MP Vote Analysis Project

NON-EXAM ASSESSMENT

Manas Rawat

Contents

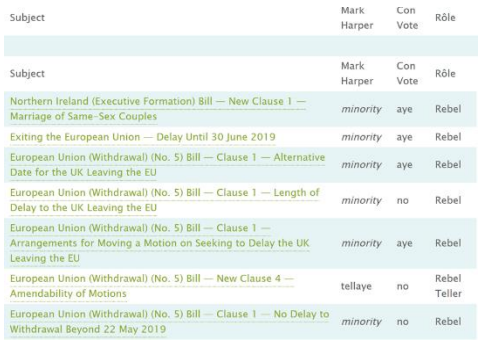
Analysis

Problem

It isn't always clear to constituents where their MPs stand on specific issues, how consistent (and thus principled) they are in how they vote, and how similar they vote to other MPs elected on the same manifesto as them. The difference between what these elected representatives say and actually do can be substantial.

Current System

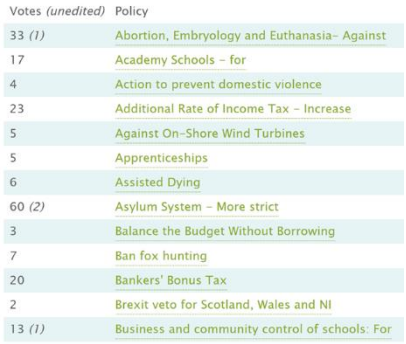
PublicWhip



Subject	Mark Harper	Con Vote	Rôle
Subject	Mark Harper	Con Vote	Rôle
Northern Ireland (Executive Formation) Bill — New Clause 1 — Marriage of Same-Sex Couples	*minority*	aye	Rebel
Exiting the European Union — Delay Until 30 June 2019	*minority*	aye	Rebel
European Union (Withdrawal) (No. 5) Bill — Clause 1 — Alternative Date for the UK Leaving the EU	*minority*	aye	Rebel
European Union (Withdrawal) (No. 5) Bill — Clause 1 — Length of Delay to the UK Leaving the EU	*minority*	no	Rebel
European Union (Withdrawal) (No. 5) Bill — Clause 1 — Arrangements for Moving a Motion on Seeking to Delay the UK Leaving the EU	*minority*	aye	Rebel
European Union (Withdrawal) (No. 5) Bill — New Clause 4 — Amendability of Motions	tellaye	no	Rebel Teller
European Union (Withdrawal) (No. 5) Bill — Clause 1 — No Delay to Withdrawal Beyond 22 May 2019	*minority*	no	Rebel

PublicWhip is a website that gets its data from Hansard, the official parliamentary record of divisions (MPs' votes). For each MP, it displays the raw division (vote) data. In the page of each MP, it shows 'interesting votes' – votes where the MPs voted differently from their party ("rebelled against the whip") or where they were assigned to count the votes. In 'free votes', where the MP's party had no policy and let their MPs vote however, they want, MPs are determined to be rebellious based on how they compare to how the majority of their party colleagues vote.

Below this, specific policies are shown (such as support for apprenticeships, support for academy schools, etc.) and next to those policies, the % of votes where the MPs voted in favour are shown. PublicWhip also has a policy-specific page, which link to pages that show the votes where the majority or minority of MPs supported said policy by either abstaining, voting for or against. Each individual vote's page contains a description of the motion and details the outcome, with a table showing MPs per party voted.



Votes *(unedited)*	Policy
33 *(1)*	Abortion, Embryology and Euthanasia– Against
17	Academy Schools – for
4	Action to prevent domestic violence
23	Additional Rate of Income Tax – Increase
5	Against On-Shore Wind Turbines
5	Apprenticeships
6	Assisted Dying
60 *(2)*	Asylum System – More strict
3	Balance the Budget Without Borrowing
7	Ban fox hunting
20	Bankers' Bonus Tax
2	Brexit veto for Scotland, Wales and NI
13 *(1)*	Business and community control of schools: For

PublicWhip calculates each MP's support for a policy using a set of 'weighting' criteria:

Voting	Points
Vote in favour in more important votes on the policy	50
Vote in favour in less important votes on the policy	10
Abstain in more important votes on the policy	25
Abstain in less important votes on the policy	2
Vote against in votes on the policy	0

Abstaining is given points because usually, rather than it coming about from MPs being unable to vote due to illness or other commitments etc, it comes from them either not having that strong of a view on the matter and supporting some parts of the policy-linked motion (be it legislation or non-binding) but not others, supporting the policy but not voting in favour due to party whip reasons (to dampen the rebellion and thus any disciplinary measures), or because they only support parts of it (e.g. the principle, but not the implementation).

The number of points accumulated by an MP in favour of a particular policy is then divided by the maximum total number of points attainable by that MP on the policy, and multiplied by 100, to calculate their % support. For example: if an MP votes has only voted thrice in favour of tuition fees, and two votes are less important while the other is more important, the maximum possible points obtainable by them on that issue is 10 x 2 + 50 = 70. If they voted in the way (with the majority or minority, in voting for/against/abstaining) that supports the policy in one less important vote but abstain on the second less important vote, they are at 10 + 2 = 12 points. If they vote in favour in the more important vote, they're now at 12 + 50 = 62 points. 62 / 70 * 100 = 89% support for policy.

PublicWhip also has JSON and XML files storing different types of data, on the types of ministers, files for a policy's votes (detailing which ways of voting – for/against/abstain – in specific votes equates to support for a particularly policy), as well as a large file that aggregates all the policy votes.

Positives

- PublicWhip goes into great detail when detailing the profiles of MPs, division votes and votes pertaining to policies.
- Frequently updates data, as it gets it all directly from Hansard
- It calculates and shows the MPs' support for policies
- It provides the data in different accessible formats – web form, XML and JSON

Negatives

- The data presentation and UI is not that user-friendly, especially to those with little knowledge on parliament
- Giving voting points to abstinence can be misleading, as MPs may have genuinely been unable to attend the vote e.g. maternity leave (prior to proxy voting introduction in 2019)
- The way by which they calculate an MP's % policy support, with different weightings for more and less important votes, is based on the developers' own assumptions

- Too many categories (policies) – 150 core policies as well as further provisional ones - makes it harder for users to gain a clear idea of an MP's overall political position
- No overall positioning of an MP politically
- No means of allowing the user to find their own positions and compare themselves to MPs

TheyWorkForYou

TheyWorkForYou is a website that gets its data, in turn, from PublicWhip. It isn't nearly as thorough; rather than going into a policy—orientated model as well, it stems first and foremost from MPs. It gets all the division data from PublicWhip for each MP, and then sorts the votes out into a maximum of (depending on how many votes the MP has participated in) the following categories (topics):

- Social Issues
- Foreign Policy and Defence
- Welfare and Benefits
- Taxation and Employment
- Business and the Economy
- Health
- Education
- Constitutional Reform
- Home Affairs
- Environmental Issues
- Transport
- Housing
- Miscellaneous

Each category is further divided into subcategories ("Equal gay rights", "Laws to promote equality and human rights", etc.) - policies. Instead of % support for policy being shown, instead, it is shown how many times the MPs voted in favour, how many time against, and how many times they abstained. The policy description on the MP's page, based on that information, states if the MP generally/consistently votes for/against or a mix for the policy. Within the subcategories, a summary of the votes (in date-descending order) is given, with less important votes initially collapsed.

Each individual vote's page gives a summary of the vote description, and visually shows how many MPs voted for/against/abstained.

Positives

- Much more user-friendly than PublicWhip in its design
- MPs' support for policies much clearer to users, as put into simple terms rather than percentage.
- Far fewer categories/more categorisation is easier to follow

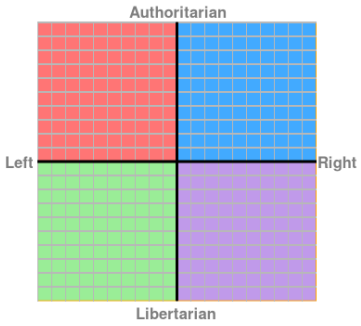
Negatives

- Still doesn't give overall picture of MP politically
- Data updates are slow, so you won't necessarily have the most up-to-date, accurate division data e.g. most recent votes, going as far back as a few months ago, aren't shown and thus aren't taken into account when determining MPs' positions on policies
- By not giving MP policy support in percentage, and using discrete descriptions instead, it can potentially group together MPs with very different view on the same policy
    - In addition, abstentions aren't counted as contributing any level of support, contrary to their usual use of indicating some level of support, thus painting the wrong picture.
- Categories are inconsistent and overlap in votes
- Also has no overall positioning of an MP politically
- Also has no means of allowing the user to pinpoint their own position and compare
- No transparency or explanation provided on how they describe and create policy categories of votes (e.g. "Laws to promote equality and human rights"), which can give rise to accusations of bias

All in all, neither of the two are really analysis-orientated nor do they provide a way for users to see how they compare with MPs in views; they're primarily focused on displaying data instead.

