```java
package edu.wsu.gridstat.router.condFunction;

/**
 * <p>Title: CondCalcInterface </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Washington State University </p>
 * @author Kjell Harald Gjermundrod
 * @version 1.0
 */
// GridStat packages.
import edu.wsu.gridstat.util.IntHashMap;
import edu.wsu.gridstat.command._CondensationSetupHolder;

public abstract interface CondCalcInterface
{
  /**
  * The <code>initialize</code> method is used to initialize this condensation calculator.
  * <BR>
  * @param setUpHolder The command that is used to set up this condensation   * function.
  * @param statusHolders The holder for the status events that will be used in the calculation.
  */
  public void initialize(_CondensationSetupHolder setUpHolder, IntHashMap statusHolders);

  /**
  * The <code>filterOutValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public boolean filterOutValue(int value);

  /**
  * The <code>filterOutValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public boolean filterOutValue(float value);

  /**
  * The <code>filterOutValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public boolean filterOutValue(boolean value);

  /**
  * The <code>filterInValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public boolean filterInValue(int value);

  /**
  * The <code>filterInValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public boolean filterInValue(float value);

  /**
  * The <code>filterInValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public boolean filterInValue(boolean value);

  /**
  * The <code>filterOutValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public abstract boolean filterOutValue(byte[] value);

  /**
  * The <code>filterInValue</code> method is used to test if the value should be filtered.
  * <BR>
  * @param value The value that is tested for filtering.
  * @return Returns <code>true</code> if the value should be filtered, <code>false</code> otherwise.
  */
  public abstract boolean filterInValue(byte[] value);

  /**
  * The <code>calculate</code> abstract method is used to do the calculation.
  * <BR>
  * @return Returns <code>true</code> if we produced a new value, <code>false</code> if no new value was produced.
  */
  public abstract boolean calculate();
}
```