Software Product Line Engineering

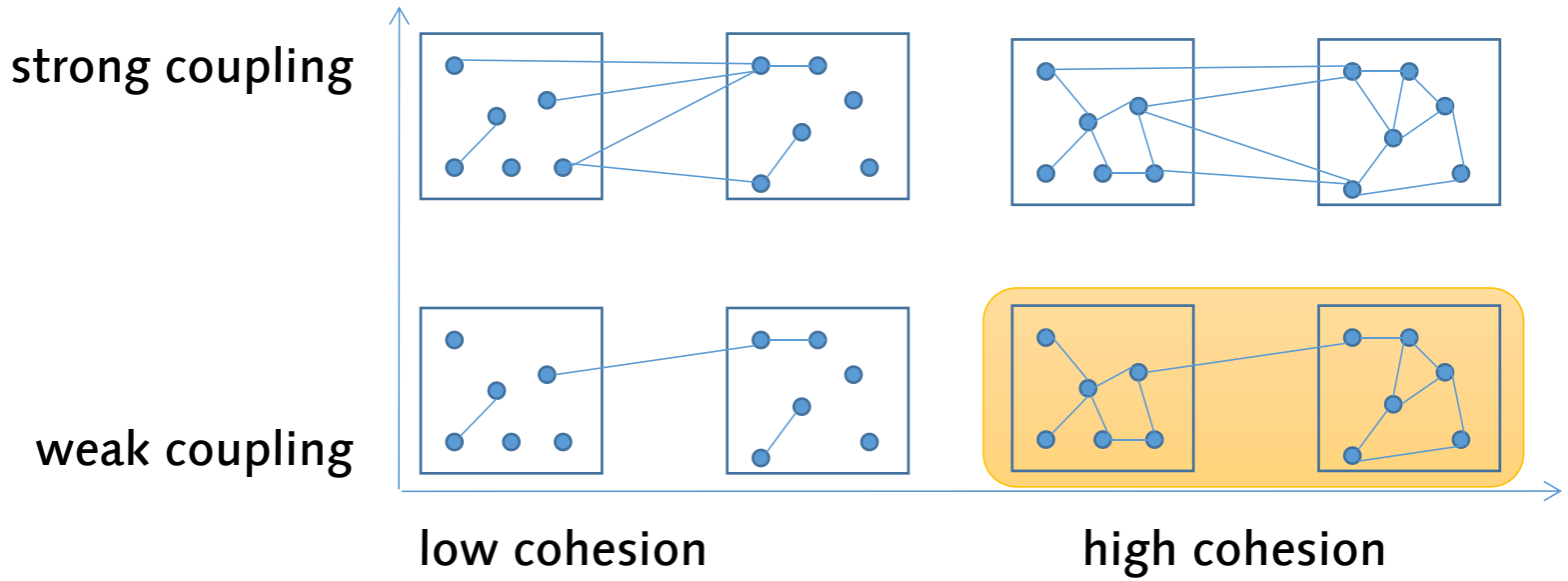
Lab Class 3

Outline

- Discussion Task 2 & 4
- Remarks on first submission

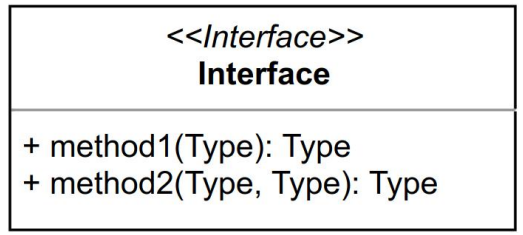
Recap Modularity

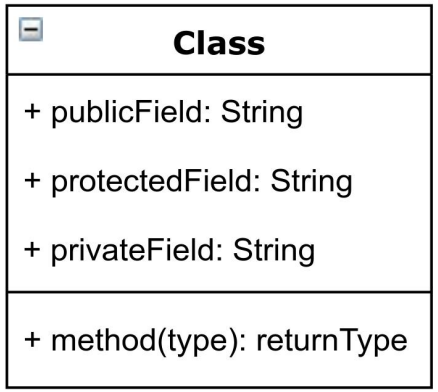
- Coupling: Connectivity *across* modules
- Cohesion: Connectivity *within* modules



