Principles of Computer Science I

Prof. Nadeem Abdul Hamid

CSC 120A - Fall 2004

Lecture Unit 8

Software Design & Implementation

Software Engineering

- Engineering: design, implementation, and analysis of a useful solution to a practical problem
  - Analysis: prototyping, testing, simulation
  - Resource constraints: time, money, safety, aesthetics, marketing, ethics…
- Software Engineering
  - Apply computer science to the benefit of mankind
  - Very young area (~30 years?)
  - No widespread consensus on best practice and engineering standards yet
  - Nonetheless, many common key ideas: abstraction, modular programming, encapsulation, specification, design, coding, testing

Why Software Engineering?

- We are becoming/have become greatly dependent on computers in many aspects of life
- We have a perception that computers (should) function reliably; but reliability depends on the care taken in writing programs
- Consequences of bad or inadequate design, coding, and testing
  - F-18 jet fighter flips upside down the first time it flies across the equator, control software error
  - Rocket launch out of control and had to be self-destructed because comma typed instead of period in program
  - Radiation therapy machine kills several patients because software error causing the machine to operate at full power when operator typed certain commands too quickly
  - Your word processor crashes, causing your term paper to be lost 3 hours before it is due

Software Design Strategies

- Functional Decomposition
  - Older methodology
  - View the solution as a task to be accomplished
  - Outline the sequence of operations needed to complete the task
  - Break the series of steps into smaller, easier-to-solve subproblems
  - Use to develop algorithms for an object's methods
- Object-Oriented Design
  - Focuses on objects in the problem (self-contained entities composed of data and related operations)
  - Identify these components and determine how they need to interact

Object-Oriented Design (OOD)

- Focus is on the entities (objects) in a problem
- Begin by identifying the classes of objects in the problem, and determining appropriate operations to be supported by those objects
- Programs are collections of objects that communicate with (send messages to = call methods of) each other
- A class defines the pattern or blueprint used when instantiating object(s) of that type
- A class generally contains private data (*fields*) and public operations (called *methods*)
  - You can also have public fields (not generally a good practice) and private methods (a common practice)

Why OOD?

- Objects within a program often model real-life objects in the problem to be solved
- The OOD concept of inheritance allows the customization of an existing class to meet particular needs
  - Reduces the time and effort needed to design, implement, and maintain large systems
  - We will discuss inheritance later

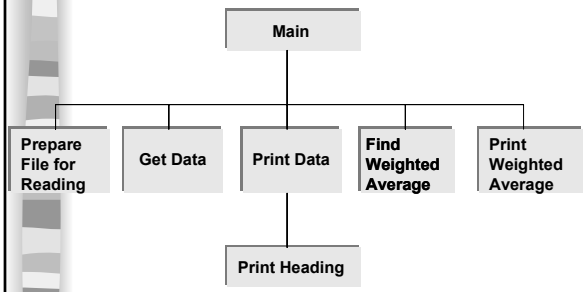
Functional Decomposition

- A technique for developing a program in which the problem is divided into more easily handled subproblems, the solutions of which create a solution to the overall problem
- Work from the abstract (a list of the major solution steps for which some implementation details remain unspecified) to the concrete (algorithmic steps for which the implementation details are fully specified)

Functional Decomposition (cont.)

- Focus on the sequence of actions (algorithms) required to solve the problem
- Begin by breaking the solution into a series of major steps
  - Process continues until each subproblem cannot be divided further or has an obvious solution
- Programs are collections of modules (methods) that solve subproblems
- Data plays a secondary role in support of actions

Module Structure Chart



Object-Oriented Design Process

- Three steps in the process
  - Identify an initial set of object classes that seem relevant to the problem
    - nouns often represent objects
    - verbs often represent actions (operations/responsibilities)
  - Filter the list, eliminating duplicates
  - Identify the responsibilities for the reduced list of objects

Step 1: Identify Possible (Classes of) Objects

- Brainstorming (by a team)
  - Identify objects
  - Propose classes
  - Write on a blackboard
- Keep going around until no one can think of any more objects
- No ideas are rejected

Step 2: Filter

- Eliminate duplicates
- Decide if the classes really do represent objects in the problem
- Look for classes that are related
- Overlooked classes may emerge
- For each class that survives the filtering stage, create a CRC card

