Introduction

- Class
- Who I am
  - Qinghua Li
- Who you are
  - Homework 0: Upload your photo to Blackboard (FirstName.LastName.jpg)

Quick Surveys

- What prog. language was used in PF I?
- What prog. language was used in PF II?
- Who is confident in Java?
- Who is confident in Python?
- Who is familiar with web development?
- Who is familiar with socket programming?

Syllabus

- Homework 1: Read the Syllabus

Chapter 1 Introduction to Java

Java How to Program, 10/e

Learning Objectives

- Describe object-oriented programming basics
- Understand different types of programming languages
- Learn a typical Java-development environment
- Test-drive a Java application

1.1 Introduction

- Java is one of the world's most widely used computer programming languages.
- You'll learn *object-oriented* programming—today's key programming methodology.

1.1 Introduction (Cont.)

- The preferred language for many enterprises' programming need is Java.
- Java is widely used for implementing Internet-based applications and network-based software systems.
- According to Oracle, 97% of enterprise desktops, 89% of PC desktops, and three billion devices run Java. (http://www.oracle.com/technetwork/articles/java/javaone12review-1863742.html)

Devices		
Airplane systems	ATMs	Automobile infotainment systems
Blu-ray Disc™ players	Cable boxes	Copiers
Credit cards	CT scanners	Desktop computers
e-Readers	Game consoles	GPS navigation systems
Home appliances	Home security systems	Light switches
Lottery terminals	Medical devices	Mobile phones
MRIs	Parking payment stations	Printers
Transportation passes	Robots	Routers
Smart cards	Smart meters	Smartpens
Smartphones	Tablets	Televisions
TV set-top boxes	Thermostats	Vehicle diagnostic systems

Fig. 1.1 | Some devices that use Java.

1.1 Introduction (Cont.)

Java Standard Edition

- Java How to Program, 10/e is based on Java Standard Edition 7 (Java SE 7) and Java Standard Edition 8 (Java SE 8)
- Java Standard Edition contains the capabilities needed to develop desktop and server applications.

1.1 Introduction (Cont.)

Java Enterprise Edition

- The Java Enterprise Edition (Java EE) is geared toward developing large-scale, distributed networking applications and web-based applications.

1.1 Introduction (Cont.)

- Java Micro Edition (Java ME)
  - a subset of Java SE.
  - geared toward developing applications for resource-constrained embedded devices, such as
    - Smartwatches
    - MP3 players
    - television set-top boxes
    - smart meters (for monitoring electric energy usage)
    - and more.

1.5 Introduction to Object Technology

- Objects
  - E.g., audio objects, vehicle objects, and people objects
  - Almost any *noun* can be represented as a software object in terms of *attributes* (e.g., name, color and size) and *behaviors* (e.g., calculating, moving and communicating).
- Objects are defined by *classes*
- Object-oriented programing is much more productive than “structured programming”
  - Easier to understand, modify, and reuse

1.5.1 The Automobile as an Object

- The Automobile as an Object
  - Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal.*
  - Before you can drive a car, someone has to *design* it.
  - A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house.
  - Drawings include the design for an accelerator pedal.
  - Pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal hides the mechanisms that slow the car, and the steering wheel "hides" the mechanisms that turn the car.

1.5.1 The Automobile as an Object (Cont.)

- Enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.
- Before you can drive a car, it must be *built* from the engineering drawings that describe it.
- A completed car has an *actual* accelerator pedal to make it go faster, but even that's not enough—the car won't accelerate on its own (hopefully!), so the driver must press the pedal to accelerate the car.

1.5.2 Methods and Classes

- In Java, we create a program unit called a class to house the attributes of car and the set of methods that perform the class's tasks.
  - A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.
- Performing a task in a program requires a method.
- The method houses the program statements that actually perform its tasks.
- Hides these statements from its user, just as the accelerator pedal of a car hides from the driver the mechanisms of making the car go faster.

1.5.3 Instantiation

- Just as someone has to *build* a car from its engineering drawings before you can actually drive a car, you must *build an object* of a class before a program can perform the tasks that the class's methods define.
- An object is then referred to as an instance of its class.

