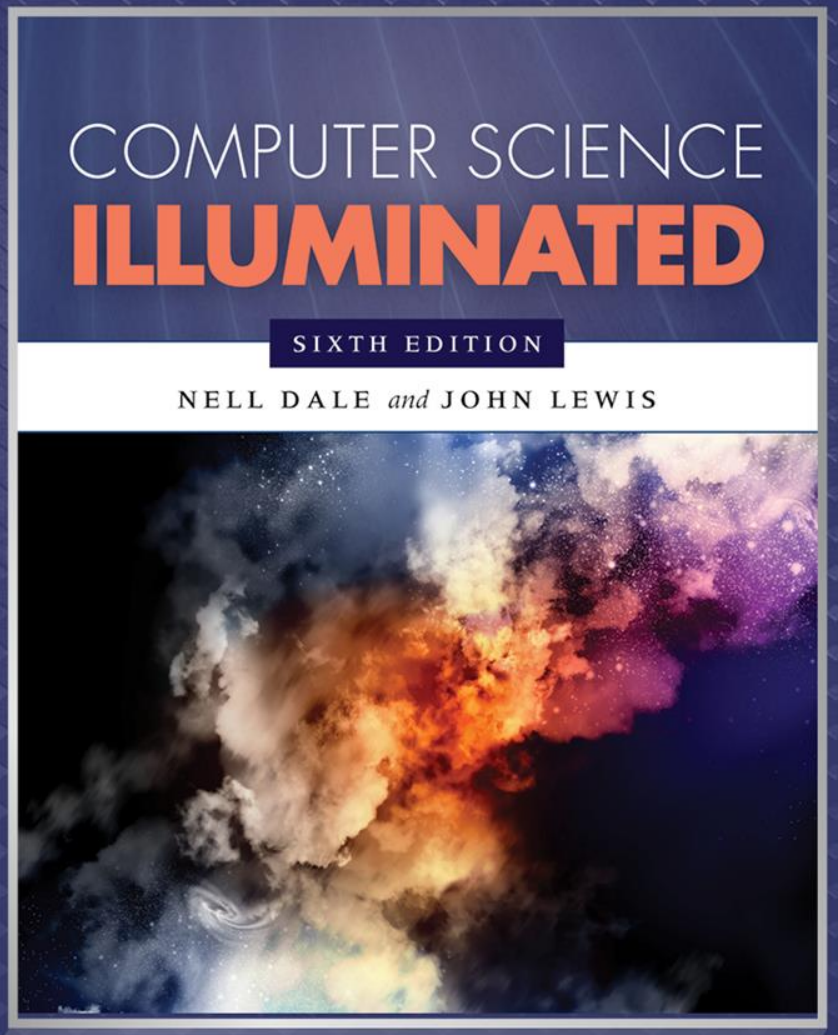
Chapter 9

Object-Oriented Design and High-Level Programming Languages



Chapter Goals

- Distinguish between functional design and object-oriented design
- Describe the stages of the object-oriented design process
- Apply the object-oriented design process
- Name, describe, and give examples of the three essential ingredients of an object-oriented language

Chapter Goals

- Describe the translation process and distinguish between assembly, compilation, interpretation, and execution
- Name four distinct programming paradigms and name a language characteristic of each
- Define the concepts of a data type and strong typing

Chapter Goals

- Understand how the constructs of top-down and object-oriented design are implemented in programming languages

Object-Oriented Design

Object-oriented Design

A problem-solving methodology that produces a solution to a problem in terms of self-contained entities called *objects*

Object

A thing or entity that makes sense within the context of the problem

For example, a *student*, a *car*, *time*, *date*

Object-Oriented Design

World View of OOD

Problems are solved by

- isolating the objects in a problem,
- determining their properties and actions (responsibilities), and
- letting the objects collaborate to solve a problem

What? Say again!

Object-Oriented Design

An analogy: You and your friend fix dinner

Objects: you, friend, dinner

Class: you and friend are people

People have name, eye color, …

People can shop, cook, …

Instance of a class: you and friend are instances of class People, you each have your own name and eye color, you each can shop and cook

You collaborate to fix dinner

Object-Oriented Design

Class (or object class)

A description of a *group* of similar objects

Object (instance of a class)

A concrete example of the class

Classes contain fields that represent the
properties (name, eye color) and
behaviors (responsibilities) (shop, cook) of the class

Method

A named algorithm that defines behavior (shop, cook)

Object-Oriented Design

Top-Down Design

decomposes problems into **tasks**

Object-Oriented Design

decomposes problems into

collaborating **objects**

Yes, but how?

