Introduction to Abstraction and Specification

EE 564

Lecture 2

Daqing Hou, Winter 2007

Today:

- Abstraction & specification
- Two abstraction mechanisms
  - abstraction by parameterization
  - abstraction by specification
- Four kinds of abstractions
  - procedural abstraction
  - data abstraction
  - iteration abstraction
  - type hierarchy

Abstraction & specification

- Abstraction is ***selective ignorance***
  - identify key elements and ignore inessential details
  - many-to-one mapping; expressive
  - ex. Abstract algebras
- Benefits
  - simplifies problem
  - enables *divide and rule*
- Suitability depends on context of use
  - ex. (5/3)*3 and 5
- Specifications make abstractions explicit and tangible

Challenge of decomposition

- Do the parts together solve the original problem?
  - ex. Car design (due to Prof. Khondker)
- How to improve my skill of doing decomposition?
  - Solving many problems
  - Knowing good abstractions

Abstraction by parameterization

Daqing Hou, Winter 2007

What does it compute?

```
int n, d;
while ***(n!=d)
{
  if (n > d)
  {
    n = n - d;
  }
  else
  {
    d = d - n;
  }
}
```

```
n =  d =
8    20
8    12
8    4
4    4
```

gcd!

- This code calculates the greatest common divisor of two positive numbers (gcd)
  - gcd(20, 8) = 4

- What if we want to obtain the gcd of two variables other than *d* and *n*
  - copy and paste may work, with some problems
  - a better way is to define a procedure gcd(int, int)

Abstraction by specification

Daqing Hou, Winter 2007

What do they compute?

```
found = false;
for (i=0;i<a.length; ++i)
{
    if (a[i] == e){
        found = true;
        break;
    }
}
```

```
found = false;
for (i=a.length-1;i>-1;--i)
{
    if (a[i] == e){
        found = true;
        break;
    }
}
```

What do they compute?

```
/**
 * search e forwards
 * found is true if e is found
 * i has index of the element
 */
found = false;
for (i=0;i<a.length; ++i)
{
    if (a[i] == e){
        found = true;
        break;
    }
}
```

```
/**
 * search e backwards
 * found is true if e is found
 * i has index of the element
 */
found = false;
for (i=a.length-1;i>-1;--i)
{
    if (a[i] == e){
        found = true;
        break;
    }
}
```

What is the original intent?

- From the code alone we do not know what is intended. It can be either of the following:
  - If array *a* has an element equal to *e*, then set *found* to true and *i* to the index of that element; otherwise set *found* to false.
  - If array *a* has an element equal to *e*, then set *found* to true and *i* to the index of the left (right)-most such element; otherwise set *found* to false.
- Specifications are needed to express intent
- One more example on expression of intent

What does this do?

```
float calc (float aFloat) {
   float ans = aFloat/2.0;
   int i=1;
   while (i < 7) {
      ans = ans - ((ans*ans - aFloat)/(2.0*ans));
      i = i + 1;
   }
   return ans;
}
```

Better

```
float sqrt (float coef) {
   float ans = coef/2.0;
   int i=1;
   while (i < 7) {
      ans = ans - ((ans*ans - coef)/(2.0*ans));
      i = i + 1;
   }
   return ans;
}
```

- Why better?

Even better

```
// REQUIRES: coef>0
// EFFECTS: returns an approximation to the square root
   of coef
float sqrt (float coef)
```

- *requires assertion* (or *precondition*)
- *effects assertion* (or *postcondition*)

- This is procedural abstraction; more on this later

Why specification?

- It can be non-trivial to examine details from code and then abstract intent
  - specification spares this effort

- Cannot always tell intent from code alone
  - May thus accidentally depend on irrelevant details

Data abstraction

- Combination of data and operations is more powerful than procedural abstraction
- More than a set of operations
  - e.g. MultiSet
- Most useful in OO design
- Other common data abstractions?

Iteration abstraction

- Allow to iterate over all the elements of a collection
- Avoid having to say more than is relevant about the flow of control in a loop
  - i.e. implementation details are protected

Type hierarchy

- Type families, supertype and subtype
- Commonalities in supertypes and differences in subtypes
- E.g. JCF (Java Collection Framework), input stream

Summary

- Abstraction & specification
- Two abstraction mechanisms
  - abstraction by parameterization
  - abstraction by specification
- Four kinds of abstractions
  - procedural abstraction
  - data abstraction
  - iteration abstraction
  - type hierarchy