1.5.4 Reuse

- Just as a car's engineering drawings can be *reused* many times to build many cars, you can reuse a class many times to build many objects.
- Reuse of existing classes when building new classes and programs saves time and effort.
- Reuse also helps you build more reliable and effective systems, because existing classes and components often have undergone extensive *testing, debugging* and *performance* tuning.
- Just as the notion of *interchangeable parts* was crucial to the Industrial Revolution, reusable classes are crucial to the software revolution that has been spurred by object technology.

Software Engineering Observation 1.1

Use a building-block approach to creating your programs. Avoid reinventing the wheel—use existing high-quality pieces wherever possible. This software reuse *is a key benefit of object-oriented programming.*

1.5.5 Messages and Method Calls

- When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster.
- Similarly, you *send messages to an object*.
- Each message is implemented as a method call that tells a method of the object to perform its task.

1.5.6 Attributes and Instance Variables

- A car has *attributes*
- Color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading).
- The car's attributes are represented as part of its design in its engineering diagrams.
- Every car maintains its ***own*** attributes.
- Each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.

1.5.6 Attributes and Instance Variables (Cont.)

- An object, has attributes that it carries along as it's used in a program.
- Attributes are specified by the class's instance variables.

1.5.7 Encapsulation

- Classes (and their objects) encapsulate, i.e., encase, their attributes and methods.
- Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves.
- Information hiding, as we'll see, is crucial to good software engineering.

Exercise: Define a Class for Personal Bank Accounts

- Attributes
- Methods

1.5.8 Inheritance

- A new class of objects can be created conveniently by inheritance—the new class (called the subclass) starts with the characteristics of an existing class (called the superclass), possibly customizing them and adding unique characteristics of its own.
- In our car analogy, an object of class "convertible" certainly *is an* object of the more *general* class "automobile," but more *specifically*, the roof can be raised or lowered.

1.5.9 Interfaces

- Interfaces are collections of related methods that typically enable you to tell objects *what* to do, but not *how* to do it.
- In the car analogy, a "basic-driving-capabilities" interface consisting of a steering wheel, an accelerator pedal and a brake pedal would enable a driver to tell the car what to do.
- Once you know how to use this interface for turning, accelerating and braking, you can drive many types of cars, even though manufacturers may implement these systems differently.

1.5.9 Interfaces (Cont.)

- A class implements zero or more interfaces, each of which can have one or more methods, just as a car implements separate interfaces for basic driving functions, controlling the radio, controlling the heating and air conditioning systems, and the like.
- Just as car manufacturers implement capabilities *differently*, classes may implement an interface's methods *differently*.

1.5.10 Object-Oriented Analysis and Design (OOAD)

- How will you create the code (i.e., the program instructions) for your programs?
- Follow a detailed analysis process for determining your project's requirements (i.e., defining *what* the system is supposed to do)
- Develop a design that satisfies them (i.e., specifying *how* the system should do it).
- Carefully review the design (and have your design reviewed by other software professionals) before writing any code.

1.5.10 Object-Oriented Analysis and Design (OOAD) (Cont.)

- Analyzing and designing your system from an object-oriented point of view is called an object-oriented-analysis-and-design (OOAD) process.
- Languages like Java are object oriented.
- Object-oriented programming (OOP) allows you to implement an object-oriented design as a working system.

1.7 Programming Languages

Programming language	Description
Fortran	Fortran (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s for scientific and engineering applications that require complex mathematical computations. It's still widely used, and its latest versions support object-oriented programming.
COBOL	COBOL (COmmon Business Oriented Language) was developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users based on a language developed by Grace Hopper, a U.S. Navy Rear Admiral and computer scientist who also advocated for the international standardization of programming languages. COBOL is still widely used for commercial applications that require precise and efficient manipulation of large amounts of data. Its latest version supports object-oriented programming.
Pascal	Research in the 1960s resulted in *structured programming*—a disciplined approach to writing programs that are clearer, easier to test and debug and easier to modify than large programs produced with previous techniques. One result of this research was the development in 1971 of the Pascal programming language, which was designed for teaching structured programming and was popular in college courses for several decades.

