```
/***********************************                 Base                 Classes
***********************************/

/** @brief Abstract base class for 2D image feature detectors and descriptor extractors
*/
class CV_EXPORTS_W Feature2D : public virtual Algorithm
{
public:
    virtual ~Feature2D();

    /** @brief Detects keypoints in an image (first variant) or image set (second variant).

    @param image Image.
    @param keypoints The detected keypoints. In the second variant of the method keypoints[i] is a
set
    of keypoints detected in images[i] .
    @param mask Mask specifying where to look for keypoints (optional). It must be a 8-bit integer
    matrix with non-zero values in the region of interest.
     */
    CV_WRAP virtual void detect( InputArray image,
                                 CV_OUT std::vector<KeyPoint>& keypoints,
                                 InputArray mask=noArray() );

    /** @overload
    @param images Image set.
    @param keypoints The detected keypoints. In the second variant of the method keypoints[i] is a
set
    of keypoints detected in images[i] .
    @param masks Masks for each input image specifying where to look for keypoints (optional).
    masks[i] is a mask for images[i].
    */
    CV_WRAP virtual void detect( InputArrayOfArrays images,
                         CV_OUT std::vector<std::vector<KeyPoint> >& keypoints,
                         InputArrayOfArrays masks=noArray() );
    /** @brief Computes the descriptors for a set of keypoints detected in an image (first variant) or
image set
    (second variant).
    @param image Image.
    @param keypoints Input collection of keypoints. Keypoints for which a descriptor cannot be
    computed are removed. Sometimes new keypoints can be added, for example: SIFT duplicates
keypoint
```

```
    with several dominant orientations (for each orientation).
    @param descriptors Computed descriptors. In the second variant of the method descriptors[i]
are
    descriptors computed for a keypoints[i]. Row j is the keypoints (or keypoints[i]) is the
    descriptor for keypoint j-th keypoint.
     */
    CV_WRAP virtual void compute( InputArray image,
                                  CV_OUT CV_IN_OUT std::vector<KeyPoint>& keypoints,
                                  OutputArray descriptors );

    /** @overload
    @param images Image set.
    @param keypoints Input collection of keypoints. Keypoints for which a descriptor cannot be
    computed are removed. Sometimes new keypoints can be added, for example: SIFT duplicates
keypoint
    with several dominant orientations (for each orientation).
    @param descriptors Computed descriptors. In the second variant of the method descriptors[i]
are
    descriptors computed for a keypoints[i]. Row j is the keypoints (or keypoints[i]) is the
    descriptor for keypoint j-th keypoint.
    */
    CV_WRAP virtual void compute( InputArrayOfArrays images,
                          CV_OUT CV_IN_OUT std::vector<std::vector<KeyPoint> >& keypoints,
                          OutputArrayOfArrays descriptors );
    /** Detects keypoints and computes the descriptors */
    CV_WRAP virtual void detectAndCompute( InputArray image, InputArray mask,
                                           CV_OUT std::vector<KeyPoint>& keypoints,
                                           OutputArray descriptors,
                                           bool useProvidedKeypoints=false );
    CV_WRAP virtual int descriptorSize() const;
    CV_WRAP virtual int descriptorType() const;
    CV_WRAP virtual int defaultNorm() const;
    CV_WRAP void write( const String& fileName ) const;
    CV_WRAP void read( const String& fileName );
    virtual void write( FileStorage&) const;
    virtual void read( const FileNode&);

    //! Return true if detector object is empty
    CV_WRAP virtual bool empty() const;
};
```