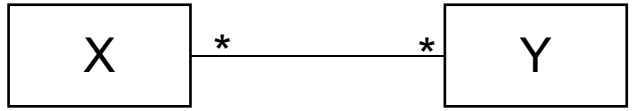
Recap UML

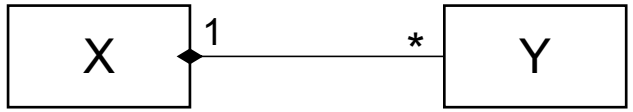
- Entities
  - Classes
  - Interfaces / Abstract Classes



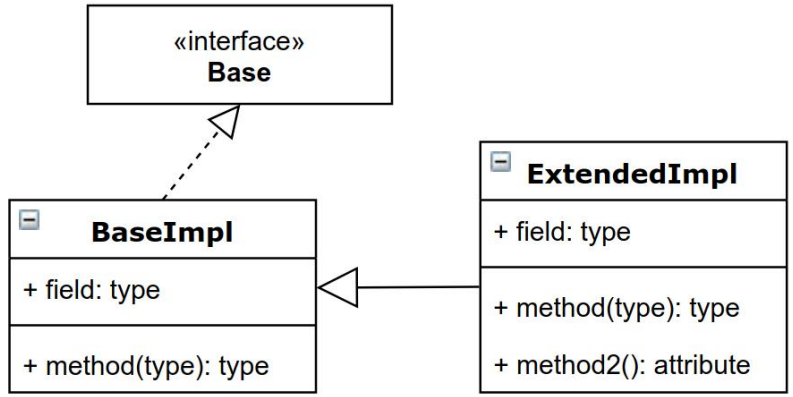


- Associations
  - normal: “x corresponds to y"
  - composition: “x(s) is/are part of y"

