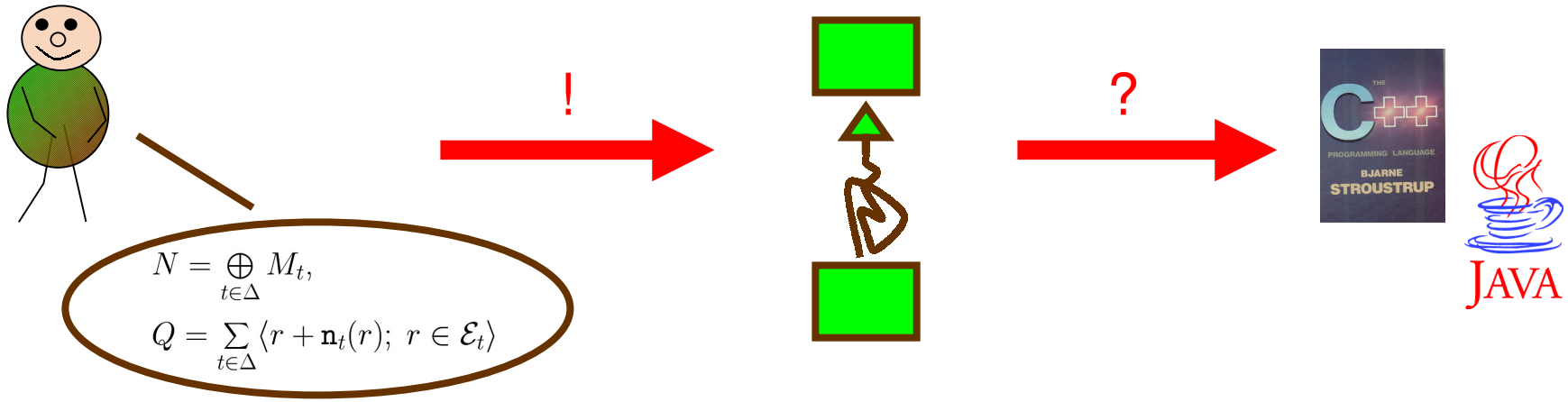
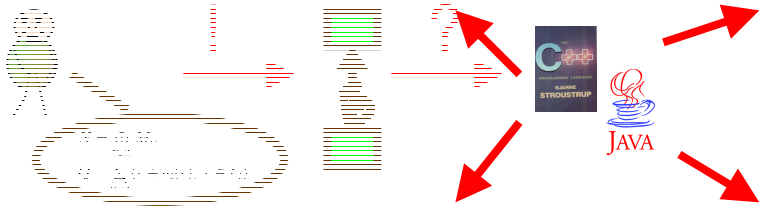
Mathematical Use Cases lead naturally to non-standard Inheritance Relationships – How to make them accessible in a "mainstream" language?

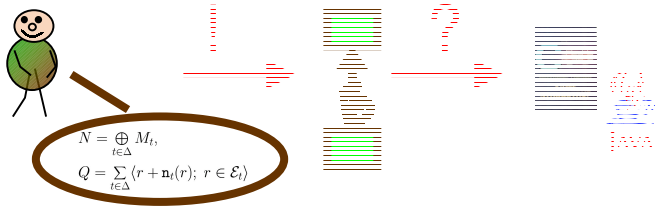


- by:
  - Marc Conrad, Tim French, Carsten Maple (University of Luton)
  - Sandra Pott (University of York)



