System(Aircraft)

- **The types**

  [PERSON] the set of all possible uniquely identified persons

  capacity: N
  YESORNO ::= yes | no
  RESPONSE ::= OK | twoErrors | onBoard | full | notOnBoard

- **The state**

Aircraft

onboard: $\mathbb{P}$PERSON

 # onboard ≤ capacity

- **Initialization operation**

Init

Aircraft'

onboard' = Ø

System(Aircraft)

Operations

$Board_0$

Δ Aircraft
p?: PERSON

$p? \notin onboard$
$\#onboard < capacity$
$onboard' = onboard \cup \{p?\}$

$Disembark_0$

Δ Aircraft
p?: PERSON

$p? \in onboard$
$onboard' = onboard \setminus \{p?\}$

System(Aircraft)

- **Enquiry operations**

Number

Ξ**Aircraft**

numOnboard!: $\mathbb{N}$

---

numOnboard! = # onboard

Onboard

Ξ **Aircraft**

p?: PERSON

reply!: YESORNO

(p? $\in$ **onboard** $\wedge$ **reply! = YES)** $\vee$

(p? $\notin$ **onboard** $\wedge$ **reply! = NO)**

System(Aircraft)

- **Dealing with errors**

$$OKMessage == [rep!: RESPONSE \mid rep! = OK]$$

BoardError

$$\Xi Aircraft$$
$$p?: PERSON$$
$$rep!: RESPONSE$$

$$(p? \in onboard \land \#onboard = capacity \land rep! = twoError) \lor$$
$$(p? \in onboard \land \#onboard < capacity \land rep! = onBoard) \lor$$
$$(p? \notin onboard \land \#onboard = capacity \land rep! = full)$$

DisembarkError

$$\Xi Aircraft$$
$$p?: PERSON$$
$$rep!: RESPONSE$$

$$p? \notin onboard \land rep! = notOnBoard$$

System(Aircraft)

- **Final version of operations**

$$\textbf{Board} == (\textbf{Board}_0 \land \textbf{OKMessage}) \lor \textbf{BoardError}$$

$$\textbf{Disembark} == (\textbf{Disembark}_0 \land \textbf{OKMessage}) \lor \textbf{DisembarkError}$$

Class-work: Student Programme of Modules

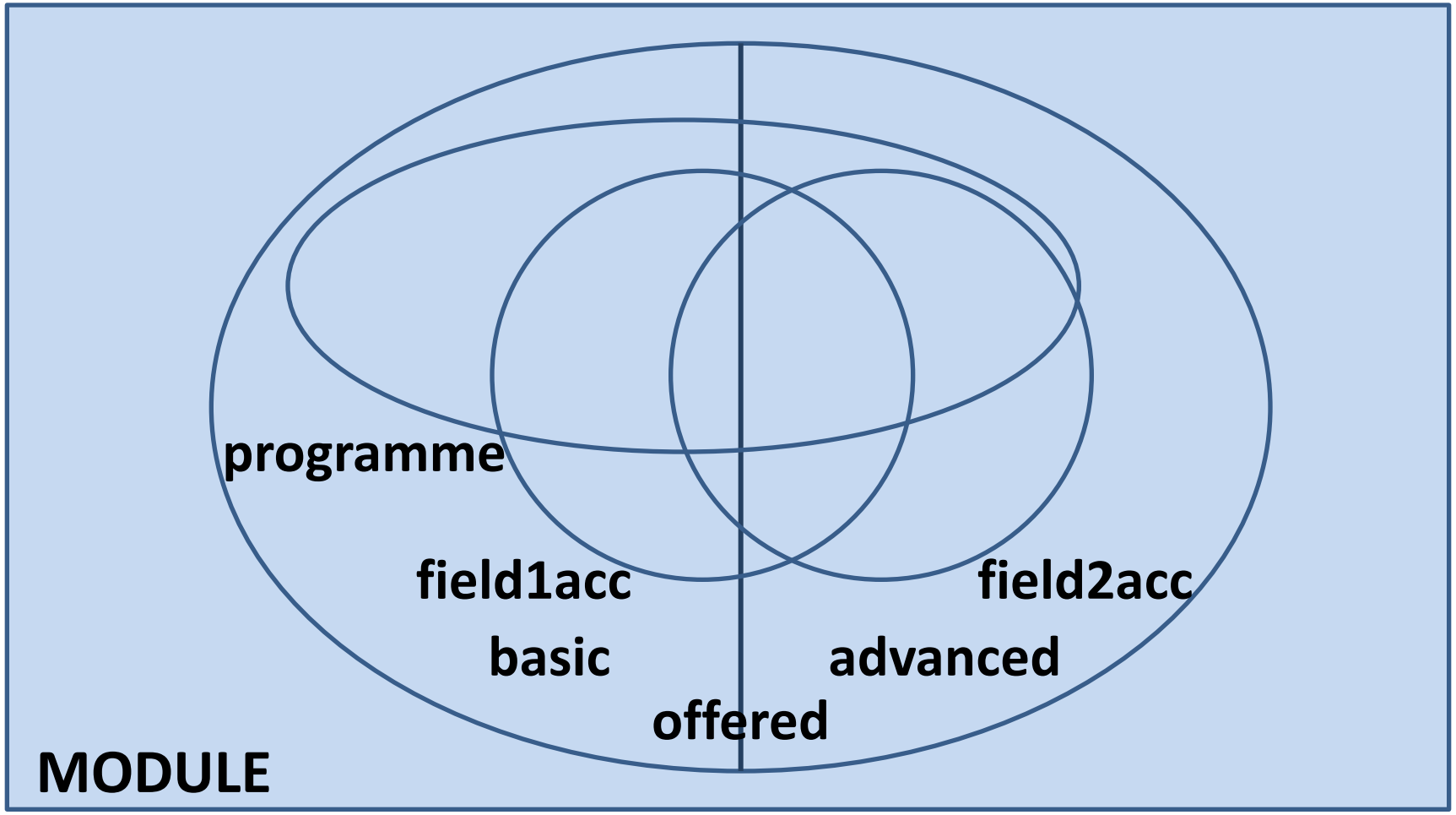
- **This specification concerns a student on a modular course. The student chooses modules from those offered and constructs a *programme* by *adding* and *deleting modules*. The programme is *viable* if it fulfils certain conditions. At least one viable programme must exist.**

