National University

Faculty of Computer Since and Technology

Object Oriented Programming

Lec 4

Introduction to Object Technology

- Objects, or more precisely, the classes objects are essentially reusable software components.
- Almost any noun can be reasonably represented as a software object in terms of attributes (e.g., name, color and size) and behaviors (e.g., calculating, moving and communicating).

➢ The Automobile as an Object

- Let’s begin with a simple analogy.
- 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.

- 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 the car 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.

➢ Methods and Classes

- 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.
- In Java, we create a program unit called a class to house 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.

➢ 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.

➢ 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 gone through extensive testing, debugging and performance tuning.

➢ Messages and Methods 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.

➢ Attributes

- 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.
- The car’s attributes are represented as part of its design in its engineering diagrams.
- Every car maintains its own attributes.

➢ Encapsulation

- Classes encapsulate (i.e., wrap) attributes and methods into objects.
- 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.

➢ Inheritance

- A new class of objects can be created quickly and conveniently by inheritance—the new class absorbs the characteristics of an existing class, possibly customizing them and adding unique characteristics of its own.

Operating Systems

- Software systems that make using computers more convenient.
- Provide services that allow each application to execute safely, efficiently and concurrently (i.e., in parallel) with other applications.
- The software that contains the core components of the operating system is called the kernel.
- Popular desktop operating systems include Linux, Windows 7 and Mac OS X.
- Popular mobile operating systems used in smartphones and tablets include Google's Android, BlackBerry OS and Apple's iOS (for its iPhone, iPad and iPod Touch devices).

➢ Examples

- Windows—A Proprietary Operating System
- Linux—An Open-Source Operating System
- Android

Java and a Typical Java Development Environment

- Microprocessors are having a profound impact in intelligent consumer-electronic devices.
- Recognizing this, Sun Microsystems funded an internal corporate research project led by James Gosling, which resulted in a C++-based object-oriented programming language 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-control devices.
- This is sometimes called “write once, run anywhere.”

➢ In 1993

- The web exploded in popularity
- Sun saw the potential of using Java to add dynamic content to web pages.
- 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.

➢ Java Class Libraries

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

- Java programs normally go through five phases
- Edit
- Compile
- Load
- Verify
- Execute.

➢ *Phase 1: Creating a Program*

- Type a Java program (source code) using the editor.
- Make any necessary corrections.
- Save the program.
- A file name ending with the .java extension indicates that the file contains Java source code.

➢ *Phase 2: Compiling a Java Program into Bytecodes*

- Java compiler translates Java source code into bytecodes that represent the tasks to execute.
- Bytecodes are executed by the Java Virtual Machine (JVM)—a part of the JDK and the foundation of the Java platform.
- 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 that it executes can be used on all those platforms
- Bytecodes are platform independent
  - They do not depend on a particular hardware platform.
- Bytecodes are portable
  - The same bytecodes can execute on any platform containing a JVM that understands the version of Java in which the bytecodes were compiled.

➢ ***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 bytecode transfers them to primary memory.
- Also loads any of the bytecodes provided by Java that your program uses.
- The bytecodes can be loaded from a disk on your system or over a network.

➢ ***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).

➢ ***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.
- A just-in-time (JIT)compiler—known as the Java HotSpot compiler—translates the bytecodes into the underlying computer's machine language.