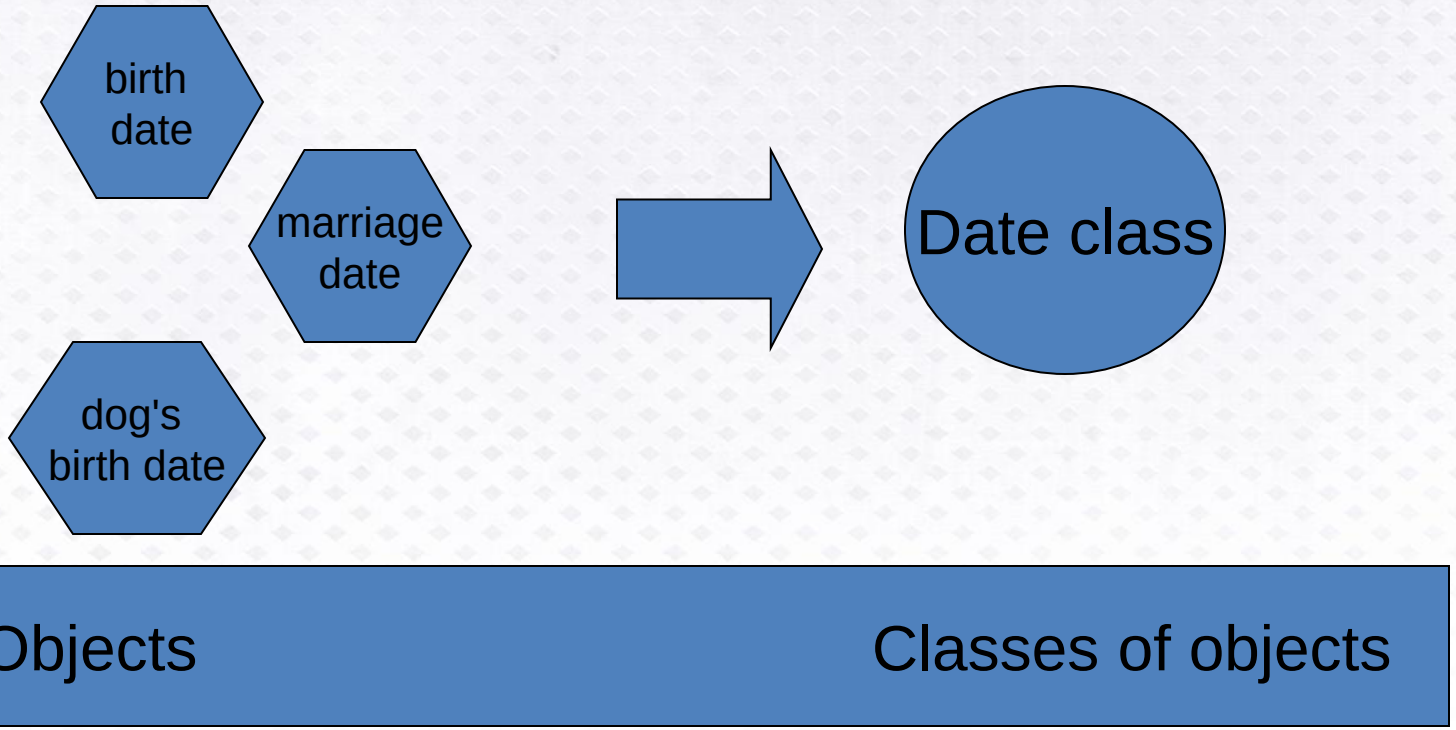
Object-Oriented Design

Steps

1. **isolate** the real-world objects in the problem
2. **abstract** the objects with like properties into groups (classes)
3. **determine** the responsibilities of the group in interacting with other groups

