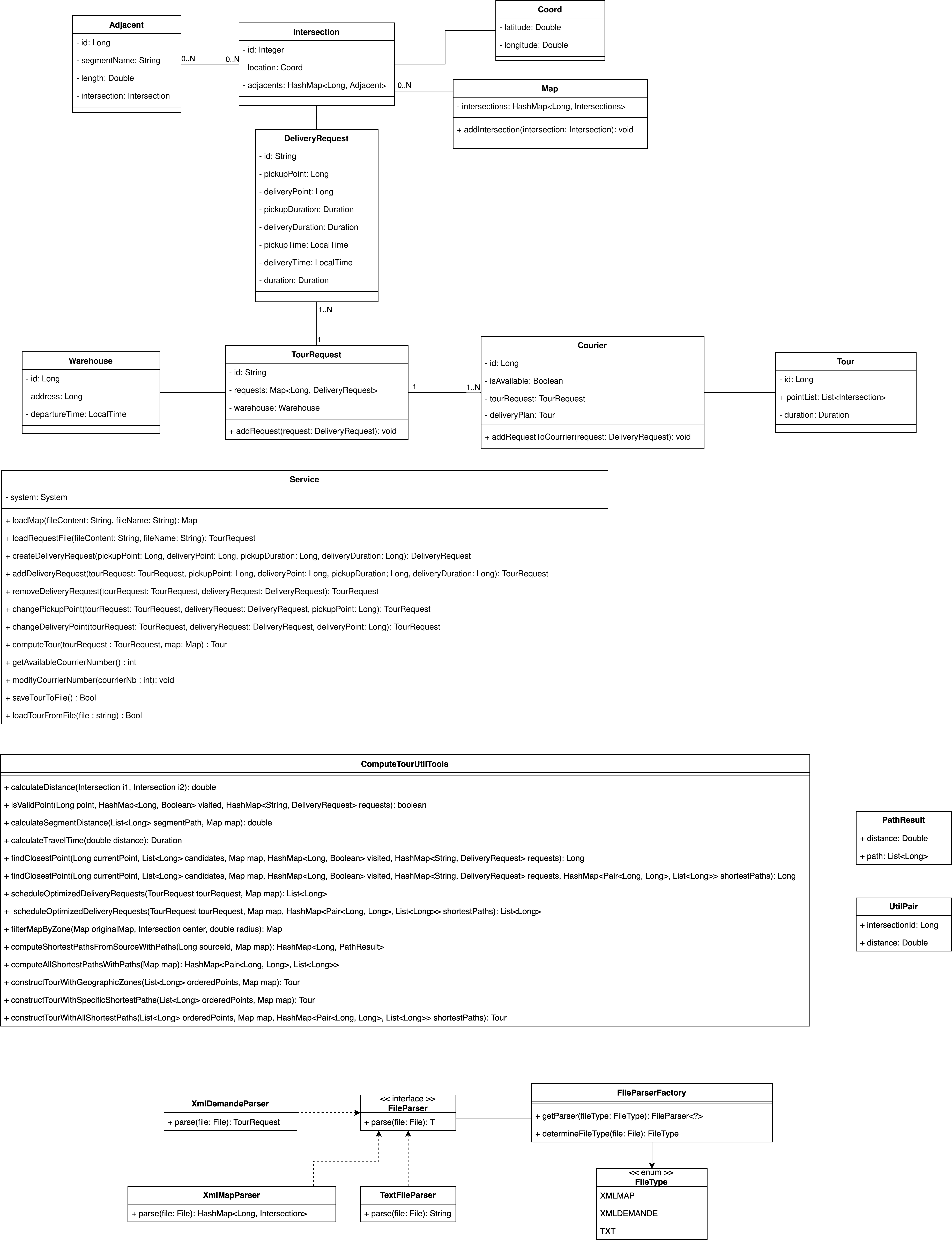
Adjacent
- id: Long
- segmentName: String
- length: Double
- intersection: Intersection
0..N
0..N
Intersection
- id: Integer
- location: Coord
- adjacents: HashMap<Long, Adjacent>
Coord
- latitude: Double
- longitude: Double
0..N
Map
- intersections: HashMap<Long, Intersections>
+ addIntersection(intersection: Intersection): void
DeliveryRequest
- id: String
- pickupPoint: Long
- deliveryPoint: Long
- pickupDuration: Duration
- deliveryDuration: Duration
- pickupTime: LocalTime
- deliveryTime: LocalTime
- duration: Duration
1..N
1
Warehouse
- id: Long
- address: Long
- departureTime: LocalTime
TourRequest
- id: String
- requests: Map<Long, DeliveryRequest>
- warehouse: Warehouse
+ addRequest(request: DeliveryRequest): void
1
1..N
Courier
- id: Long
- isAvailable: Boolean
- tourRequest: TourRequest
- deliveryPlan: Tour
+ addRequestToCourrier(request: DeliveryRequest): void
Tour
- id: Long
+ pointList: List<Intersection>
- duration: Duration
Service
- system: System
+ loadMap(fileContent: String, fileName: String): Map
+ loadRequestFile(fileContent: String, fileName: String): TourRequest
+ createDeliveryRequest(pickupPoint: Long, deliveryPoint: Long, pickupDuration: Long, deliveryDuration: Long): DeliveryRequest
+ addDeliveryRequest(tourRequest: TourRequest, pickupPoint: Long, deliveryPoint: Long, pickupDuration; Long, deliveryDuration: Long): TourRequest
+ removeDeliveryRequest(tourRequest: TourRequest, deliveryRequest: DeliveryRequest): TourRequest
+ changePickupPoint(tourRequest: TourRequest, deliveryRequest: DeliveryRequest, pickupPoint: Long): TourRequest
+ changeDeliveryPoint(tourRequest: TourRequest, deliveryRequest: DeliveryRequest, deliveryPoint: Long): TourRequest
+ computeTour(tourRequest : TourRequest, map: Map) : Tour
+ getAvailableCourrierNumber() : int
+ modifyCourrierNumber(courrierNb : int): void
+ saveTourToFile() : Bool