Background Research

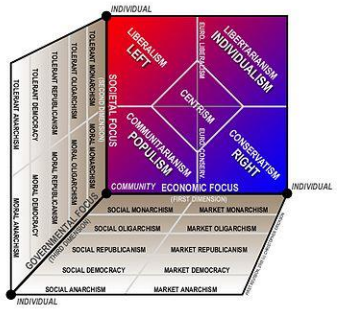
In order to make voting data more accessible, I knew that I'd require a means of aggregating this data. In order to do this, I researched various types of political compasses.



I started with the most well-known political compass: the 2D Model. This model is represented by a 4-quadrant graph, comprising of an economic (x) axis and a social (y) axis.

Benefits: simple, easy for any user to understand, most recognisable model, enough categories for the dataset

Drawbacks: some policies cannot be put into socio-economic categories. E.g. internationalism/secularism may have reasoning that can be split into left/right and authoritarian/libertarian categories, but they are their own distinct category.



The Vosem chart is a 3-axis political model. Alongside the social and economic axis, it adds a governmental axis. Though this greater categorisation may appear to be of greater use in pinpointing votes, it's actually counter-productive. This isn't a new category that increases precision, but rather something that can be folded into the social axis anyway. In addition, having more categories for the same set of data can actually reduce categoric precision; more categories necessitate more data to preserve precision.

There is also the simpler, left/right-only axis. People tend to classify themselves mostly on just this axis, and our politics is divided primarily down this line. However, this would be an inefficient means of categorising, as policies just can't be categorised down this sole lane. There is a significant difference between left-wing authoritarians (usually communists) and left-wing/centre-left libertarians (the more mainstream left-wing preached by the majority of the UK Labour Party and Liberal Democrats).

Having studied these various models, I came to the conclusion that the 2D-model is by far the best. These two axis can cover almost all electoral topics. This may be harder in the case of foreign policy, which will depend on patterns being found between the axis and various foreign policy decisions. E.g. support for foreign intervention correlates strongly with the right. However, there are exceptions that correlate neither with left/right exclusively but are split between e.g. EU membership.

Users

The client for this project will be a Year 13, while the end users will be anyone of voting age. The client is close to being an end user.

Interview

Using the existing websites (TheyWorkForYou and PublicWhip), would you say they give you a good idea of your MP in terms of their voting?

"No, not really. I'm not into politics, so I can't really tell"

What sort of features would make it easier?

"Some way of kind of getting their position based on how they've voted, and other MPs', as well a means of showing if they're like trustworthy"

Trustworthy?

"Um I don't know, um I guess I like mean consistent, so their votes line up"

Would you vote based on the MP or party they represent? Are their any features relating to that?

“Oh yea, definitely the party. Couldn’t care less about the one on the ballot. So yea, good idea to show if they’re voting the right way”

Objectives

- To pinpoint the political positions of MPs
    - To use votes on specific ‘topics’ to identify how much an MP agrees with a specific topic
    - To use all their votes, by summing and averaging for the socioeconomic axis, to find their overall political compass position (how left/right wing or/and authoritarian/libertarian the MP is)
- Show how consistent MPs are when voting
    - Show both overall and broken down into topics
- Compare their voting record to other MPs of their party (not by comparing to whip – how party wants them to vote, but by comparing raw voting records)
    - Contrast their votes on each policy with their party %
- Allow users to find their own political compass
    - Create a quiz that uses the same topics the MPs’ votes are categorised into
    - Use the users’ results to show how far off they were from the position they stated themselves to be in prior to the test
- See which MP(s) the user is closest to (both overall and broken down on specific issues)
- Ensure data is updated to make the analysis as accurate as possible

Limitations

- AI cannot factor in due to limited amount of data (votes) available, and so can’t predict how MPs may vote in future votes on any topic asked
- Only works with UK Members of Parliament
- Limited to 2 axes for categorisation, so not all votes (e.g. EU policy) can influence overall position

Platform and language

Client-side

Platform	Core language	Additional languages	IDE	Other technologies (libraries etc.)
Website	JavaScript/Python	HTML & CSS (UI) and SQL (DB)	IntelliJ IDEA or none (HTML & CSS files can be tested by running to	Django/Flask and JSON, possibly jQuery

			local machine)	
Android app	Java	XML (UI) and SQL (DB)	IntelliJ IDEA or Android Studio	JSON
Windows app	C#	SQL (DB)	JetBrains Rider or Visual Studio	JSON

Choosing

If the way of TheyWorkForYou and PublicWhip is followed in making a website, JavaScript would be necessitated for making the interactive user-quiz. Either Python or JavaScript can be used, but Python would be best given it is the primary language with Flash and Django. This website would be dynamic, not static. For elements such as the interactive quiz on the UI-side, HTML and CSS wouldn't be enough; JS, as well as potentially jQuery to the master the DOM, would be necessitated. However, with the requirement of both a server for the dynamic website (to function and store the Sqlite data created, which can't be stored offline) and the connection to the Heroku Dropbox files, as well as privacy concerns arising from data caching, given that I'm more experienced with app development, I discarded this option.

Using C#, a windows form or WPF (windows presentation form) app would be the base. An advantage of this is that is works particularly well with SQL, given their seamless integration in the .NET framework. This would also function faster than the website given the client and the option to cache the data both being offline. Android, meanwhile, would have the same benefits in terms of speed and an offline client app, as well as seamless integration with SQLite. However, between the two, Android is the better option. This is because, nowadays, people tend to use their phones more frequently, and so making the app on Android will be the more accessible option.

Solution

Java Android application that scrapes web data directly from the 86MB all-policies JSON file on PublicWhip (to ensure faster functionality, compared with connecting to multiple individual policy files). Due to the JSON file's large size, which Android cannot maintain a connection to, it will have to be broken down into 650 files (one for each MP). The Jackson JSON library will be used in order to read and create JSON. In order for these 650 files to be updated automatically like the live source file, the breakdown program will be run on Heroku, which will periodically re-run the program. For each vote of a policy: a vote for represents 1, a vote against represent 0, while an abstention represents 0.5. For each policy, all these vote valuations will then be averaged in order to show how much the MP agrees with said policy. Each of the 150 policy will be pre-determined in terms of their socioeconomic 'coordinate' (x-axis corresponds to economic axis; y-axis corresponds to social axis). All these vote-coordinates will then be added together and averaged to give the MPs' positions. The GUI is initially done in XML, but dynamically expanded on by Java. The Drobox API will be used to upload the 650 files post-breakdown, so that the app can access

them. As the source file was broken down into 650, at the same time the majority vote for each party was stored. This will be compared to the MPs' votes to show how close they vote with their party. A socioeconomic (social, economic) position will also be calculated for the parties, so that MPs can be compared to their parties' general position as well as the user's.

When the user opens the app for the first time, they will be asked to do a quiz. This quiz will be based on policies from the JSON files are sorted into being translated into questions, and similar methods to those used for MPs will be used to find the user's positions. The app will also show the user which MPs are closest to them, both socially and economically.

Presentation comparison

Public Whip

What it is	Source	Destination
MP name	Parliament API (via Hansard parliamentary record)	PublicWhip (HTML page) table row item
MP votes	Parliament API (via Hansard parliamentary record)	PublicWhip (HTML page) table row item

TheyWorkForYou

What it is	Source	Destination
MP name	PublicWhip (JSON file(s))	TheyWorkForYou HTML anchor/div tag
MP votes	PublicWhip (JSON file(s))	TheyWorkForYou HTML list item

Big Tent

What it is	Source	Destination
MP name	TheyWorkForYou*	TextView in cardView under recyclerView or textView in appbar toolbar
MP votes	PublicWhip policies.json	TextView in cardView under recyclerView

Hardware and software

Android Studio will be used to develop the application, while Heroku is used to host the breakdown program, which was developed using IntelliJ IDEA. This app can run on any Android device, and also on laptop and desktop environments via the Google Chrome ARC Welder extension or Bluestacks.