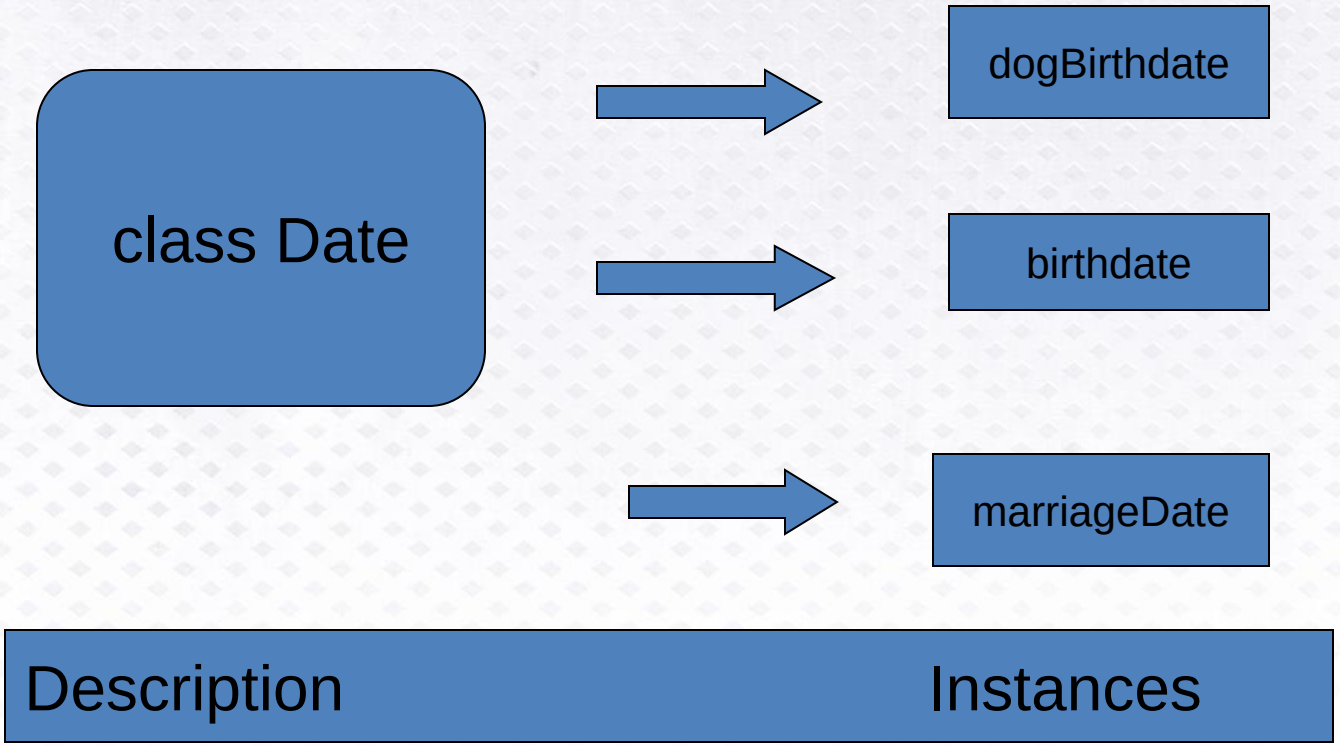
Object-Oriented Design

Think of design as a mapping from real world objects to classes of objects



Object-Oriented Design

Program World simulates these groups



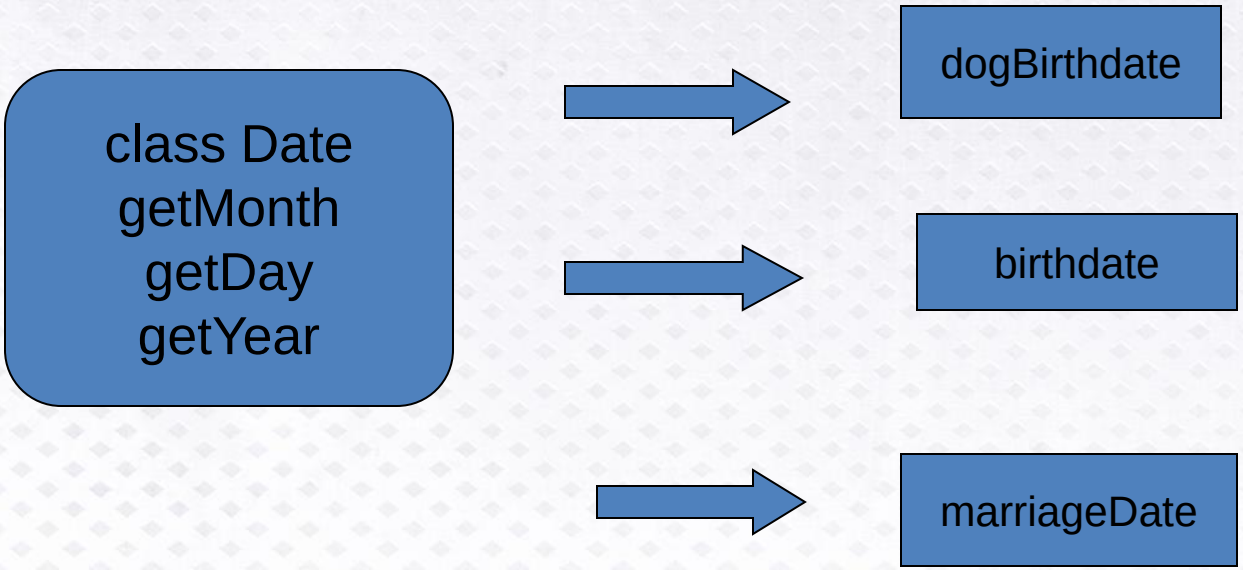
Object-Oriented Design

We call an object's interactions with other objects its **responsibilities**

Create itself
Know the state of its fields
Compare itself to another date
Return a date a number of days hence

Object-Oriented Design

Responsibilities become **methods** in the Program World



Object-Oriented Design Methodology

Four stages to the decomposition process

- Brainstorming to locate possible classes
- Filtering the classes to find duplicates or remove unnecessary ones
- Scenarios are tried to be sure we understand collaborations
- Responsibility algorithms are designed for all actions that classes must exhibit

Brainstorming

A group problem-solving technique that involves the spontaneous contribution of ideas from all members of the group

- All ideas are potential good ideas
- Think fast and furiously first, and ponder later
- A little humor can be a powerful force

Brainstorming is designed to produce a list of candidate classes

Filtering

Determine which are the core classes in the problem solution

There may be two classes in the list that have many common attributes and behaviors

There may be classes that really don't belong in the problem solution

Scenarios

Assign responsibilities to each class

There are two types of responsibilities

- What a class must know about itself (**knowledge** responsibilities)
- What a class must be able to do (**behavior** responsibilities)

Scenarios

Encapsulation

The bundling of data and actions in such a way that the logical properties of the data and actions are separated from the implementation details

Each class **encapsulates** its data but shares their values through knowledge responsibilities

Responsibility Algorithms

The algorithms must be written for the responsibilities

- Knowledge responsibilities usually just return the contents of one of an object's variables
- Action responsibilities are a little more complicated, often involving calculations

CRC Cards

CRC cards are a notational device to record information about a class, what it must do and with whom it must collaborate

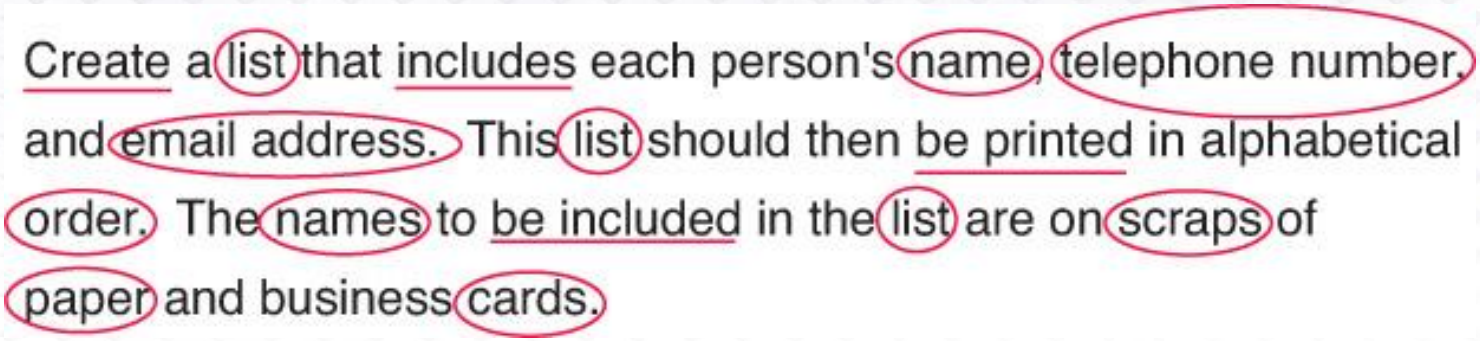
Class Name: SortedList (from library)	Superclass:	Subclasses:
Responsibilities	Collaborations	
Insert (person)	Person	
Print itself	Person	

