Technology in Action

Chapter 8

Behind the Scenes: Software Programming
Chapter Topics

• Understanding Software Programming
  – The Importance of Programming
  – The Life Cycle of an Information System
  – The Life Cycle of a Program

• Programming Languages
  – Many Languages for Many Projects
  – Exploring Programming Languages
Understanding Software Programming

• Some tasks are complex
  – Requires creative thought
  – Requires human touch

• Some tasks are candidates for automation
  – Repetitive
  – Work with electronic information
  – Follow a series of clear steps
The Importance of Programming

• A career in programming offers
  – Plentiful jobs
  – Strong salaries
  – The opportunity to telecommute in some cases

• Computer programs exist for many tasks

• Programming is necessary when there is no existing software for the task
The Importance of Programming (cont.)

• Having a basic knowledge of programming, you can
  – Add features that support your personal needs
  – Create macros and add custom commands
  – Create custom applications from scratch and successfully complete your projects
The Life Cycle of an Information System

• System
  – A collection of pieces working together to achieve a common goal

• An information system includes
  – Data
  – People
  – Procedures
  – Hardware
  – Software
The System Development Life Cycle

• To create modern software, an entire team is needed
• Needs to be an organized process
• Must be available for multiple operating systems and work over networks
• Must be free of errors and well supported
Six Steps in the SDLC

Each step must be completed before you can progress to the next step.
Problem and Opportunity Identification

- Corporations attempt to break into new markets and launch new products
- Corporations serve existing customers
  - Greater efficiency
  - Respond to problems
- Decide which projects to take forward based on available resources
  - Personnel and funding
Analysis

• Analysts explore problem to be solved
• Develop program specifications
  – Clear statement of goals and objectives of project
• Feasibility assessment is performed
• User requirements are defined
• Analysts recommend a plan of action
Design

- Generates a detailed plan for programmers
- Flowcharts and data-flow diagrams are used for the proposed system
  - Flowcharts are visual diagrams of a process
  - Data-flow diagrams trace all data from the point at which data enters the system to its final place
Design (cont.)

Data-flow diagram

- Concert Attendee
- Ticket Request
- Ticket Reservation System
  - Customer Ticket Information
  - Customer Preferences
  - Customer Ticket Information
- Concert Hall Ticket Agent
  - Ticketing Information

Customer Preferences

Concert Hall
Development

• Actual programming takes place
• First phase of the program development life cycle (PDLC)
• Documentation work is begun by technical writers
Testing and Installation

• Program is tested to ensure it works properly
• Program is installed for official use
Maintenance and Evaluation

• Performance of the system is monitored
• Corrections and modifications to the program are made
• Additional enhancements that users request are evaluated
• Appropriate program modifications are made
• Scope creep is an ever-changing set of requests for additional features
The Life Cycle of a Program

• Programming
  – Process of translating a task into a series of commands a computer will use to perform that task
  – Involves identifying which parts of a task the computer can perform
  – Describes tasks in a highly specific and complete manner
  – Translates this description into a language understood by the computer’s CPU
Program Development Life Cycle

1. **STEP 1: Describing the Problem** (The Problem Statement)
2. **STEP 2: Making a Plan** (Algorithm Development)
3. **STEP 3: Coding** (Speaking the Language of the Computer)
4. **STEP 4: Debugging** (Getting Rid of Errors)
5. **STEP 5: Testing and Documentation** (Finishing the Project)
Step 1: Describing the Problem

- Programmers develop a complete description of problem
- Problem statement identifies task to be automated
- Statement describes how software will behave
Step 2: Making a Plan

- Problem statement is translated into a set of specific, sequential steps known as an algorithm.
- Describes what the computer must do to complete the work.
- Algorithm is written in natural, ordinary language such as English.
Step 3: Coding

- Algorithm is translated into programming code
- Programming code is friendlier to humans but is highly structured
- Programmers must think in terms of operations that a CPU can perform
Step 4: Debugging

• Code goes through process of debugging
• Programmers repair any errors found in code
Step 5: Testing and Documentation