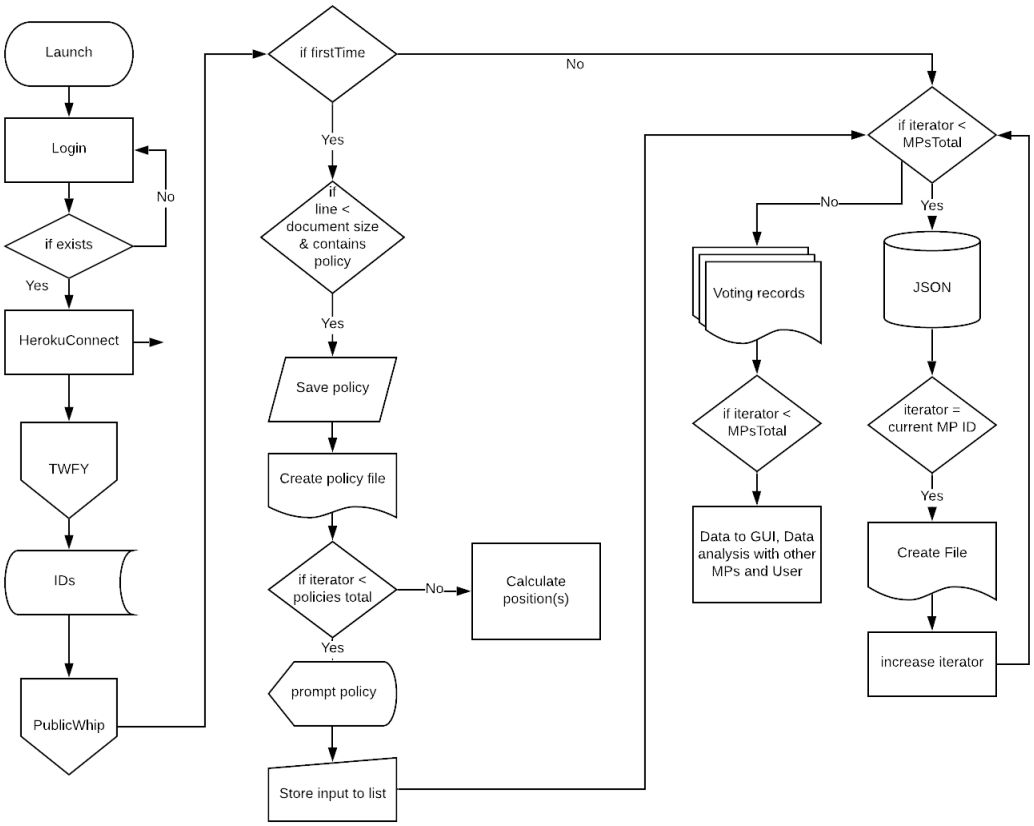
Technological requirements:

- Heroku cloud application platform: Android cannot establish a connection to the source file (too large); this necessitates the breakdown of this file into smaller, ‘profile’ files via JSON.
- Android Studio was initially used to develop the app. IntelliJ IDEA was not used because it has developed problems with rendering android apps in the test phase, However, upon switching to new hardware, IntelliJ became more proficient for usage.
- IntelliJ IDEA was however always used to create the breakdown algorithm for the JSON files
- An Android device of minimum API level 4.3 is required to run the said app
- Android SDK is of course required, being the framework on which such applications are developed
- JSON is used to store data from MPs, while Android SharedPreferences stores user data
- WiFi connection is a necessity for the file breakdown program to function; however, storage can be cached in the Android app, so as to save time, increase speed and reduce reliance on the existence of an internet connection.

IPSO

Input	Process	Storage	Output
Data from JSON file	Breakdown and format; aggregated to create graphical positions, per-party analysis and policy % agreement	Drobox	GUI
Dropbox connection code	Encrypted and decrypted via randomly generated key	SharedPreferences on Android	Allows in or rejected
User quiz selections	Aggregated to create graphical position and per-policy % agreement; compared with JSON data	SharedPreferences	Relayed onto GUI

Flowchart


Launch
Login
if exists
No
Yes
HerokuConnect
TWFY
IDs
PublicWhip
if firstTime
No
Yes
if line < document size & contains policy
Yes
Save policy
Create policy file
if iterator < policies total
No
Calculate position(s)
Yes
prompt policy
Store input to list
if iterator < MPsTotal
No
Yes
Voting records
if iterator < MPsTotal
Data to GUI, Data analysis with other MPs and User
JSON
iterator = current MP ID
Yes
Create File
increase iterator

Design

File structure

Android

- App: contains all the folders in the Android component
    - manifests/AndroidManifest.xml: contains metadata for the other files. Added permissions to enable the app to enable internet access and reading/storing file
    - Java folder contains all the Java files
        - Activity: start-up code. Manages GUI-upon-click functionality and dynamic management of the user interface. Links all the other Java files. Instead of using multiple activities, the same one (and the same RecyclerView) is reused in different states, so as to save space and prevent repetitive code.
        - HeapSort: sorts the data
        - Item: encapsulated class object, used to populate the RecyclerView with list items – be it the MPs or their votes, depending on which state the project activity is in
        - RecyclerAdapter and DataLoader: facilitate the functionality and calling of the RecyclerView GUI (in all its activity states)
        - DataRetriever: retrieves and sorts the voting data from DropBox
    - Resource file contains the XML files which statically define the user interface of the application. These are dynamically modified and given functionality in Java
        - Main activity XML provides the base of the app's core layout.
        - Member XML provides the base layout for each RecyclerView item (given same activity and RecyclerView are reused for different states, this isn't used for just members but also the different applications e.g. votes)
        - Sort spinner layout XML provides the base layout of the RecyclerVew-sorter, which is dynamically populated by Java
        The previous 3 were layout files. The following differ:
        - Menu xml file sets the core layout for the filterer (by party or policy)
        - Mipmap folder stores icon in different resolutions
        - Values folder contains files that define and store properties such as accent colours, theme and app title.

File Breakdown/Upload Program

- Main.java: the bulk of the code is contained in this file: establishes a connection to the online-hosted JSON source file, JSON-traverses it to break that down into 650 or however many there are and uploads them to Dropbox one-by-one
- Socioeconomic.java: object to facilitate positioning MPs

- A folder contains the external libraries: for the Dropbox Core and JSON API JAR files, which enable their utilisation in the application
- Text file holding policies and their socioeconomic values

