Classes and Objects

2 – 4 pm Tuesday 7/1/2008 @JD2211

1

Agenda

- The Background of the Object-Oriented Approach
- Class
- Object
- Package and import

2

Quiz

- Who was the oldest profession in the world?

1. Physician
2. Civil Engineer
3. Computer Scientist and Software Engineer

3

Complexity

- A physician, a civil engineer, and a computer scientist were arguing about what was the oldest profession in the world.
- The physician remarked, "Well, in the Bible, it says that God created Eve from a rib taken out of Adam. This clearly required surgery, and so I can rightly claim that mine is the oldest profession in the world."

4

Complexity

- The civil engineer interrupted, and said, "But even earlier in the book of Genesis, it states that God created the order of the heavens and the earth from out of the chaos. This was the first and certainly the most spectacular application of civil engineering. Therefore, fair doctor, you are wrong: mine is the oldest profession in the world."

5

Complexity

- The computer scientist leaned back in his chair, smiled, and then said confidently, "Ah, but who do you think created the chaos?"
- *Object-Oriented Analysis and Design with Applications*, by Grady Booch

6

Example of Complexity -- Feasibility of Testing to Specification

- Two inputs
  - One has five values
  - The other has seven values
  - How many test cases are needed
  - 5 X 7 = 35
- 30 inputs
  - Each input has four different values
  - How many test cases are required?
  - If a program has 1.1 X 10^{18} possible inputs and one test can be run every microsecond, how long would it take to execute all of the possible inputs?

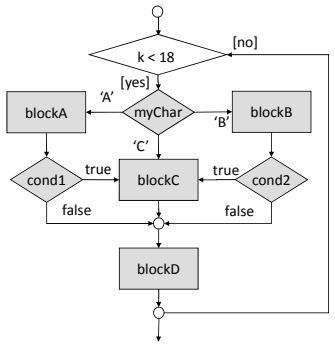
7

Example of Complexity -- Feasibility of Testing to Code

```
Read (kmax)          // kmax is an integer between  1 and 18
for (k = 0; k < kmax; k++) do
{
        read (myChar)       // myChar is the character A, B, or C
        switch (myChar)
        {
                case 'A':
                        block A;
                        if (cond1) blockC;
                        break;
                case 'B':
                        block B;
                        if (cond2) blockC;
                        break;
                case 'C':
                        block C:
                        break;
        }
}
```

8

Example of Complexity -- Feasibility of Testing to Code



9

Software – Essential Difficulties

- "No Silver Bullet: Essence and Accidents of Software Engineering" by Frederick P. Brooks, Jr., 1986
  - http://www.csun.edu/~twang/Java/NoSilverBullet.pdf
  - http://en.wikipedia.org/wiki/No_Silver_Bullet
- Essential difficulties
  - High complexity
  - Low conformity
  - Changeability

10

Past Breakthrough (as of 1986)

- High-level language
- Time-sharing
- Unified programming environment

11

Hope for the Silver (as of 1986)

- Ada and other high-level language advance
- Object-oriented programming
- Artificial intelligence
- Expert system
- Graphical programming
- Program verification
- Environments and tools
- Workstation

12

Hope for the Silver (since1990's)

- Design pattern
- Architecture pattern
- Application frameworks
- Software product lines
- Software factory
- Component-based software
- Service-oriented architecture

13

Foundations of the Object Model

- Classes and objects are basic building blocks
  - Structured design methods use algorithms or functions as their fundamental building blocks
- The object model has been influenced by a number of factors
  - not just to programming languages, but also the design of database, user interface, and even computer architecture

14

Elements of the Object Model

- Abstraction
- Encapsulation
- Object
- Class
- Attribute
- Methods
- Relationships
- Inheritance
- Polymorphism

15

Abstraction

- Denotes the essential characteristics of an object that distinguish it from all other kinds of object, and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

16

Encapsulation

- Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation
- Encapsulation is often achieved through information hiding

17

Abstraction vs. Encapsulation

- Abstraction and encapsulation are complementary concepts
- Abstraction focuses upon the observable behavior of an object
- Encapsulation focuses upon the implementation that gives rise to this behavior

18

Object

- Is anything crisply defined
- An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain
- An object has state, behavior, and identity; the structure and behavior of similar objects are defined in tier common class

19

Attribute

- The object's attributes are the values it stores internally, which may be represented as primitive data or as other objects
- Collectively, the values of an object's attributes define its current state

20

Method

- The methods of an object define its potential behaviors
- A method is a group of programming statements that is given a name
- When a method is invoked, its statements are executed
- A set of methods is associated with an object

21

Class

- An object is defined as a class
- A class is the model or blueprint from which an object is created
- In general, a class contains no space to store data.
- Each object has space for its own data
- Once a class has been defined, multiple objects can be created from that class

22

Class

- In Java, there are *ready-made classes* and classes we have written ourselves
- When you use ready-made class, you do not normally need to know how the class methods are implemented, you just have to know how they are called