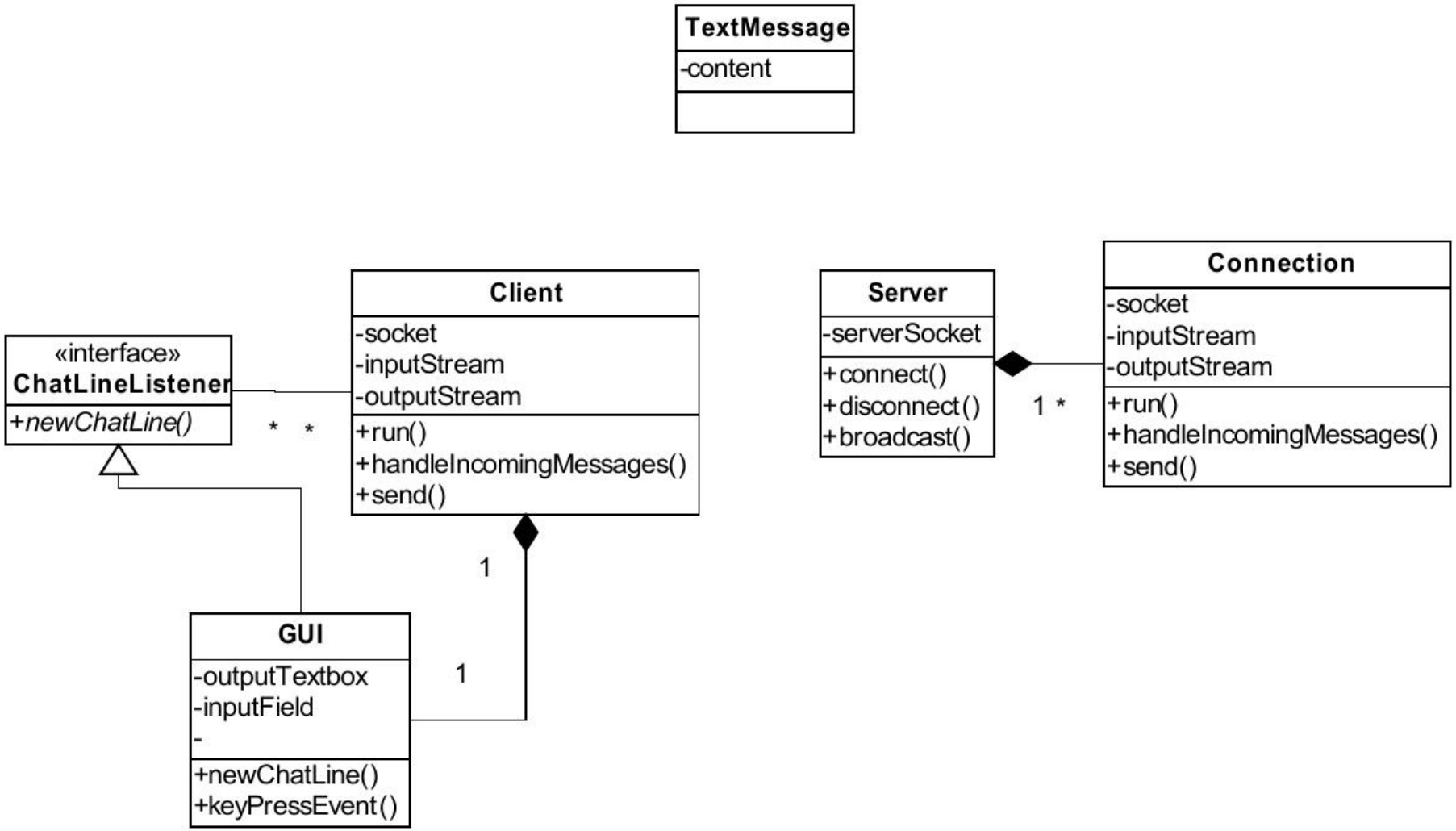




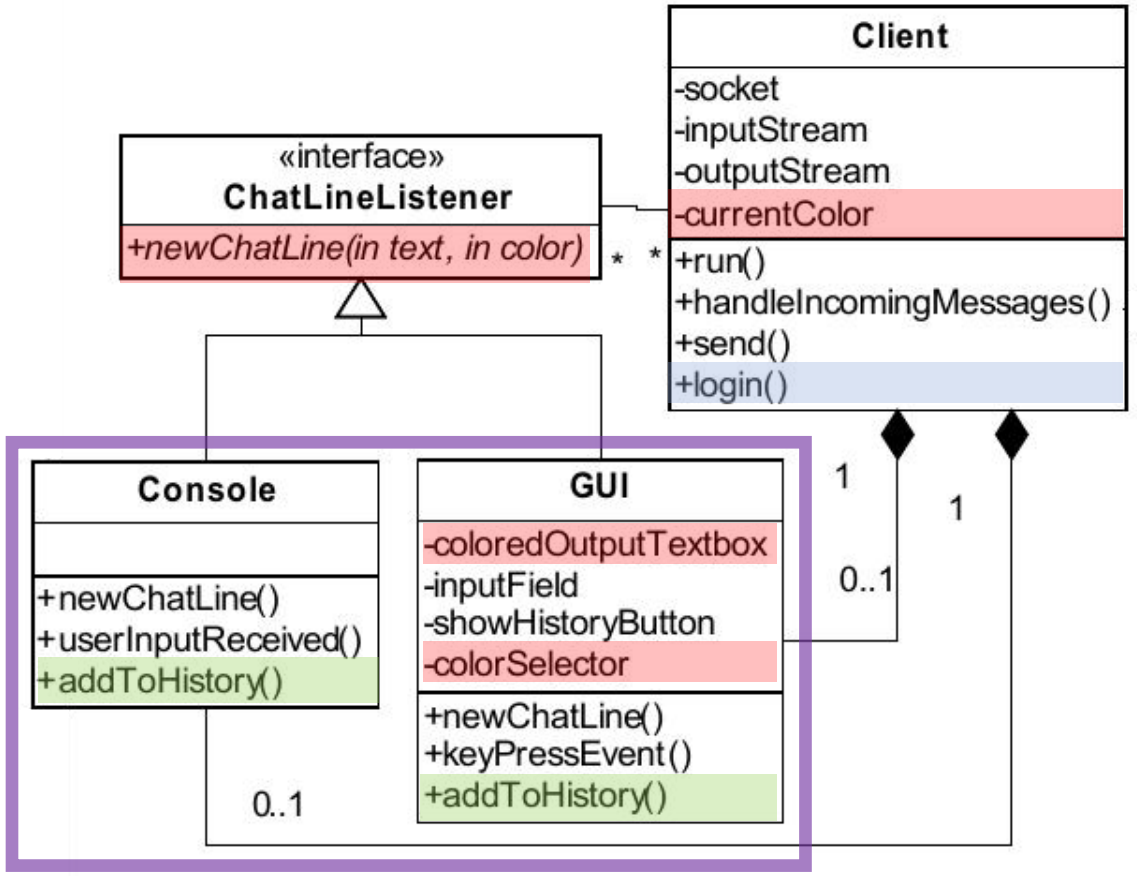
- Generalization
  - Implementation of interfaces
  - Inheritance of (abstract) classes

