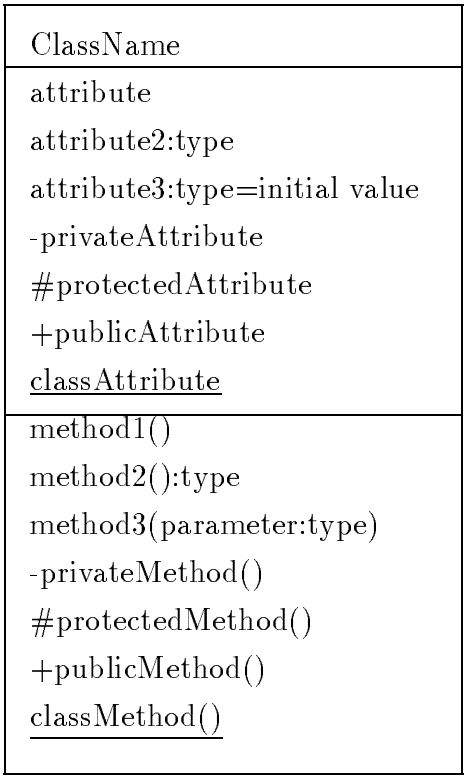
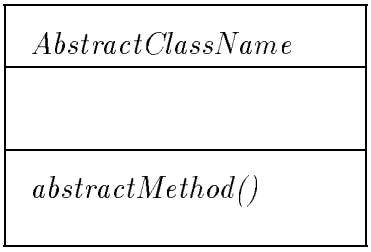
Class Definition:



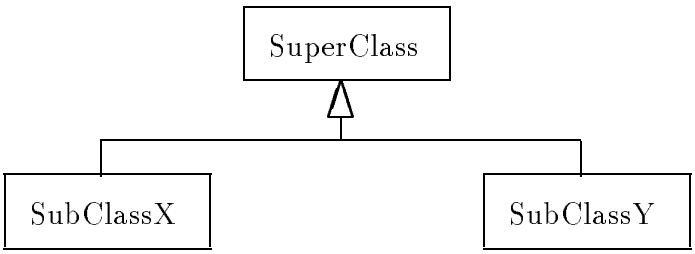

```
class ClassName {
  int attribute2;
  int attribute3=10;
  private int privateAttribute;
  protected int protectedAttribute;
  public int publicAttribute;
  static int classAttribute;

  void method1() { }
  int method2() {return 0;}
  void method3(int parameter) {}
  private void privateMethod() {}
  protected void protectedMethod() {}
  public void publicMethod() {}
  static void classMethod() {}
}
```

Abstract Class Definition:



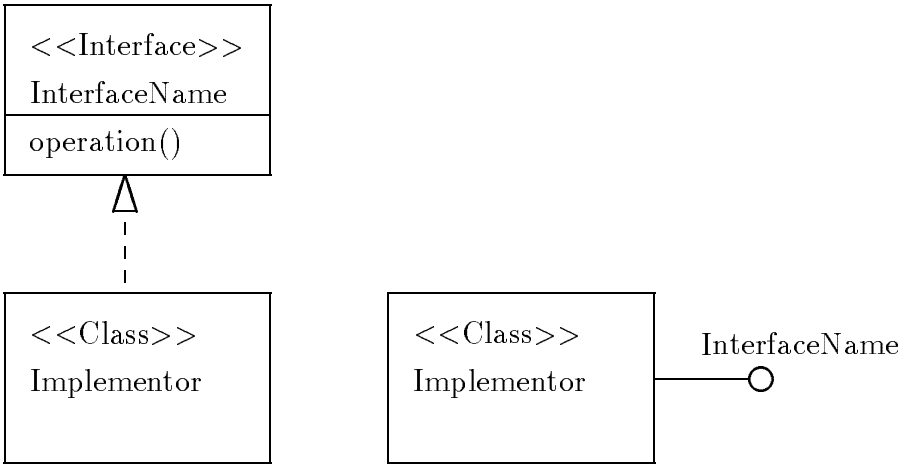

```
abstract class AbstractClassName {
  abstract void abstractMethod();
}
```

Generalization/Specialization ("is a"):



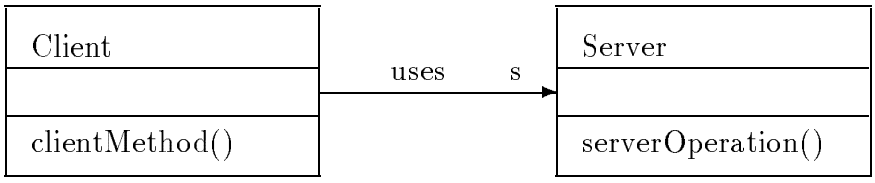

```
class SuperClass {}
class SubClassX extends SuperClass {}
class SubClassY extends SuperClass {}
```

Realization("realizes"):



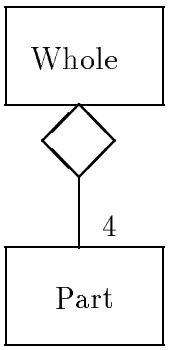

```
interface InterfaceName {
  public void operation();
}
class Implementor implements InterfaceName {
  public void operation() {}
}
```

