Type Highlighting.
A Client-Driven Visual Approach for Class Hierarchies Reengineering

Petru Florin Mihancea

LOOSE Research Group
“Politehnica” University of Timișoara, Romania
petru.mihancea@cs.upt.ro

Class Hierarchies and Polymorphism

Class Hierarchies and Polymorphism

- Keys for increased extensibility

Class Hierarchies and Polymorphism

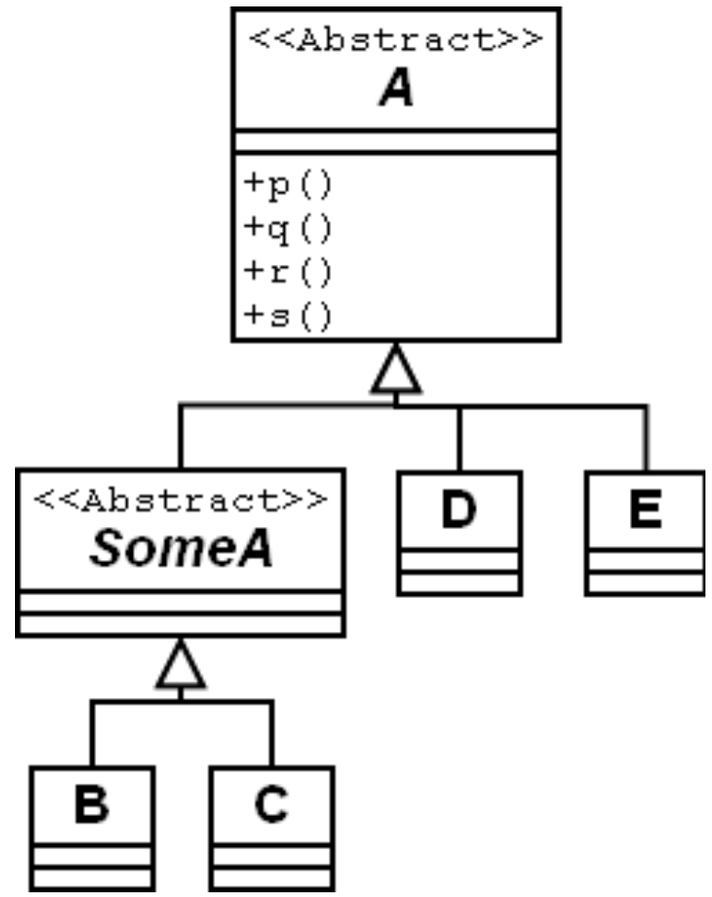
- Keys for increased extensibility
- Raise supplementary understandability issues

Class Hierarchies and Polymorphism

- Keys for increased extensibility
- Raise supplementary understandability issues
- How to understand the role of a legacy hierarchy?

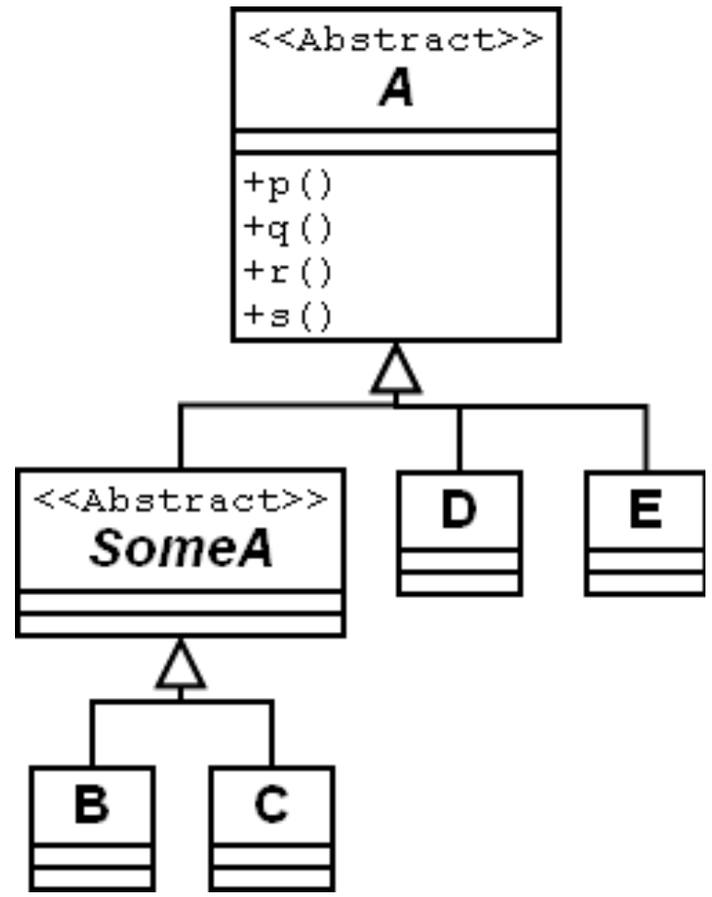
Look at the Clients of Hierarchies

Look at the Clients of Hierarchies



Look at the Clients of Hierarchies

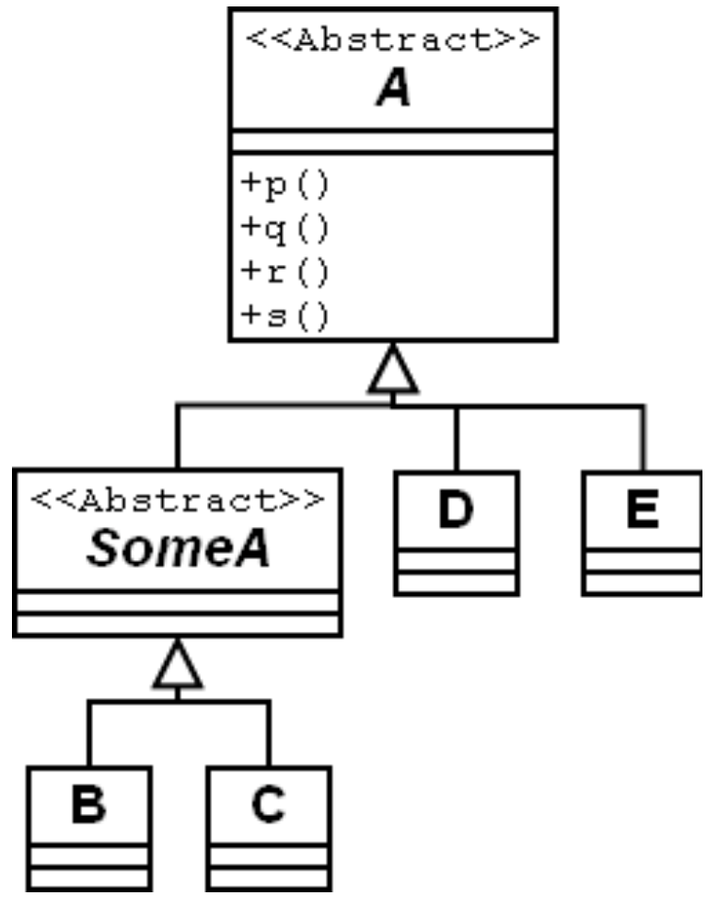
```
void polymorphic(A x) {
    x.p();
}
```



