Status of the J-PET analysis framework

Wojciech Krzemień, Michał Silarski, Karol Stola, Damian Trybek

Simposium on Positrion Emission Tomography
Kraków, September 19th – 22nd, 2013

Plan of the presentation

- Objectives
- Some technical details
- General architecture
- Data flow
- Current status
- Outlook

Aims

- Provide a simple framework for development of „lab“ analyses and calibration procedures.
- Automatize and standarize some common operations like input/output operations, event by event analysis etc.
- Implement data structure hierarchy (hits, signals, LORs ...)

Longer-term aims

- Develop a reconstruction module for the PetController.
- Create stable reconstruction and calibration procedures at different stages of the analysis.

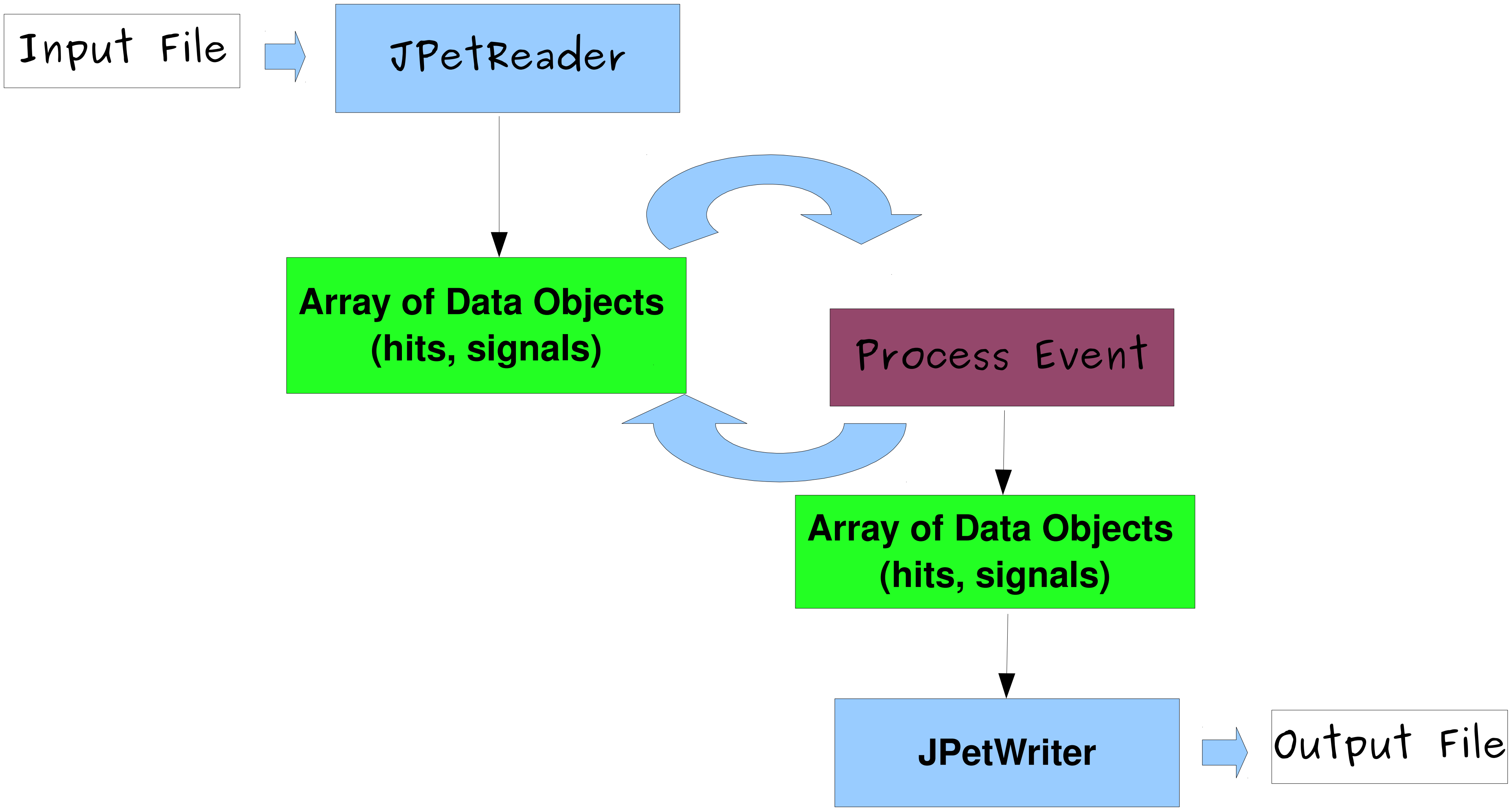
Some technical details

- language: C++, Standard Template Library
- operating system: Linux (Ubuntu, Suse),
- installation: Makefiles
- Subversion control system: **git** repository & **Redmine** for task managment,
- Unit testing based on **BOOST** library,
- code documentation in **Doxygen**,
- part of the system based on the **ROOT** framework.