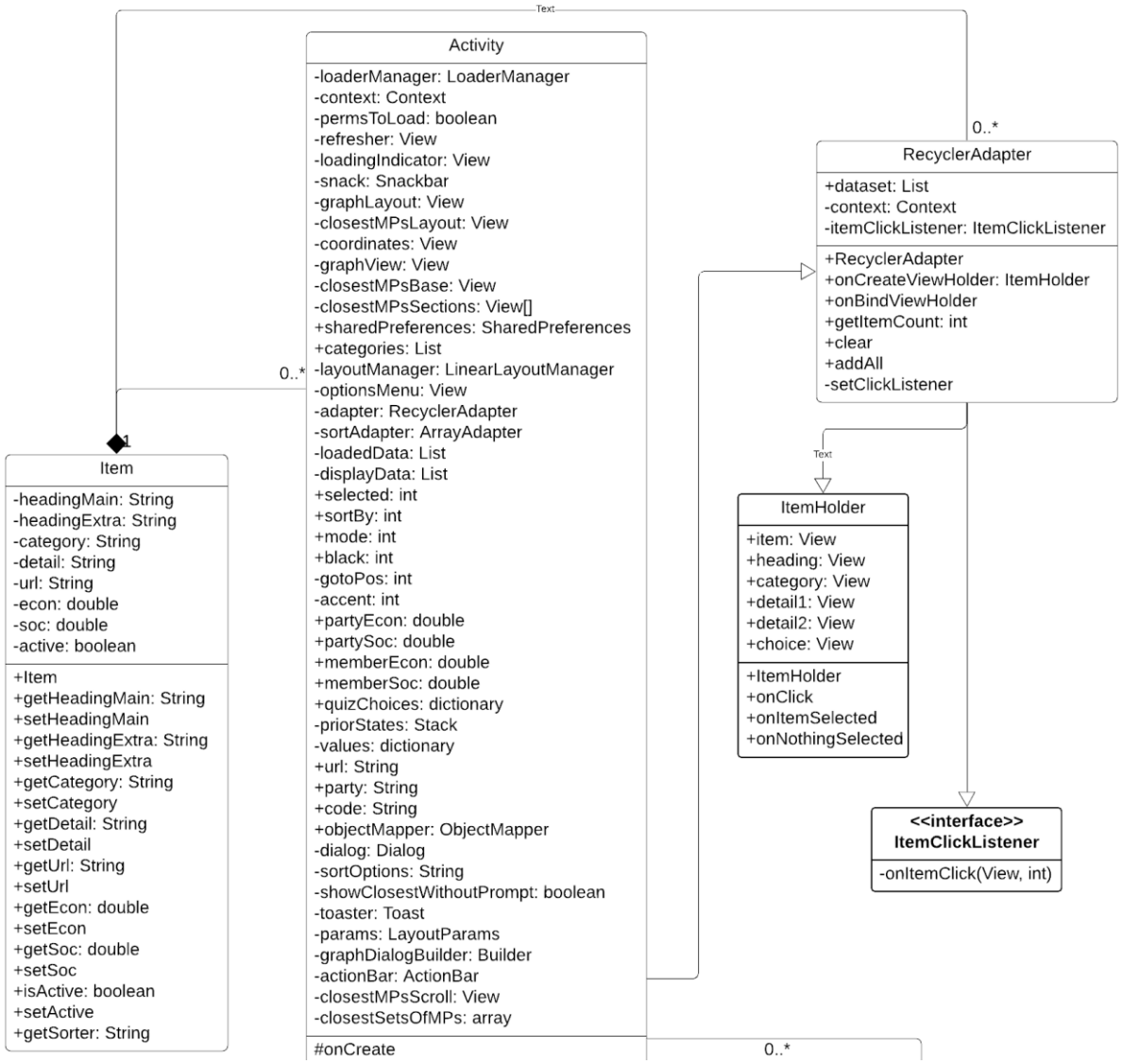
Data volumes

The online-hosted source JSON file is 86MB. The file breakdown program breaks this down into 650 MP-specific files + 1 party-specific file (shows what majority of each party's MPs support). The total size of all these files under a folder is 68.5MB. Range of 2 to 178 KB, with the party-policy one being 1.3MB. The Android app size is 4.1MB. The file breakdown program is 8KB.

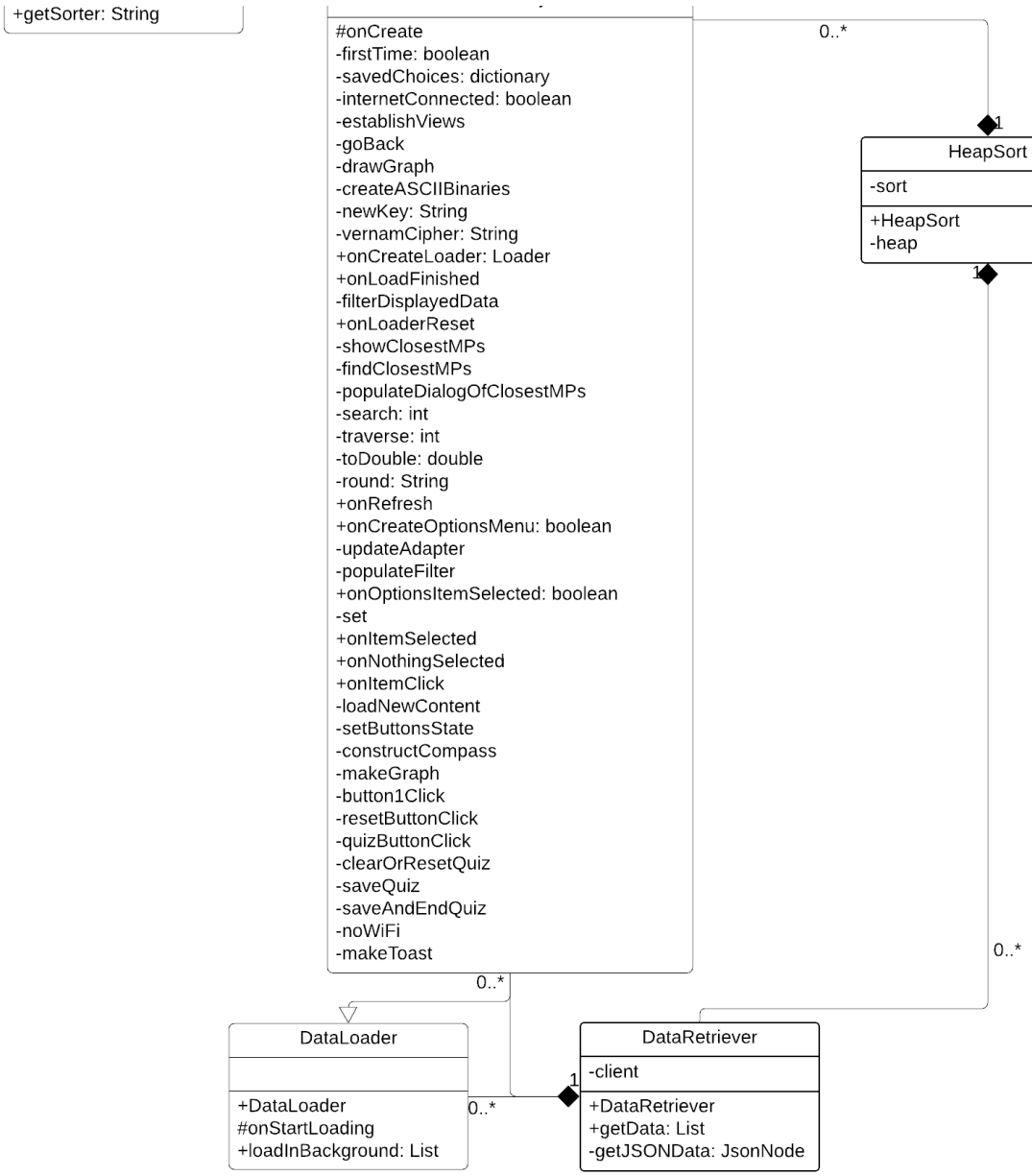
Class hierarchy diagram

The following is of the Android app component:

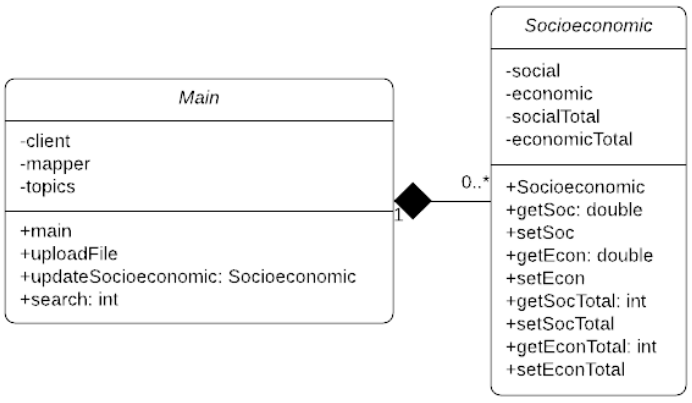
First half:



Second half:



JSON Breakdown and Creation program:



The following is the class hierarchy of the used, built-in Android classes:

java.lang.Object

↳ android.view.View

↳ android.view.ViewGroup

↳ androidx.recyclerview.widget.RecyclerView

↳ android.content.Context

↳ android.content.ContextWrapper

↳ android.view.ContextThemeWrapper

↳ android.app.Activity

Data dictionary

Field name	Field purpose	Field type	Field size	Example data	Validation (REGEX)
First name	Stores the first name of the MP	String	-	John	
Last name	Stores the last name of the MP	String	-	Smith	
ID	Extracted from TheyWorkForYou to identify the MPs on policies.json	String	5	45713	\\d{5}
Constituency	Stores the seat name of the MP	String	-	West Berka	
Party	Stores the party item of the MP	String	-	SNP	
AgreementPercent	Stores how much an MP agrees with a policy	String (created from StringBuilder to truncate integer)	If (second last character is 0): 4 Else: 6	56.7%	

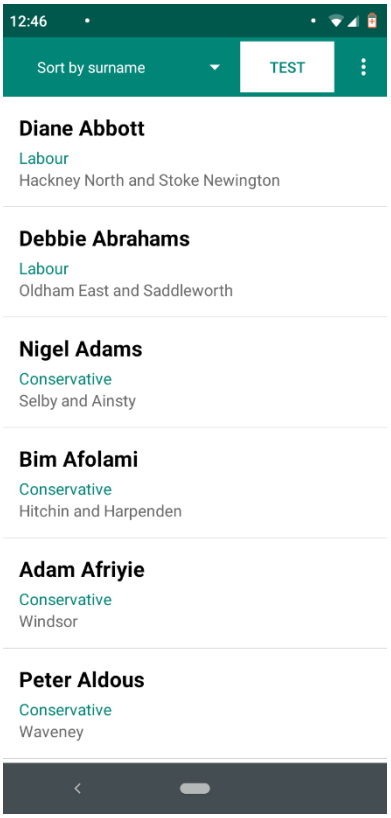
Field name	Field type	Start value	Description
Policy	String	NULL	This specifies the policy

SupportFor	Double (will remain integer though)	3	How many policy votes the user backs
Abstain	Double (multiple of 0.5)	1.5	How many policies the user it not sure on
PolicyVotes	Double (will remain integer)	5	How many policies there
OverallSupport	String (Double formatted to 3d.p. %)	37.4%	Policy percentage support

Classes explained

Prototype image shown

Android app



MainActivity: this class is the 'UI' class i.e. dynamically manages the design and user experience of the application. It is tied to the corresponding XML and handles functionality such as refreshing the displayed list.