Look at the Clients of Hierarchies

```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

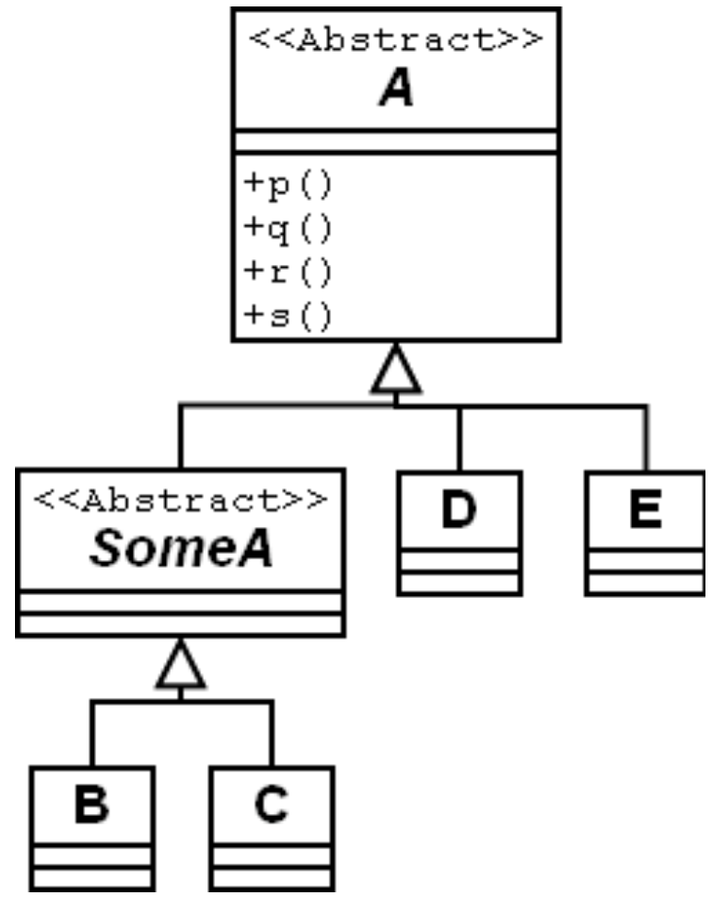


Look at the Clients of Hierarchies

```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```



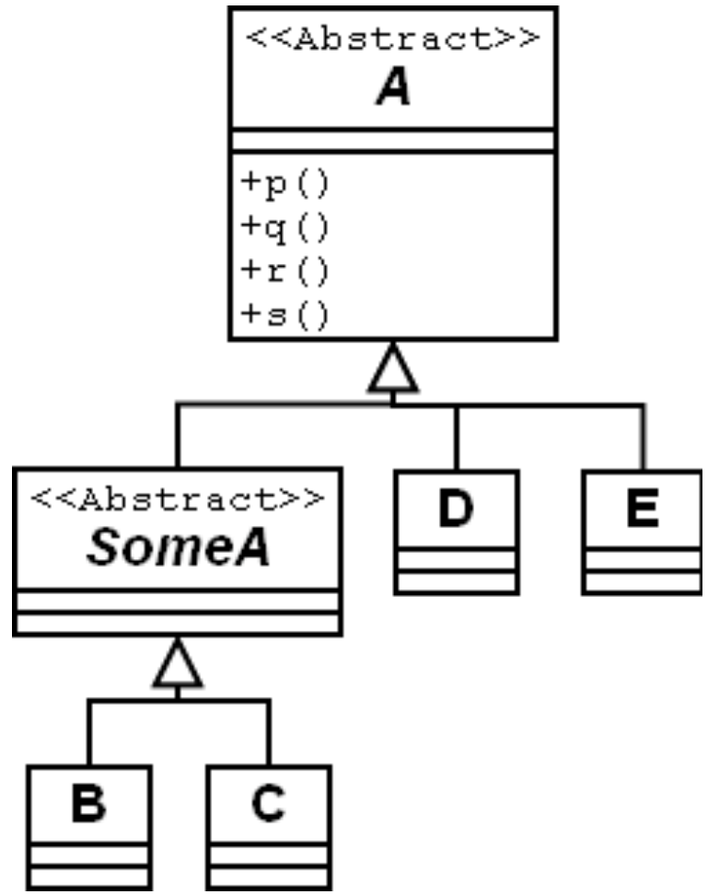
Look at the Clients of Hierarchies

```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```

```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    } else {
        x.r();
    }
}
```



Look at the Clients of Hierarchies

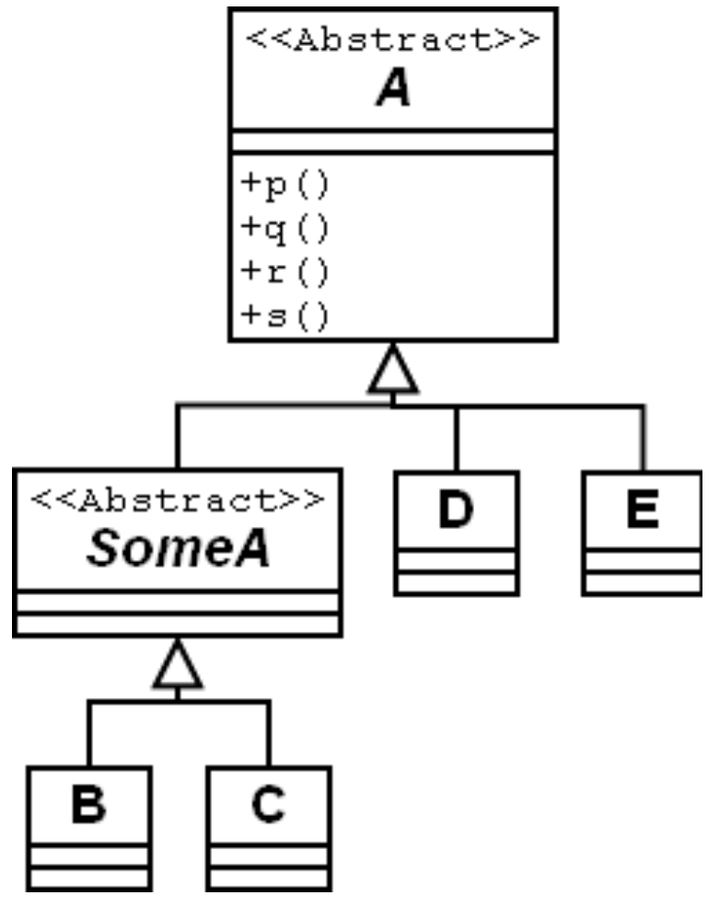
```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```

```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    } else {
        x.r();
    }
}
```

```
void indirectClient(Intermediate y) {
      A x = y.getAnyAObject();
      x.p();
}
```



