Introduction to Software Engineering

Coursework 2

Nikoleta Kuneva s1643102
Ivan Karaslavov s1632798

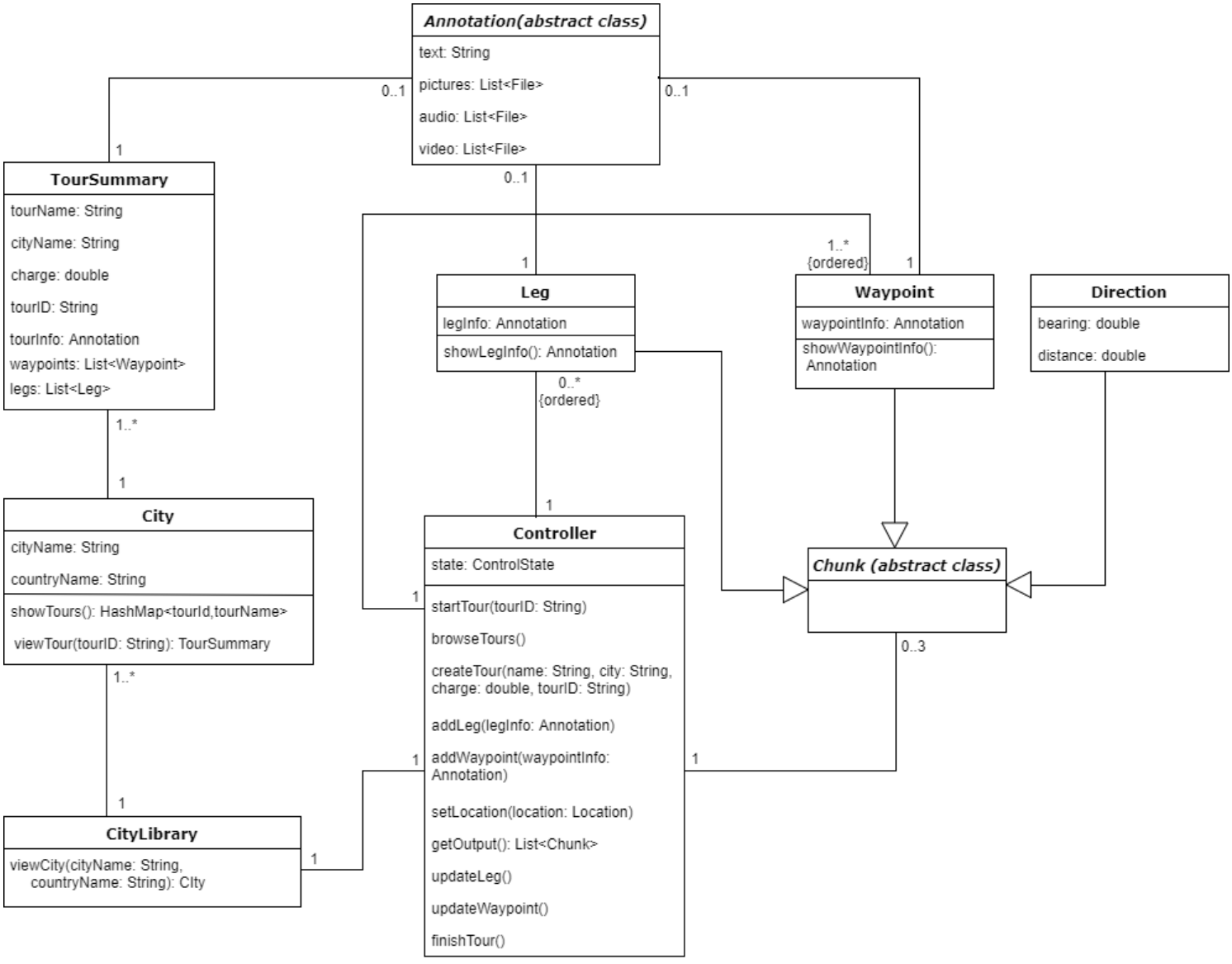
November 2017

1.Introduction

The app offers a variety of walking tours in different cities around the world. It presents information about the different attractions of each tour, highlighting features of interest. It contains three modes for browsing, creating and following a tour. This document includes information about the design of the aforementioned system. For more information visit: https://www.inf.ed.ac.uk/teaching/courses/inf2c-se/Coursework/2017/cw1.pdf and

https://www.inf.ed.ac.uk/teaching/courses/inf2c-se/Coursework/2017/cw2.pdf.

2.UML class model



3.High-level description

1. Description

The Controller class serves as the driving force behind our application. Its function is to take in messages coming from the outside world and pass them on to the internal components of the system. It contains methods related to the three main modes of the app – Browse Tours, Follow Tour, Author Tour, and thus enables the user to choose a tour from a tour library, to start or create a tour and to repeatedly add waypoints and legs. It also gives access to information regarding the chosen tour (text, pictures, video, audio). This is realized by the getOutput() method which presents chunks of data to the app user. The Controller class also automatically updates the current leg and waypoint and the tourist's location by receiving set location messages from a Location Service actor. These messages carry as arguments information about the current location.