Member: this encapsulated class serves as the object that holds the data required to be displayed (depending on the activity 'mode') on the scrollable, list "RecyclerView". I.e. if displaying the list of MPs, an instance of this class would hold an MP's name, party, seat and URL ID (to identify them in the JSON). If displaying the votes of an MP, it would hold the policy itself, how the MP voted on that in percentage figures, and the topic that policy comes under.

MemberAdapter: this class manages and sets the associated functionality of the RecyclerView. It sets methods that can be called to e.g. clear the list of data (Members) held by the RecyclerView, add Members, get the data count, etc. It also sets the values of specific graphical elements of each XML 'Member' item layout to the corresponding Java Member's values.

- MemberHolder class: composed inside, this class specifies the functionality for what happens if an item of the RecyclerView is clicked.

MemberLoader: runs, as a background task, the methods of QueryUtils that require an internet connection

QueryUtils: contains the methods that retrieve, format and sort the online data (JSON files as well as MP IDs from TheyWorkForYou via HTML web scraping).

Built-in classes used:

- List<Member> is used with encapsulation to manage the recyclerview
- JSONObject is used as a wrapper for the entre JSON document
- JSONArray is used for multiple JSON objects within the JSON document JSONObject

File breakdown and upload program

Single class program that establishes web connection to the online-hosted JSON file (after connecting to list of MPs on TheyWorkForYou and HTML-scraping their respective IDs to identify them in the file) and reads it via JSON library and breaks it down into 650 files, one for each MP. As this is done, files are both uploaded to Dropbox (at the same time a link is created for each) and in addition how each MP of their party voted per policy is counted, to create a party-policy-vote file, which is uploaded at the end.

Data Structures

Lists: feature frequently in the project, across both the Android app and the file breakdown program. Lists of different variable types, mainly of Strings and the custom Member class (to hold all the data associated with all the MPs e.g. name and constituency, to display on the GUI). JSONArrays, from the JSON library, is also used in the file breakdown program.

SharedPreferences

- Stores user-pin-pointing quiz answers
- Also stores user's calculated political position
- Also stores the Dropbox code ciphertext and key, which are re-generated each time the app is started (key randomly generated each time)

Key:

For social: +1 means authoritarian, -1 means libertarian

For economic: +1 means right-wing, -1 means left-wing

Policy	Social	Economic
Social issues		
Abortion, Embryology and Euthanasia- Against	+1	0
Action to prevent domestic violence	-1	0
Corporal punishment of children - Against	-1	0
Gambling - Against permissiveness	+1	-1
Homosexuality - Equal rights	-1	0
Human Rights and Equality	-1	0
Same Sex Marriage - for	-1	0

Teach children about drugs, sexuality and health	-1	0
Termination of pregnancy - against	+1	0
Transexuality - Against legal recognition	+1	0
Smoking ban - In favour	+1	-1
Foreign and defence		
Deployment of UK armed forces in Afghanistan	+1	0
Do more to help refugees inclding children	-1	-1
European Union - For	0	0
For the UK to Remain a Member of the EU	0	0
Free trade	0	+1
Hold a UK referendum on Lisbon EU Treaty	0	0
Iraq 2003 - For the invasion	+1	0
Iraq Investigation - Necessary	-1	0
Military Action against Daesh / ISIL	+1	0
Nuclear power - For	+1	+1
Preserve Environmental Protection on EU Withdrawal	0	0
Referendum on any EU withdrawal arrangements	0	0
Referendum on UK's EU membership -For -Pre 2016	0	0
Right for EU Citizens in the UK to Stay	-1	0
Stronger Military Covenant	0	-1
The UK should not ratify the Lisbon Treaty	0	0
Trident replacement - In favour	0	0
Use of UK Military Forces Overseas	+1	0
Welfare		
Excess Bedroom Benefit Reduction - Social Tenants	0	+1
Jobs Guarantee for Long Term Young Unemployed	0	-1
Localise Council Tax Support	0	+1
More Generous Benefits for Ill and Disabled	0	-1
Reduce Spending on Welfare Benefits	0	+1
Welfare benefits ought rise in line with prices	0	-1
Cap or Reduce Civil Service Redundancy Payments	0	+1
Pension auto-enrolment - For	0	-1
Prevent abuse of zero hours contracts	0	-1
Promote Occupational Pensions	0	-1

Woman's pension age increase - slow transition	0	-1
Assisted Dying	-1	0
Foundation hospitals - In favour	0	+1
GP Commissioning in the NHS	0	+1
Limit NHS Foundation Trust Private Patient Income	0	-1
More Emergency Service Workers	0	-1
More funds for social care	0	-1
Phase out of Tenancies for Life	+1	-1
Academy Schools - for	0	+1
Apprenticeships	0	0
Business and community control of schools: For	0	+1
End support for some 16-18 yr olds in education	0	+1
Replace Higher Education Grants with Loans	0	+1
Schools - Greater Autonomy	0	0
Tuition fees - Set Upper Limit at £9,000 per Year	0	+1
University education fees - Should be free	0	-1
University Tuition Fees - For	0	+1
The Economy		
Additional Rate of Income Tax - Increase	0	-1
Bankers' Bonus Tax	0	-1
Employee Shareholder Status	0	+1
Higher Pay for Public Sector Workers	0	-1
Higher taxes on alcoholic drinks	0	+1
Higher taxes on banks	0	-1
Higher taxes on sugary drinks	0	+1
Increase Air Passenger Duty	0	+1
Increase the income tax - tax free allowance	0	+1
Increase VAT	0	+1
Inheritance Tax	0	-1
Mansion Tax	0	-1
Member trustees on pension boards	0	-1
Minumum Wage	0	-1
Reduce capital gains tax	0	+1
Balance the Budget Without Borrowing	0	+1
Measures to reduce tax avoidance.	0	-1
Reduce the rate of Corporation Tax	0	+1
Require Pub Companies to Offer Rent Only Leases	0	-1
Tax Incentives for Companies Investing in Assets	0	+1

Post office - in favour of Government policy	0	+1
Post office closures - against	0	-1
Privatise Royal Mail	0	+1
Right to strike	-1	-1
Trade Union Regulation	+1	+1
Encourage and incentivise saving	0	+1
Extend Right to Buy to Housing Associations	0	+1
Make High Earners Pay Market Rent for Council Home	0	-1
Reduce taxes on domestic property transactions	0	+1
Regulate letting agent fees	0	-1
The Constitution		
Brexit veto for Scotland, Wales and NI	0	0
Delegate more powers to government ministers	+1	0
English Votes on English Laws etc.	0	0
Equal Number of Electors Per Constituency – for	0	0
Fixed Term Parliaments	0	0
Fully Elected House of Lords	+1	0
Further devolution to Northern Ireland	-1	0
Further devolution to Scotland	-1	0
Further devolution to Wales	-1	0
Make it easier to trigger a new election for an MP	0	0
More powers for local councils	-1	0
MPs decide if to approve a withdrawal agreeement	-1	0
No Polls Clash With MP Election System Referendum	0	0
Proportional Representation Voting System - For	0	0
Reduce central funding for local government	0	+1
Reducing the number of MPs - for	0	0
Referendum on Alternative Vote for MP Elections	0	0
Referendums for Directly Elected City Mayors	0	0
Remove Hereditary Peers from the House of Lords	-1	0
Restrict 3rd party campaigners during elections	+1	0
Retain funds from council house sales locally	0	0
Retention of Business Rates by Local Government	0	0