• Software is tested by
  – Programming team
  – People who will use program

• Results of entire project are documented for
  – Users
  – Development team

• Users are trained to use program efficiently
Describing the Problem: The Problem Statement

- A problem statement is necessary because it:
  - Is a starting point of programming work
  - Clearly describes tasks the computer program must accomplish
  - Describes how program will execute tasks and respond to unusual situations
  - Helps to better understand goals of programming efforts
Computer programs solve several problems

- Sophisticated problems can be broken down into series of steps
- Simulation software can perform “dry” tests
- Most promising choices are tested in “wet” laboratory
Describing the Problem: The Problem Statement (cont.)

• Computers programs can not:
  – Act with intuition
  – Be spontaneously creative
  – “Think” like humans

• Computers only follow instructions and algorithms
Describing the Problem:
The Problem Statement (cont.)

• Programmers create problem statements which interact with users to describe three relevant things

1. Data: Raw input users have at the start
2. Information: Result users require
3. Method: Process of how program converts the inputs to correct outputs
Describing the Problem: The Problem Statement (cont.)

- Programmers handle bad inputs through error handling through a testing plan
  - Lists specific input numbers expected of users
  - Lists specific output numbers that program would return
  - Uses these lists to determine whether program works
Describing the Problem: The Problem Statement (cont.)

• Testing plan does not cover every possible use of the program, instead it:
  – Works with users to identify categories of inputs
  – Specifies kind of output to be generated
  – Describes how errors would be managed or output generated for each input category
There is a standard format for a problem statement
  – Most companies have their own format
  – Basic components include
    • Inputs
    • Outputs
    • Processing
    • Error handling
    • Testing plan
# Problem Statement Example

## Complete Problem Statement for the Parking Garage Example

<table>
<thead>
<tr>
<th>PROBLEM STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program Goal</strong></td>
</tr>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td><strong>Output</strong></td>
</tr>
<tr>
<td><strong>Process</strong></td>
</tr>
<tr>
<td><strong>Error Handling</strong></td>
</tr>
</tbody>
</table>

## Testing Plan

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8*7.50</td>
<td>Testing positive input</td>
</tr>
<tr>
<td>3</td>
<td>3*7.50</td>
<td>Testing positive input</td>
</tr>
<tr>
<td>12</td>
<td>8<em>7.50 + 4</em>11.25</td>
<td>Testing overtime input</td>
</tr>
<tr>
<td>-6</td>
<td>Error message/ask user to reenter value</td>
<td>Handling error</td>
</tr>
</tbody>
</table>
Making a Plan: Algorithm Development

• The process starts by developing a detailed algorithm
  − Set of specific, sequential steps
  − Describe in natural language what program must do
Making a Plan: Algorithm Development (cont.)

• Algorithms:
  – Are used in daily life
  – Lay out specific plan that encapsulates all choices
  – Show specific sequence in which tasks will occur
Making a Plan: Algorithm Development (cont.)

• Algorithms are limited
  – Not all problems can be described as a fixed sequence of predetermined steps
  – Some involve random, unpredictable events
Making a Plan: Algorithm Development (cont.)

• Programmers represent an algorithm through:
  – Flowcharts: Provide a visual representation of patterns
    • Specific shape symbols indicate program behaviors
    • Microsoft Visio is a popular flowcharting program
  – Pseudocode: Text-based approach
    • Words describe the actions
    • Organized like an outline
Pseudocode

• Text-based approach to documenting an algorithm
• Words describe actions the algorithm will take
Pseudocode (cont.)

• Organized like an outline
  – Differing levels of indentation indicate flow of actions within the program
  – No standard vocabulary
  – Use a combination of common and special words as commands
Developing the Algorithm: Decision Making and Design

• Programmers handle complex algorithms
  – By the use of a list of choices
  – Through decision points which are dependent on value of input
Decision Points

- Ask for # of Hours Worked
- Read # of Hours Worked
- Is Number of Hours Worked \( \leq 8 \)?
  - Yes: Total Pay = \$7.50 \times \text{Number of Hours Worked}
    - Done
  - No: Program executes different set of steps if answer is No
- Total Pay = \$7.50 \times 8 + \$11.25 \times (\text{Number of Hours Worked} - 8)
  - Done
Developing the Algorithm: Decision Making and Design (cont.)

