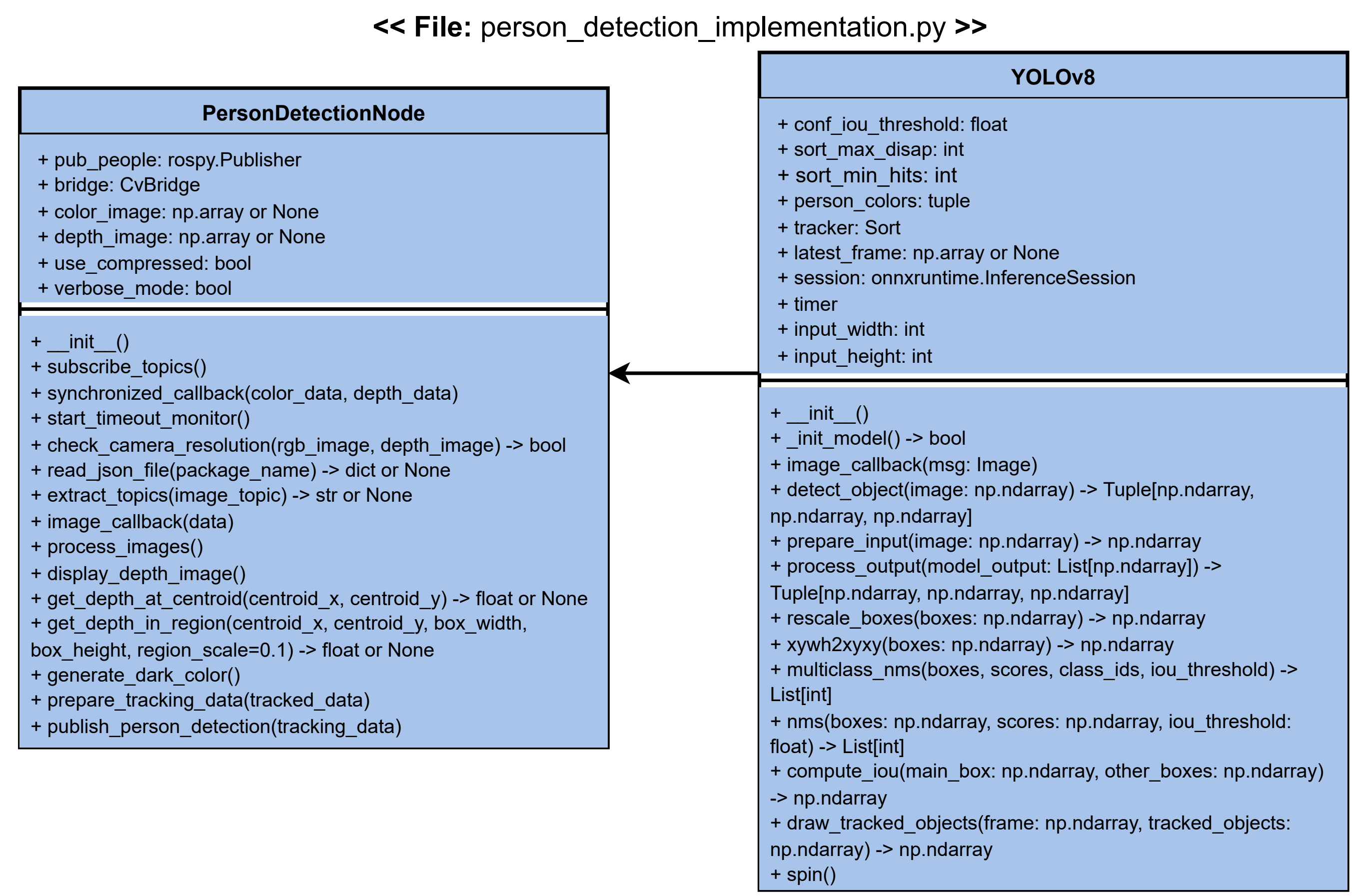

<< File: person_detection_implementation.py >>
PersonDetectionNode
+ pub_people: rospy.Publisher
+ bridge: CvBridge
+ color_image: np.array or None
+ depth_image: np.array or None
+ use_compressed: bool
+ verbose_mode: bool
+ __init__()
+ subscribe_topics()
+ synchronized_callback(color_data, depth_data)
+ start_timeout_monitor()
+ check_camera_resolution(rgb_image, depth_image) -> bool
+ read_json_file(package_name) -> dict or None
+ extract_topics(image_topic) -> str or None
+ image_callback(data)
+ process_images()
+ display_depth_image()
+ get_depth_at_centroid(centroid_x, centroid_y) -> float or None
+ get_depth_in_region(centroid_x, centroid_y, box_width, box_height, region_scale=0.1) -> float or None
+ generate_dark_color()
+ prepare_tracking_data(tracked_data)
+ publish_person_detection(tracking_data)
YOLOv8
+ conf_iou_threshold: float
+ sort_max_disap: int
+ sort_min_hits: int
+ person_colors: tuple
+ tracker: Sort
+ latest_frame: np.array or None
+ session: onnxruntime.InferenceSession
+ timer
+ input_width: int
+ input_height: int
+ __init__()
+ _init_model() -> bool
+ image_callback(msg: Image)
+ detect_object(image: np.ndarray) -> Tuple[np.ndarray, np.ndarray, np.ndarray]
+ prepare_input(image: np.ndarray) -> np.ndarray
+ process_output(model_output: List[np.ndarray]) -> Tuple[np.ndarray, np.ndarray, np.ndarray]
+ rescale_boxes(boxes: np.ndarray) -> np.ndarray
+ xywh2xyxy(boxes: np.ndarray) -> np.ndarray
+ multiclass_nms(boxes, scores, class_ids, iou_threshold) -> List[int]
+ nms(boxes: np.ndarray, scores: np.ndarray, iou_threshold: float) -> List[int]
+ compute_iou(main_box: np.ndarray, other_boxes: np.ndarray) -> np.ndarray
+ draw_tracked_objects(frame: np.ndarray, tracked_objects: np.ndarray) -> np.ndarray
+ spin()