Role of MPs in the House of Commons - Strengthen	-1	0
Transparency of Parliament	-1	0
Voting age - Reduce to 16	-1	0
War - Parliamentary authority not necessary	0	0
Home affairs		
Asylum System - More strict	+1	0
Closed Material Proceedure	+1	0
Control Orders	+1	0
Freedom of Information Bill 2000 - Strengthen	-1	0
Identity cards - For introduction	+1	0
In Favour of Mass Surveillance	+1	0
Incentivise membership of press regulator	+1	0
Labour's Terrorism laws - For	+1	0
Mass Retention of Communications Data	+1	0
Merge Police and Fire under Police & Crime Cmmr	0	0
Ministers Can Intervene in Coroners' Inquests	+1	0
No detention without charge or trial	-1	0
Openness and Transparency - In Favour	-1	0
Police and Crime Commissioners	0	0
Protesting near Parliament - Unrestricted	-1	0
Recreational drugs - Against legalization	+1	0
Register of Lobbyists	+1	0
Restrict Scope of Legal Aid	0	+1
Tougher on illegal immigration	+1	0
Environment and transport		
Against On-Shore Wind Turbines	0	+1
Civil aviation pollution - For limiting	0	-1
Cull Badgers	0	+1
Energy Prices - More Affordable	0	0
Ban fox hunting	+1	0
Fox hunting - Ban	+1	0
HS2 - In Favour	0	0
Incentivise Low Carbon Electricity Generation	0	-1
Regulation of Shale Gas Extraction	0	-1
Sell England's Public Forests	0	+1
Stop climate change	0	-1
Lower taxes on petrol & diesel for motor vehicles	0	-1
Crossrail - In favour	0	0

Heathrow Third Runway - In Favour	0	0
Public Ownership of Railways	0	-1
Rail Fares - Lower	0	0
State control of bus services	0	+1
Misc		
Coalition Programme for Government - For	0	0
Decamp from Palace of Westminister During Works	0	0

Determination

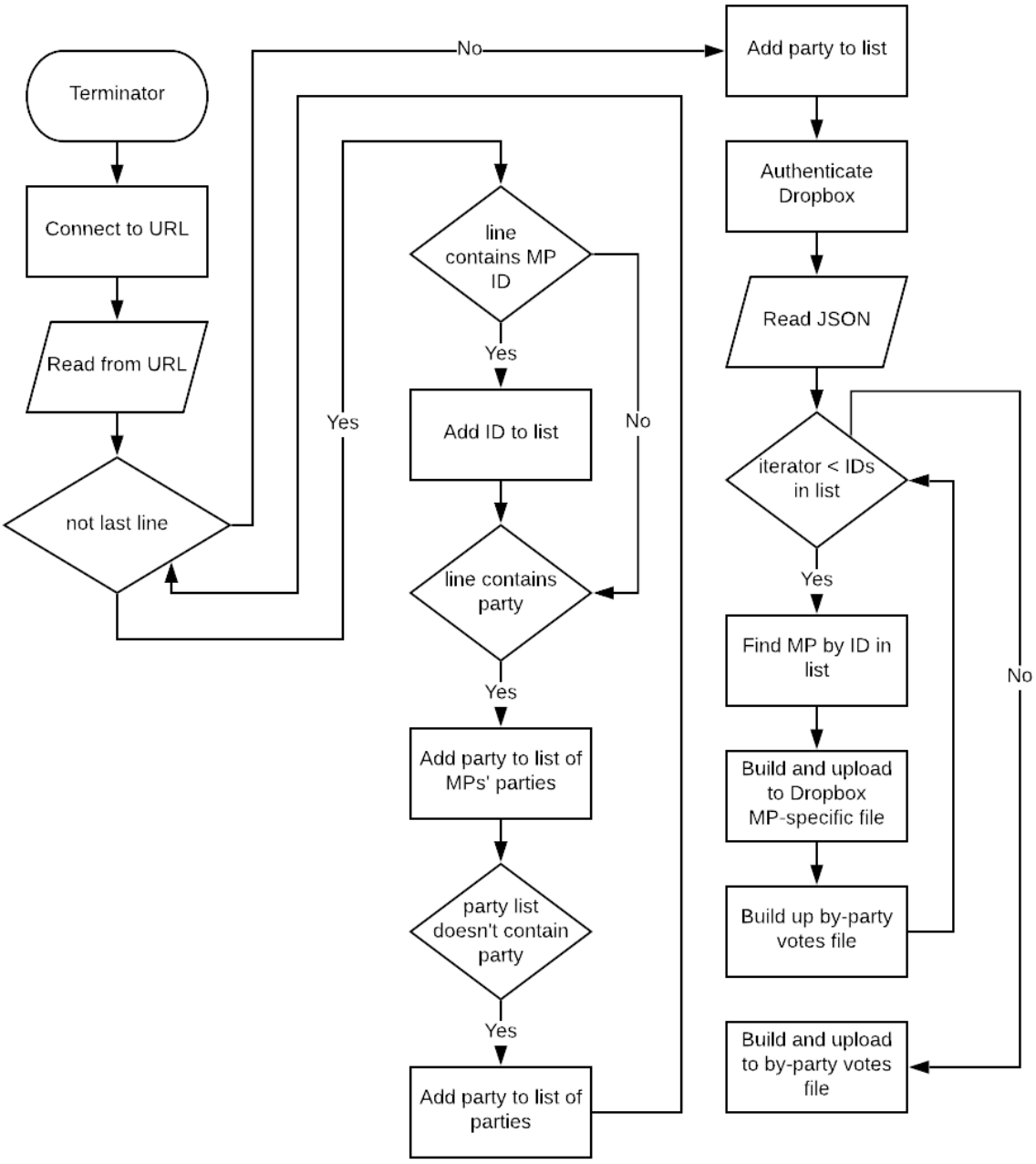
In terms of categorising the various policies into topics, I looked at how TheyWorkForYou did it. This is helpful due to how TheyWorkForYou gets its vote data from PublicWhip as well, and so their topics' policies' pages link directly to PublicWhip. As for the socioeconomic valuation of each policy, these adhered to strict and universally accepted principles.

For economic:

- Measures increasing regressive (indirect) taxation are +1 (right-leaning), while measures increasing proportional (direct) taxation are -1 (left-leaning). Voting against
- Measures increasing state intervention (including welfare state) and maintaining/creating public services are -1, while measures that involve privatisation and reducing state intervention/welfare state are +1.

Important Algorithms

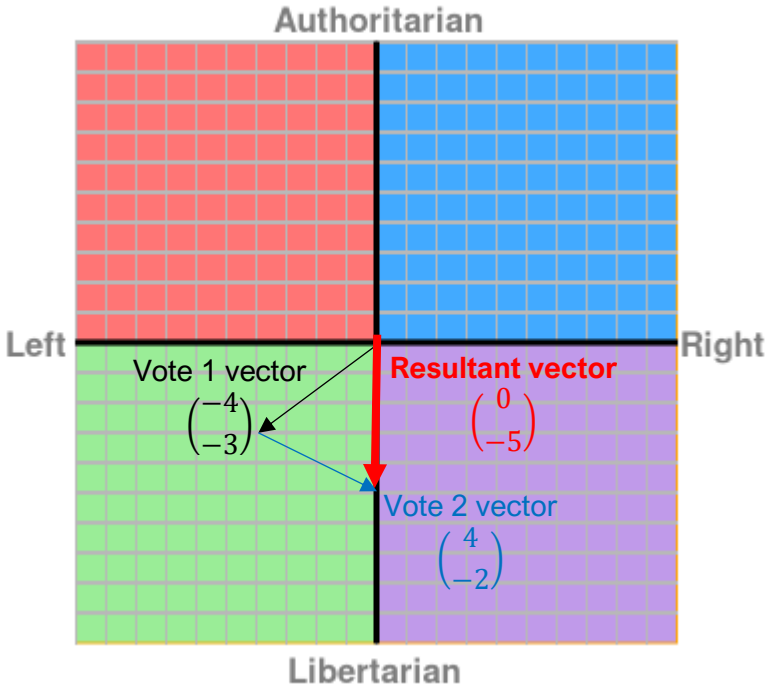
File breakdown program overview



MP-specific JSON file compilation

Note: FOR loops' counter's start value is inclusive, end value is exclusive

```
allTheVotes <- JSONObject[]
parties <- labels of grouped (party) or individual MPs
MPs <- all current members of parliament, grouped by party
FOREACH party IN parties
     partysMPs <- MPs.get(party)
     FOR count <- 0 TO partysMPs.size:
          ID <- partyMPs[count]
          memberObj <- JSONObject
          FOR count2 <- 0 TO JSON.length:
               level <- 0
               votesIn <-- 0
               policy <- JSON[count2].title
               topic <- getTopic(policy)
               aspects[] <- JSON[count2].aspects
               FOR count3 <- 0 TO aspects.length:
                    item[] <- JSONObject[]
                    motion[] <- aspects[count3].motion
                    policy_vote <- motion.policy_vote
                    motionId <- motion.id
                    ...
                    votes[] <- motion.vote_events.votes
                    FOR count4 <- 0 TO votes.length:
                         IF votes[count4].id = ID THEN
                              IF memberObj doesn't contain topic
                              THEN memberObj.put(topic, JSONObject)
                              ENDIF
                              votesIn <- votesIn + 1
                              option <-votes[count4].option
                              IF option = policy_vote THEN
                                   level <- level + 1
                              ELSEIF option = "absent" THEN
                                   level <- level + 0.5
                              ENDIF
                          #Party-specific generalisation calculation
                              BREAK LOOP
                         ENDIF
                    ENDFOR
               ENDFOR
               IF votesIn != 0 THEN
                    percent = level / votesIn * 100
                    memberObj.get(topic).put(policy, percent)
                    #MP policy support calculation algorithm
                    #(with some variables declared beforehand)
               ENDIF
          ENDFOR
          #Calculate socioeconomic vectors (add to memberObj)
          file <- createFile(ID, memberObj)
          Dropbox.upload(file)
     ENDFOR
ENDFOREACH
```



