## Software UML Diagram

## **ProfileResult**

+ name : std::string

+ elapsedTime: std::chrono::milliseconds::rep

Struct to store profiler results

## Timer

- name\_ : std::string
- stopped\_ : bool
- startTimepoint\_:std::chrono::time\_point
- <std::chrono::steady\_clock>
- + static profileResult: ProfileResult
- + Timer(name\_ : const char\*)
- + Stop : void
- + ~Timer()

## ObjectTracker

- minConfidence\_ : float
- minOverlap\_ : float
- objectClasses\_: std::unordered\_set<std::string>
- P\_ : cv::Matx34f
- colors\_: std::map<std::string, cv::Scalar>
- network\_: cv::dnn::DetectionModel
- datasetLabels\_ : std::vector<std::string>
- + localizeObjects(cv::Mat) : std::vector<cv::Point3f>
- + localize(const cv::Point2i&) : cv::Point3f
- + detectObject(cv::Mat) : std::tuple<std::vector<float>,

std::vector<float>, std::vector<cv::Rect>>

- parseFile(const std::string&) : std::vector<string>
- visualize(frame: cv::Mat, classId: int, confidence: float, box: const cv::Rect&, objectKeypoint: const cv::Point2i&,

objectLocation: const cv::Point3f&, distance:float)

- + ObjectTracker(objectClasses: const std::unordered\_set<std::string>& extP: cv::Matx34f&, intP: cv::Matx34f&, minConfidence: float, minOverlap: float, detectionModel: const std::string&)
- + datasetLabels() const : std::vector<std::string>

y uses

cv::dnn::DetectionModel

openCV API for Deep Learning based object de