Computer Example

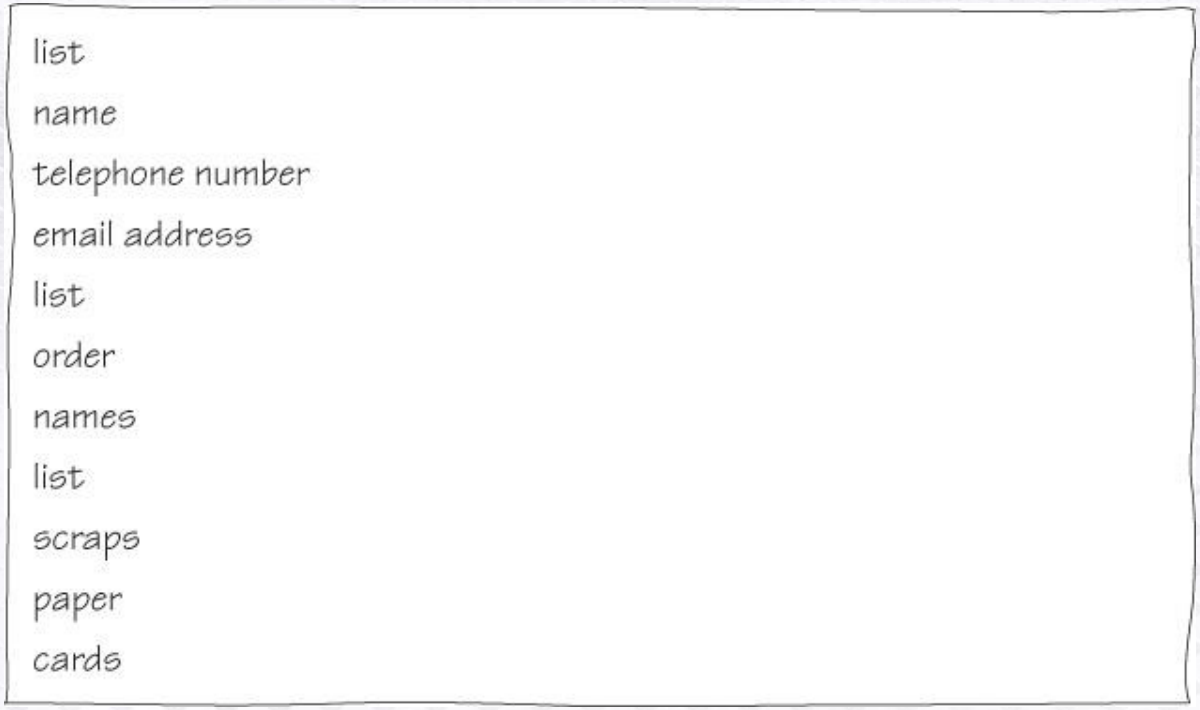
Let's examine the problem-solving process for creating an address list

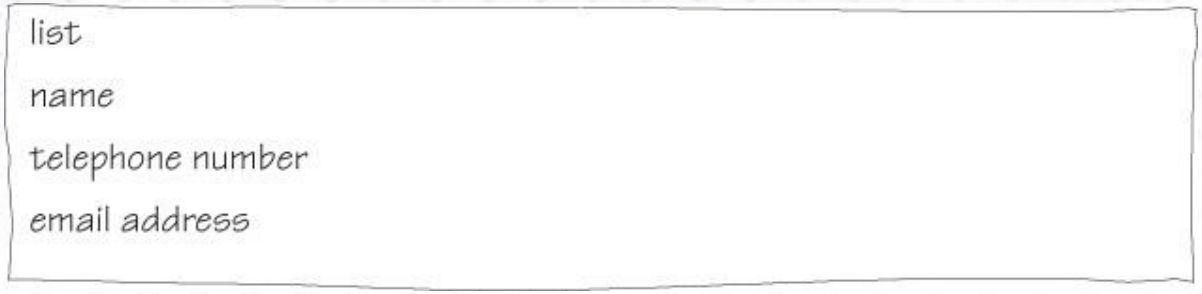
Brainstorming and filtering

- Circling the nouns and underlining the verbs is a good way to begin



Computer Example





CRC Cards

Class Name: Person	Superclass:	Subclasses:
Responsibilities	**Collaborations**	
Initialize itself (name, telephone, email)	Name, String	
Print	Name, String	
GetEmail	String	
GetName	Name, String	
Get Telephone	String	

Can you think of any other useful responsibilities?

CRC Cards

Class Name: Name	Superclass:	Subclasses:
Responsibilities		**Collaborations**
Initialize itself (firstName, lastName)		String
Print itself		String
GetFirstName		String
GetLastName		String

Can you think of any other useful responsibilities?

CRC Cards

Class Name: SortedList (from library)	Superclass:	Subclasses:
Responsibilities	**Collaborations**	
Insert (person)	Person	
Print itself	Person	

How is this class different from Name and Person?

Responsibility Algorithms

Person Class

Initialize

Tells name to initialize itself

```
name.initialize()
Write "Enter phone number; press return."
Get telephone number
Write "Enter email address; press return."
Get email address
```

Print

Tells name to print itself

```
name.print()
Write "Telephone number: " + telephoneNumber
Write "Email address: " + emailAddress
```

Responsibility Algorithms

Name Class

Initialize

"Enter the first name; press return."
Read firstName
"Enter the last name; press return."
Read lastName