  Certain modules are *offered*. An offered module is either *basic* or *advanced* (not both). Certain offered modules are deemed to be *acceptable* to *field1* and certain to *field2*. A module may be acceptable to more than one field or to none.

  For there to be at least one *viable* programme of modules, there must at least 18 offered modules, at least 7 offered that are advanced and acceptable to *field1,* at least seven offered that are advanced and acceptable to *field2*, and at least 16 offered that are advanced and acceptable to the field combination.

Class-work: Student Programme of Modules

Venn diagram: the relationships between the sets in this specification



Class-work: Student Programme of Modules

- Type
- State
- Initialization operation
- Operations
  - Adding a module
  - Deleting a module
- Enquiries
  - Viable programme
- Error operations
  - Error in adding
  - Error in deleting
- Final versions of operations

Class-work: Student Programme of Modules

- **Type**

[MODULE] the set of all possible modules (module identifications)

$$
\begin{array}{l}
\text{offered, advanced, basic,} \\
\text{field1acc, field2acc: } \mathbb{P}\text{MODULE} \\
\hline
\text{advanced} \cap \text{basic} = \emptyset \\
\text{advanced} \cup \text{basic} = \text{offered} \\
\text{field1acc} \subseteq \text{offered} \\
\text{field2acc} \subseteq \text{offered} \\
\#\,\text{offered} \geq 18 \\
\#\,(\text{field1acc} \cap \text{advanced}) \geq 7 \\
\#\,(\text{field2acc} \cap \text{advanced}) \geq 7 \\
\#\,((\text{field1acc} \cup \text{field2acc}) \cap \text{advanced}) \geq 16
\end{array}
$$

Class-work: Student Programme of Modules

State

Student

programme: $\mathbb{P}$ MODULE

---

programme $\subseteq$ offered

Initialization operation

Init

Student'

---

programme' = Ø

Class-work: Student Programme of Modules

Operations

Add_0

Δ Student

m?: MODULE

---

m? $\in$ Offered

m? $\notin$ programme

programme' = programme $\cup$ {m?}

$Delete_0$

Δ Student

m?: MODULE

---

m? $\in$ programme

programme' = programme $\setminus$ {m?}

Class-work: Student Programme of Modules

Enquiries

YESORNO ::= yes | no

Viable
Ξ Student
reply!: YESORNO

(# programme ≥ 18 ∧
#(programme ∩ field1acc ∩ advanced) ≥ 7 ∧
#(programme ∩ field2acc ∩ advanced) ≥ 7 ∧
#(programme ∩ (field1acc ∪ field2acc) ∩ advanced) ≥ 16 ∧
reply! = yes)
∨
(¬ (# programme ≥ 18 ∧
#(programme ∩ field1acc ∩ advanced) ≥ 7 ∧
#(programme ∩ field2acc ∩ advanced) ≥ 7 ∧
#(programme ∩ (field1acc ∪ field2acc) ∩ advanced) ≥ 16) ∧
reply! = no)

Class-work: Student Programme of Modules

Error operations

RESPONSE ::= OK | noSuchModule | alreadyRegistered | notRegistered

AddError

Ξ Student
m?: MODULE
resp!: RESPONSE

(m? ∉ offered ∧ resp! = noSuchModule)
∨
(m? ∈ programme ∧ resp! = alreadyRegistered)

DeleteError

Ξ Student
m?: MODULE
resp!: RESPONSE

---

m? ∉ programme ∧ resp! = notRegistered

Class-work: Student Programme of Modules

- **Final version of operations**

$$OKMessage == [resp!: RESPONSE \mid resp! = OK]$$

$$Add == (Add_0 \land OKMessage) \lor AddError$$

$$Delete == (Delete_0 \land OKMessage) \lor DeleteError$$