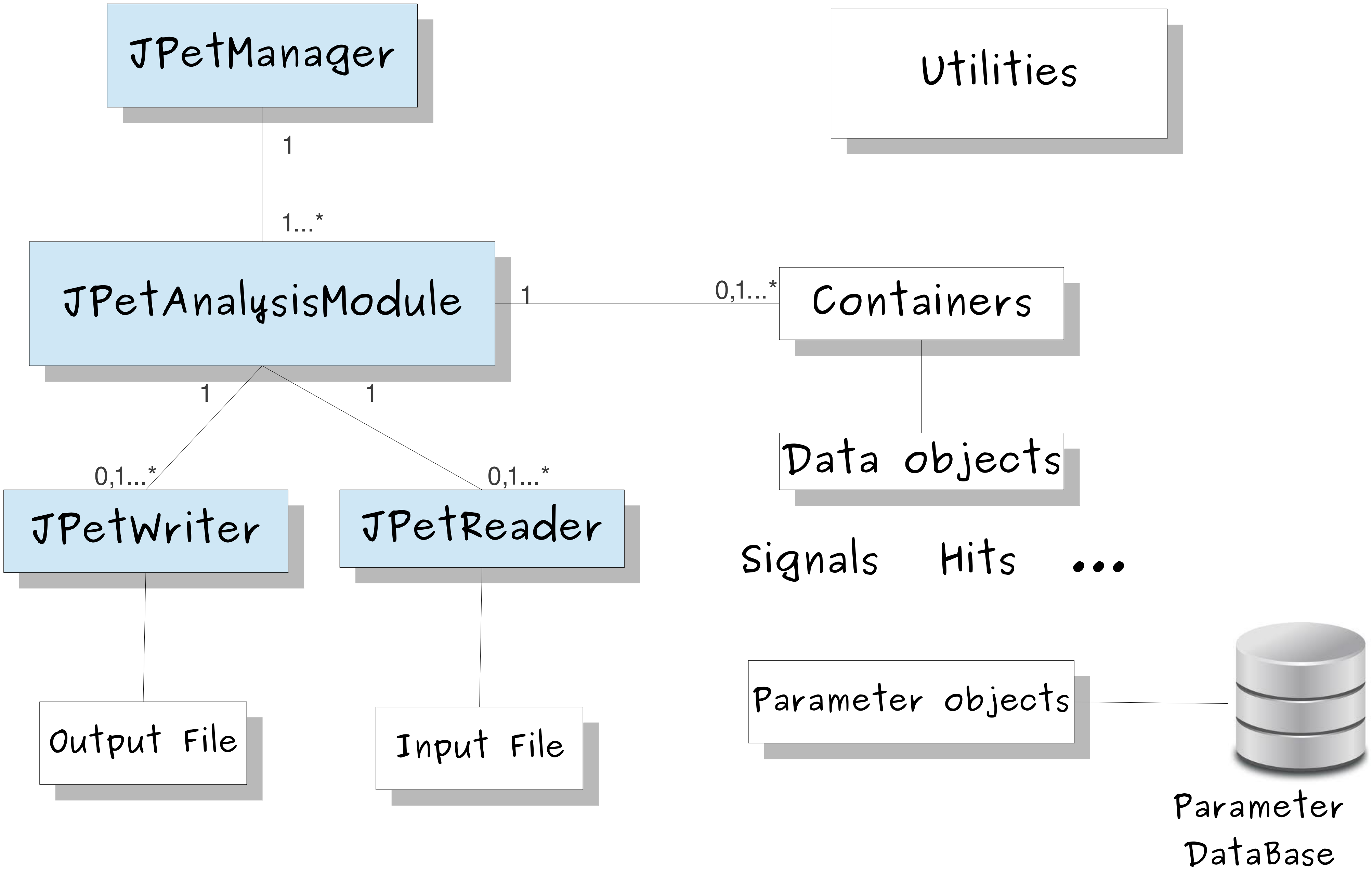
Data levels

Level	Objects	Files
raw experimental data		tslot.hld
Level 0	channels	tslot.unp
Level 1	time slot	tslot.raw tslot.cal
Level 2	signals	phys.sig
Level 3	hits	phys.hit