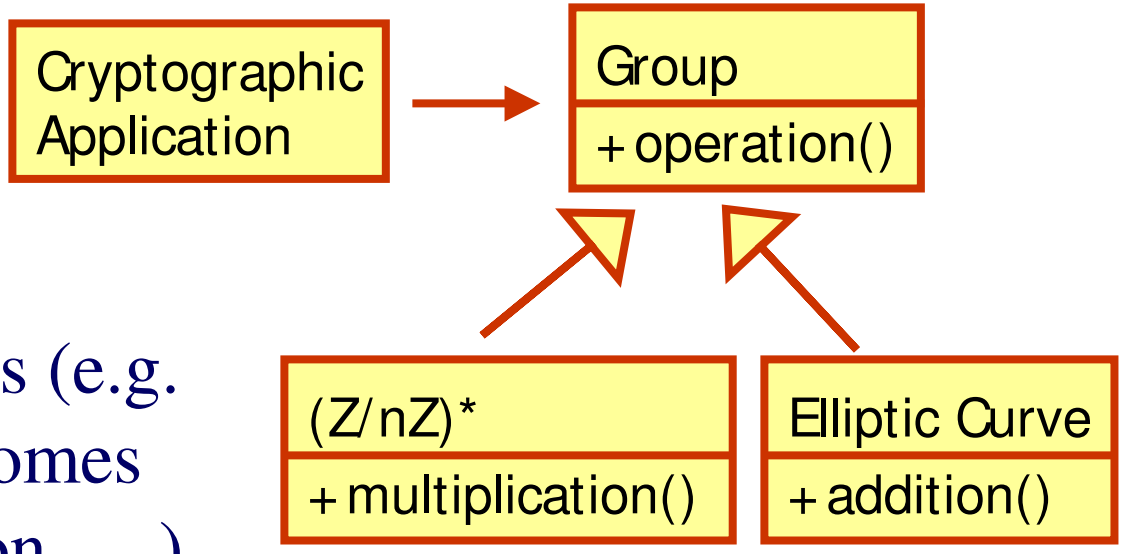
Possible Strategies

- Adding new features to the mainstream language itself (negotiations with the “owner of the language”).
- Providing an “add-on” to the mainstream language (library, pre-processor, package, …).
- Developing an extension of the mainstream language.
- Developing a new language and educating possible users.

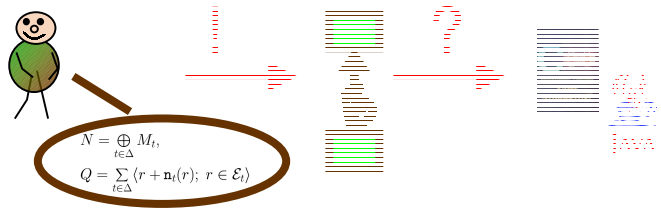


Use Case I – Overriding & Renaming

- (G,o) a group with an operation. Can be *additive (+)* or multiplicative (x).



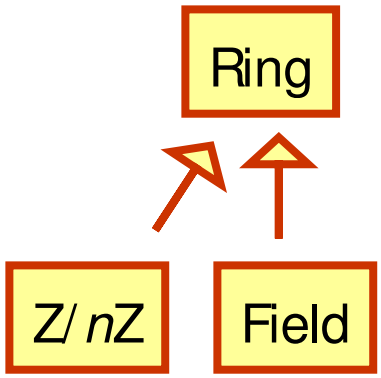
- Also other examples (e.g. endomorphism becomes matrix multiplication, …)



