StormWare

Group 15
Anirudh Verma
David Hospital
Sijie Zhou
Zijing Chen

April 11, 2018

McMaster University

Department of Computing and Software

COMPSCI 2XB3

Contents

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals is the group is their exclusive work as a group and we consent to make available the application developed through CS-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purpose.

1 Team Members and Role Assignment

Team Members	**Student Number**	**Roles**
Anirudh Verma	400039737	Owner
David Hospital	400015029	Developer
Sijie Zhou	400038163	Quality Assurance
Zijing Chen	400020376	Designer

2 Contribution

Name	Role	Contribution	Comment
Anirudh Verma	Owner	Parser module	
David Hospital	Developer	Proposal	
David Hospital	Developer	Data and DisasterEvent module	
Sijie Zhou	Quality Assurance	Requirement spec	
Sijie Zhou	Quality Assurance	Presentation slide	
Sijie Zhou	Quality Assurance	Design spec	
Zijing Chen	Designer	MainActivity module	
All	Group 15	Design spec	The design spec is not finished by an individual. Its a group effort.

3 Revision History

Timestamp	Originator	Version Number	Comment
180209T1014	All	#1	Topic selected
180209T1014	All	#1	Team roles decided
180209T1014	All	#1	Setup github repo
180209T1014	All	#1	Implementation platform decided
180216T1107	All	#1	Create log file
180216T1107	All	#1	Specification decided: Using google map API and possibly machine learning for data parsing
180228T1405	All	#1	Completed requirements specification
180302T1100	All	#2	Start prototype
180302T1100	All	#2	Discussed prototype design and requirements
180305T0200	David Hospital	#2	Create Android Studio project and commit it to github repo
180307T1300	All	#3	Discussed relevance of machine learning and abandon machine learning
180309T1000	All	#3	Continued work on Parser.java and created Test.java.src/.../Parser.java
180316T1000	All	#3	Discussed project objective and how much data to display at once
180323T1000	All	#4	Final version confirmed and requirements modified

4 Executive Summary

The application is an Android mobile program using Google Map API which will help user to have an overall view of natural disasters in America. Users can choose one exact disaster type and the output will show by the form of heatmap.

5 Internal Evaluation

Overview: In this document we have presented a full report on the work undertaken to design and build the application in response to all the requirements of the stakeholders, and to the specification and plan provided in the brief.

The application is an Android mobile program using Google Map API which will help user to have an overall view of natural disasters in America. Users can choose one exact disaster type and the output will show by the form of heatmap. Problems faced and Improvements:

- Parser: The parser that we had implemented initially was creating performance issues (it took 24 seconds with our first parser) and we decided to implement a second parser which allowed for faster processing by 80%. It reduced the loading time to 8 seconds.
- Search Bar: For our interface, we decided to implement a search bar which allowed users to search locations and disaster events by type. Following that, the team felt that the user interface needed more functionality, for which we added a horizontal scroll view listing disaster types that can be directly selected to output the heat map, instead of typing the type. We decided to keep the search bar as it allows users to select locations on map as a normal search bar does.

Review Questions:

- Does it meet the design need or situation?

  Yes, the design needs for the project have been met.

- Does it fit the purpose for which it is intended?

  Yes, as per our initial planning, we have been able to make a project which covers the needs of our intended user base which includes (but is not limited to) governmental agencies for planning location sensitive things, researchers, weather networks and anyone who would be interested in knowing/working with the history of disaster events in a given area(in the US).

Concluding Remarks: The team felt that if time and skill set permitted, we would have gone ahead with integrating machine learning in our project to employ neural networks for processing patterns of disaster events that we are currently displaying in our app.

