Information System Design (IT60105)

Lecture 26

Object-Oriented System Testing

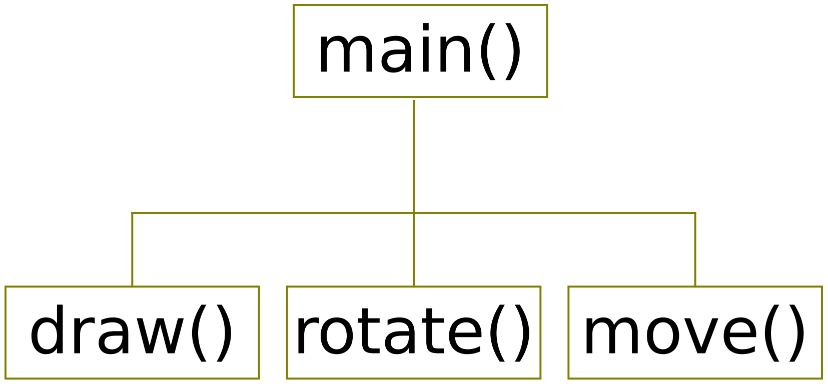
Lecture #23

- Procedural vs OO paradigms
- Why not Traditional Testing?
- Issues
- Methodology

Procedural Vs OO paradigms

- **Procedural Vs OO programs**
- **Procedural Vs OO Software development**

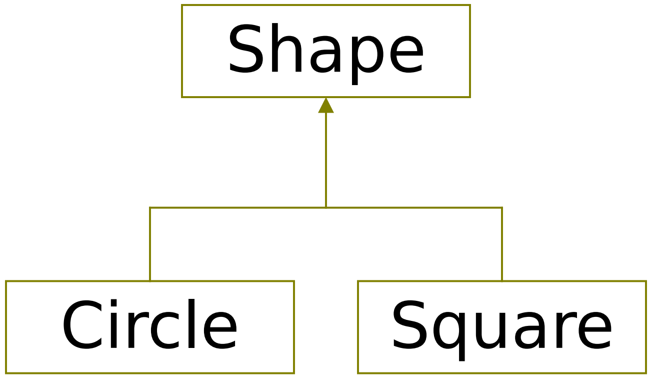
Example: Procedural Program



This is a procedural program for graphical tool. Two shapes are there: circle, square. Each have functions draw(), rotate(), move()

```
int x, y, radious, length, sh_type;
float angle;
enum { CIRCLE=0, SQUARE=1 }
void draw (sh_type){
    case CIRCLE:
    /* code to draw circle */ break;
    case SQUARE:
    /* code to draw square*/ break;
    case default: break;  }
void rotate (int sh_type, int dgree)
    {
    case CIRCLE:
    /*code to rotate circle */ break;
    case SQUARE:
    /*codeto rotate square*/ break;
    case default: break;  }
void move (int x_off,y_off){
    x= x+x_off; y= y+y_off;  }
```

Example: OO Program



```
public class Shape(){
      public int x, y;  float angle;
       public abstract void draw ( );
       public void rotate(int degree);
       public void move(int x_off,
      y_off){
      x= x+x_off; y= y+y_off; }}
public class Circle extends Shape{
      private int radious;
      public void draw(){/* code*/ }
      public void rotate( int dgree){
      draw();  }}
public class Square extends
      Shapes{
      private int length;
      public void draw(){/* code*/ }
      public void rotate(int degree){
      /* code*/ }}
```

This is an Object-oriented program for graphical tool. Two shapes are there: circle, square. Each have functions draw(), rotate(), move()

Note:

1. Methods in oo programs are shorter in oo programs
2. Arbitrary sequence of class methods can call because of reuse

Procedural vs OO programs

Consists of functions and global data	Consists of classes and objects
Communication through function calls and global data	Communication through message passing and inheritance
Control flow	Both data & control flow
Functional decomposition	Recursion and inheritance
Reuse is incidental	Reuse is central

Procedural vs OO Software Development

Primarily waterfall model	Incremental iterative and recursive model
Units=functions	Units=classes
Data & call coupling	Inheritance & message coupling
ER diagrams	Semantic nets
Data flow diagrams	Control flow diagrams
Structured charts	Interaction diagrams
Primarily top-down development	Top-down & bottom-up development
White box testing emphasized	Block box testing emphasized

Analyze a little ,Design a little, Code a little, Test a little

Why not Traditional Testing Adequate

Why not Traditional Testing?