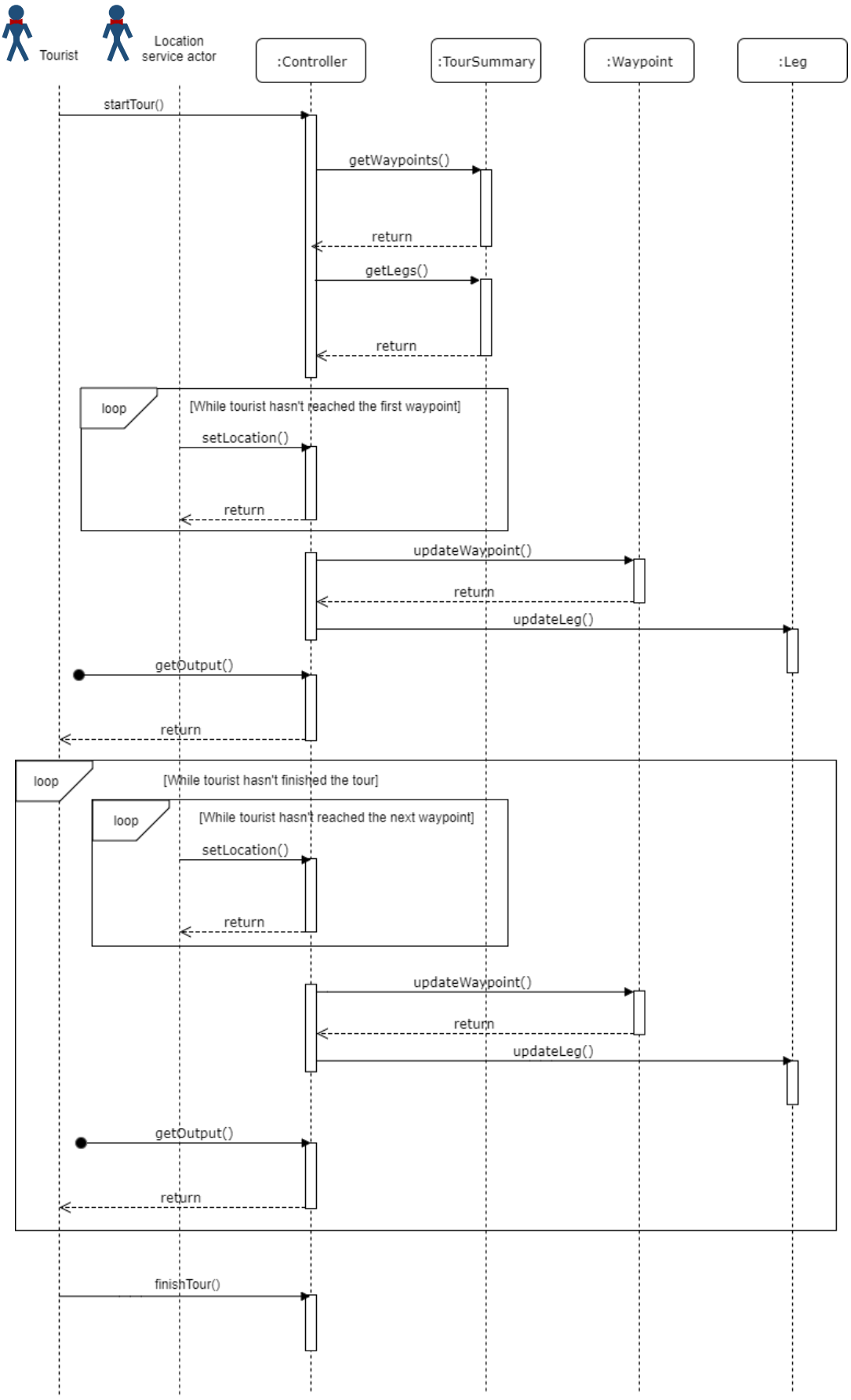
The CityLibrary class consists of a list of cities. Each city is accessed by the viewCity method which redirects the user to a list of tours available in the chosen city. For each tour in the list only its ID and name are displayed. After picking a tour, additional information about the selected tour appears. This information is held in the TourSummary class and includes the charge and the annotation. The annotation consists of text, pictures, audio and video files. Apart from the TourSummary, the Leg and Waypoint classes also contain Annotation type. The class Chunk is inherited by Leg, Waypoint and Direction.

When creating the UML class diagram, we considered the possibility of creating separate classes for each mode of the app. However, we chose to create a Controller class instead in which we put the main control functionality for each mode. Even though this leads to a lower cohesion, it improves encapsulation and reduces coupling. This makes the overall design simpler. Using the UML property text "{ordered}" we made sure that the waypoint and legs are kept in the same order as the destinations the tourist wants to visit. Using the good design principles, we named our methods and fields in a reasonable and recognizable way so there are no ambiguities.

2. Assumptions

1. There is at least one waypoint. A leg of the tour is a stage of the tour between two adjacent waypoints so if there is only one waypoint, there won't be a leg from our current point to the first waypoint (however, there might be navigation to the first waypoint).
2. The city library contains only cities with at least one tour. Cities are identified by their name and their country.
3. We assume that the getOutput() method can return a list of at most 3 Chunk objects (leg annotation, waypoint annotation and direction information). We show this constraint by making the multiplicity 0..3.

4. UML sequence diagram



5. Behaviour descriptions

1. Browse tours

aTourist - - browseTours() - -> theController

theController - - viewCity() - -> theCityLibrary

theCityLibrary - - showTours() - -> aCity

IF <tourist selects a tour>

aCity - - viewTour() - -> aTourSummary

IF <tourist decides to follow the tour>

aTourist - -> startTour() - -> theController (*the first step of the Follow Tour use case*)

ELSE

aCity < - - - - - - - aTourSummary

ENDIF

ELSE

theCityLibrary < - - - - - - - aCity

theController < - - - - - - - theCityLibrary

ENDIF

2. Author tour

aUser - - createTour() - -> theController

LOOP [while user continues to add waypoints]

aUser - - addWaypoint() - -> theController

ENDLOOP

LOOP [while user continues to add legs]

aUser - - addLeg() - -> theController

ENDLOOP