Level 4	Events(LORs)	phys.eve

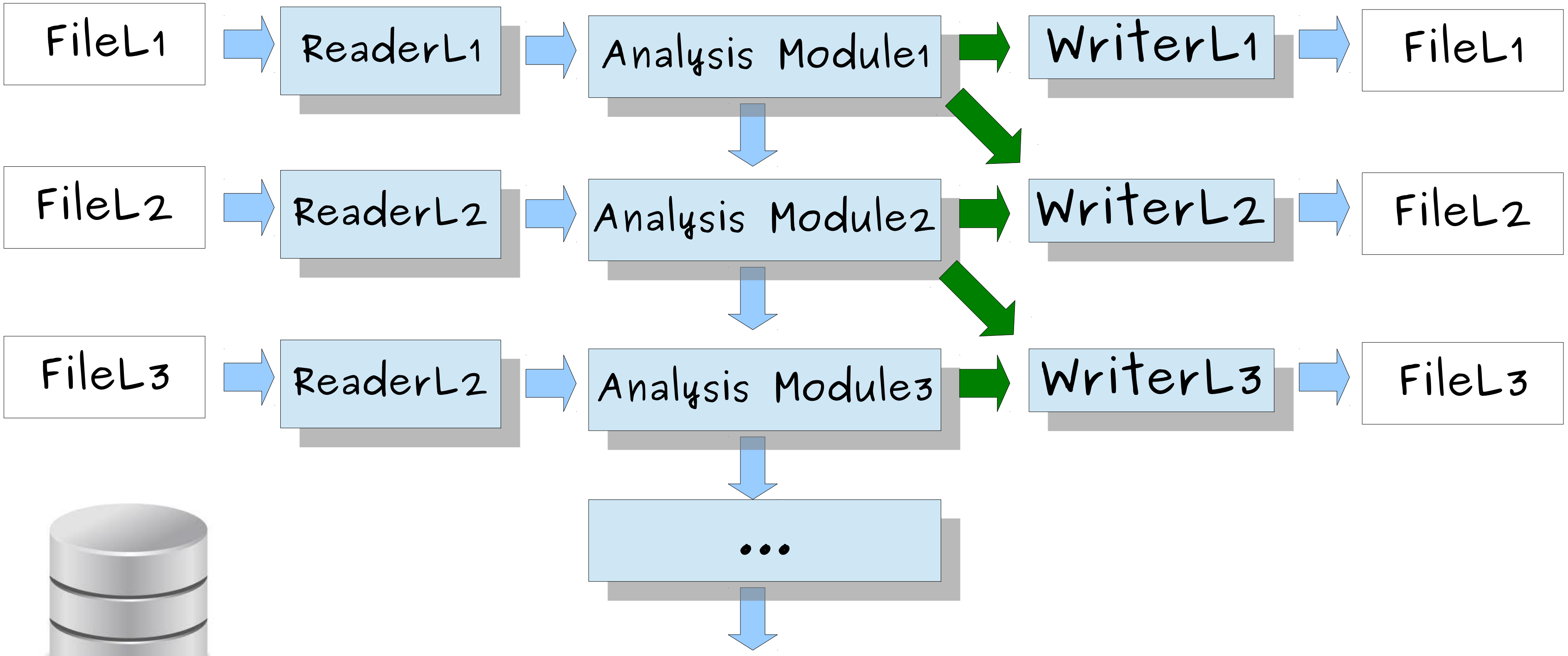
Idea


Input File
JPetReader
Array of Data Objects (hits, signals)
Process Event
Array of Data Objects (hits, signals)
JPetWriter
Output File