• There are several kinds of decision points
  – Binary decisions
    • Like a “fork in the road”
    • Can only be answered in one of two ways
  – Loops
    • Question is asked
    • If yes, set of actions is performed
    • Question is asked again
    • If no, moves to step that follows loop
• There are three features in a loop
  – Beginning point, or initial value, is the default value
  – Set of actions to be performed
  – Test condition checks to see if the loop is completed
Developing the Algorithm: Decision Making and Design (cont.)

Start on Monday
Start with Total Pay for Week = 0.00

Initial value

If test condition passes, loop continues

Are we still in the same week?

Yes

Calculate Pay for the Day

Calculate Total Pay for Week So Far

Bump to Next Day

Update

No

If test condition fails, break out of loop and move to next step

Print Weekly Paycheck
Developing the Algorithm: Decision Making and Design (cont.)

- Programmers create algorithms for specific tasks by using:
  - Top-down design
  - Object-oriented analysis
Top-Down Design

• Problem is broken into a series of high-level tasks
• Detailed subtasks are created from high-level tasks
Top-Down Design (cont.)

• How top-down design is used in programming

```
if (NumberHoursWorkedToday <= 8)
    Pay = $7.50 * NumberHoursWorkedToday
else
    Pay = $7.50 * 8 + $11.25 * (NumberHoursWorkedToday - 8)
```
Object-Oriented Analysis

- Programmers identify categories of inputs
  - Classes (categories of inputs) are identified
  - Classes are defined by information (data) and actions (methods or behaviors)
  - Algorithm enables objects to interact
Object-Oriented Analysis (cont.)

Data (Information)
- Name
- Address
- Social Security #
- Pay Grade
- Pay Rate

Methods (Actions)
- GoToWork()
- LeaveWork()
- CollectPay()

Objects (Specific Employees)
- John Doe
  - Address: 1313 Mockingbird
  - SSN: 011-11-0000
  - Pay Grade: 5
  - Pay Rate: $43.02

- Jane Doe
  - Address: 1060 West Addison
  - SSN: 999-09-0909
  - Pay Grade: 10
  - Pay Rate: $57.33

- Bill McGillicutty
  - Address: 7 Freedom Square
  - SSN: 123-45-6789
  - Pay Grade: 4
  - Pay Rate: $39.80
Object-Oriented Analysis (cont.)

• A developer selects the object-oriented approach over top-down design for several reasons
  – Leads to reusability
  – Forces programmers to think in general terms
  – Can reuse existing classes
  – Can produce new code quickly
Object-Oriented Analysis (cont.)

- A programmer takes advantage of reusability
  - Hierarchies can be built quickly
  - Inheritance - new class picks up data and methods of an existing class
    - Extends and customizes to fit its own need
      - Original class is called the base class
      - New, modified class, is called the derived class
• Ideas are translated into CPU instructions
  – Select best programming language
  – Coding: Translate algorithm into that language
  – Highly precise format using few keywords, but consistent structure
Recipe

Sugar Cookies
1 cup sugar
1 tsp. vanilla
1 egg
2 cups flour

Chocolate Sugar Cookies
ADD
2 squares unsweetened chocolate
1 tsp. cinnamon

Mini-Chip Sugar Cookies
ADD
1 1/2 cups mini chocolate chips
1/2 cup brown sugar
• Programmers move from algorithm to code by:
  – Identifying key pieces of information
  – Identifying flow of each step
  – Converting algorithm into computer code
Coding: Speaking the Language of the Computer (cont.)