Print

Print "First name: " + firstName
Print "Last name: " + lastName

Object Oriented Problem Solving and Implementation Phases

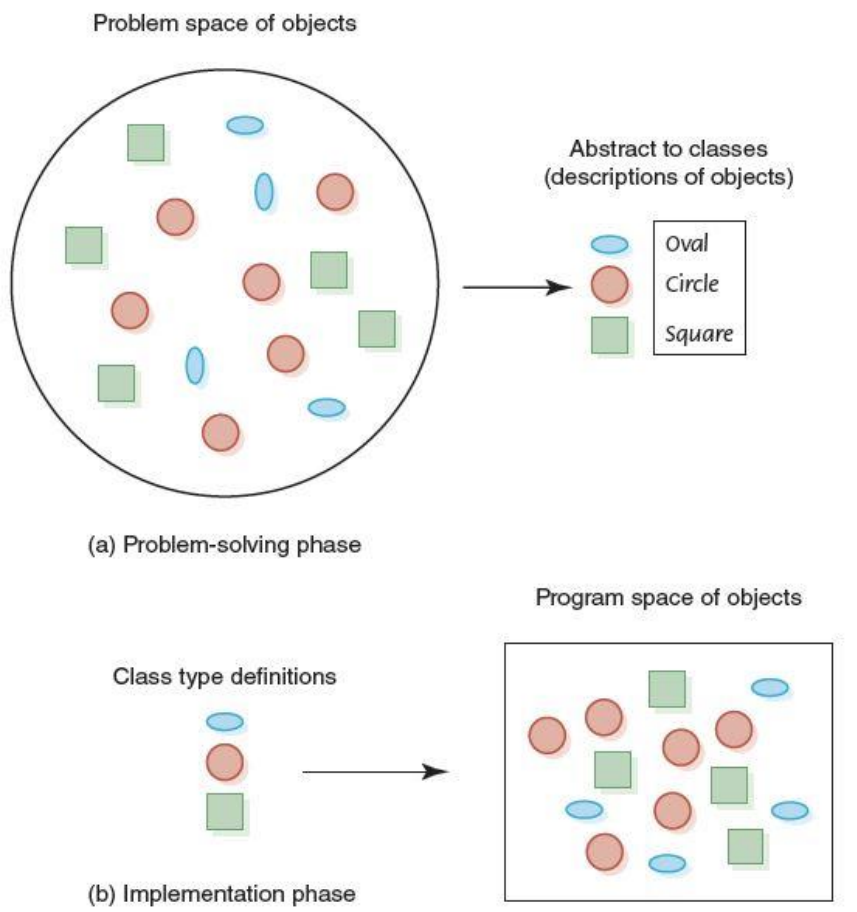


FIGURE 9.1 Mapping of a problem into a solution

Translation Process

A program written in a high-level language must be translated into machine code

The machine code is then executed

Compilers and Interpreters are software tools employed to help with the translation process

Compilers

High-level language

A language that provides a richer (more English-like) set of instructions

Compiler

A program that translates a high-level language program into machine code

Compilers

FIGURE 9.2 Compilation process

How does this differ from the assembly process?

Interpreters

Interpreter

A translating program that translates and executes the statements in sequence

- Assembler or compiler produce machine code as output, which is then executed in a separate step
- An interpreter translates a statement and then immediately executes the statement
- Interpreters can be viewed as *simulators*

Java

- Introduced in 1996 and became instantly popular
- Portability was of primary importance
- Java is compiled into a standard machine language called Bytecode
- A software interpreter called the JVM (Java Virtual Machine) takes the Bytecode program and executes it

Portability

Portability

The ability of a program to be run on different machines

Compiler portability

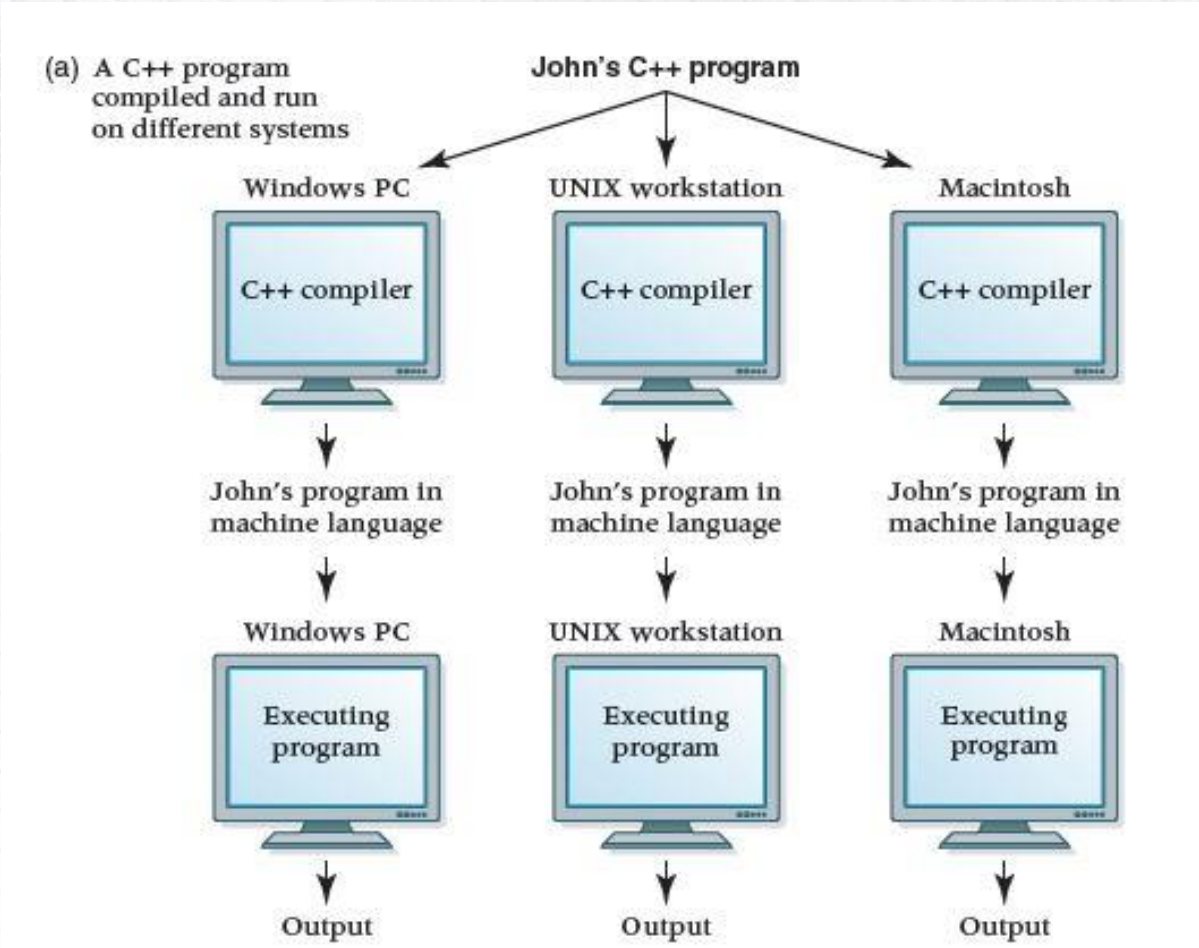
A program in a standardized language can be compiled and run on any machine that has the appropriate compiler

Bytecode portability

A program translated into Bytecode can be run on any machine that has a JVM

Do you understand the difference?