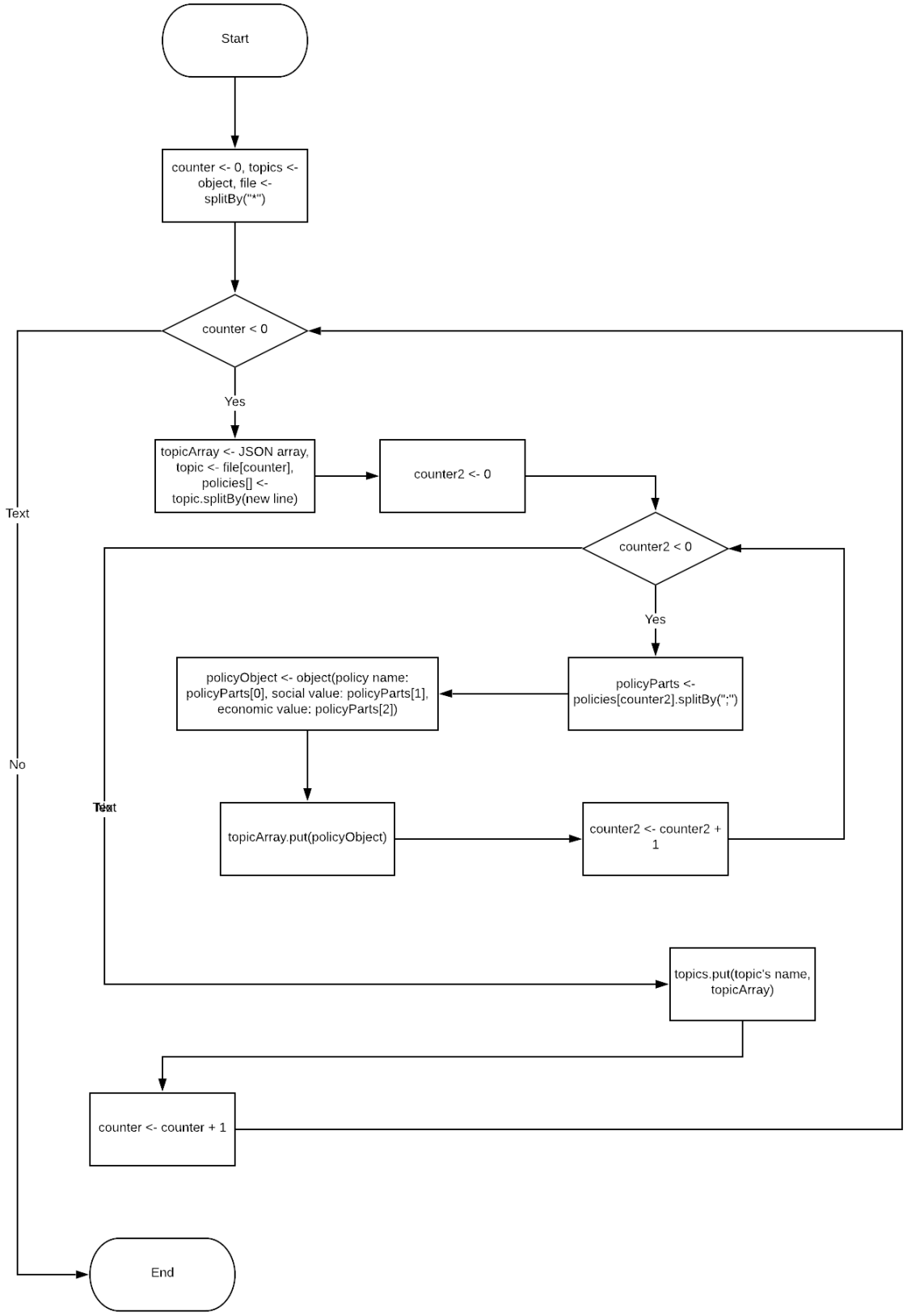
In same quadrant and x-difference of 4 means ~consistent in x-axis measure (will have numerical consistency measure)

A 2D, graphical coordinated-based algorithm pin-points the position of MPs on the political compass from their voting record. The axis will be left/right (x-axis) and libertarian/authoritarian (y-axis). Thus, from each recorded vote, a vector movement (x/y) will be made on their personal political compass, depending on the political nature of their cast vote.

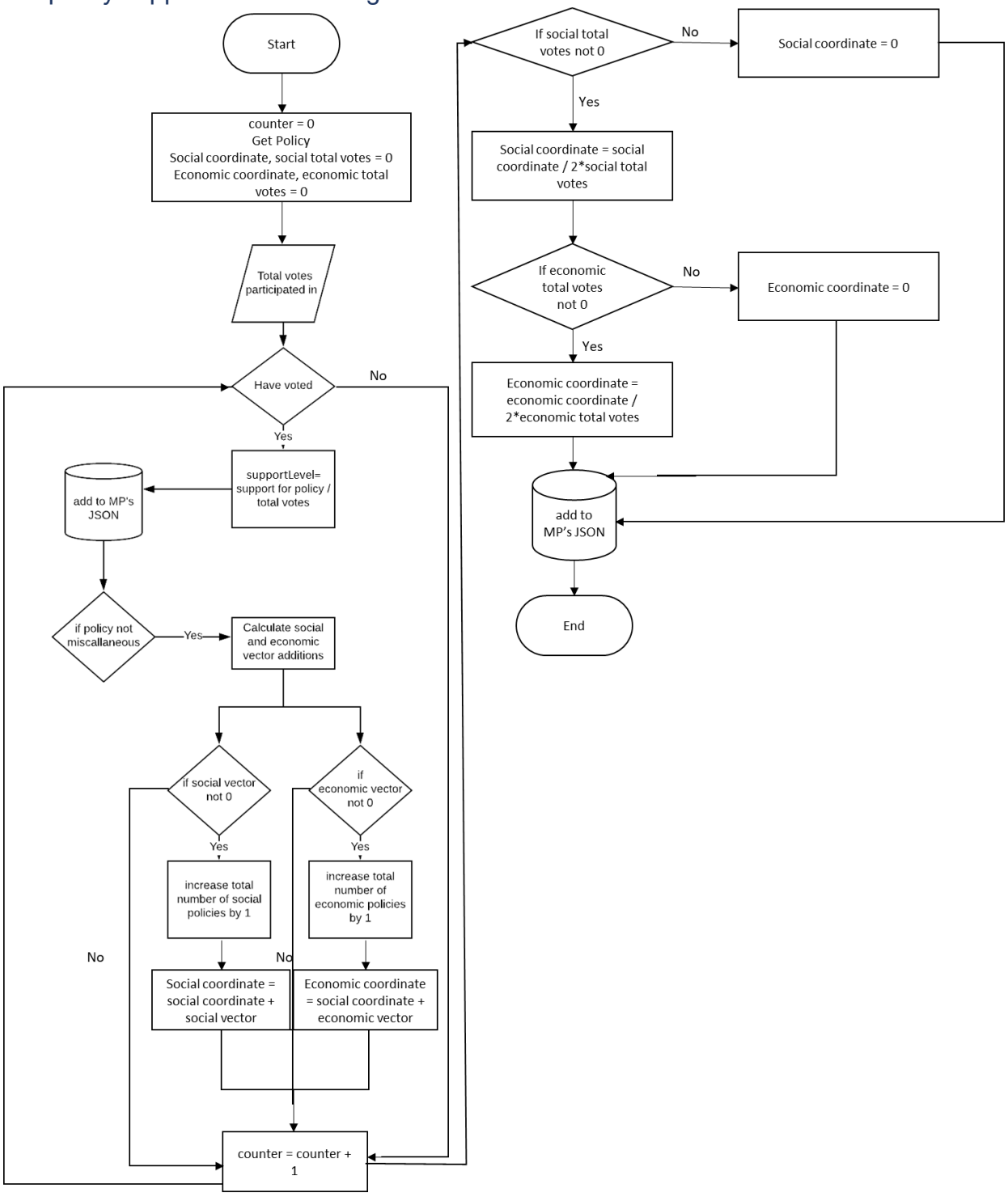
- As each MP's JSON file is constructed, their votes are added to a JSON 'Array' for their party. This builds up a file which shows how many of each party's MPs voted on each vote. This file can be retrieved to compare with the specific MP.