- Programming language
  - Kind of “code” for instructions the CPU knows how to perform
  - Languages use special words and strict rules
  - Allows control of CPU without knowing hardware details
There are various kinds of programming languages

- Classified in major groups called generations
- Early languages required programmer to know how computer was constructed and how it stored data
- Programming is easier now as languages more closely match human thinking
• Evolution of modern programming languages
  – First-generation (1GL): Actual machine language – sequence of bits CPU understands
  – Second-generation (2GL): Known as Assembly language – uses short, English-like commands
  – Third-generation (3GL): Uses symbols and commands; easier for humans to read
Coding: Speaking the Language of the Computer (cont.)

- Evolution of modern programming languages (cont.)
  - Fourth-generation (4GL): Database query languages and report generators
  - Fifth-generation (5GL): Most natural; problem is presented as a series of facts
### Sample Code for Different Language Generations

<table>
<thead>
<tr>
<th>GENERATION</th>
<th>EXAMPLE</th>
<th>SAMPLE CODE</th>
</tr>
</thead>
</table>
| 1GL        | Machine | Bits describe the commands to the CPU.  
1110 0101 1001 1111 0000 1011 1110 0110 |
| 2GL        | Assembly| Words describe the commands to the CPU:  
ADD Register 3, Register 4, Register 5 |
| 3GL        | FORTRAN, BASIC, C, Java | Symbols describe the commands to the CPU:  
Total Pay = Pay + Overtime Pay |
| 4GL        | SQL     | More powerful commands allow complex work to be done in a single sentence.  
SELECT isbn, title, price, price*0.06 AS salesTax  
FROM books WHERE price > 100.00 ORDER BY title; |
| 5GL        | PROLOG  | Programmers can build applications without specifying an  
algorithm. Find all the people who are Mike’s cousins as:  
7-cousin (Mike, family) |
• Programmers do not have to use a higher-level programming language
  – Experienced programmers can write directly in assembly language
  – Higher-level languages allow programmers to focus on problem to be solved
  – Higher-level languages offer portability
• When a program is first written
  – Each input and output (variables) are announced early in program
  – Memory space needs to be set aside for inputs and outputs, known as variable declaration
  – Input and output values need to be stored in RAM
• Programmers can leave notes to themselves inside a program
  – Comments can be included to:
    • Explain the purpose of a section of code
    • Indicate the date the program was written
    • Include other important information
  – Comments are written in plain English
  – Compilers ignore comments
  – Intended to be read by programmers
Example of completed code
• Ways in which programmers make their code more useful for the future
  – “Containers” are sections of code that can be used repeatedly
  – Referred to as functions, methods, procedures, subroutines, modules, or packages
We can use the ComputePay function again and again in programs:

\[
\text{BrettsPay} = \text{ComputePay}(40, 7.50, 11.25);
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Brett} & \text{Hours} & \text{Base pay} & \text{Overtime pay} \\
\hline
\text{40} & \text{$7.50} & \text{$11.25}$ \\
\hline
\end{array}
\]

\[
\text{MarinasPay} = \text{ComputePay}(20, 10.50, 15.75);
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Marina} & \text{Hours} & \text{Base pay} & \text{Overtime pay} \\
\hline
\text{20} & \text{$10.50} & \text{$15.75}$ \\
\hline
\end{array}
\]
Compilation

• Program language code must be converted to 1s and 0s for the CPU
  – Compilation is the process by which code is converted into machine language
  – Compiler understands syntax of language and structure of CPU
  – Source Code – the instructions programmers have written in high level language
• Not every programming language has a compiler – they use an interpreter
  – The interpreter translates source code into a line-by-line intermediate form
  – Each line is executed as it is translated
  – Programmers do not have to wait for the entire program to be recompiled each time they make a change
  – Programmers can immediately see the results of changes as they are making them
Coding Tools: Integrated Development Environments

• Tools make the coding process easier
  – Integrated development environment (IDE) helps programmers test programs
  – One IDE can be configured to support many languages
Coding Tools: Integrated Development Environments
Coding Tools: Integrated Development Environments (cont.)