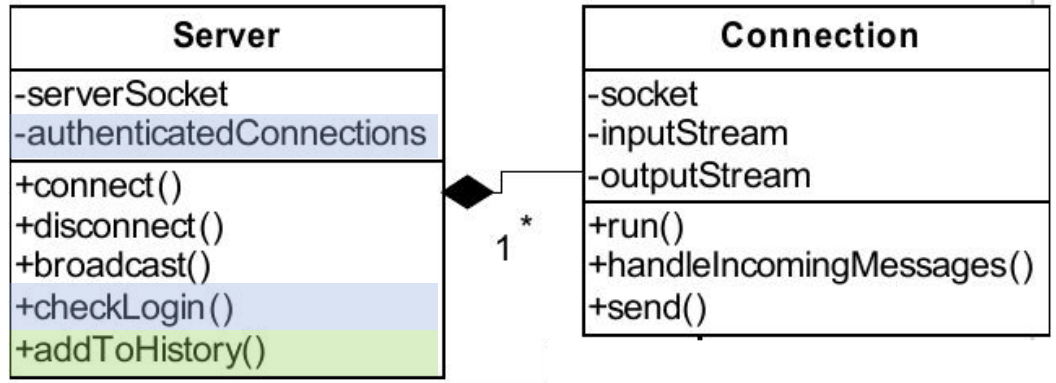


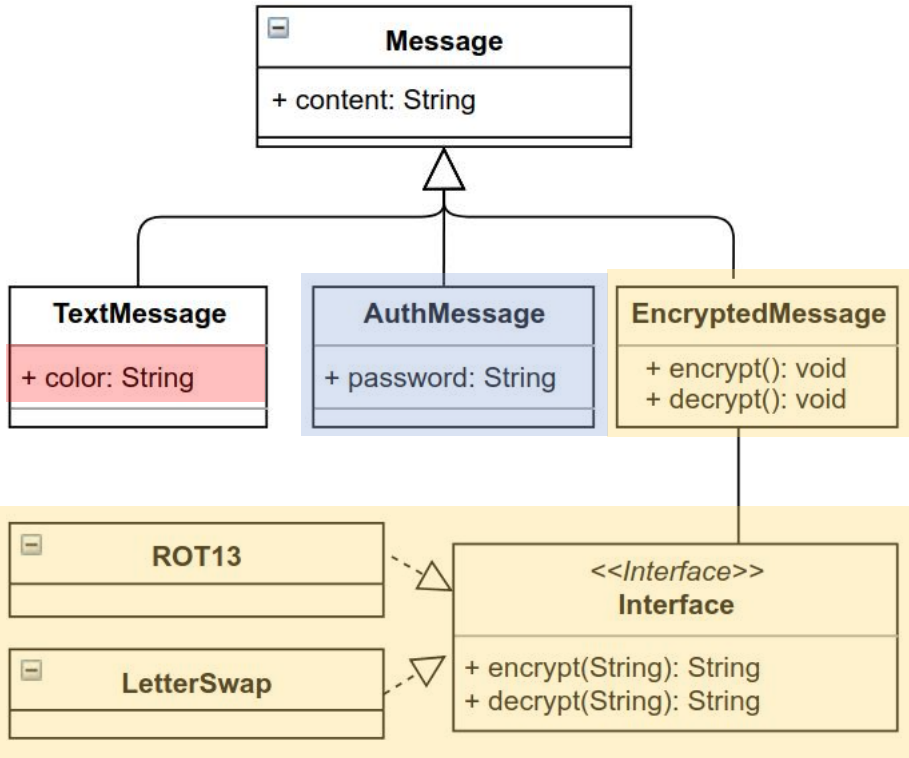
Chat base implementation



Chat with features

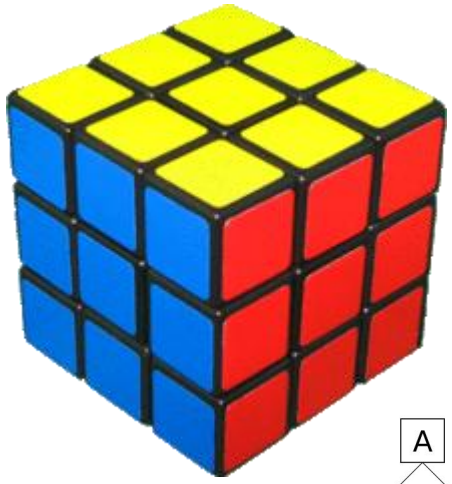


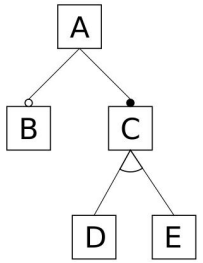




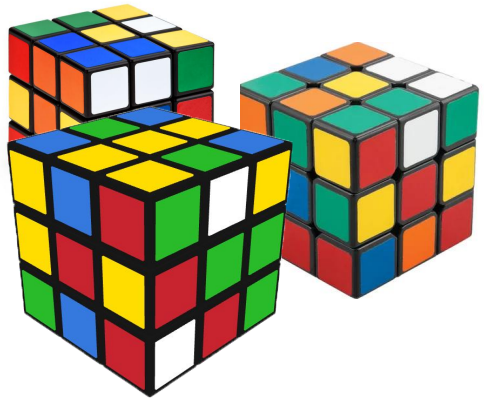
Interfaces
Color
Authentication
Encryption
Logging

Scattered and Tangled Code





Modeling



Implementation

```
class Graph {
 Vector nv = new Vector(); Vec
 Edge add(Node n, Node m) {
  Edge e = new Edge(n, m);
  nv.add(n); nv.add(m); ev.add(e);
  if (Conf.WEIGHTED) e.weight = new Weight();
  return e;
 }
 Edge add(Node n, Node m, Weight w)
  if (!Conf.WEIGHTED) throw RuntimeException();
  Edge e = new Edge(n, m);
  nv.add(n); nv.add(m); ev.add(e);
  e.weight = w; return e;
 }
 void print() {
  for(int i = 0; i < ev.size(); i++) {
   ((Edge)ev.get(i)).print();
  }
 }
}
```

Code Scattering

```
class Node {
  w Color();
  if (Conf.COLORED) Color.setDisplayColor(color);
  System.out.print(id);
 }
}
```

```
class Edge {
 Node a, b;
 Color color = new Color();
 Weight weight;
 Edge(Node _a, Node _b) { a = _a; b = _b; }
 void print() {
  if (Conf. COLORED) Color.setDisplayColor(color);