General architecture



Data flow



Analysis Module e.g.:

- Matching procedure
- Reconstruction procedure
- Calibration procedure

JpetReader/ JPetWriter

- General interfaces for input/output operations at any level of the analysis.

```
    void OpenFile(const char* filename);
    void CloseFile();
    long long GetEntries()const;
    int GetEntry(int number);
```

File format: tslot.Unp

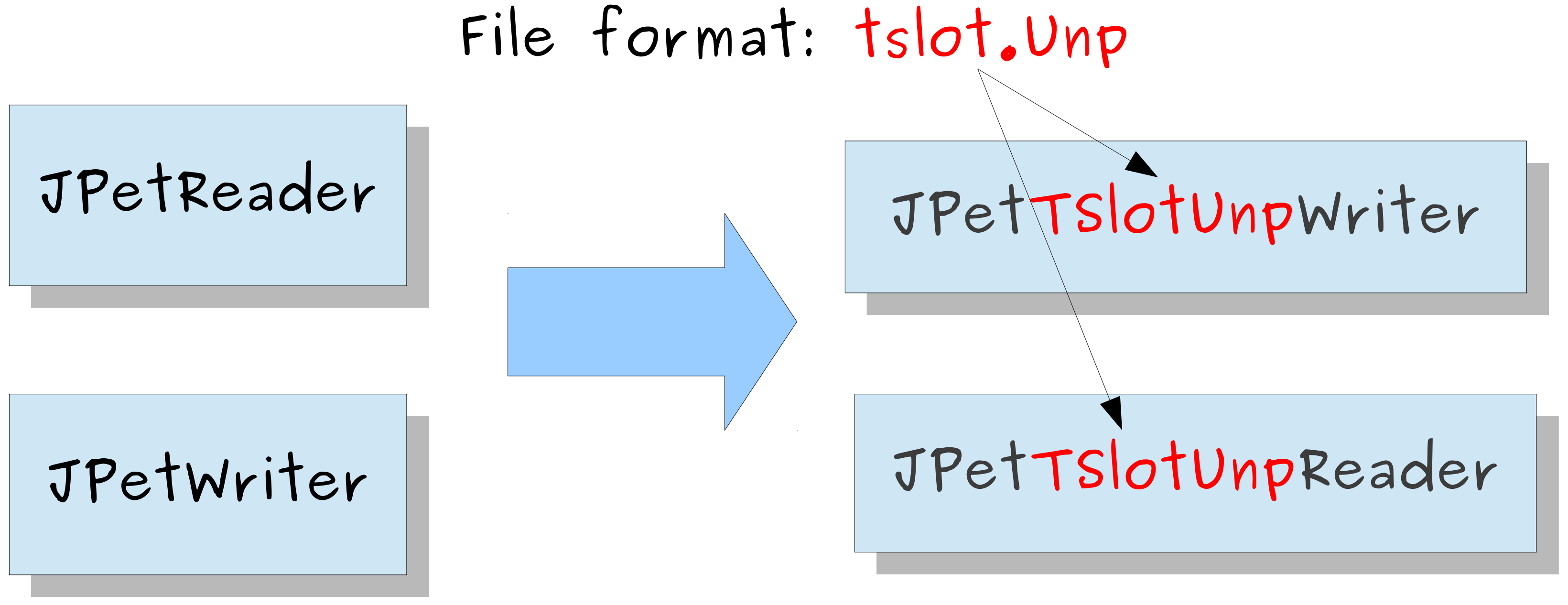
JPetReader

JPetWriter

JpetReader/ JPetWriter

- General interfaces for input/output operations at any level of the analysis.

```
    void OpenFile(const char* filename);
    void CloseFile();
    long long GetEntries()const;
    int GetEntry(int number);
```



