Scanning MC-files with automatic event-weighting

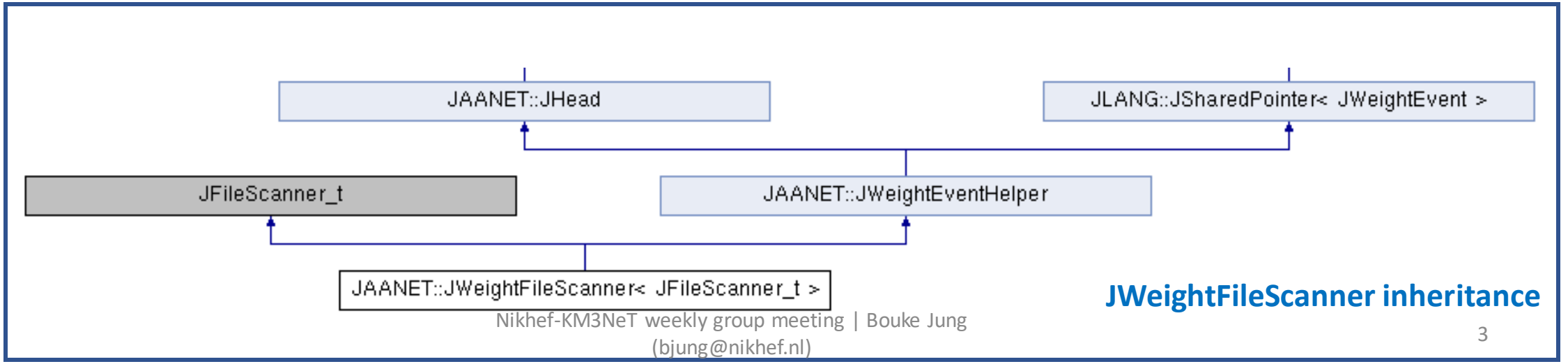
In a nutshell

- Running a program over many different (types) of MC files is possible... But not always practical nor intuitive at the moment:
  - Requires knowledge which of the (many different) files should be joined together
  - Requires you to know how to weight each combined file type
- A general Jpp-framework is made available to accomplish the above and avoid any inconsistensies
  - Inspired by Maarten's JHeadSet application and Alba and Alfonso's prior work

Software Structure

- Top datastructure called "JWeightFileScannerSet"
  - Ordered set of JWeightFileScanner objects
  - Ordering guaranteed by JHead::less operator

- A JWeightFileScanner object consists of:
  1. A filescanner which can be used to loop over all events corresponding to one MC-header type
  2. An event-weight helper, which does the bookkeeping on the combined header info and assigns the correct weight to each event correspondingly

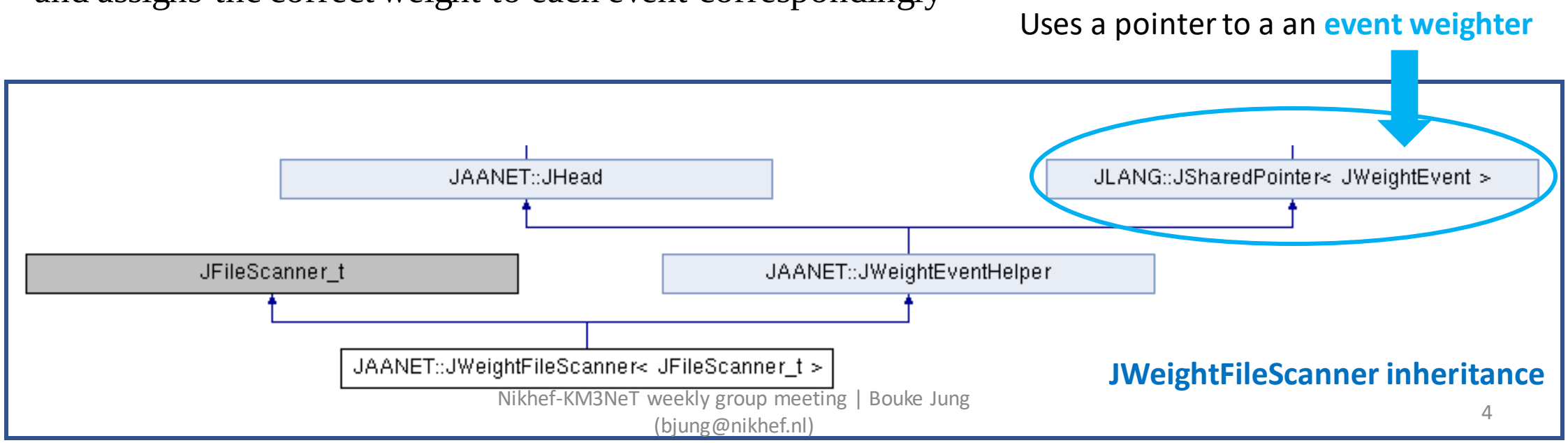


JWeightFileScanner inheritance

Software Structure

- Top datastructure called "JWeightFileScannerSet"
  - Ordered set of JWeightFileScanner objects
  - Ordering guaranteed by JHead::less operator