  a.print(); b.print();
  if (!Conf.WEIGHTED)  weight.print();
 }
}
```

```
class Color {
 static void setDisplayColor(Color c) { ... }
}
```

```
class Weight { void print() { ... } }
```

22

```
class Graph {
 Vector nv = new Vector(); Vector ev = new Vector();
 Edge add(Node n, Node m) {
  Edge e = new Edge(n, m);
  nv.add(n); nv.add(m); ev.add(e);
  if (Conf.WEIGHTED) e.weight = new Weight();
  return e;
 }
 Edge add(Node n, Node m, Weight w)
  if (!Conf.WEIGHTED) throw RuntimeException();
  Edge e = new Edge(n, m);
  nv.add(n); nv.add(m); ev.add(e);
  e.weight = w; return e;
 }
 void print() {
}
```

Code Tangling

```
class Node {
 int id = 0;
 Color color = new Color();
 void print() {
  if (Conf.COLORED) Color.setDisplayColor(color);
  System.out.print(id);
 }
}
```

```
class Edge {
 Node a, b;
 Color color = new Color();
 Weight weight;
 Edge(Node _a, Node _b) { a = _a; b = _b; }
 void print() {
  if (Conf. COLORED) Color.setDisplayColor(color);
  a.print(); b.print();
  if (!Conf.WEIGHTED)  weight.print();
 }
}
```

```
class Color {
 static void setDisplayColor(Color c) { ... }
}
```

```
class Weight { void print() { ... } }
```

23