• An IDE helps programmers when typing the code
  – Code editing is when programmers type code into the computer
  – An editor is a special tool that helps programmers as code is entered
• Code editor
  – Highlights keywords
  – Alerts programmers to typos
  – Automatically indents code correctly
  – Aligns sections of code
  – Applies color to comments
  – Provides auto-completion of code
  – Suggests solutions to common errors
• The IDE helps programmers after code editing is finished
  – Compilation begins
  – Shows progress
  – Identifies syntax errors
  – Number of warnings that have been generated
  – Allows programmer to repair syntax errors quickly
Debugging: Getting Rid of Errors

• Debugging
  – Process of running program over and over
  – Helps to find errors
  – Makes sure the program behaves the way it should
Debugging: Getting Rid of Errors (cont.)

• The testing plan helps programmers know the program has solved the problem
  – Testing plan clearly lists input and output values
  – Shows how users expect the program to behave
  – Needs to contain specific examples to test every part of the program
Debugging: Getting Rid of Errors (cont.)

• The compiler functions even if the testing plan reveals errors
  – Compiler can’t decide if code is logical
  – Compiler only verifies specific rules of language are followed
  – Logical errors (when what the programmer wrote is not logical) - only caught when the program executes
  – Runtime errors (forbidden operations) are also caught (Ex. divide by zero errors)
There tools that help programmers find logic errors

- Debugger pauses the program so that values of all variables can be examined
- Can run the program in slow motion
- Can isolate the precise place of the logical error
- When correct, can recompile the program
Testing and Documentation: Finishing the Project

• First round of testing a program
  – Internal testing: A group within the software company uses program in every way possible
  • Differences in how the program responds are reported back to the programmer
  – External testing: Eventual users work with program to determine whether it matches original vision
• Additional testing - Beta version
  – Provided free or at reduced cost
  – Programmers collect information about remaining errors
  – Results in final revisions before officially releasing the program
  – Often available months before the official public release
• Solving problems after beta testing
  – Manufacturer will make changes before final release
  – Called release to manufacturers (RTM)
  – After RTM is issued, product is in general availability (GA)
  – Problems are addressed in updates or service packs

• Service packs repair identified errors in the code
Testing and Documentation: Finishing the Project (cont.)

• To finish the project:
  – Technical writers produce internal documentation
    • Development and technical details of software
    • How the code works
    • How the user interacts with the program
  – User documentation is produced
  – Software trainers teach others how to use program
Many Languages for Many Projects

• Programmers want solutions that meet several objectives
  – Run quickly
  – Be reliable
  – Be simple to expand later when demands on system change
  – Be completed on time
  – Be finished for the minimum possible cost
Many Languages for Many Projects (cont.)

• Programming languages have been developed to balance conflicting goals
  – Share common characteristics
  – Each language has specific traits
    • Allow it to be the best for certain types of projects
  – Programmer needs to understand each language to match program needs with language
Many Languages for Many Projects (cont.)