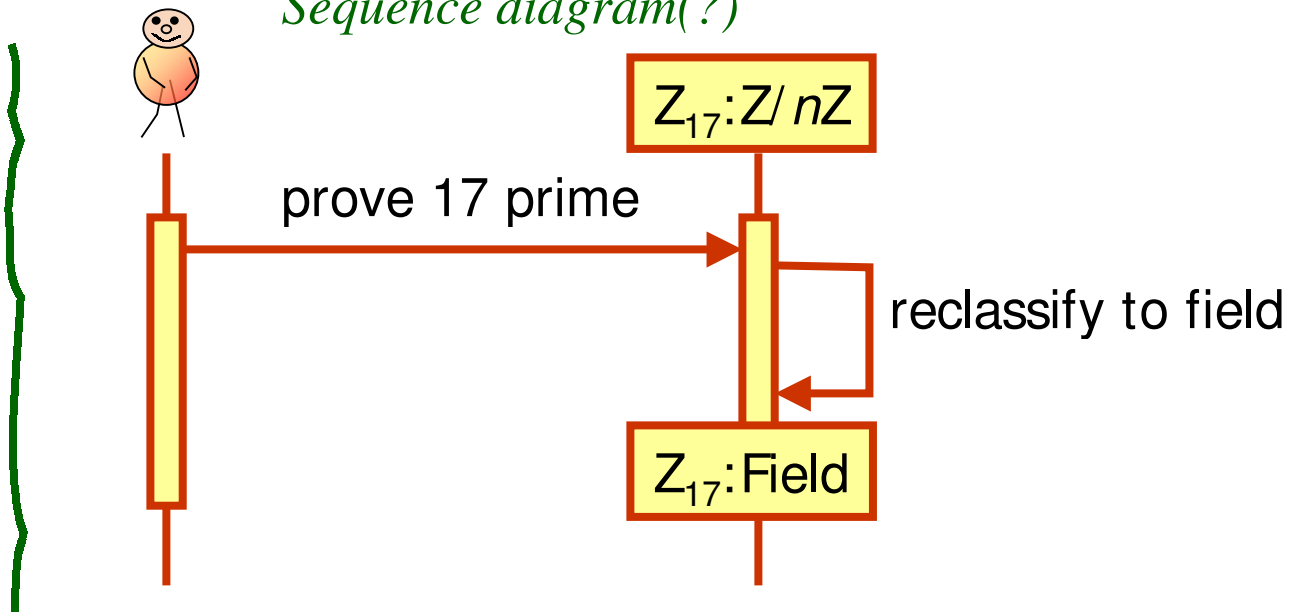
Use Case II – Reclassification

- $\mathbb{Z}/n\mathbb{Z}$ can be a *field* or a *ring* (depending on n).

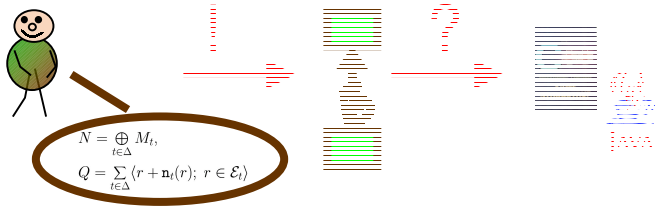
Class diagram



Sequence diagram(?)

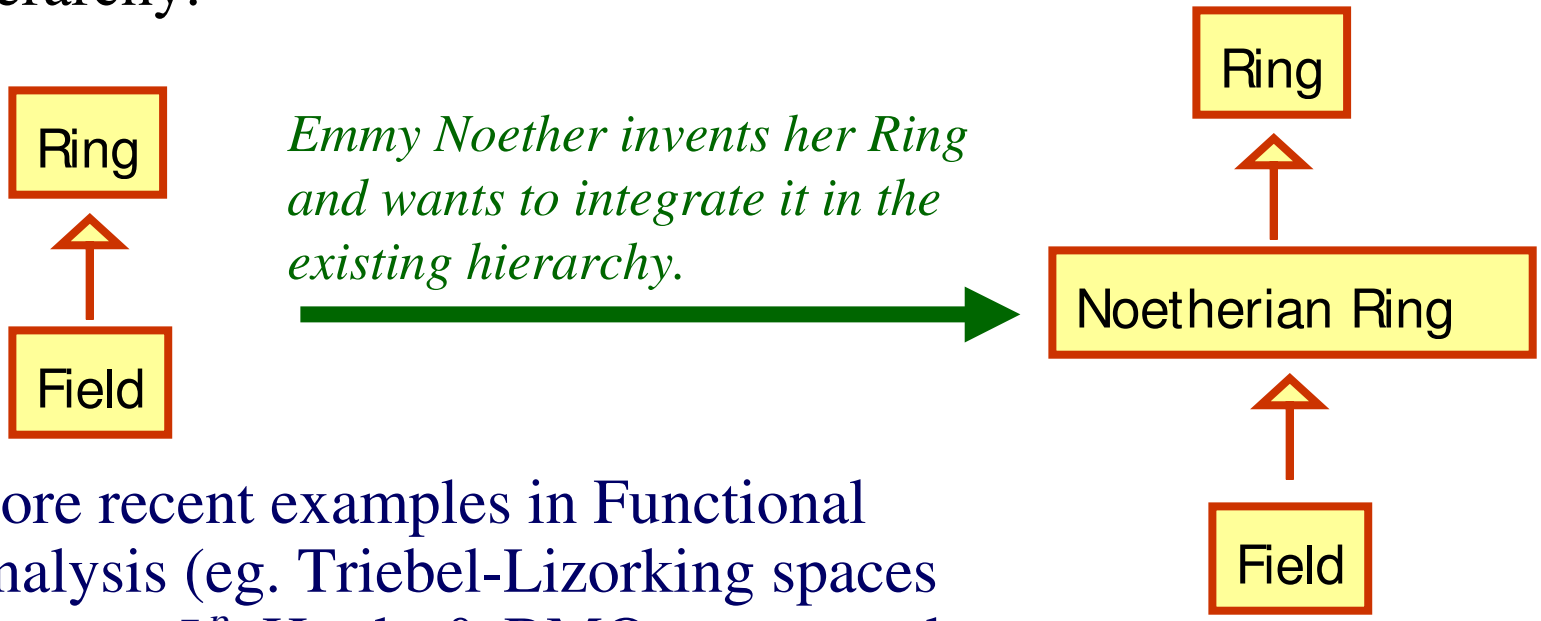


- Also other examples (Euclidian Ring vs. UFD vs. Noetherian Ring …, abelian groups, …)