Look at the Clients of Hierarchies

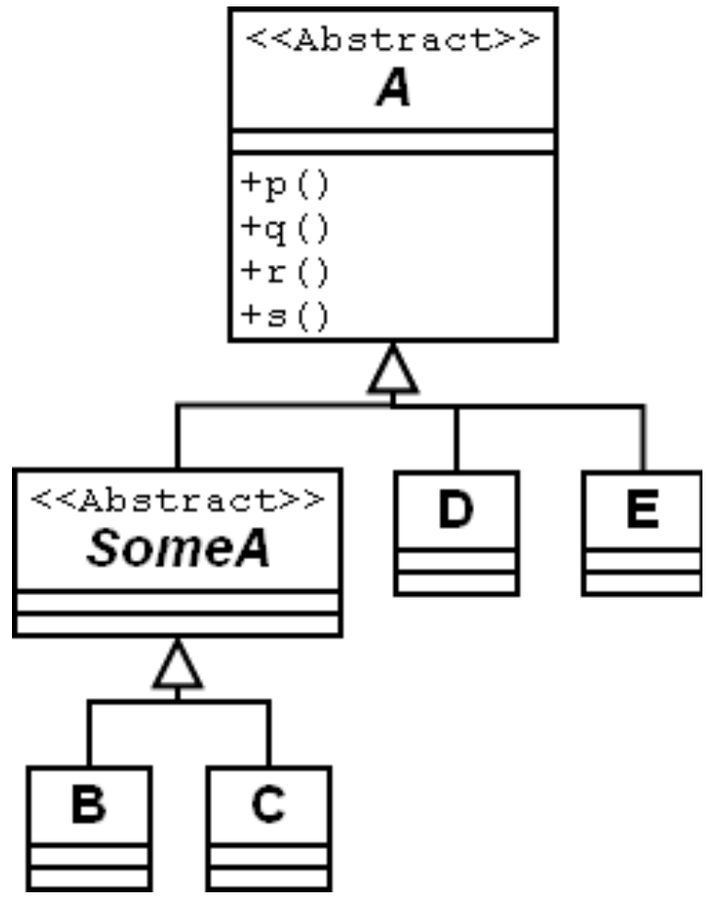
```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```

```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    } else {
        x.r();
    }
}
```

```
void indirectClient(Intermediate y) {
      A x = y.getAnyAObject();
      x.p();
}
```



Look at the Clients of Hierarchies

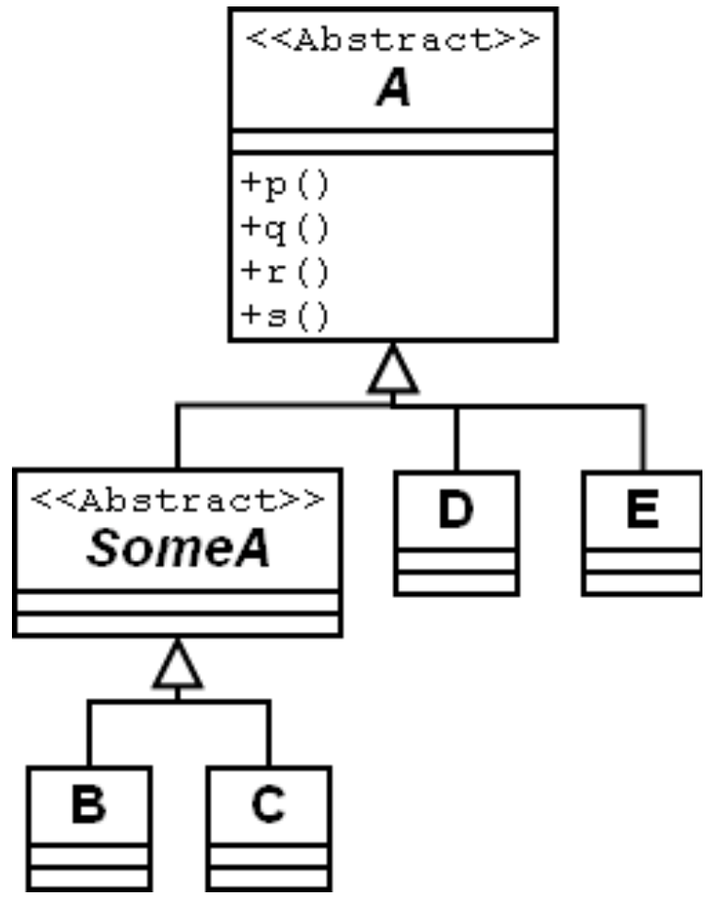
```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```

```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    } else {
        x.r();
    }
}
```

```
void indirectClient(Intermediate y) {
    A x = y.getAnyAObject();
    x.p();
}
```