6 UML Class Diagram

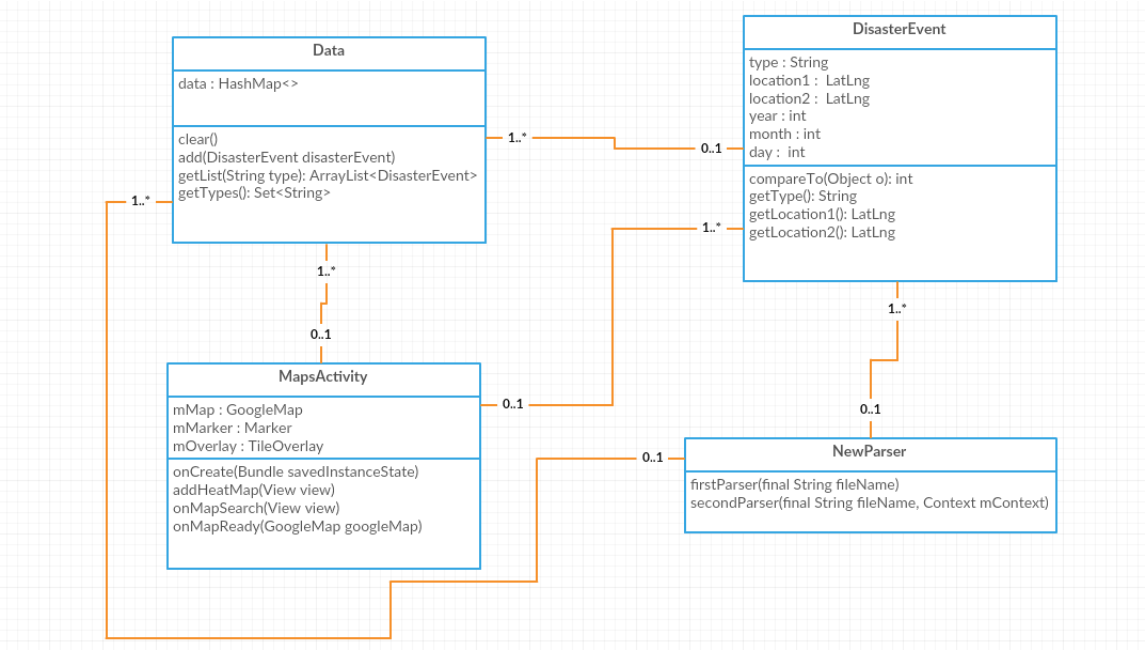

Figure 1: *DisasterEvent uses nothing. Data uses DisasterEvent. NewParser uses data and DisasterEvent. MapsActivity uses NewParser, Data and DisasterEvent.*

DisasterEvent:

- Module for creating DisasterEvent objects, used to store information about each event.
- Has start and end location, time (year, month, and day), and event type variables.

Data:

- Module for Storing and sorting a large collection of DisasterEvent objects.
- Uses a HashMap to partition events into different lists, separated by their type.

- Lists can be accessed by using the event type as a key in the HashMap.
- When a event is added and the HashMap does not contain a list for that type, it creates a new list and appends the event to it.

NewParser:

- Module for parsing data from specifically designed csv files directly into the Data module
- The method firstParser is responsible for condensing the raw data file with over 40 columns into a smaller file with just 8 columns.
- The method secondParser is responsible for parsing the condensed data file and creating a DisasterEvent object for each row. These events are added to the Data module using Data.add.
- The parsing time for the secondParser is 80% faster than firstParser, making it a valuable improvement to the module.

MapsActivity:

- Module for handling the controller and view components of the program.
- Uses the Android framework as a backbone for handling most of the input and output events.
- The onCreate method is called when the program starts and is responsible for loading the google map view onto the screen.
- The addHeatMap method is called whenever one of the type buttons is pressed (UI event). It removes the old heatmap object if there is one, and then adds a new one, by getting the list of DisasterEvent objects from the given type from the Data module. It then passes that list to the google map framework to create a heatmap and displays it on the map.
- The onMapSearch method is called whenever the user presses the search button (UI event). The text that is currently in the search bar is sent to the google map framework which returns a list of possible addresses that it might match. The first one is picked (best result). A marker is created (the old one is removed) on the map at that result and the camera is translated to focus on the new marker.