• Popular Programming languages
  – C/, C++, and Java in most demand
  – COBOL - Banking and insurance
  – Tiobe Index uses different techniques to see which languages are popular
  – Pascal – designed as a teaching language
Many Languages for Many Projects (cont.)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>↑</td>
<td>C</td>
<td>18.331%</td>
<td>+1.05%</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>↓</td>
<td>Java</td>
<td>16.067%</td>
<td>-3.16%</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>↑↑↑↑↑↑</td>
<td>Objective-C</td>
<td>9.335%</td>
<td>+4.15%</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>C++</td>
<td>9.118%</td>
<td>+0.10%</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>↓</td>
<td>C#</td>
<td>6.66%</td>
<td>+0.45%</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>↑</td>
<td>(Visual) Basic</td>
<td>5.695%</td>
<td>+0.59%</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>↓</td>
<td>PHP</td>
<td>5.012%</td>
<td>-1.17%</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>=</td>
<td>Python</td>
<td>4.000%</td>
<td>+0.42%</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>=</td>
<td>Perl</td>
<td>2.053%</td>
<td>-0.28%</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>↑↑↑↑↑↑</td>
<td>Ruby</td>
<td>1.768%</td>
<td>+0.44%</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>↓</td>
<td>JavaScript</td>
<td>1.454%</td>
<td>-0.79%</td>
<td>A</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>↑↑↑↑↑↑</td>
<td>Delphi/Object Pascal</td>
<td>1.157%</td>
<td>+0.27%</td>
<td>A</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>=</td>
<td>Lisp</td>
<td>0.997%</td>
<td>+0.09%</td>
<td>A</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>↑</td>
<td>Transact-SQL</td>
<td>0.954%</td>
<td>+0.15%</td>
<td>A</td>
</tr>
<tr>
<td>15</td>
<td>25</td>
<td>↑↑↑↑↑↑</td>
<td>Visual Basic.NET</td>
<td>0.917%</td>
<td>+0.43%</td>
<td>A</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>=</td>
<td>Pascal</td>
<td>0.837%</td>
<td>+0.17%</td>
<td>A</td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>↑</td>
<td>Ada</td>
<td>0.689%</td>
<td>+0.14%</td>
<td>B</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>↑</td>
<td>Lua</td>
<td>0.684%</td>
<td>-0.89%</td>
<td>B</td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>↑</td>
<td>PL/SQL</td>
<td>0.645%</td>
<td>+0.10%</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>26</td>
<td>↑↑↑↑↑</td>
<td>MATLAB</td>
<td>0.639%</td>
<td>+0.19%</td>
<td>B</td>
</tr>
</tbody>
</table>
Many Languages for Many Projects (cont.)

• Factors in selecting the right language
  – Space available – some applications (cell phones) require space-efficient programs
  – Speed required – sometimes more important than size of code
  – Organizational resources available – high turnover in programmers requires easier language
  – Type of target application – supports specific environments (UNIX, Windows)
Many Languages for Many Projects (cont.)

• Programs for Windows environment
  – Extremely popular
  – Have common features (scroll bars, title bars, text boxes, buttons, expanding or collapsing menus)
  – Several languages include customized controls to make it easy for programmers
  – Same is true of OS X operating system
Many Languages for Many Projects (cont.)

• Visual programming
  – Mouse is used to lay out the screen
  – Code is written automatically
  – Helps programmers produce final application more quickly
Many Languages for Many Projects (cont.)
Visual Basic

• Advantages of Visual Basic
  – Prototyping is a form of rapid application development (RAD)
  – Developers create prototype first then generate system documents
  – RAD is an alternative to the waterfall approach
Visual Basic (cont.)

- Advantages of Visual Basic (cont.)
  - Visual Basic (VB) is a programming language used to build Windows applications
  - VB is simple and quick and is used to build object-oriented applications
Visual Basic (cont.)

• Microsoft .NET Framework helps programmers
  – Allows websites “talk” to each other easily
  – Web services make information available to other websites
  – Provides a standard way to interact
Visual Basic (cont.)

• C and C++ are natural choices for learning a first language
  – Neither intended as a teaching language
  – Relatively difficult to master
  – In high demand though because code runs fast and uses a small amount of memory
  – Basic components are common to many other languages
Java and C#

- Programmers use various languages when information needs to be collected from networked computers
  - Java is an object-oriented language
    - Is a good choice for these applications
    - Popular because it uses a large set of existing classes
    - Classes exist for many graphical objects
• Java applications work on many types of computer
  – Java is architecture neutral
    • Needs to be compiled only once
  – Can run on many CPUs and with many operating systems
  – Target computer runs a Java Virtual Machine (VM)
  – Java applet is a small Java-based program
Java and C# (cont.)

```java
for (i = 1; i < 20; i++)
    System.out.println(i);
```
Objective C

- The most popular language for writing Mac OS X applications
  - Object oriented
  - Superset of C language
  - Adds more keywords and features
  - Used with library called Cocoa
  - Supports gesture recognition
Objective C (cont.)