Blank CRC Card

Class Name:	Superclass:	Subclasses:
Responsibilities	**Collaborations**	

Responsibilities and Collaborations

- Responsibilities: Actions/operations that an implementation of an object must be capable of performing
- Collaboration: An interaction between objects in which one object requests that another object carry out one of its responsibilities

Step 3: Determine Responsibilities

- Initial responsibilities
  - Know and return the states of the object
  - Called Knowledge Responsibilities ("Accessor methods")
- Action responsibilities
  - Use scenario walk-throughs, a role-playing technique,
    - to explore the actions of a class
    - to explore the interactions of classes

TicTacToe Board CRC Card

Class Name: Board	Superclass:	Subclasses:
Responsibilities	**Collaborations**	
Initialize itself (set up blank board)		
Tell the mark at a given position		
Set a mark at a given position		
Determine if win for given player		
Display board on the screen	Output class (System.out)	

Categories of Responsibilities (Methods)

- Constructor An operation that creates a new instance of a class
- Copy constructor An operation that creates a new instance by copying an existing instance, possibly altering its state in the process
- Transformer/Mutator An operation that changes the state of an object
- Observer/Accessor An operation that allows us to observe the state of an object without changing it

Methods

- Each responsibility is coded as a method of the class
- May define additional utility methods (usually kept private so that they cannot be accessed by code outside the class)
  - Like findRandomPos() in ComputerPlayer

Java Classes Revisited

- Class specifies a __(?)_____ for creating objects
- Once class is define, we can create (instantiate) objects using the **new** operator
  - The **new** operator returns a pointer to the newly created object, which is usually stored in a variable

```
class Name {
  …
}
…
Name testName = new Name();
```

- Class defines data (fields) and operations (methods)

Three Kinds of Data (Variables)

- Instance data ("fields") is the internal representation of a specific object
  - Records the object's state.
- Class data is accessible to all objects of a class
- Local data is specific to a given call of a method

Instance Data (Variables)

- Instance data is the internal representation of a specific object

```
public class Name {
   // Instance variables
   String first;
   String middle;
   String last;
   . . .

  }
```


Class Data (Variables)

- Accessible to all objects of a class
- Fields declared as static belong to the class rather than to a specific instance (object)

```
public class Name {
     // Class constant
     static final String PUNCT = ", ";
          . . .

  }
```


Local Data (Variables)

- Specific to a given call of a method
- JVM allocates space for this data when the method is called and deallocates it when the method returns

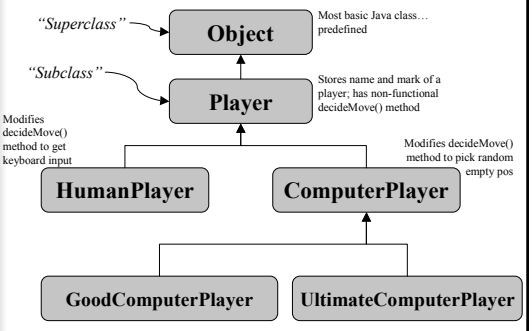
```
public class Name {
   ...

   public int compareTo(Name otherName) {
     int result;    // Local variable
     . . .
     return  result;
   }
}
```


Inheritance

- A mechanism that enables us to define a new class by adapting the definition of an existing class
  - Reuse or extend existing functionality
  - Define new functionality
- Example on next slide
- Detailed explanation in Chapter 7
  - We will probably jump to Chapter 10 (Arrays) and then come back to Chapter 7 and 9

TicTacToe Player Objects Inheritance Hierarchy



Programming and Design

- From textbook, page 300:

  In the 1960s, IBM developed a major new operating system called OS/360, which was one of the first true examples of programming in the large. After the operating system was written, more than 1,000 significant errors were found. Despite years of trying to fix these errors, the programmers never did get the number of errors below 1,000; and sometimes the "fixes" produced far more errors than they eliminated…

  Hindsight analysis showed that the code was badly organized and that different pieces were so interrelated that no one could keep all of it straight. A seemingly simple change in one part of the code caused several other parts of the system to fail. Eventually at great expense, an entirely new system was created using better organization and tools…

  Fortunately, most large, life-critical, software development efforts today use a combination of good methodology, appropriate [programming] language, and extensive organizational tools–that is, *software engineering*.