23

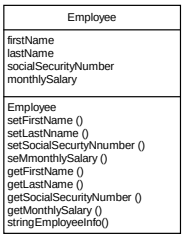
Examples of Classes

Class	Attributes	Operations
Student		
Flight		
Employee		

24

Employee Class

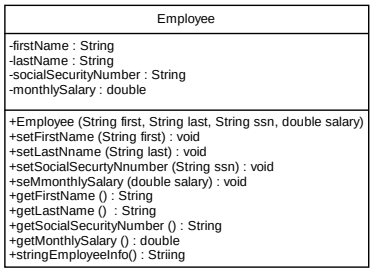
- Attributes
  - First name
  - Last name
  - Social security number
  - Monthly salary
- Operations
  - Set first name
  - Set last name
  - Set social security number
  - Set monthly salary
  - Get monthly salary
  - Print employee information



Analysis-level Employee UML class diagram

25

Employee Class



Design-level Employee UML class diagram

26

```
public class Employee {
   private String firstName;
   private String lastName;
   private String socialSecurityNumber;
   double monthlySalary;

   //constructor

   public Employee (String first, String last, String ssn, double salary)
   {

   }

   public void setFirstName(String first)
   {

   }

   public void setLastNname(String last)
   {

   }

   public void setSocialSecurtyNnumber(String ssn)
   {

   }
```

27

```
    public String getFirstName()
    {

    }

    public String getLastName()
    {

    }

    public String getSocialSecurityNumber()
    {

    }

    public double getMonthlySalary()
    {

    }

    public String stringEmployeeInfo()
    {

    }
}
```

28

Create a "Main" Class

- A main Class contains the main method, that contains a series of instructions to create objects and to tell those objects to do things.

```
public class EmployeeDemo
{
  public static void main (String args[])
  {
    Employee myEmployee = new Employee("George", "Bush", "111-22-3333", 50000.00);
    ........
  }
}
```

29

Creating Objects -- String

- The standard class String is an important example of a ready-made class
- String is a special class
- `String s1 = "Java"`
- s1 is a reference variable which refers to String object, "Java"
- `String s1 = new String ("Java");`

30

The new Operator and Constructor

- Has the same name as the class
- Returns a reference to a newly created object: Are not allowed to have a return type
- There may be several of them but then they must have a different number of parameters, or parameters of different types
- Unless we define our own constructors, a constructor without parameter is defined automatically (and this does nothing).

31

Package

- Java program is supported by a standard library
- A class library is a set of classes that supports the development of programs
- A complier or development environment often comes with a class library

32

Package (cont'd)

- Class libraries can also be obtained separately though third-party vendor
- The String class is not an inherent part of the Java language. It is part of the Java standard class library.
- The class library is made up of several clusters of related classes, called Java APIs or application programming interfaces

33

Package (cont'd)

- The classes of the Java standard class library are grouped into packages. Each class is part of particular package
- The String class is part of the `java.lang` package
- The System class is part of the `java.lang` package as well

34

Some Packages in the Java Standard Class Library

Package	Provides support to
`java.awt`	Draw graphics and create graphical user interfaces; AWT stands for Abstract Windows Toolkit
`Java.io`	Perform a wide variety of input and out functions
`java.lang`	General support; it is automatically imported into all Java programs
`Java.math`	Perform calculations with arbitrary high precision
`java.sql`	Interact with databases
`java.text`	Format text for output
`java.util`	General utilities
`javax.swing`	Create graphical user interfaces with components that executed the AWT capabilities

35

The `import` Declaration

- The classes of `java.lang` package are automatically available for use when writing Java program
- To use classes from any other package, however, we must use an `import` declaration
- For example, `import java.io.*` is used for data input

36

The `Random` Class

- The need for random numbers occurs frequently when writing programs
  - Games
    - The roll of a die, the shuffle of a deck of cards
  - Flight simulators
- The Random class, which is a part of the `java.util` package, represents a random number generator

37

Some Methods of the `Random` Class

- `Random()`
  - Constructor
- `float nextFloat()`
  - Returns a random number between 0.0 (inclusive) and 1.0 (exclusive)
- `int nextInt()`
  - Returns a random number that ranges over all possible `int` value
- `int nextInt (int num)`
  - Returns a random number in the range 0 to num-1

38

Lab Assignments

- Write a program to produces several random numbers in various ranges: 1) over all possible `int` value; 2) from 0 to 9; 3) from 1 to 10; 4) from 20 to 34; 5) from -10 to 9; 6) between 0.0 and 1.0, and 7) between 1.0 and 6.0
- Complete the `Employee` class in the slide and implement the main class to create an `Employee` object and print out the object.
- Construct a class Person. A person should have a name, an address and an age. Make use of the class String. Form a constructor and some appropriate methods for class Person (Exercise 2 on page 79). Also create a person object and print it by implementing the main class.

39