Look at the Clients of Hierarchies

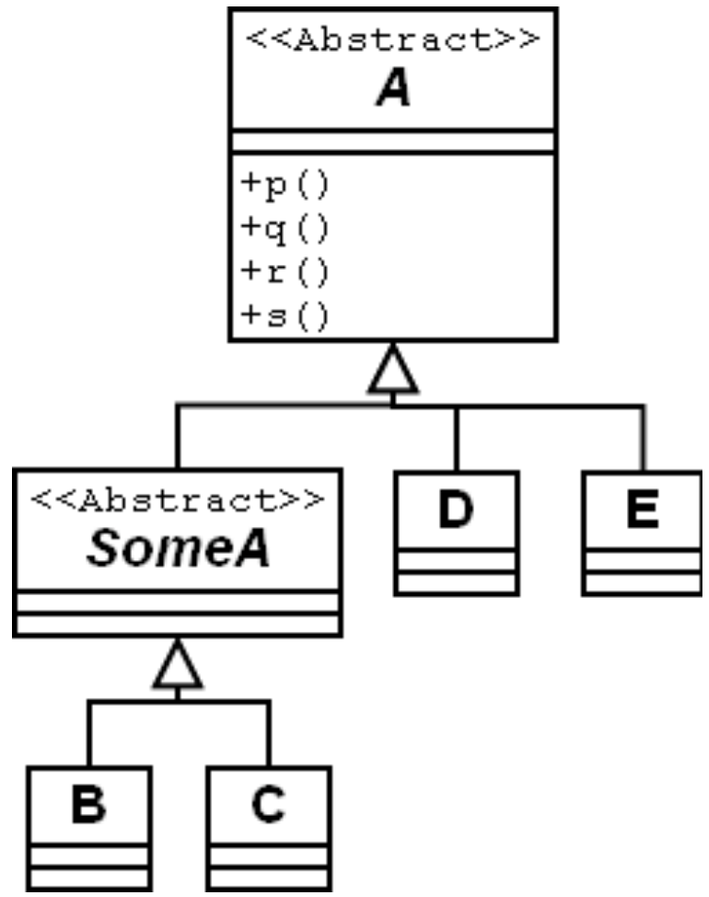
```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```

```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    } else {
        x.r();
    }
}
```

```
void indirectClient(Intermediate y) {
      A x = y.getAnyAObject();
      x.p();
}
```



Look at the Clients of Hierarchies

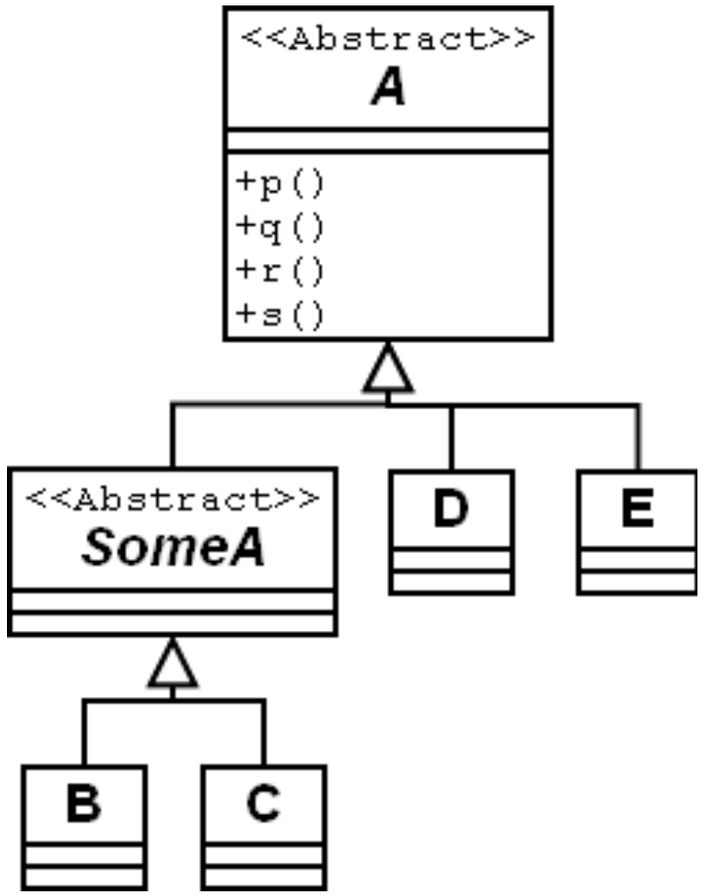
```
void polymorphic(A x) {
   x.p();
}
```

```
void pPolymorphic(SomeA x) {
   x.p();
}
```

```
void concrete(B x) {
   x.p();
}
```

```
void mixed(A x) {
   if(x instanceof SomeA) {
      x.p();
   } else if(x instanceof D) {
      x.q();
   } else {
      x.r();
   }
}
```

```
void indirectClient(Intermediate y) {
      A x = y.getAnyAObject();
      x.p();
}
```



Look at the Clients of Hierarchies

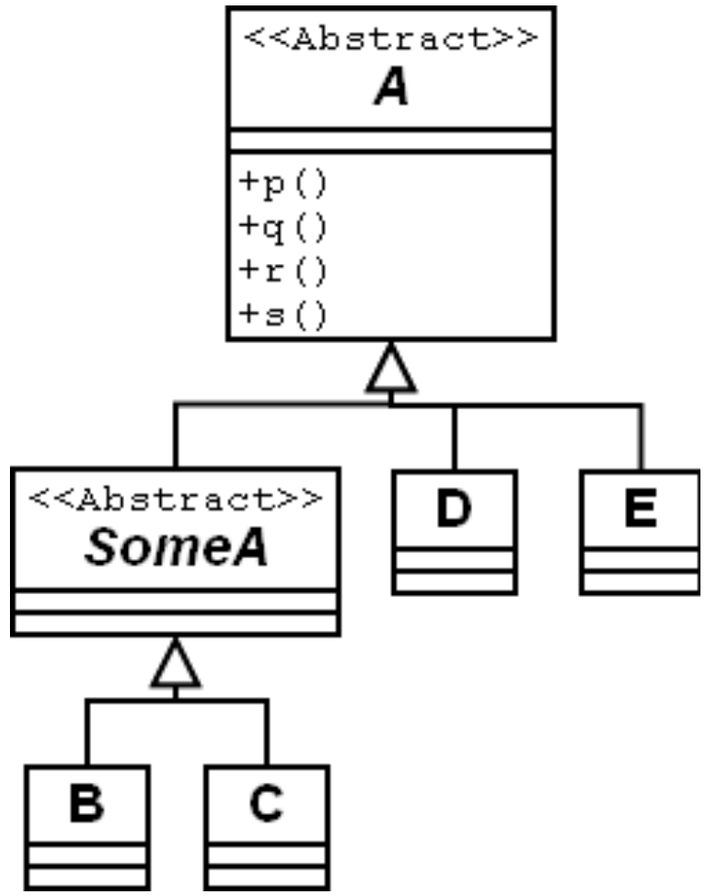
```
void polymorphic(A x) {
   x.p();
}
```