Portability



Portability

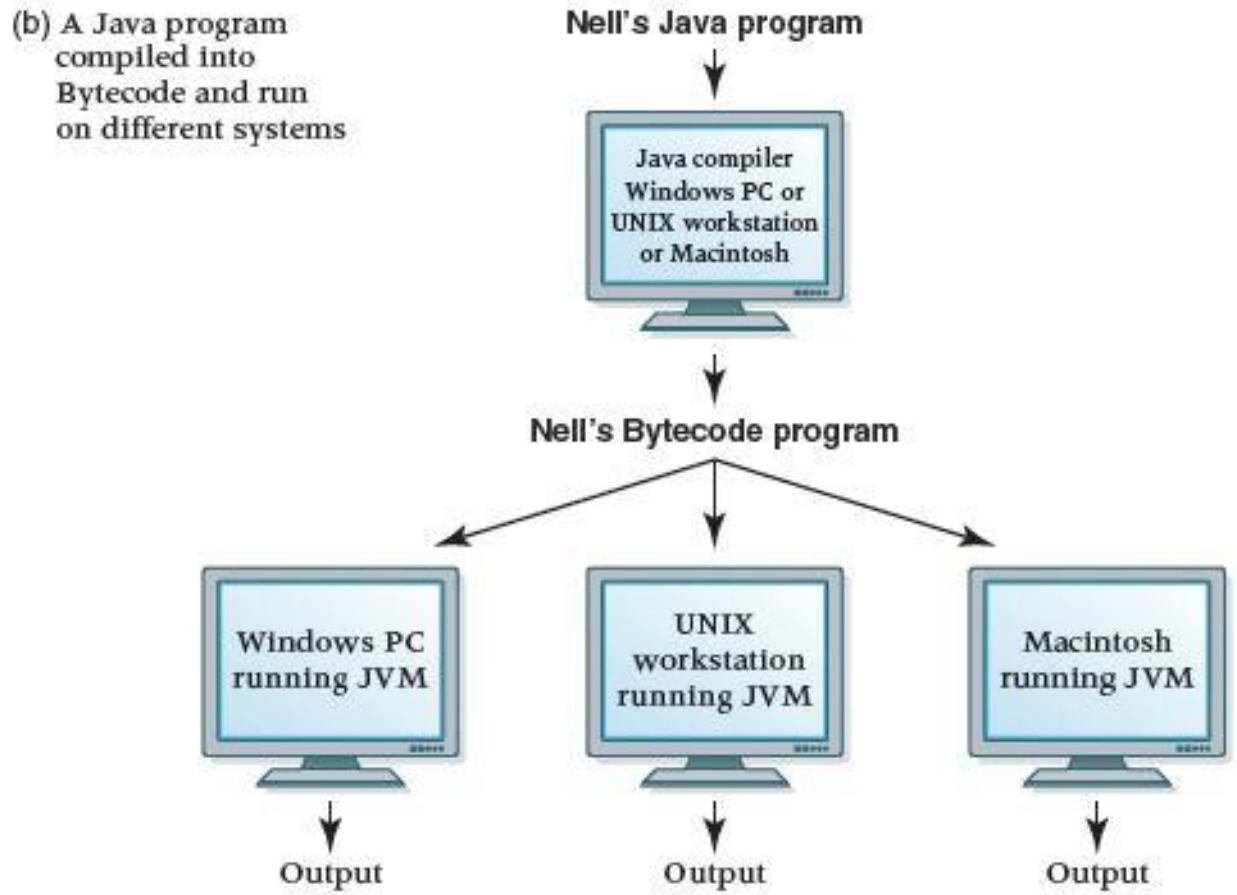


FIGURE 9.3 Portability provided by standardized languages versus interpretation by Bytecode

Programming Language Paradigms

Imperative Paradigm

Program describes the processing

Declarative Paradigm

Program describes the results

Each of these major paradigms have distinct subparadigms

Programming Language Paradigms

- Imperative
  - Procedural
    - Characterized by sequential instructions
    - A program in which statements are grouped into a hierarchy of subprograms
    - Fortran, C, C++
  - Object-oriented model
    - Program consists of a set of objects and the interactions among the objects
    - Python, Java, Smalltalk, Simula

Programming Language Paradigms

C++ is a procedural language with some object-oriented features

Java is an object-oriented language with some procedural features

Programming Language Paradigms

- Declarative
  - Functional
    - Based on the mathematical concept of a function
    - Lisp, Scheme, and ML
  - Logic
    - Based on principles of symbolic logic
    - Types of statements
      - declares facts about objects and relationships
      - defines rules about objects
      - asks questions about objects
    - PROLOG

Scheme

```
#;> (* 3 4)
12
#;> (+ (* 5 4)(+ 1 4))
25
#;> (length '(2 4 6 8 10))
5
#;> (max 2 5 1 3)
5
```

Scheme

```
#;> (define factorial
#;> (lambda(n)
#;>   (if
#;>   (= n 0)
#;>  1
#;>  (* n (factorial (- n 1))))))
#;> (factorial 7)
5040
```

Compare to pseudocode algorithm

PROLOG

Pets to owners

```
owns(mary,bo).
owns(ann,kitty).
owns(bob,riley).
owns(susy,charlie).

?-owns(mary,bo)
yes
?-owns(bo,mary)
no
?-owns(susy,bo)
no
```

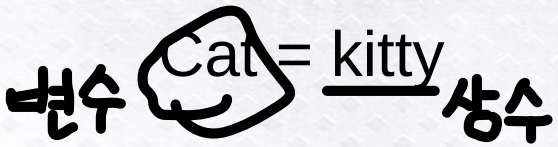
States facts

Asks questions

PROLOG

?-owns(ann, Cat).

Cat = kitty



?-owns(Name,charlie).

