

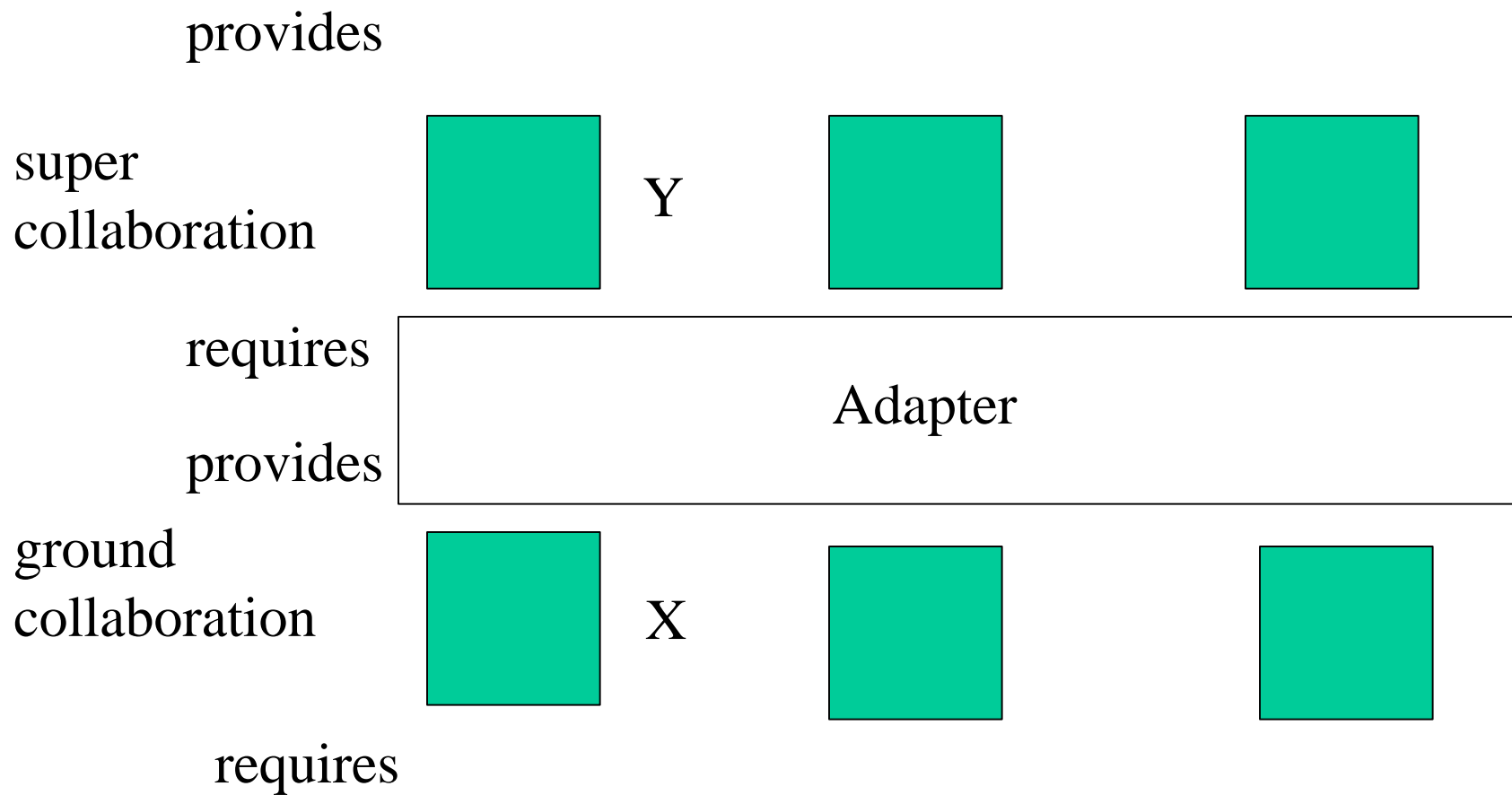
Plug & Play Adapters

```
Adapter = "adapter" AdapterName "{ "  
    <ground> CollabName <super> CollabName  
    List(MemberDefinition)  
    List(AdaptClause) " }".
```