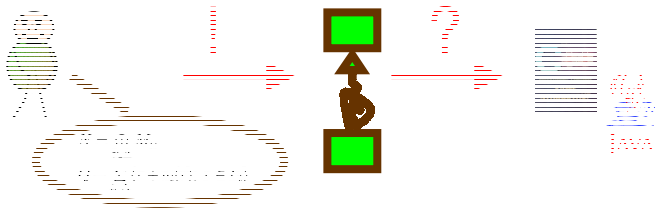


Use Case III – Interclassing

- New structures in Mathematics are discovered or defined (or simply "non-standard") but belong inside an existing inheritance hierarchy.



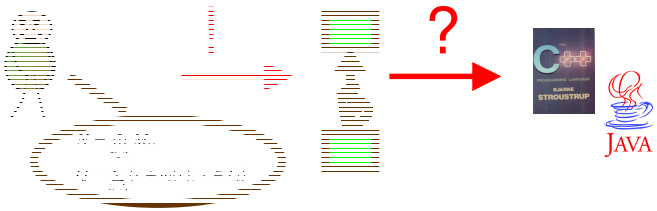
- More recent examples in Functional Analysis (eg. Triebel-Lizorking spaces between L^p, Hardy & BMO spaces and Banach spaces, …), …



Discussion

- *These features are not new!*
- Overriding with renaming: Eiffel, Python
- Reclassification: Cecil (predicate classes), Self, the Darwin Project, Fickle
- Interclassing:OFL (hyper-generic parameters), Shadows(?)

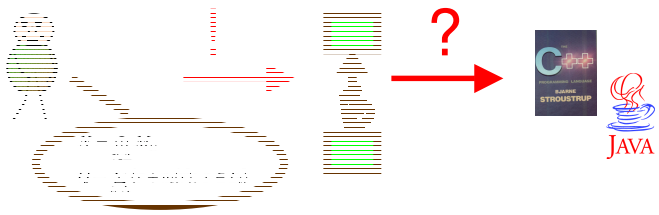
- Common to all these languages, architectures, solutions:
  - They are mostly unrelated to each other.
  - They are not usually applied in Mathematical context
  - They are unknown to the intended target user population (i.e. Mathematicians).



Possible Java Syntax of Overriding & Renaming

- Implementation could be straightforward, eg. using a keyword *override* (cf. C#).

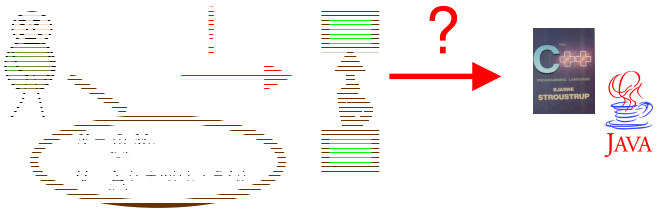
```
public class Group {
        public abstract Elt operation( Elt a, Elt b );
}
public class EllipticCurve extends Group {
        public override operation Elt add( Elt a, Elt b) { …}
…
}
```



Possible Java Syntax of Reclassification

- Define a method in the java.lang.Object class (cf. clone()).

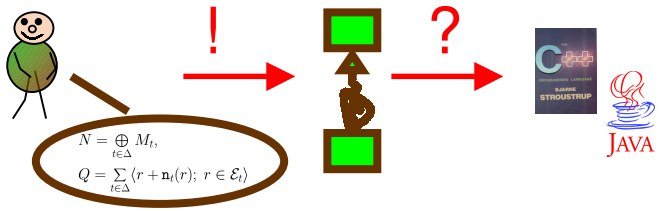
```
// module=17: Z/17Z is instantiated:
Ring Z_nZ = new ModularIntegerRing(module);

// …
// Z/17Z is reclassified because it is a Field:
if(Z_nZ.getModule.isPrime() ) {
  Z_nZ.reclassify(
      Class.forName("com.perisic. FinitePrimeField"));
}
```



Possible Java Syntax of Interclassing

- Any suggestions?

?



Summary

- Solutions to these problems *do* exist.
- But not in Java! (C++, C#, …)
- In Mathematics these "esoteric" features exist *naturally*.
- *So they should be naturally accessible in mainstream languages ("Interclassing for Dummies").*
- Maybe similar problems in other application areas.

- If a language feature cannot be used it is useless.