Fig. 1.5 | Some other programming languages. (Part 1 of 4.)

1.7 Programming Languages

Programming language	Description
Ada	Ada, based on Pascal, was developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s. The DOD wanted a single language that would fill most of its needs. The Ada language was named after Lady Ada Lovelace, daughter of the poet Lord Byron. She's credited with writing the world's first computer program in the early 1800s (for the Analytical Engine mechanical computing device designed by Charles Babbage). Ada also supports object-oriented programming.
Basic	Basic was developed in the 1960s at Dartmouth College to familiarize novices with programming techniques. Many of its latest versions are object oriented.
C	C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most of the code for general-purpose operating systems is written in C or C++.
C++	C++, which is based on C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides several features that "spruce up" the C language, but more important, it provides capabilities for object-oriented programming.

Fig. 1.5 | Some other programming languages. (Part 2 of 4.)

1.7 Programming Languages

Programming language	Description
Objective-C	Objective-C is another object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It has become the key programming language for the OS X operating system and all iOS-powered devices (such as iPods, iPhones and iPads).
Visual Basic	Microsoft's Visual Basic language was introduced in the early 1990s to simplify the development of Microsoft Windows applications. Its latest versions support object-oriented programming.
Visual C#	Microsoft's three object-oriented primary programming languages are Visual Basic (based on the original Basic), Visual C++ (based on C++) and Visual C# (based on C++ and Java, and developed for integrating the Internet and the web into computer applications).
PHP	PHP, an object-oriented, open-source scripting language supported by a community of users and developers, is used by millions of websites. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports many databases, including the popular open-source MySQL.

Fig. 1.5 | Some other programming languages. (Part 3 of 4.)

1.7 Programming Languages

Programming language	Description
Perl	Perl (Practical Extraction and Report Language), one of the most widely used object-oriented scripting languages for web programming, was developed in 1987 by Larry Wall. It features rich text-processing capabilities.
Python	Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam (CWI), Python draws heavily from Modula-3—a systems programming language. Python is "extensible"—it can be extended through classes and programming interfaces.
JavaScript	JavaScript is the most widely used scripting language. It's primarily used to add dynamic behavior to web pages—for example, animations and improved interactivity with the user. It's provided with all major web browsers.
Ruby on Rails	Ruby, created in the mid-1990s, is an open-source, object-oriented programming language with a simple syntax that's similar to Python. Ruby on Rails combines the scripting language Ruby with the Rails web application framework developed by 37Signals. Their book, *Getting Real* (`gettingreal.37signals.com/toc.php`), is a must read for web developers. Many Ruby on Rails developers have reported productivity gains over other languages when developing database-intensive web applications.

Fig. 1.5 | Some other programming languages. (Part 4 of 4.)

1.8 Java

- Microprocessors have had a profound impact in intelligent consumer-electronic devices.
- 1991
  - Recognizing this, Sun Microsystems funded an internal corporate research project led by James Gosling, which resulted in a C++-based object-oriented programming language that Sun called Java.
  - Key goal of Java is to be able to write programs that will run on a great variety of computer systems and computer-controlled devices.
  - This is sometimes called "***write once, run anywhere***."

1.8 Java (Cont.)

- 1993
  - The web exploded in popularity
  - Sun saw the potential of using Java to add *dynamic content* to web pages.
- Java drew the attention of the business community because of the phenomenal interest in the web.
- Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices and for many other purposes.

1.8 Java (Cont.)

Java Class Libraries

- Rich collections of existing classes and methods
- Also known as the Java APIs (Application Programming Interfaces).

Performance Tip 1.1

Using Java API classes and methods instead of writing your own versions can improve program performance, because they're carefully written to perform efficiently. This also shortens program development time.

1.9 A Typical Java Development Environment

- Normally there are five phases
  - edit
  - compile
  - load
  - verify
  - execute.
- See the Before You Begin section for information on downloading and installing the JDK on Windows, Linux and OS X.

