeat
# until two numbers have been entered
for iteration in range(2):
    text = raw_input("Enter number: ")
    total = total + float(text)

# Print the sum
print "The sum of the numbers you entered is", total
```

Comments are preceded by a pound symbol #.

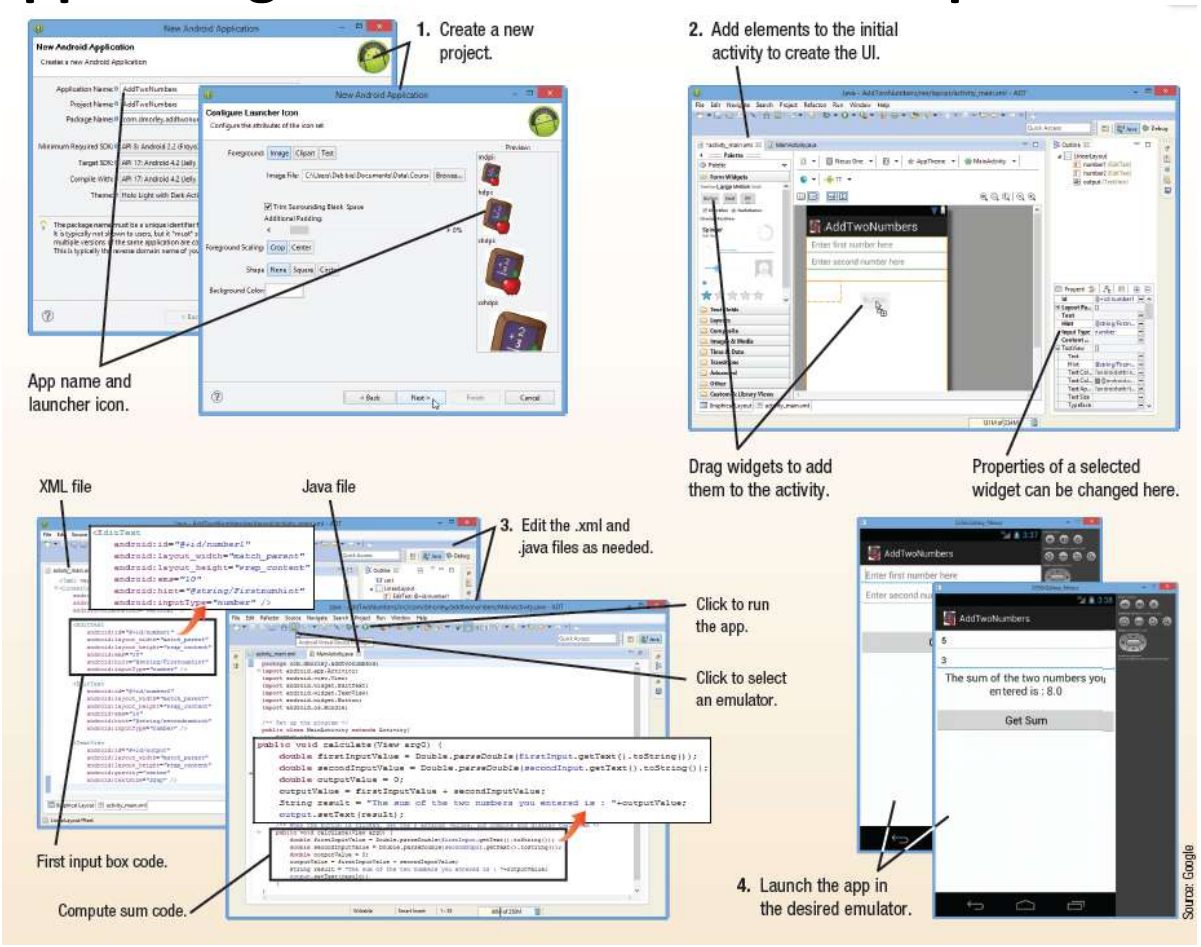
The indented statements in this for statement will be executed two times.

Copyright © 2015 Cengage Learning®

FIGURE 13-27
The adding-two-numbers program written in Python.

How It Works Box

Creating Apps Using the Android SDK and Eclipse



Quick Quiz

1. An example of a high-level programming language is _____.
   a. Pascal
   b. Assembly language
   c. Machine language
2. True or False: Visual Basic is an object-oriented version of COBOL.
3. Java applets are small programs written in the _____ programming language.

Answers:

1) a; 2) False; 3) Java

Summary

- Approaches to Program Design and Development
- The Program Development Life Cycle (PDLC)
- Tools for Facilitating Program Development
- Programming Languages