- Favorite IDE for Objective C
  - Xcode is shipped with OS X
  - Can be purchased and downloaded from App Store
Most basic language for developing web applications

- A document for the web must be written using tags
- Tags control how a browser will display the content in Hypertext Markup Language (HTML)
- HTML isn’t a programming language, just a series of tags
• Tools help programmers write in HTML
  – Dreamweaver and Microsoft Expression are web page designers similar to word processing programs
  – Can quickly insert several design elements
  – Program automatically inserts HTML tags
  – Static web pages require no programming
• Evolution of HTML
  – Next release will be called HTML5
  – Move toward standardizing HTML and adapting to the times
  – Will eliminate need to install third-party browser plug-ins
  – Will support drag-and-drop and document editing
JavaScript and VBScript

• Programmers use several languages to make complex web pages
  – Programmers use scripting languages
  – Scripting languages are limited to performing specialized tasks
  – Allow decisions to be made and calculations to be performed
JavaScript and VBScript (cont.)

Adobe Dreamweaver (popular tool for creating web pages)
• JavaScript is a scripting language used to add interactivity to web pages
  – Not as fully featured as Java
  – Syntax, keywords, data types, and operators are subsets of Java’s
  – Has a set of classes that represent objects used on web pages
  – Includes behaviors which can respond to user actions
• Other scripting languages besides JavaScript
  – VBScript is a subset of Visual Basic
  – Dynamic decision making means any page can decide how to display itself based on reader’s choices
  – PHP, a very efficient open-source language, is popular and runs on multiple platforms
• How interactive web pages are built
  – Several languages adapt the HTML page to user’s selections
    • Active Server Pages (ASP)
    • JavaServer Pages (JSP)
    • PHP (Hypertext Preprocessor)
  – User supplies information that is translated into a request using database query language
• Programming brings additional features to a web page
  – Interact with the user
  – Collect information
  – Customize the content based on user feedback
• Creating a web page that includes sophisticated animation
  – Adobe Flash: Web-based multimedia
    • ActionScript is similar to JavaScript in keywords, operators, and classes
  – Microsoft Silverlight: Supports rich multimedia and interactive web applications
• Creating a web page that includes sophisticated animation (cont.)
  – Other advances include updating information without requiring user to do a page refresh
  • AJAX (Asynchronous JavaScript and XML)
  • HTML5
Flash, AJAX, and XML (cont.)

• XML helps websites gather information from other sites
  – eXtensible Markup Language (XML) enables designers to define data-based tags
  – Makes it easier for a website to transfer key information to another site
  – Formatting controls is important to people
  – Groups can agree on standard system of tags
Flash, AJAX, and XML (cont.)

**STEP 1:** User requests information on red bikes.

**STEP 2:** Server sends request to database computer.

**STEP 3:** Database computer returns list.

**STEP 4:** Server’s ASP program writes HTML page.
Flash, AJAX, and XML (cont.)

a. **ASP code**

```vbscript
<%Dim I%>

<%For I = 1 To 5
  ' Output our HTML and text using the value of I as the FONT TAG's SIZE attribute.
  %>  
  &lt;FONT SIZE=""&lt;%= I %&gt;">Hello World&lt;/FONT><BR>
  <%
Next
' Continue the loop 5 times
%
```

b. **HTML code**

```html
&lt;FONT SIZE="1">Hello World&lt;/FONT><BR&gt;
&lt;FONT SIZE="2">Hello World&lt;/FONT><BR&gt;
&lt;FONT SIZE="3">Hello World&lt;/FONT><BR&gt;
&lt;FONT SIZE="4">Hello World&lt;/FONT><BR&gt;
&lt;FONT SIZE="5">Hello World&lt;/FONT><BR&gt;
```

c. **HTML Displayed on User's Screen**

```
Hello World
Hello World
Hello World
Hello World
Hello World
```