Navigability ("has a") where the Association is "uses":




```
class Client {
  Server s;
  void clientMethod() {
    s.serverOperation();
  }
}
class Server {
  void serverOperation() {}
}
```

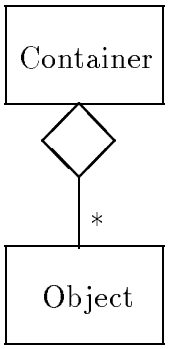
Composition (should be dark diamond):




```
class Part {}
class Whole {
  Part part1;
  Part part2;
  Part part3;
  Part part4;
}
```

- Composition creates/destroys sub-parts
- Part can only belong to one parent (Whole)
- Example: Car (Whole) to Carburetor (Part)

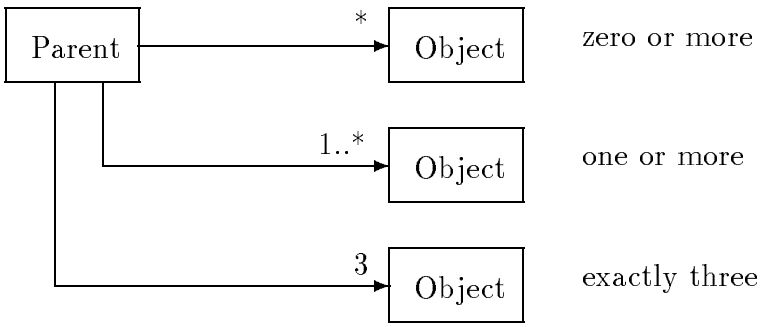
Aggregation (is a clear diamond):




```
class Container {
  java.util.Vector v;

  void addChild(Object o) { v.addElement(o); }
}
```

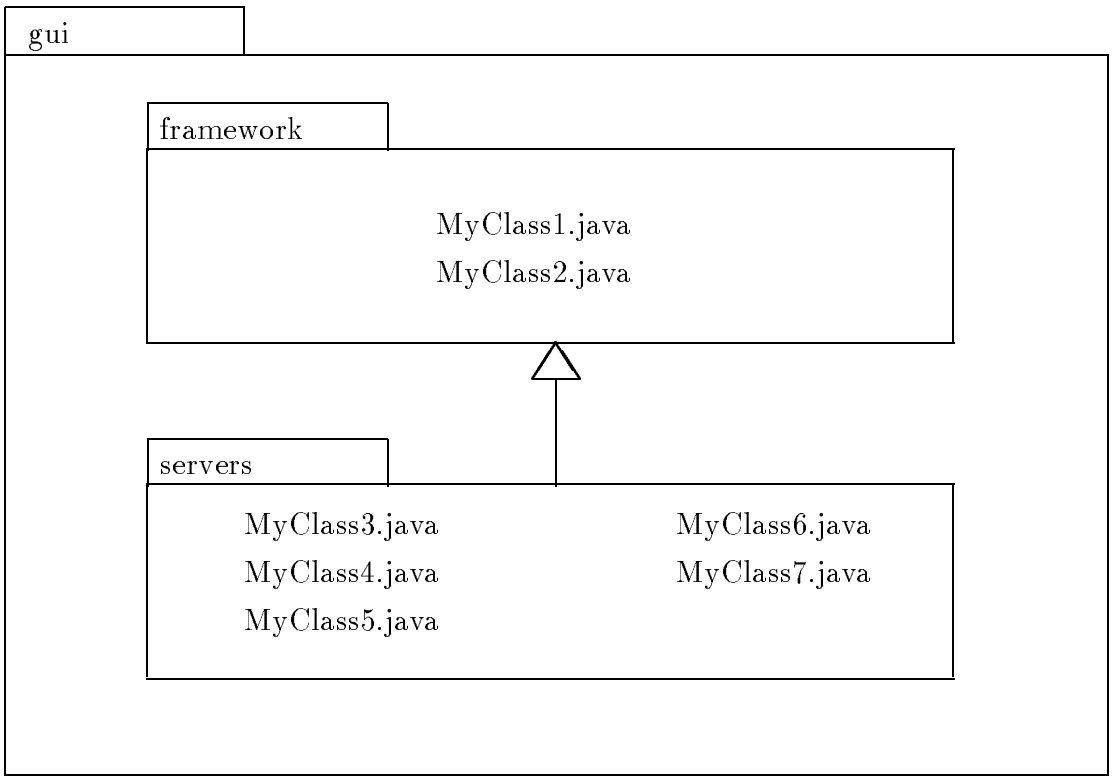
- Aggregation receives Objects from the outside
- Object can belong to more than on parent, and live longer than parent
- Use Vector (or Hash, etc.) to store collection of many Objects
- Example: Pond (Container) to Duck (Object)
- Composition is a stronger form of Aggregation

Multiplicity:




```
class Parent {
  java.util.Vector v1;
  java.util.Vector v2;
  Object o1;
  Object o2;
  Object o3;
}
```

Package:




```
package gui.framework;
class MyClass1 {}

package gui.servers;
import gui.framework;
class MyClass3 extends MyClass1 {}
```