```
void pPolymorphic(SomeA x) {
   x.p();
}
```

```
void concrete(B x) {
   x.p();
}
```

```
void mixed(A x) {
   if(x instanceof SomeA) {
      x.p();
   } else if(x instanceof D) {
      x.q();
   } else {
      x.r();
   }
}
```

```
void indirectClient(Intermediate y) {
    A x = y.getAnyAObject();
    x.p();
}
```



Look at the Clients of Hierarchies

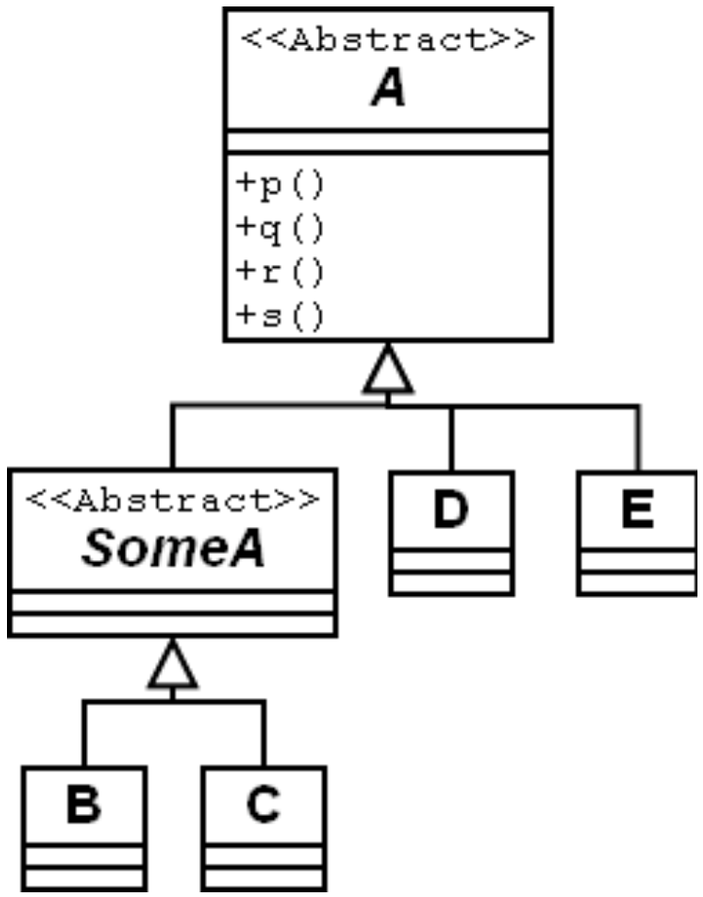
```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

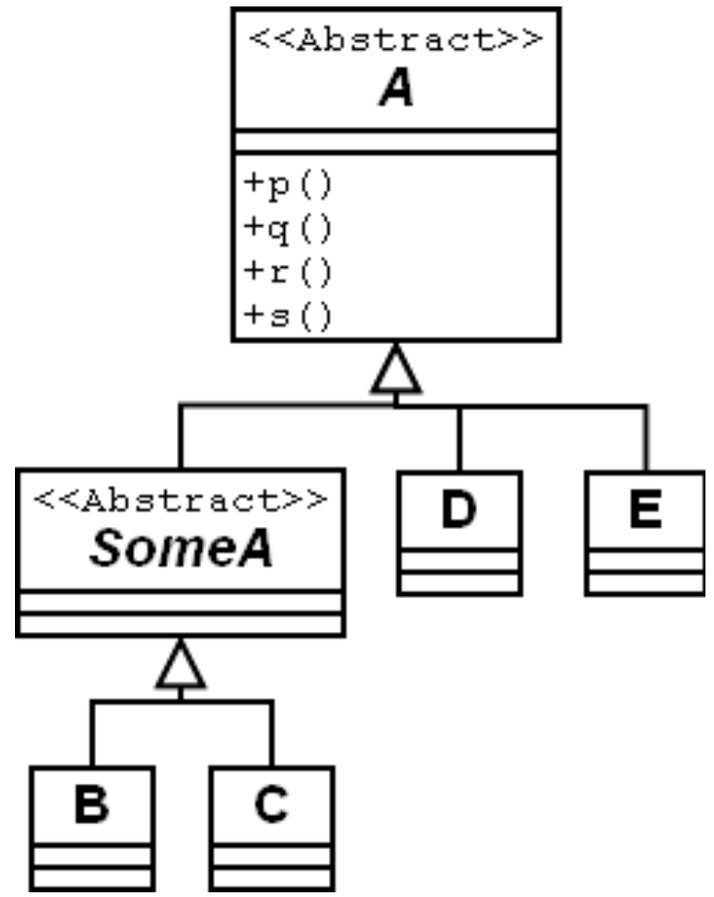
```
void concrete(B x){
    x.p();
}
```

```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    }else {
        x.r();
    }
}
```

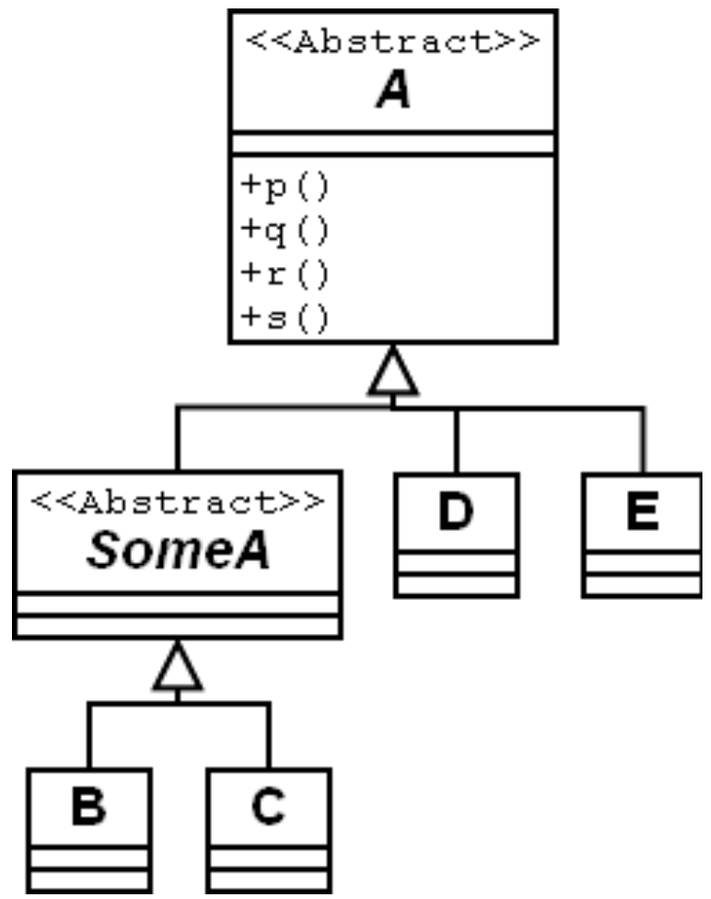
```
void indirectClient(Intermediate y){
    A x = y.getAnyAObject();
    x.p();
}
```