- Uses `For` loops
- Writes five different HTML statements with a different number for font size
- HTML document is program output
- HTML document output in web browser
<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Features</th>
<th>Typical Setting</th>
</tr>
</thead>
</table>
| C/C++ and C#             | • Can create compact code that executes quickly  
                            • Provides high- and low-level access                                      | • Used in industrial applications such as banking and engineering               |
| Java                     | • Is architecture neutral  
                            • Is object-oriented                                                        | • Used to create applets that can be delivered over the web                     |
| Objective C              | • Has a framework for writing iOS applications                            | • Used to create applications for OS X and Apple mobile devices                 |
| Visual Basic             | • Is easy to learn and use  
                            • Is object-oriented  
                            • Has a drag-and-drop interface                                              | • Used in prototype development  
                                                                                   | • Used to design graphical user interfaces                                    |
| Web Technologies         |                                                                           |                                                                                |
| AJAX                     | • Uses a combination of existing technologies like JavaScript, CSS, and XML | • Creates websites that can update without the user refreshing the page        |
| HTML5                    | • Latest version of HTML but currently in development                      | • Introduces tags like <video> and supports drag-and-drop                      |
| VBScript                 | • Is similar in syntax to Visual Basic  
                            • Has classes that represent buttons, drop-down lists, and other web page components | • Creates code that lives on the client machine and adds interaction to web pages |
| XML                      | • Enables users to define their own tags  
                            • Facilitates exchange of information between websites                        | • Used in the construction of web services                                      |
Mobile Applications

• Applications for mobile devices
  – Special languages and tools are needed
  – Need to be able to take advantage of features
    • GPS capability
    • Compasses
    • Software keyboards
    • Touch-sensitive screens
  – Need to take smaller screen size into account
Mobile Applications (cont.)

• Creating mobile apps for Apple’s iOS platform
  – Detailed prototype organizes and links elements of app smoothly
    • Screens needed
    • User interface elements
    • Content
Mobile Applications (cont.)

• Creating mobile apps for Apple’s iOS platform (cont.)
  – Mock-App uses PowerPoint or Keynote to construct a working simulation
  – Interface Builder can rapidly create a prototype
Mobile Applications (cont.)

• Creating mobile apps for Apple’s iOS platform (cont.)
  – When time to write code programmers use Objective C and Apple Xcode toolset
  – Lets designers code and debug and simulate behavior of application
  – Application can be profiled for speed, memory usage, and other problems
Mobile Applications (cont.)
Mobile Applications (cont.)

• Tools for building apps for Android devices
  – Android software development kit (SDK) is required
  – Uses well-known IDEs with special plug-ins
  – Latest version and other resources are available at www.developer.android.com
Mobile Applications (cont.)

• Creating a simple app for non-programmers
  – With tools like Corona, even someone with little experience can produce games and apps quickly
  – Corona supports a wide range of features through simple programming syntax
  – Magmito supports developing a simple app and requires no programming knowledge
Mobile Applications (cont.)

• Adapting an application for other kinds of mobile device
  – Corona and Magmito support several different devices and save time for simple applications
  – For specific features and ultimate performance, custom programming is still required
• The next great programming language
  – Never easy to predict
  – As projects grow in size so does compiling time
  – Interpreted languages (Python, Ruby, and Smalltalk) take virtually no compile time, so will be even more important in coming years
The convergence of languages

- Most modern languages have common features
- Forcing a language to work for any task
  - Makes it slower to compile
  - Produces larger executables
  - Requires more memory to run
- Having a variety of languages is more efficient
The Next Great Language (cont.)

• Learning languages that will be relevant in the future
  – No set of languages is best to learn and no best sequence to learn them
  – Need a core set of mathematical and programming skills
  – Geography might indicate which languages are in demand in your area
  – Having an understanding of how software is created will be helpful in many IT careers
Chapter 10 Summary Questions

1. Why do I need to understand how to create software?
Chapter 10 Summary Questions

2. What is a system development life cycle, and what are the phases in the cycle?
3. What is the life cycle of a program?
4. What role does a problem statement play in programming?
5. How do programmers create algorithms and move from algorithm to code?
Chapter 10 Summary Questions

6. What steps are involved in completing the program?
Chapter 10 Summary Questions

7. How do programmers select the right programming language for a specific task?
Chapter 10 Summary Questions

8. What are the most popular programming languages for different types of application development?
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

Copyright © 2014 Pearson Education, Inc.
Publishing as Prentice Hall