1.9 A Typical Java Development Environment (Cont.)

- Phase 1 consists of editing a file with an *editor program*
  - Using the editor, you type a Java program (source code).
  - Make any necessary corrections.
  - Save the program.
  - Java source code files are given a name ending with the .java extension indicating that the file contains Java source code.

1.9 A Typical Java Development Environment (Cont.)

- Linux editors: `vi` and `emacs`.
- Windows provides Notepad.
- OSX provides TextEdit.
- Many freeware and shareware editors available online:
  - Notepad++ (`notepad-plus-plus.org`)
  - EditPlus (`www.editplus.com`)
  - TextPad (`www.textpad.com`)
  - jEdit (`www.jedit.org`).
- Integrated development environments (IDEs)
  - Provide tools that support the software development process, such as editors, debuggers for locating logic errors (errors that cause programs to execute incorrectly) and more.

1.9 Java and a Typical Java Development Environment (Cont.)

- Popular Java IDEs
  - Eclipse (`www.eclipse.org`)
  - NetBeans (`www.netbeans.org`)
  - IntelliJ IDEA (`www.jetbrains.com`)
- On the book's website at `www.deitel.com/books/jhtp10`
  - Dive-Into® videos that show you how to execute this book's Java applications and how to develop new Java applications with Eclipse, NetBeans and IntelliJ IDEA.

1.9 A Typical Java Development Environment (Cont.)

- Phase 2: Compiling a Java Program into Bytecodes
  - Use the command javac (the Java compiler) to compile a program. For example, to compile a program called `Welcome.java`, you'd type
    - `javac Welcome.java`
  - If the program compiles, the compiler produces a .class file called `Welcome.class` that contains the compiled version.

1.9 A Typical Java Development Environment (Cont.)

- Java compiler translates Java source code into bytecodes that represent the tasks to execute.
- The Java Virtual Machine (JVM)—a part of the JDK and the foundation of the Java platform—executes bytecodes.
- Virtual machine (VM)—a software application that simulates a computer
  - Hides the underlying operating system and hardware from the programs that interact with it.
- If the same VM is implemented on many computer platforms, applications written for that type of VM can be used on all those platforms.

1.9 A Typical Java Development Environment (Cont.)

- Bytecode instructions are *platform independent*
- Bytecodes are portable
  - The same bytecode instructions can execute on any platform containing a JVM that understands the version of Java in which the bytecode instructions were compiled.
- The JVM is invoked by the java command. For example, to execute a Java application called `Welcome`, you'd type the command
  - `java Welcome`

1.9 A Typical Java Development Environment (Cont.)

- Phase 3: Loading a Program into Memory
  - The JVM places the program in memory to execute it—this is known as loading.
  - Class loader takes the `.class` files containing the program's bytecodes and transfers them to primary memory.
  - Also loads any of the `.class` files provided by Java that your program uses.
- The `.class` files can be loaded from a disk on your system or over a network.

1.9 A Typical Java Development Environment (Cont.)

- Phase 4: Bytecode Verification
  - As the classes are loaded, the bytecode verifier examines their bytecodes
  - Ensures that they're valid and do not violate Java's security restrictions.
- Java enforces strong security to make sure that Java programs arriving over the network do not damage your files or your system (as computer viruses and worms might).

1.9 A Typical Java Development Environment (Cont.)

- Phase 5: Execution
  - The JVM executes the program's bytecodes.
  - JVMs typically execute bytecodes using a combination of interpretation and so-called just-in-time (JIT) compilation.
  - Analyzes the bytecodes as they're interpreted
  - A just-in-time (JIT) compiler—such as Oracle's Java HotSpot™ compiler—translates the bytecodes into the underlying computer's machine language.

1.9 A Typical Java Development Environment (Cont.)

- When the JVM encounters these compiled parts again, the faster machine-language code executes.
- Java programs go through *two* compilation phases
- One in which source code is translated into bytecodes (for portability across JVMs on different computer platforms) and
- A second in which, during execution, the bytecodes are translated into *machine language* for the actual computer on which the program executes.