Look at the Clients of Hierarchies



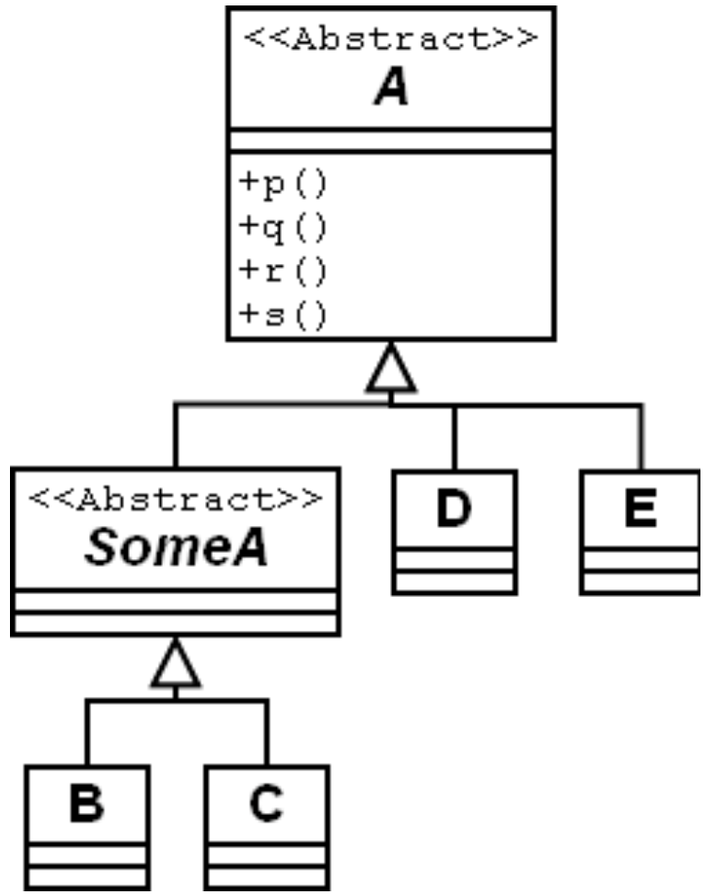
Look at the Clients of Hierarchies



Look at the Clients of Hierarchies

Level of Abstraction View

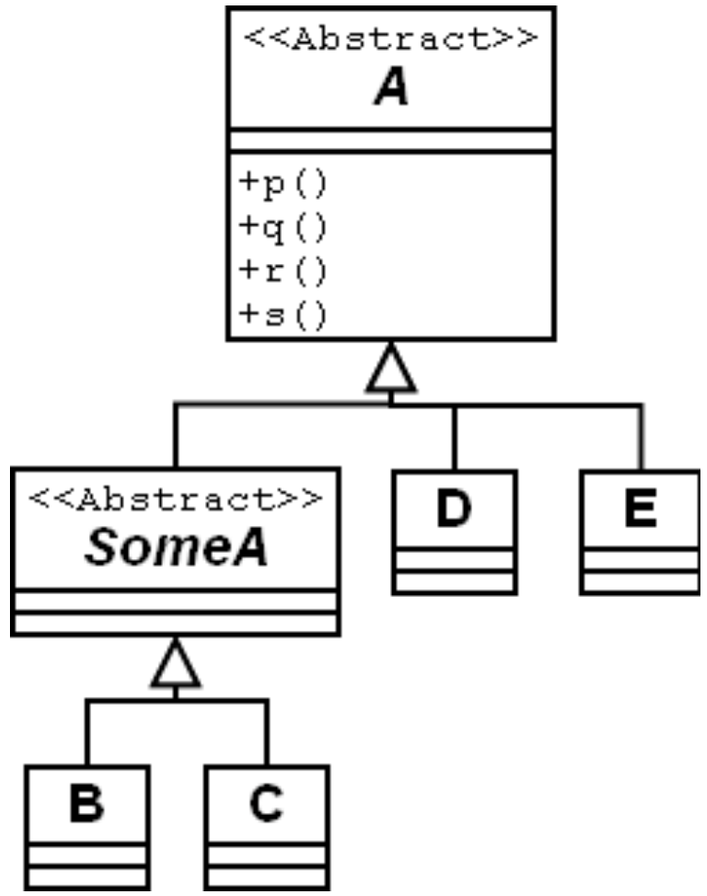
- The extent of polymorphic manipulation
- Fast identification of polymorphic clients
- Detects client-type checking design flaws
- etc.



Look at the Clients of Hierarchies

Level of Abstraction View

- The extent of polymorphic manipulation
- Fast identification of polymorphic clients
- Detects client-type checking design flaws
- etc.