JpetReader/ JPetWriter

- General interfaces for input/output operations at any level of the analysis.

```
    void OpenFile(const char* filename);
    void CloseFile();
    long long GetEntries()const;
    int GetEntry(int number);
```

file format: phys.hit

JPetReader

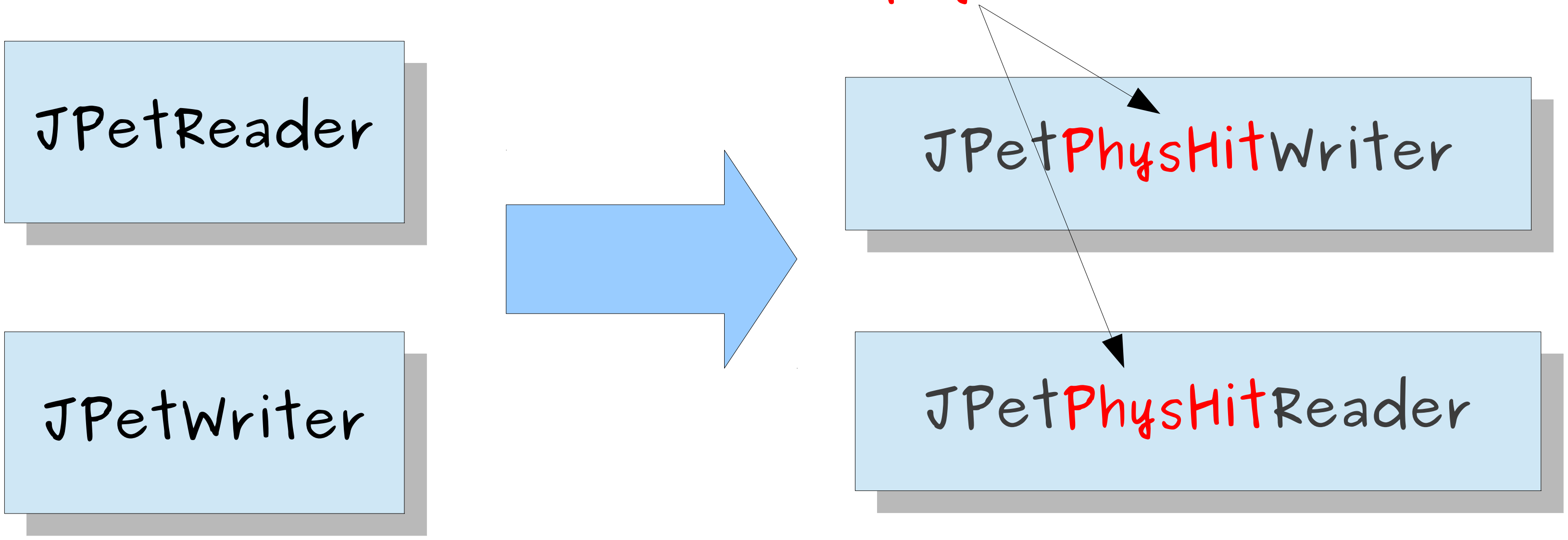
JPetWriter

JpetReader/ JPetWriter

- General interfaces for input/output operations at any level of the analysis.

```
    void OpenFile(const char* filename);
    void CloseFile();
    long long GetEntries()const;
    int GetEntry(int number);
```

File format: phys.hit



Analysis module

- General „template“ class that can be used at any level of the analysis.
- Predefined functions filled by user's code

```
   virtual void CreateInputObjects()=0; //
   virtual void CreateOutputObjects()=0; //
   virtual void Exec()=0; // called for every event
   virtual void Terminate()=0; // called once when analysis
 terminates
```

Analysis module

- Every analysis module is registered by the Manager.
- Analysis modules can contain e.g:
  - Matching procedures
  - reconstruction algorithms,
  - calibration procedures,
  - ...

Current status

- Backbone architecture is implemented.
- Data object classes are defined (hits, signals, events, scintillators, photomultipliers).
- Simple example including reading the experimental file and extracting some basic information is provided.
- We are currently adding Readers and Writers on other levels.

Outlook

- Information from parameter DataBase needs to be included.
- The matching algorithms/calibration algorithms for all levels should be implemented.