- A JWeightFileScanner object consists of:
  1. A filescanner which can be used to loop over all events corresponding to one MC-header type
  2. An event-weight helper, which does the bookkeeping on the combined header info and assigns the correct weight to each event correspondingly



JWeightEvent

- Simple interface containing three methods
  - A configure-function, which sets the global event weight given a (combined) header
  - A check-method to check consistency of a header with this event weighter
  - A GetWeight-function which returns the correct weight, given a MC-event

```
struct JWeightEvent :
  public JClonable<JWeightEvent>
{
  /**
   * Virtual destructor.
   */
  virtual ~JWeightEvent()
  {}


  /**
   * Configuration.
   *
   * \param  header            header
   */
  virtual void configure(const JHead& header) = 0;


  /**
   * Check whether header is consistent with this event weighter.
   */
  virtual bool check(const JHead& head) const = 0;


  /**
   * Get weight of given event.
   *
   * \param  evt               event
   * \return                   weight        [Hz]
   */
  virtual double getWeight(const Evt& evt) const = 0;


  /**
   * Get weight of given event.
   *
   * \param  evt               event
   * \param  flux              neutrino flux [m^-2 s^-1 sr^-1 GeV^-1]
   * \return                   weight        [Hz]
   */
  virtual double getWeight(const Evt&   evt,
                           const double flux) const
  {
    return getWeight(evt);
  }
};
```

JWeightEvent

- Simple interface containing three methods
  - A configure-function, which sets the global event weight given a (combined) header
  - A check-method to check consistency of a header with this event weighter
  - A GetWeight-function which returns the correct weight, given a MC-event
- So far, three implementations
  1. Mupage weighter
  2. GSeaGen weighter
  3. KM3BUU weighter

```
struct JWeightEvent :
  public JClonable<JWeightEvent>
{
  /**
   * Virtual destructor.
   */
  virtual ~JWeightEvent()
  {}


  /**
   * Configuration.
   *
   * \param  header            header
   */
  virtual void configure(const JHead& header) = 0;


  /**
   * Check whether header is consistent with this event weighter.
   */
  virtual bool check(const JHead& head) const = 0;


  /**
   * Get weight of given event.
   *
   * \param  evt               event
   * \return                   weight       [Hz]
   */
  virtual double getWeight(const Evt& evt) const = 0;


  /**
   * Get weight of given event.
   *
   * \param  evt               event
   * \param  flux              neutrino flux [m^-2 s^-1 sr^-1 GeV^-1]
   * \return                   weight       [Hz]
   */
  virtual double getWeight(const Evt&   evt,
                           const double flux) const
  {
    return getWeight(evt);
  }
};
```

JEventWeightHelper

- Bookkeeper for header information and event weighter

- Six methods

  - A configuration-, getWeight- and header-check-function, which calls the underlying methods of JWeightEvent
  - An addition-function, which combines a given header with the current header info (if check OK) and updates the global event weight
  - Two static methods, which can be used to:

    1. Get the correct event weighter, given a specific header
    2. Get the default header corresponding to a given event weighter

JWeightFileScanner

- Two 'put'-methods for adding single or multiple files

- The configuration method, takes a file list and assigns the correct header and event-weighter

- Note:
  - only files consistent with the first header in a given list, will be added

```
/**
 * Put list of files.
 *
 * \param  input              file list
 * \return                    number of added files.
 */
size_t put(const JFileScanner_t& input)
{
  size_t N = 0;

  for (typename JFileScanner_t::const_iterator i = input.begin(); i != input.end(); ++i) {

    N += size_t(this->put(*i));
  }

  return N;
}


/**
 * Put file.
 *
 * \param  input              file name
 * \return                    true if successfully added; else false.
 */
bool put(const std::string& input)
{
  const JHead head = JSUPPORT::getHeader(input);

  if (this->check(head)) {

    JWeightEventHelper::add(head);
    JFileScanner_t::push_back(input);

    return true;

  } else {

    return false;
  }
}
```

JWeightFileScannerSet

- = The crown to top it off

- Takes a list (any mixture) of different MC-files and

  1. Combines the files of consistent header-types into individual JWeightFileScanner objects
  2. Orders the file-scanners according to the header-information

- Available methods:

  - Two put methods, to insert additional file(s)
  - A get-function to retrieve the JWeightFileScanner object corresponding to a specific header
  - A setLimit-function to limit the number of events read from each scanner (optional)

What's Next

- Try using the framework on gSeaGen and GiBUU input data (cooperation with Johannes)
- Allow scanning of DAQ-files also
  - When combined with MC-info, may be used for inspection of e.g. trigger rate
- Advertise!