Implementation

- Levels of red = LA metric values
- Computed using Static Class Analysis

A Real Example

A Real Example

UserDataContainer Hierarchy

- Height - 8
- Descendants - 63
- External clients - 121

A Real Example

UserDataContainer Hierarchy

- Height - 8
- Descendants - 63
- External clients - 121

Fast findings

A Real Example

UserDataContainer Hierarchy

- Height - 8
- Descendants - 63
- External clients - 121

Fast findings

- One relevant polymorphic client

A Real Example

UserDataContainer Hierarchy

- Height - 8
- Descendants - 63
- External clients - 121

Fast findings

- One relevant polymorphic client
- Several “splitable” big clients

A Real Example

UserDataContainer Hierarchy

- Height - 8
- Descendants - 63
- External clients - 121

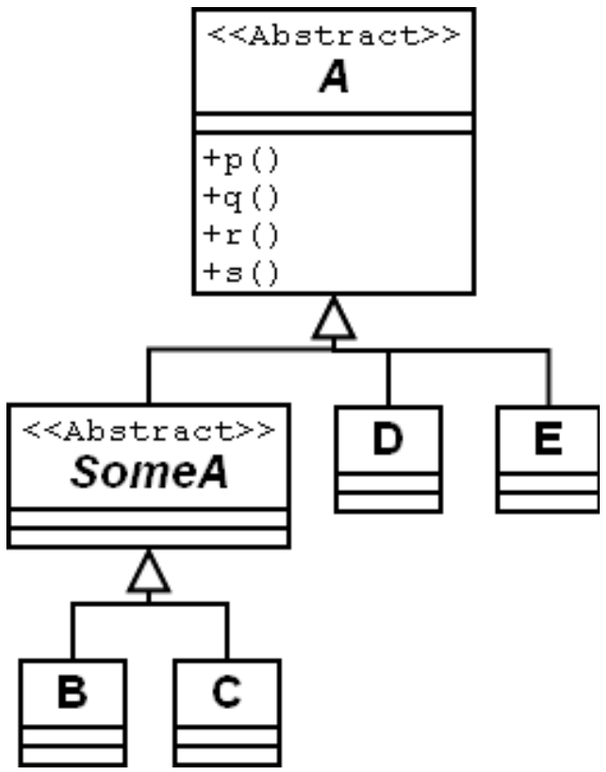
Fast findings

- One relevant polymorphic client
- Several “splitable” big clients
- Not polymorphic-intense

Group Discrimination View

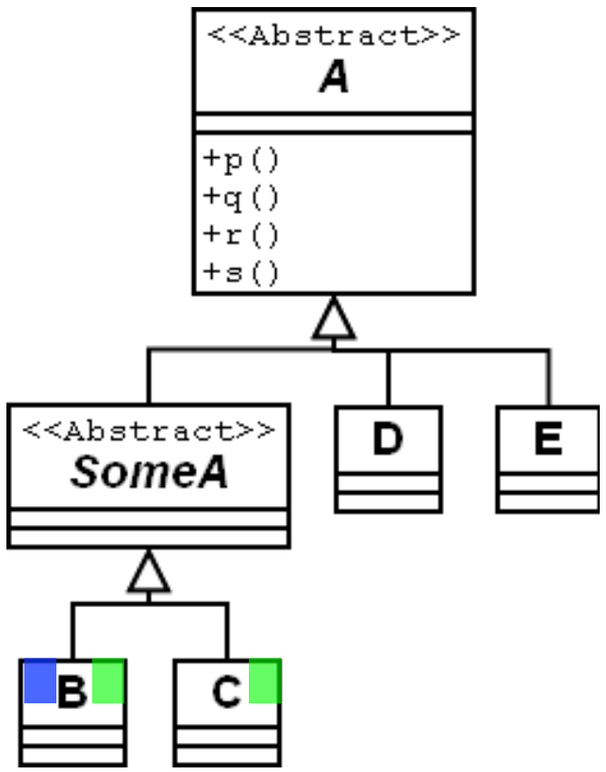
Group Discrimination View

```
void aClient(SomeA x) {
        if(x instanceof B) {
            x.p();
        }
        x.q();
}
```



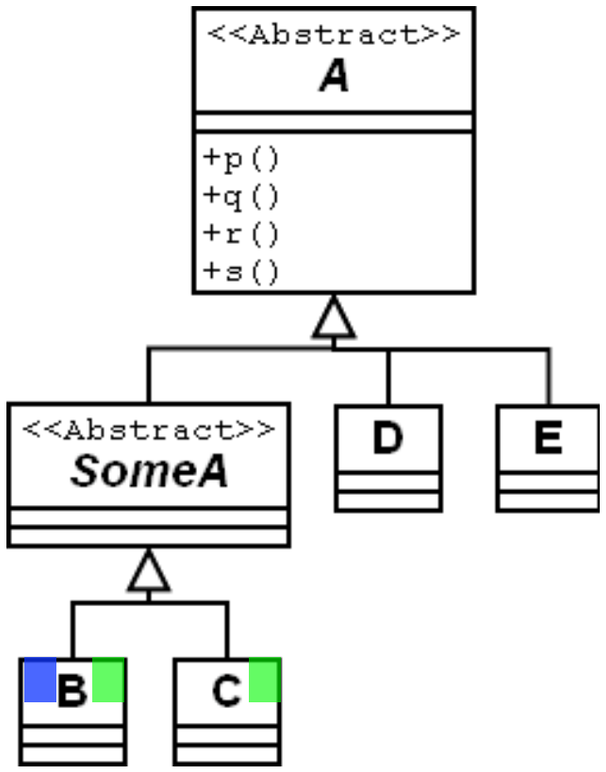
Group Discrimination View

```
void aClient(SomeA x) {
        if(x instanceof B) {
            x.p();
        }
        x.q();
}
```

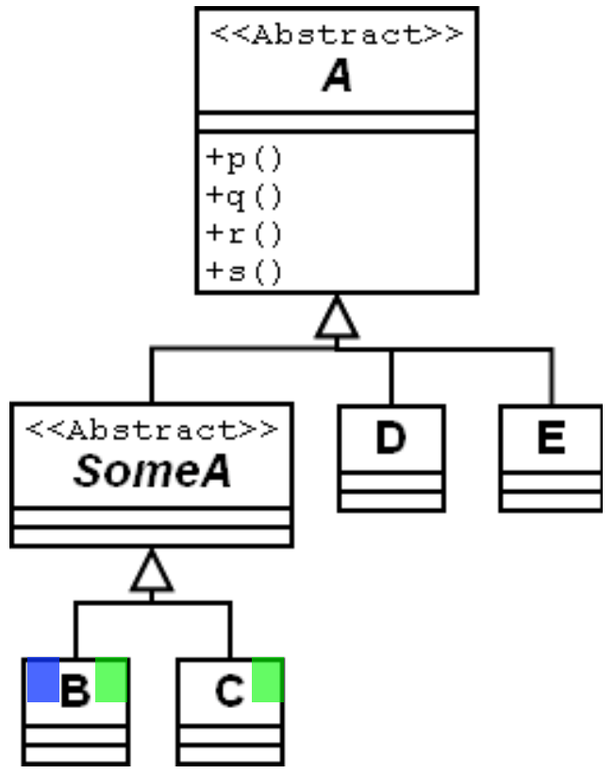


Group Discrimination View

```
void aClient(SomeA x) {
        if(x instanceof B) {
            x.p();
        }
        x.q();
}
```