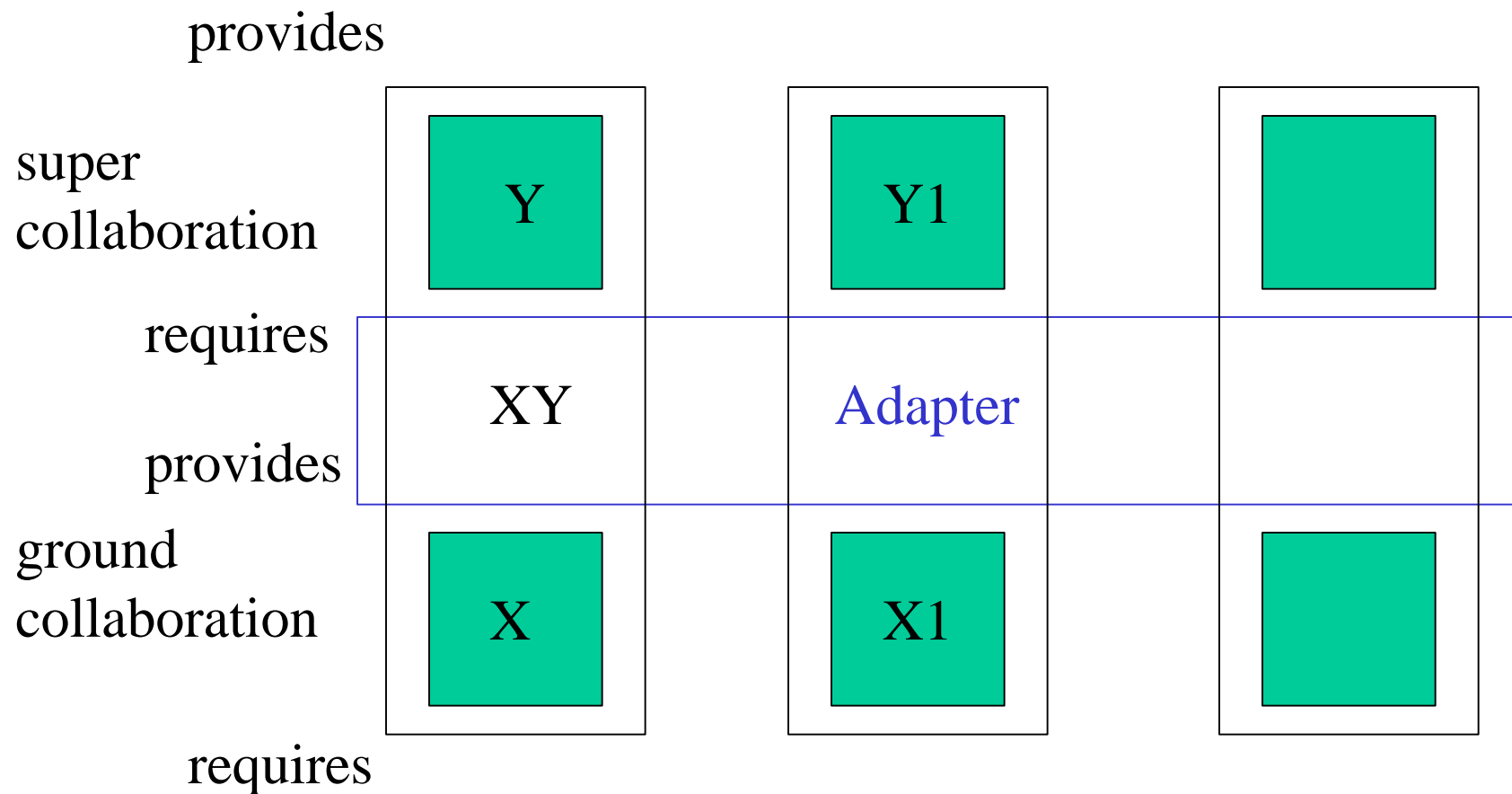
```
AdaptClause = <ground> ClassName "as"  
    <super> ClassName AdaptationBody.
```

```
AdaptationBody = List(MethodDefinition).
```

Plug & Play Adapters



Plug & Play Adapters: they are partitioned



Why partitioning?

- When we implement the required interface of Y , we might want to use the provided interface of $X1$ and $Y1$. Why is this not needed?
- Mira writes: in XY part of connector all public provided X and Y definitions are visible. Why cannot we use the provided $X1$ definitions? Why are they not needed?

Why partitioning?

- I argue that implementing the required interface of Y may require services of multiple participants of the ground collaboration. Example (in DJ):
- `class Y {Z required() { return TraversalGraph.compute(cd, new Strategy("from Y to Z")).fetch();}` may traverse through several participants.

Personalities

- By partitioning the adapters we say that $X + XY$ is a personality that requires exactly what Y provides. Therefore Y personifies $X + XY$.

Subclasses

X as Y XY is equivalent to

```
class X extends Y
```

```
{ Xbody; XYbody }
```

- X gets the provided interface of Y
- The methods that Y requires are provided in subclass X.

Proposal

- In *XY* part of connector all public provided *Y* methods and all provided methods of ground collaboration are visible.

Collaboration Refinement

