This Class

- Review
  - summarize previous lectures
  - point out key concepts, questions, and techniques
  - connect them to exam coverage
- Practice problems
  - exam sample problems

Qualities, Principles & Process

- SE history (software crisis)
- SE definitions*
- Differences between SE & programming
- Software nature* (intangible, malleable, human intensive)
- Software qualities
  - classification of software qualities*
  - correctness, reliability, and robustness
  - performance, maintainability (three categories)*, and usability
- Software principles (seven)**
  - rigor and formality, separation of concerns, modularity, abstraction, anticipation of change, generality, incrementality (lots of examples)

Qualities, Principles & Process

- What is a software process
- Waterfall model *
  - major steps
  - pros & cons
- Agile software development**
  - XP*
  - Scrum*

Design

- Objectives of software design
- Modular design*
  - TDN & GDN
  - relations among modules
  - desired properties

Design

- UML
  - class diagram
  - basic relationships among classes*
- Design patterns *
  - categories: creational, structural, and behavioral
  - simple factory, dependency injection, singleton, observer

Frameworks & Tools

- See full list under Course Materials > Tech Presentation Slides on iLearn
- Understand the basics

Testing

- Testing: nature and goals
- Theoretical foundations of testing (D, R, OR, test case/set)
- Complete coverage principle**
- Whitebox testing*
  - four coverage criteria*
  - control flow graph (CFG)*
- Blackbox testing*
  - decision table testing*
  - boundary value analysis*

Quiz Question Review

Quiz: Agile Methods

- Consider a developer is implementing a feature for the business owner. He realizes he needs a DB to make the feature work.
- His plan: stop work on the feature and build a robust DB to support it and other features (in the future).
- Is his plan agile (from Scrum's view)? What would you do?

Answer: Typically, this is not agile from Scrum's perspective. The developer should postpone this proposal to the next Sprint Planning Meeting. It is not preferred to make major changes to the plan in the middle of a sprint.

credit: Mark Shead

Quiz: Observer

```
#include <iostream>
#include <vector>
using namespace std;

class Subject {
    // 1. "independent" functionality
    vector < class Observer * > views;
    int value;
  public:
    void attach(Observer *obs) {
        views.push_back(obs);
    }
    void setVal(int val) {
        value = val;
        notify();
    }
    int getVal() {
        return value;
    }
    void notify();
};
```

```
void Subject::notify() {
  // 5. Publisher broadcasts
  for (int i = 0; i < views.size(); i++)
    views[i]->update();
}
```

```
class Observer {
    // 2. "dependent" functionality
    Subject *model;
    int denom;
  public:
    Observer(Subject *mod, int div) {
        model = mod;
        denom = div;
        // 4. Observers register themselves with the Subject
        model->attach(this);
    }
    virtual void update() = 0;
  protected:
    Subject *getSubject() {
        return model;
    }
    int getDivisor() {
        return denom;
    }
};
```

Quiz: Observer

```
class DivObserver: public Observer {
  public:
    DivObserver(Subject *mod, int div): Observer(mod, div){}
    void update() {
        // 6. "Pull" information of interest
        int v = getSubject()->getVal(), d = getDivisor();
        cout << v << " div " << d << " is " << v / d << '\n';
    }
};

class ModObserver: public Observer {
  public:
    ModObserver(Subject *mod, int div): Observer(mod, div){}
    void update() {
        int v = getSubject()->getVal(), d = getDivisor();
        cout << v << " mod " << d << " is " << v % d << '\n';
    }
};
```

```
int main() {
  Subject subj;
  DivObserver divObs1(&subj, 4);
  DivObserver divObs2(&subj, 3);
  ModObserver modObs3(&subj, 3);
  subj.setVal(14);
}
```

Output ?

Answer:

```
14 div 4 is 3
14 div 3 is 4
14 mod 3 is 2
```

Quiz

- **Question**: Is path-coverage criterion stronger than all the other three criteria (statement/edge/condition)?

Answer: No.

```
read(x)
read(z)

if(x ≠ 0 && z > 1)
   x = x / z
else
   x = x * z
```

<x=5, z=2>

<x=0, z=2>

Practice Problems

Practice Problem

Problem: (Theoretical Foundation of Testing) Consider the following program. Please indicate its input domain **D**, output range **R**, and output requirement **OR**.

```
enum E = {NONE=-1, FQ=0, SQ=1, TQ=2, HQ=3};
E quartiles (int x)  {
   if (x==25)
      return FQ;
   else if (x==50)
      return SQ;
   else if (x==75)
      return TQ;
   else if (x==100)
      return HQ;
   else
      return NONE;
}
```

Practice Problem

Problem: (White-box Testing) Consider the following program:

1) Please indicate the input domain **D**;
2) Draw the control flow graph (CFG);
3) Give a minimal test set that ensures full *statement* coverage;
4) Label the edges in the CFG, then list all the *paths* in the CFG.
5) Is there a minimal test set that ensures full *path* coverage? If so, provide one; otherwise, explain the reasons.

Tips: draw a coordinate system to help figure out the ranges (areas) that satisfy the conditions of different branches.

```
test_sign (int x, int y)
{
   if (x < 0 && y < 0)
      write ("both negative");
   else
      write ("at least one positive");

   if (x ≥ 0 && y ≥ 0)
      write ("none is negative");

   write("done");
}
```