- Traditional testing considers only static binding. so execution order is to be predefined but this not happen in oo programs
- Traditional white box testing not adequate:
  - Traditional testing Consider only on intra-procedural logic and control flow
  - Traditional testing do not Consider interactions among method calls in a class

Why not Traditional Testing? cont

- Traditional block box testing not adequate:
  - Basic OO program code structure is different
  - In oo testing, exhaustive testing is impossible. Because infinite number of method sequences can be possible.
  - E.g.: Observationally equivalent objects may contain variables with different values. Because
    - Object may contain variables that are not important in the given state
    - Object May contain variable invariants

Why not Traditional Testing? cont

- Traditional dataflow testing not adequate:
  - Can be applied both to individual methods in a class and to methods in a class that interact through messages
  - But do not consider dataflow interactions that arise when users of a class invoke sequences of methods in an arbitrary order.

Issues in Testing OO Programs

- **OO Paradigms**
- **Language-Specific features**

OO Paradigms

- State Dependent Behavior
  - The behavior of objects depends on their state. so stateless behavioral testing in not sufficient for oo programs.
- Encapsulation
  - This giving observability problem (private attributes access is not allowed for outside of class). but test oracles required access of all attributes of class

OO Paradigms cont

- Inheritance
  - Subclass can invoke constructors of super class, so constructors should consider in testing
  - Testing of subclass from scratch is expansive. So reuse of superclass tests should consider in testing.
- Polymorphism and Dynamic binding
  - Tests should exercise all possible method bindings of polymorphic method call.
  - Undesidability problem because of dynamic binding

OO Paradigms cont

- Abstract Classes
  - Non-instantiation problem: abstract classes are incomplete. So they can’t be instantiate directly.
  - These can be part of interface elements of libraries or components, so these classes should be test.
- Exception Handling
  - Textual distance between the point where an exception is thrown and the point where it is handle and Dynamic determination of binding should be consider.
- Concurrency
  - Deadlock and race conditions should be consider.

Language-Specific features

- Different languages in using different OO paradigms
- Language specific hazards:
  - C++
    - Naming pollution, friend function, no type safe (dynamic array, pointer, casting), this and new problem and etc.
    - Implicit type coercion with overloaded operators
  - Java
    - Incompatible on different Java Virtual Machines or an executing user's environment.
    - No thread scheduling policy

OO Testing Methodology

Phases of Testing

- Intra Class or Class Testing (Unit)
  - It deals with classes in isolation
  - It includes
    - Method testing by Traditional testing methods
    - Message testing
    - Inheritance testing
    - Exception testing (local)
    - Polymorphism testing (local)
    - Abstract classes testing

Phases of Testing cont

- Inter Class Testing (Integration)
  - In this phase, class interactions are considered
  - It includes
    - Exception testing
    - Polymorphism testing
  - Integration is not hierarchical in OO
    - Coupling is not via subroutine
    - ‘Top-down’ , ‘Bottom-up’ have little meaning
  - Integration Testing can be done in 2 ways
    - Thread-based
    - Use-based (dependent & independent classes)

Phases of Testing cont

- Integration Testing
  - Thread-based testing
    - Integrates the set of classes required to respond to one input or event
    - Integrate one thread at a time
  - Use-based testing
    - Integrate/test independent classes
    - Then, test next layer of (dependent) classes that use the independent classes (layer by layer, or cluster-based)
    - Repeat adding/testing next layer of dependent classes until entire system is constructed
    - Driver classes or methods required to test lower layers

Phases of Testing cont

- System Testing
  - It considers the software as a whole independently from its internal structure
  - Traditional system and acceptance testing techniques can be applied.

UML diagrams in OO Testing

- UML diagrams plays vital role in each OO testing phase.
- Class testing
  - Statechart diagram, class diagram
- Interclass testing
  - Class diagram, activity diagram, interaction diagram
- System testing
  - Use case diagram