Name = susy

Upper case is variable; lower case is constant

Functionality of High-Level Languages

We examine procedural and object-oriented languages in the rest of this chapter by looking at the functionality provided in these languages

We give examples in different languages to show how syntax used to provide the functionality

Functionality of Imperative Languages

Sequence

Executing statements in sequence until an instruction is encountered that changes this sequencing

Selection

Deciding which action to take

Iteration (looping)

Repeating an action

Do these concepts sound familiar?
Let's review them

Boolean Expressions

Boolean expression

A sequence of identifiers, separated by compatible operators, that evaluates to *true* or *false*

A Boolean expression can be

- A Boolean variable
- An arithmetic expression followed by a relational operator followed by an arithmetic expression
- A Boolean expression followed by a Boolean operator followed by a Boolean expression

Boolean Expressions

Remember the relational operators?
List them!

Strong Typing

Data type

A description of the set of values and the basic set of operations that can be applied to values of the type

Strong typing

The requirement that only a value of the proper type can be stored into a variable

Data Types

Integer numbers

Real numbers

Characters

Boolean values

Strings

Give examples of each

Integers

What determines the range of an integer value?

Is the range of an integer value the same in all languages?

What operations can be applied to integers?

Reals

How are real values like integer values?

How do real values differ from integer values?

Characters

Do you remember

ASCII?

Extended ASCII?

UNICODE?

How many characters in Extended ASCII?

How many characters in UNICODE mapping?

What does a relational operator between two characters mean?

Boolean and Strings

What values can a Boolean variable be?

For what are Boolean expressions used?

What is a string?

What operations can be applied to strings?

Declarations

Declaration

A statement that associates an **identifier** with a **variable**, an **action**, or some other **entity** within the language that can be given a name; the programmer can refer to that item by name

Reserved word

A word in a language that has special meaning

Case-sensitive

Uppercase and lowercase letters are considered the same

Declaration Example

Language	Variable Declaration
Python	None required

VB .NET	`Dim sum As Single = 0.0F ' set up word with 0 as contents`
	`Dim num1 As Integer ' set up a two byte block for num1`
	`Dim num2 As Integer ' set up a two byte block for num2`
	`Dim num3 As Integer ' set up a two byte block for num3`
	`...`
	`Num1 = 1`

C++/Java	`float sum = 0.0; // set up word with 0 as contents`
	`int num1; // set up a two byte block for num1`
	`int num2; // set up a two byte block for num2`
	`int num3; // set up a two byte block for num3`
	`...`
	`Num1 = 1;`

Assignment statement

Assignment statement

An action statement (not a declaration) that says to evaluate the expression on the right-hand side of the symbol and store that value into the place named on the left-hand side

Named constant

A location in memory, referenced by an identifier, that contains a data value that cannot be changed

Remember?

Input/Output Structures

Pseudocode algorithms used the expressions *Read* or *Get* and *Write* or *Print*

High-level languages view input data as a stream of characters divided into lines

Key to the processing

The data type determines how characters are to be converted to a bit pattern (input) and how a bit pattern is to be converted to characters (output)

Input/Output Structures

Read name, age, hourlyWage

name is a string;
age is an integer;
hourlyWage is a real

The data must be a string, an integer, and a real in that order.

Input/Output Structures

Language	Input and Output Statements
C++	`cin >> name >> age >> hourlyWage;` `cout << name << age << hourlyWage;`

Java	`Scanner inData;`
	`inData = new Scanner(system.in);`
	`name = inData.nextLine();`
	`age = inData.nextInt();`
	`hourlyWage = inData.nextFloat();`
	`System.out.println(name, ' ',`
	`  age, ' ', hourlyWage);`

Python	`name = input()`
	`age = input()`
	`hourlyWage = input()`
	`print name, age, hourlyWage`

VB .NET	Uses windowing

Control Structures

Control structures

An instruction that determines the order in which other instructions in a program are executed

Can you name the ones we defined in the functionality of pseudocode?

Selection Statements

The *if* statement allows the program to test the state of the program variables using a Boolean expression

Language	*if* Statement

Python	`if temperature > 75:`
	`  print "No jacket is necessary"`
	`else:`
	`  print "A light jacket is appropriate"`
	`# Idention marks grouping`

VB .NET	`If (Temperature > 75) Then`
	`  MsgBox("No jacket is necessary")`
	`Else`
	`  MsgBox("A light jacket is appropriate")`
	`End If`

C++	`if (temperature > 75)`
	`  cout << "No jacket is necessary";`
	`else`
	`  cout << "A light jacket is appropriate";`

Java	`if (temperature > 75)`
	`  System.out.print ("No jacket is necessary");`
	`else`
	`  System.out.print ("A light jacket is appropriate");`

Looping Statements

Language	Count-Controlled Loop with a *while* Statement

Python	`count = 0`
	`while count < limit:`
	`    ...`
	`    count = count + 1`
	`# Indention marks loop body`

VB .NET	`Count = 1`
	`While (Count <= Limit)`
	`    ...`
	`    Count = Count + 1`
	`End While`

C++/Java	`count = 1;`
	`while (count <= limit)`
	`{`
	`    ...`
	`    count = count + 1;`
	`}`

Subprogram Statements

We can give a section of code a name and use that name as a statement in another part of the program

When the name is encountered, the processing in the other part of the program halts while the named code is executed

Remember?

Subprogram Statements

Language	Subprogram declaration

VB .NET	`Public Sub Example(ByVal one As Integer,`
	`  ByVal two As Integer,`
	`  ByRef three As Single)`
	`...`
	`End Sub`

C++	`void Example(int one, int two, float& three)`
	`{`
	`...`
	`}`

Nested Logic

```
Set sum to 0              // Initialize sum
Set posCount to 0         // Initialize event
WHILE (posCount <= 10)    // Test event
    Read a value
    IF (value > 0)        // Update event?
        Set posCount to posCount + 1
        // Update event
        Set sum to sum + value
// Statement(s) following loop
```

IF within a WHILE

```
Set weekCount to 1
WHILE (weekCount<= 52)
    Set weekSum to 0
    Set dayCount to 1
    WHILE (dayCount <= 7)
        Read rainfall
        Set weekSum to weekSum + rainfall
        Set dayCount to dayCount + 1
    Write "Week " + weekCount + "total: " +
        weekSum
    Set weekCount to weekCount +
```

WHILE within a WHILE

```
Set weekCount to 1
WHILE (weekCount<= 52)
    Set weekSum to CalculateWeekSum(weekCount)
    Write "Week " + weekCount + " total: " +
        weekSum
    Set weekCount to weekCount +
CalculateWeekSum(weekCount)
…..
```

Which is easier to read?