7 Disaster Event Module

7.1 Module

DisasterEvent

7.2 Uses

LatLng (android)

7.3 Syntax

7.3.1 Exported Types

DisasterEvent = ?

7.3.2 Exported Access Programs

Routine name	In	Out	Exceptions
DisasterEvent	String, LatLng, LatLng, $\mathbb{Z}$, $\mathbb{Z}$, $\mathbb{Z}$	DisasterEvent	
compareTo	DisasterEvent	$\mathbb{Z}$	
getType		String	
getLocation1		LatLng	
getLocation2		LatLng	
getYear		$\mathbb{Z}$	
getMonth		$\mathbb{Z}$	
getDay		$\mathbb{Z}$	

7.4 Semantics

7.4.1 State Variables

type: String
location1: LatLng
location2: LatLng
year: $\mathbb{Z}$
month: $\mathbb{Z}$
day: $\mathbb{Z}$

7.4.2 State Invariant

None

7.4.3 Assumptions

The constructor DisasterEvent is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.

7.4.4 Access Routine Semantics

DisasterEvent(t, l_1, l_2, y, m, d):

- transition: $type, location1, location2, year, month, day := t, l_1, l_2, y, m, d$
- output: $out := self$

compareTo($other$):

- output: $out := year < other.year \Rightarrow -1 | year > other.year \Rightarrow 1 | (month < other.month \Rightarrow -1 | month > other.month \Rightarrow 1 | (day < other.day \Rightarrow -1 | day > other.day \Rightarrow 1 | 0))$

getType():

- output: $out := type$

getLocation1():

- output: $out := location1$

getLocation2():

- output: $out := location2$

getYear():

- output: $out := year$

getMonth():

- output: $out := month$

getDay():

- output: $out := day$

8 Data Module

8.1 Module

Data

8.2 Uses

DisasterEvent

8.3 Syntax

8.3.1 Exported Types

None

8.3.2 Exported Access Programs

Routine name	In	Out	Exceptions
clear			
add	DisasterEvent		
getList	String	Sequence of DisasterEvent	
getTypes		Set of DisasterEvent	

8.4 Semantics

8.4.1 State Variables

data: HashMap of (String, Sequence of DisasterEvent)

8.4.2 State Invariant

None

8.4.3 Access Routine Semantics

clear():

- transition: $data$:=<>

add(de):

- transition: $data$.getList(de.getType) = $data$.getList(de.getType)$\| < de >$

getList(key):

- output: out := $data$.get(key)

getTypes():

- output: out := $data$.keySet

9 New Parser Module

9.1 Module

NewParser

9.2 Uses

Data, DisasterEvent

9.3 Syntax

9.3.1 Exported Types

None

9.3.2 Exported Access Programs

Routine name	In	Out	Exceptions
firstParser	String		
secondParser	String		

9.4 Semantics

9.4.1 State Variables

None

9.4.2 State Invariant

None

9.4.3 Access Routine Semantics

firstParser(s):

- Opens a file f with name s. For each row, let r be an array of strings, representing the columns defined in f. Let $year, month, day, type, lat1, lng1, lat2, lng2 :=$ r[0].substring(0, 4), r[0].substring(4, 6), r[1], r[12], r[44], r[45],r[46], r[47]. Open a second file f' with name "c" $\|s$. For each row in f, write to f' the line:
  $year\|month\|day\|type\|lat1\|lng1\|lat2\|lng2$

secondParser(s):

- Used to parse files created from firstParser
- Opens a file f with name s. For each row, let r be an array of strings, representing the columns defined in f. Let $year, month, day, type, lat1, lng1, lat2, lng2 :=$ r[0], r[1], r[2], r[3], r[4], r[5],r[6], r[7]. For each row in f,
  let $de :=$ DisasterEvent($year, month, day, type, lat1, lng1, lat2, lng2$). Data.add($de$)