Mutable and Immutable Objects

- Fields in a class definition specify the state of an object of that class
- State: the information stored in an object at any given time
- Immutable objects: state cannot be changed once the object has been created and initialized with a constructor
  - String objects
- Mutable objects: state can be changed
  - Board object (in the TicTacToe game)

Bogus Class I

```
public class BogusI {
    private int a;
    private int b;
    public int c;
    public static int n = 0;

    public BogusI(int aa, int bb, int cc) {
       a = aa;  b = bb;  c = cc;  n++;
    }

    public void foo(int x) {
       a = a + x;
       if (x < 3) return;
       bar(x*2);
       c = a * b;
    }

    private void bar(int x) {
       b = b - x;
    }

}
```

```
public class MainBogus {
    public static void main(String[] args)
    {
        BogusI b1 = new BogusI(1,2,3);
        BogusI b2 = new BogusI(4,5,6);

        // error: System.out.println(b1.a);
        System.out.println(b1.c);
        System.out.println(b1);
        b1.foo(4);
        System.out.println(b1);

        System.out.println(b2);
        b2.foo(1);
        System.out.println(b2);

        System.out.println(b1.n);
        System.out.println(b2.n);
    }
}
```


Bogus Class II

```
public class BogusII {
    private final int a;
    private final int b;
    public final int c;
    public static int n = 0;

    public BogusII(int aa, int bb, int cc) {
       a = aa;  b = bb;  c = cc;  n++;
    }

    public BogusII foo(int x) {
       int newa = a + x;
       if (x < 3)
          return new BogusII(newa, b, c);
       BogusII b2 = bar(x*2);
       return
          new BogusII(newa, b2.b,
                      newa * b2.b);
    }

    private BogusII bar(int x) {
       return new BogusII(a, b - x, c);
    }
}
```

```
public class MainBogus {
    public static void main(String[] args)
    {
        BogusII b1 = new BogusII(1,2,3);
        BogusII b2 = new BogusII(4,5,6);

        System.out.println(b1);
        BogusII b3 = b1.foo(4);
        System.out.println(b1);
        System.out.println(b3);

        System.out.println(b2);
        b3 = b2.foo(1);
        System.out.println(b2);
        System.out.println(b3);

        System.out.println(b1.n);
        System.out.println(b2.n);
    }
}
```


Ethics and Responsibilities in Computing

- Ethics: the rules and standards governing the conduct of an individual with others
- Why computer ethics?
  - Growth of WWW has created novel legal issues
  - new questions that older laws cannot answer
  - traditional laws outdated/anachronistic
  - more coherent body of law needed to govern Internet and computers
- Three pressing examples
  - Copyright
  - Privacy
  - Censorship
  - (many others…)

"Squatting"

- Domain names (www.whateverishere.com) being bought and sold to the highest bidder
  - E.g., recently man bought the name www.drugs.com and auctioned it off
- Many people purposely buying up company names and selling them to those companies at outrageous prices.
- Former President Clinton called these people 'Squatters' and wanted to pass a law forbidding them to buy up already existing company names.
  - Is this fair?
  - Do people have a right, under capitalism, to make money this way?
  - Or is it near-blackmail?

Spam

- Many people have been tricked my e-mail scams, claiming that they will make you a millionaire
  - Same sort of pyramid scheme that exists over telephone or mail, but no laws covered it for a while
- Also, many email chain letters have allowed urban legends to spread at an accelerated rate and created alarm over hoaxes concerning many food and drug products
- These emails only bog down email systems and servers, but do not seem to be ending
- Advertisers also email unsolicited ads to email users, in a practice known as spamming
  - How to deal with this?

Copyright and Software Piracy

- The unauthorized copying of software (or source code) for either personal use or use by others
- For companies like IBM, Apple, Microsoft, software piracy results in billions of dollars of losses
- Is it legally wrong? Is it ethically wrong?
- Free Software Foundation: Open Source
  - "Free as in freedom", not price
  - GNU operating system (Linux)
  - Richard Stallman, www.gnu.org

Why the Future Doesn't Need Us

- Bill Joy, Sun Microsystems, "Edison of the Internet"
  - "Our most powerful 21st-century technologies – robotics, genetic engineering, and nanotech – are threatening to make humans an endangered species."
  - "We are being propelled into this new century with no plan, no control, no brakes. Have we already gone too far down the path to alter course? I don't believe so, but we aren't trying yet, and the last chance to assert control – the fail-safe point – is rapidly approaching…"