Asynchronous Processing

Asynchronous processing

Not synchronized with the program's action

- *Clicking* has become a major form of input to the computer
- Mouse clicking is not within the sequence of the program
- A user can click a mouse at any time during the execution of a program

Functionality of OOPs

Encapsulation

A language feature that enforces information hiding

Classes

Different meanings in different places (See next slide)

Inheritance

A property that allows a class to inherit the data and actions of another class

Polymorphism

An ability to handle the ambiguity of duplicate names

Functionality of OOPs

Object class (problem-solving phase)

An entity or thing that is relevant in the context of a problem

Object class (class) (problem-solving phase)

A description of a group of objects with similar properties and behaviors

Class (implementation phase) A pattern for an object

Object (implementation phase) An instance of a class

Class Definition

A class encapsulates both data and actions

```
public class Person   // Name the class
    // Declare Class variables
    Name name
    String telephone
    String email
```

Class Definition

```
// Declare Class Methods
Initialize()        // Code for Initialize
public Print()      // Code for Print
public Name GetName()
    RETURN Name
public String GetEmail()
    RETURN email
public String GetTelephone()
    RETURN telephone
```

Class Definition

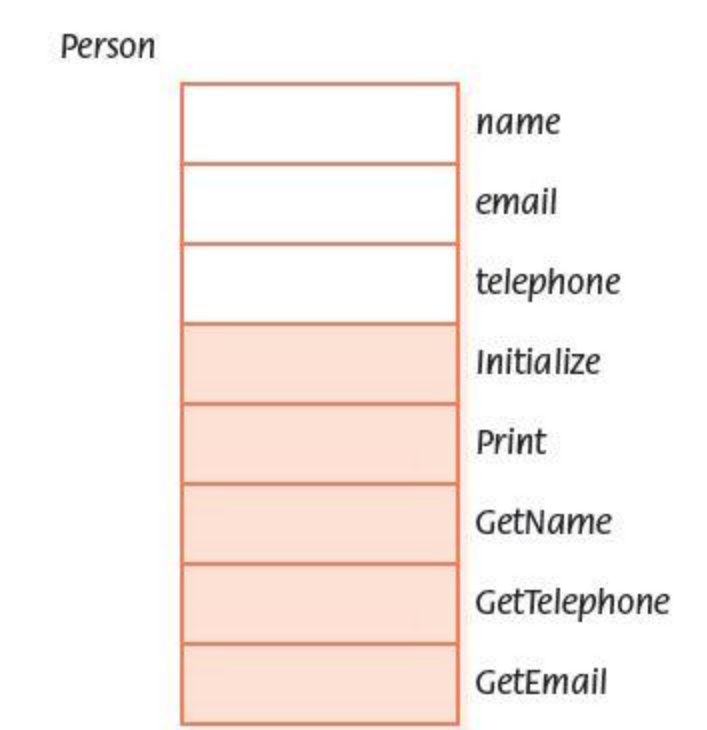


FIGURE 9.4 Class Person

Class Definition

```
Name aName = new Name()
aName.Initialize("Frank", "Jones")
Person aPerson = new Person()
aPerson.Initialize(aName, telephone, email)
aPerson.Print()
Write "Name: ", aPerson.GetName().Print()
Write " Telephone: ", aPerson.GetTelephone()
Write " Email: ", a Person.GetEmail()
```

Class Definition

To get an object of a class, we must ask that one be created (instantiated). The new operator does this for us

Person myPerson = new Person()

Student myStudent = new Student()

myPerson.Initialize(…)

myStudent.Initialize(…)

myPerson.Print()

myStudent.Print()

Inheritance and Polymorphism

Inheritance

A construct that fosters reuse by allowing an application to take an already-tested class and derive a class from it that inherits the properties the application needs

Polymorphism

The ability of a language to have duplicate method names in an inheritance hierarchy and to apply the method that is appropriate for the object to which the method is applied

Inheritance and Polymorphism

Inheritance and polymorphism work together

How?

They combine to allow the programmer to build useful hierarchies of classes that can be put into a library to be reused in different applications

Top-Down vs OO Designs

Top-down Solution

Data structures needed in solution are determined

Subprograms are written to manipulate the the data structures

Main program declares data structure

Main program calls to the subprograms, passing data structures as parameters

Top-Down vs OO Designs

Object-oriented Solution

ADTs needed in solution are determined

ADTs are written only if not in library

Data structure is encapsulated within the class that implements the ADT

Main program is instructions to ADTs to perform the necessary tasks

Ethical Issues

Gambling on the Internet

Have you ever visited an Internet gambling site?

Should Internet gambling be outlawed?

Should Internet gambling be legalized and regulated?

Should Internet gambling be taxed?

Ethical Issues

Computer Hoaxes and Scams

What is the principal difference between a hoax and a scam?

What are the most common complaints of Internet users about computer scams and hoaxes?

What are the most serious crimes perpetrated on the Web?

Why is it so difficult to police these schemes?

Who am I?

Courtesy of Staci Norman, UTCS, The University of Texas at Austin

I am best known for structured programming. Can you define it?
I am also known for my wit. Can you recall some of my witty sayings?

Do you know?

How are computers used in tennis tournaments?

What does 78% of software downloaded from websites and peer-to-peer networks contain?

What predated the functionality of Bytecode?

What does the word "paradigm" mean? How has its meaning changed over time?

How many definitions can you think of for "bow"?