Common Programming Error 1.1

Errors such as division by zero occur as a program runs, so they're called ***runtime errors*** *or* ***execution-time errors****.* ***Fatal runtime errors*** *cause programs to terminate immediately without having successfully performed their jobs.* ***Nonfatal runtime errors*** *allow programs to run to completion, often producing incorrect results.*

1.10 Test-Driving a Java Application

- ***Checking your setup.*** Read the Before You Begin section to confirm that you've set up Java properly on your computer, that you've copied the book's examples to your hard drive and that you know how to open a command window on your system.

1.11 Internet and the World Wide Web

- In the late 1960s, ARPA—the Advanced Research Projects Agency of the Department of Defense—rolled out plans to network the main computer systems of approximately a dozen ARPA-funded universities and research institutions.
- ARPA implemented what quickly became known as the ARPAnet, the precursor of today's Internet.
- Its main benefit proved to be the capability for quick and easy communication via e-mail.
- This is true even on today's Internet, with e-mail, instant messaging, file transfer and social media such as Facebook and Twitter, enabling billions of people worldwide to communicate quickly and easily.

1.11 Internet and the World Wide Web (Cont.)

- The protocol (set of rules) for communicating over the ARPAnet became known as the Transmission Control Protocol (TCP).
- TCP ensured that messages, consisting of sequentially numbered pieces called *packets*, were properly routed from sender to receiver, arrived intact and were assembled in the correct order.

1.11.1 The Internet: A Network of Networks

- In parallel with the early evolution of the Internet, organizations worldwide were implementing their own networks for both intraorganization (that is, within an organization) and interorganization (that is, between organizations) communication.
- One challenge was to enable these different networks to communicate with each other.
- The Internet Protocol (IP) created a true "network of networks," the current architecture of the Internet.
- The combined set of protocols is now called TCP/IP.

1.11.2 The World Wide Web: Making the Internet User-Friendly

- The World Wide Web (simply called "the web") is a collection of hardware and software associated with the Internet that allows computer users to locate and view multimedia-based documents (documents with various combinations of text, graphics, animations, audios and videos) on almost any subject.

1.11.2 The World Wide Web: Making the Internet User-Friendly (Cont.)

- In 1989, Tim Berners-Lee of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via "hyperlinked" text documents.
- Berners-Lee called his invention the HyperText Markup Language (HTML).
- He also wrote communication protocols such as HyperText Transfer Protocol (HTTP) to form the backbone of his new hypertext information system, which he referred to as the World Wide Web.

1.11.2 The World Wide Web: Making the Internet User-Friendly (Cont.)

- In 1994, Berners-Lee founded the World Wide Web Consortium (W3C), devoted to developing web technologies.
- One of the W3C's goals is to make the web accessible to everyone regardless of disabilities, language or culture.

1.11.3 Web Services and Mashups

- Mashup is an applications-development methodology in which you can rapidly develop powerful software applications by combining (often free) complementary web services and other forms of information feeds.

Web services source	How it's used
Google Maps	Mapping services
Twitter	Microblogging
YouTube	Video search
Facebook	Social networking
Instagram	Photo sharing
Foursquare	Mobile check-in
LinkedIn	Social networking for business
Groupon	Social commerce
Netflix	Movie rentals
eBay	Internet auctions
Wikipedia	Collaborative encyclopedia
PayPal	Payments
Last.fm	Internet radio
Amazon eCommerce	Shopping for books and many other products
Salesforce.com	Customer Relationship Management (CRM)

Fig. 1.17 | Some popular web services (`www.programmableweb.com/apis/directory/1?sort=mashups`). (Part 1 of 2.)

Web services source	How it's used
Skype	Internet telephony
Microsoft Bing	Search
Flickr	Photo sharing
Zillow	Real-estate pricing
Yahoo Search	Search
WeatherBug	Weather

Fig. 1.17 | Some popular web services (`www.programmableweb.com/apis/directory/1?sort=mashups`). (Part 2 of 2.)

1.11.4 Ajax

- Ajax helps Internet-based applications perform like desktop applications
- A difficult task, given that such applications suffer transmission delays as data is shuttled back and forth between your computer and server computers on the Internet.
- Applications like Google Maps have used Ajax to achieve excellent performance and approach the look-and-feel of desktop applications.