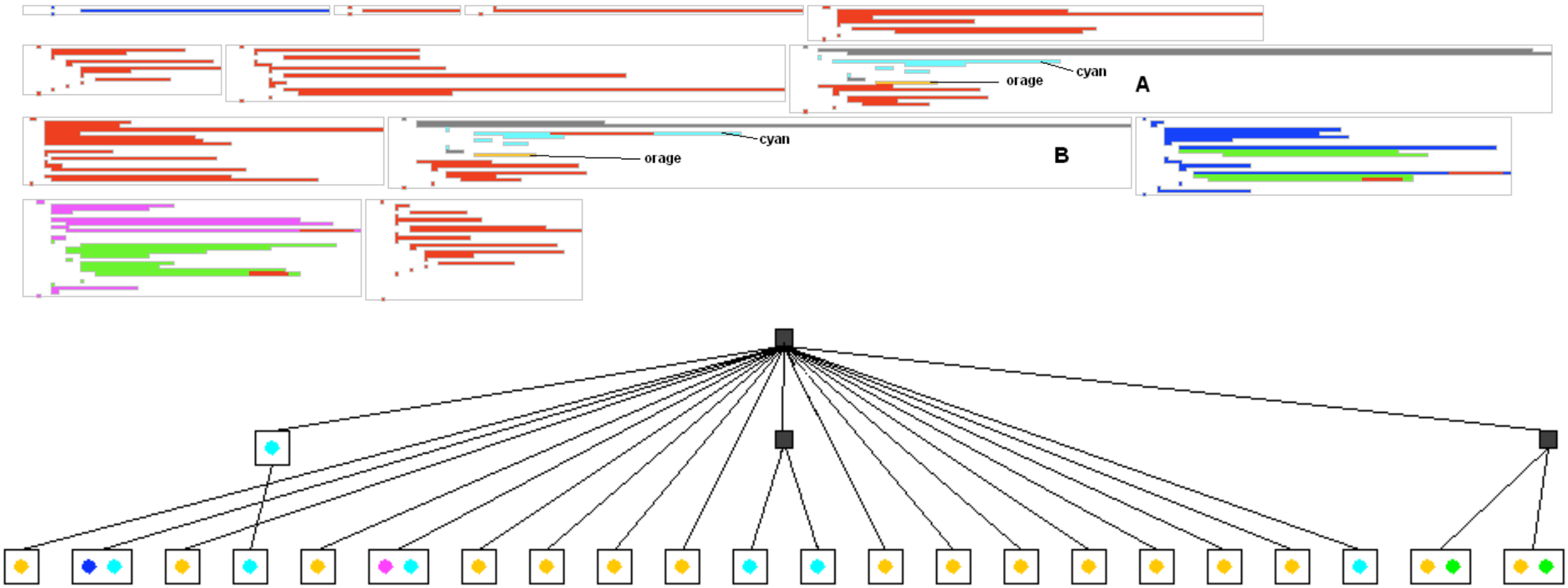


Group Discrimination View

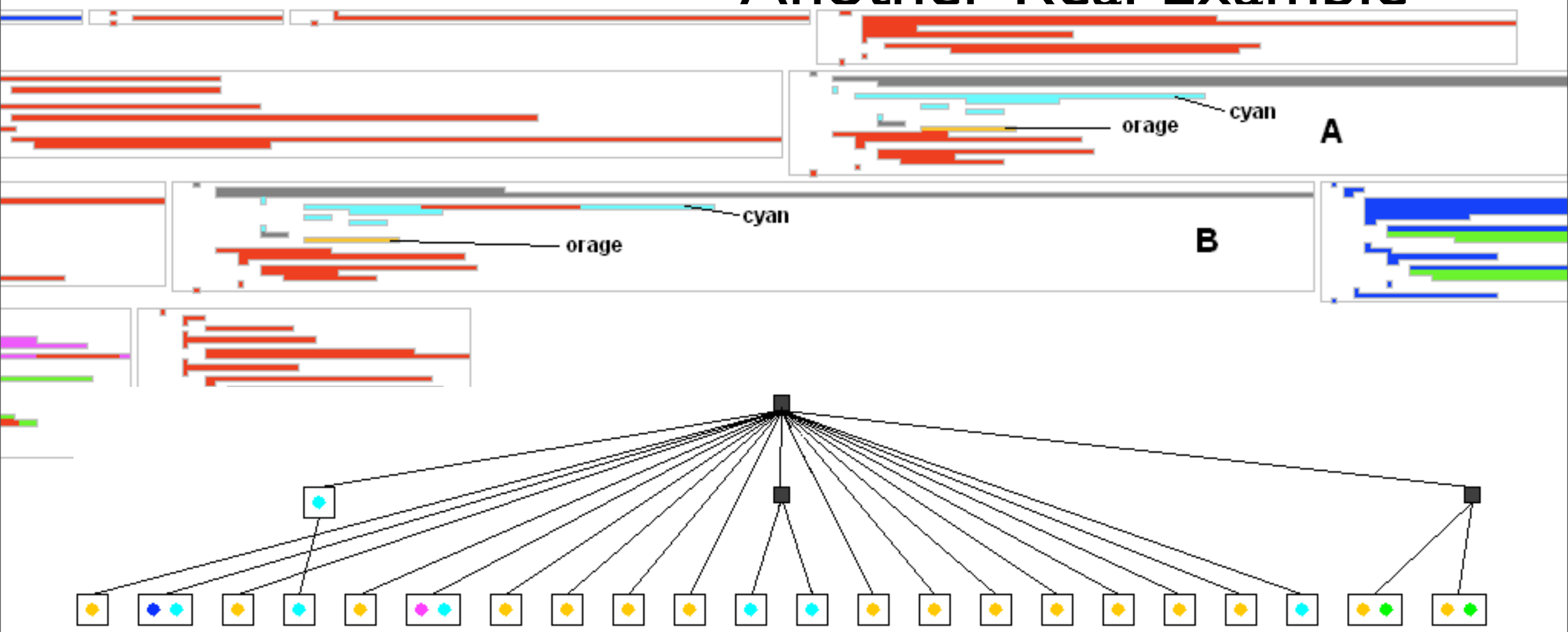


Another Real Example

Another Real Example



Another Real Example



Support for

- Planning the client/self-type checking reengineering pattern
- etc.

Conclusions and Future Work

Conclusions and Future Work

Type highlighting views are promising for

- Program comprehension
- Planning/estimating restructuring actions

Questions

- How to detect high-level policies in legacy code ?
- How many client methods should we have ?

My apologies to color blind people

Type Highlighting.
A Client-Driven Visual Approach for Class Hierarchies Reengineering

Petru Florin Mihancea

LOOSE Research Group
“Politehnica” University of Timișoara, Romania
petru.mihancea@cs.upt.ro

Looking forward to seeing you all in Timișoara
icsm2010.upt.ro