+ loadTourFromFile(file : string) : Bool
ComputeTourUtilTools
+ calculateDistance(Intersection i1, Intersection i2): double
+ isValidPoint(Long point, HashMap<Long, Boolean> visited, HashMap<String, DeliveryRequest> requests): boolean
+ calculateSegmentDistance(List<Long> segmentPath, Map map): double
+ calculateTravelTime(double distance): Duration
+ findClosestPoint(Long currentPoint, List<Long> candidates, Map map, HashMap<Long, Boolean> visited, HashMap<String, DeliveryRequest> requests): Long
+ findClosestPoint(Long currentPoint, List<Long> candidates, Map map, HashMap<Long, Boolean> visited, HashMap<String, DeliveryRequest> requests, HashMap<Pair<Long, Long>, List<Long>> shortestPaths): Long
+ scheduleOptimizedDeliveryRequests(TourRequest tourRequest, Map map): List<Long>
+ scheduleOptimizedDeliveryRequests(TourRequest tourRequest, Map map, HashMap<Pair<Long, Long>, List<Long>> shortestPaths): List<Long>
+ filterMapByZone(Map originalMap, Intersection center, double radius): Map
+ computeShortestPathsFromSourceWithPaths(Long sourceId, Map map): HashMap<Long, PathResult>
+ computeAllShortestPathsWithPaths(Map map): HashMap<Pair<Long, Long>, List<Long>>
+ constructTourWithGeographicZones(List<Long> orderedPoints, Map map): Tour
+ constructTourWithSpecificShortestPaths(List<Long> orderedPoints, Map map): Tour
+ constructTourWithAllShortestPaths(List<Long> orderedPoints, Map map, HashMap<Pair<Long, Long>, List<Long>> shortestPaths): Tour
PathResult
+ distance: Double
+ path: List<Long>
UtilPair
+ intersectionId: Long
+ distance: Double
XmlDemandeParser
+ parse(file: File): TourRequest
<< interface >>
FileParser
+ parse(file: File): T
FileParserFactory
+ getParser(fileType: FileType): FileParser<?>
+ determineFileType(file: File): FileType
XmlMapParser
+ parse(file: File): HashMap<Long, Intersection>
TextFileParser
+ parse(file: File): String
<< enum >>
FileType
XMLMAP
XMLDEMANDE
TXT