Get policies' economic and social values

In order to store the various topics' various policies and their respective social and economic value as JSON objects and arrays, for programmatic use:



MP policy support calculation algorithm



Calculate socioeconomic vectors

```
topic <- listOfTopics.get(topicIndex)
policySocialValue <- topic.get(BinarySearch(topic,
policyToFind, 0, topic.size - 1))
policySocialVector = (supportLevel * policySocialValue - (1 -
supportLevel) * policySocialValue) * 10
```

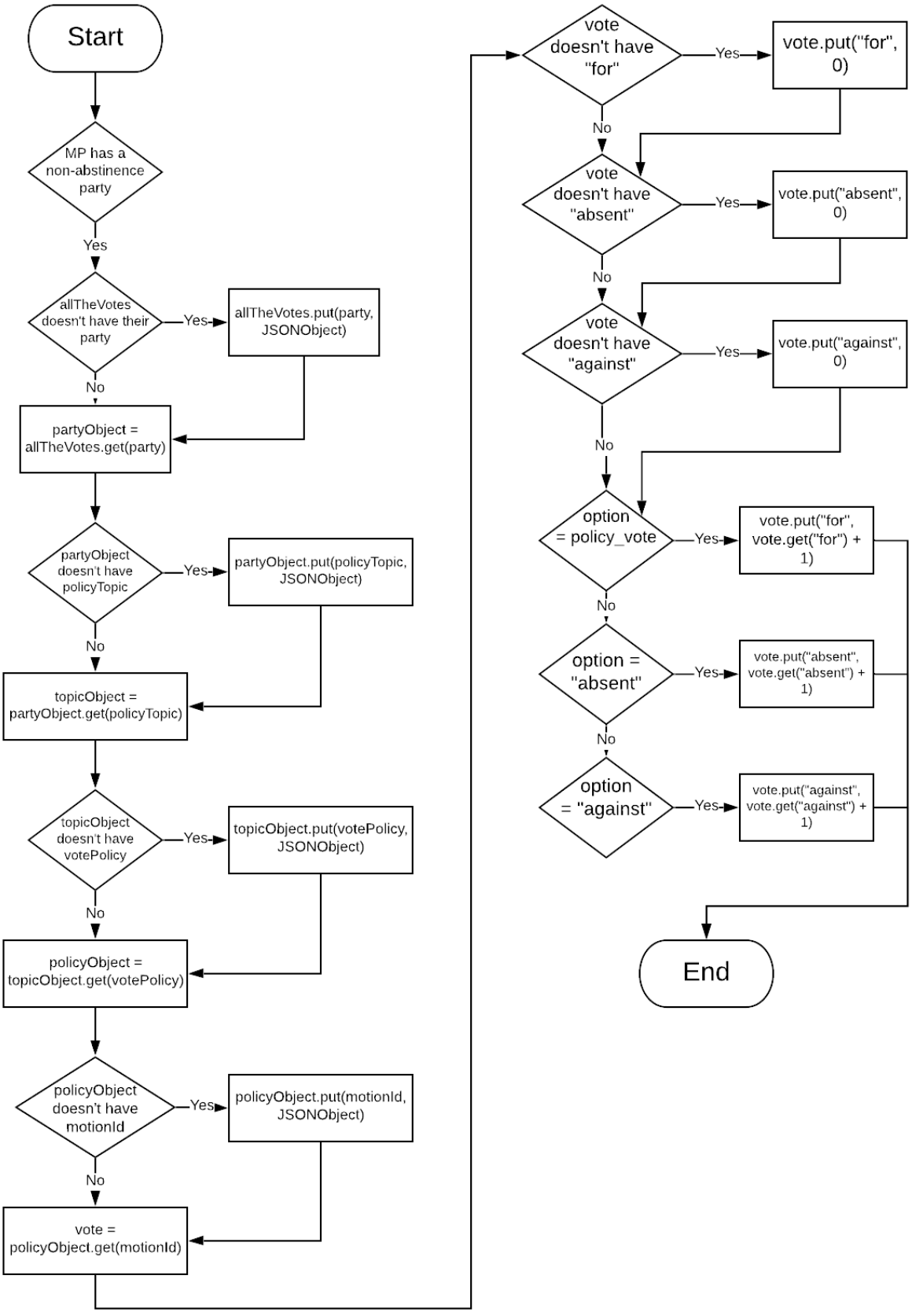
Same method for economic (`policyEconomicValue` and `policyEconomicVector`)

This can be simplified down to equalling: (supportLevel – (1 - supportLevel)) * policySocialValue * 10 = (2 * supportLevel - 1) * policySocialValue * 10

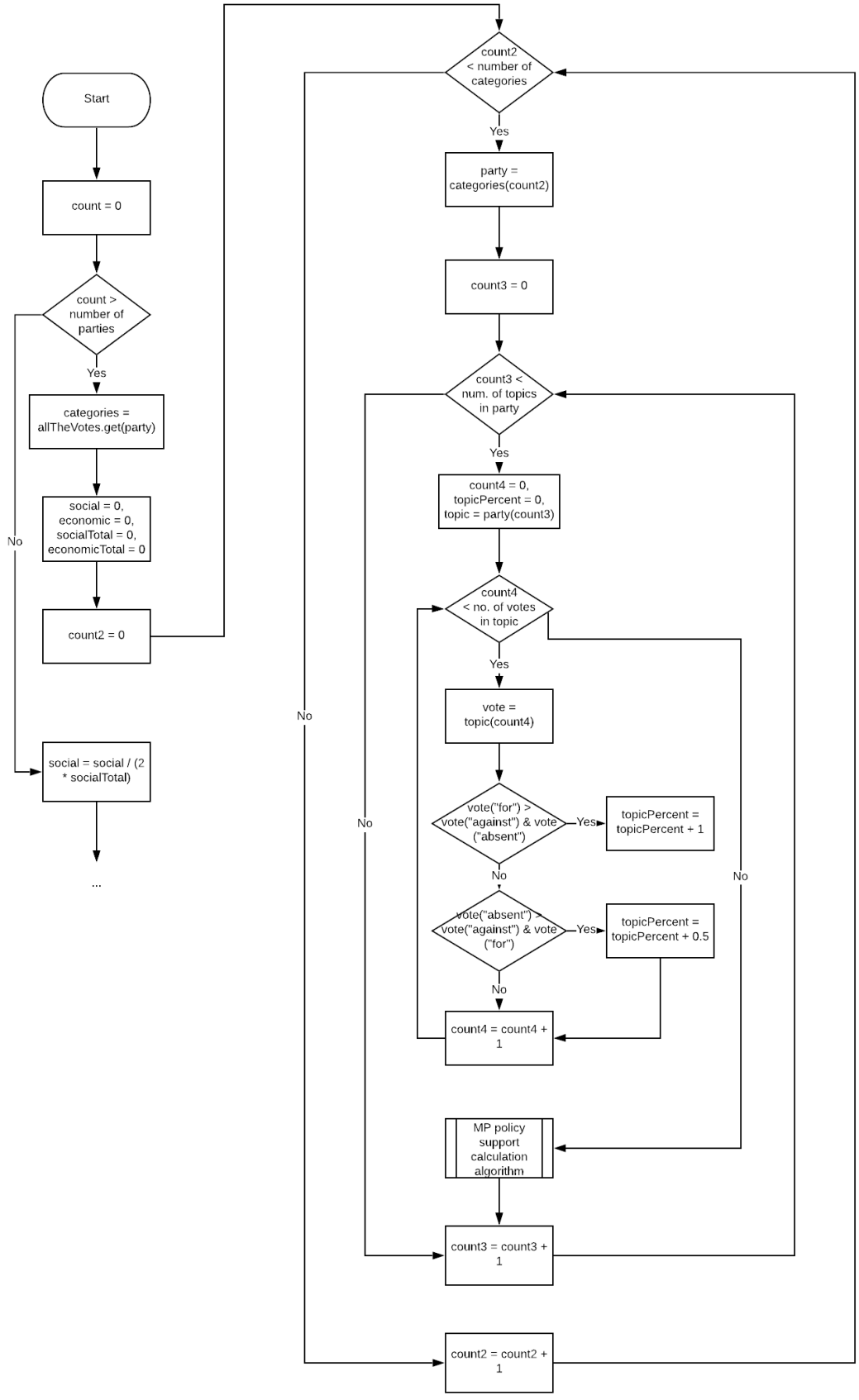
Why?

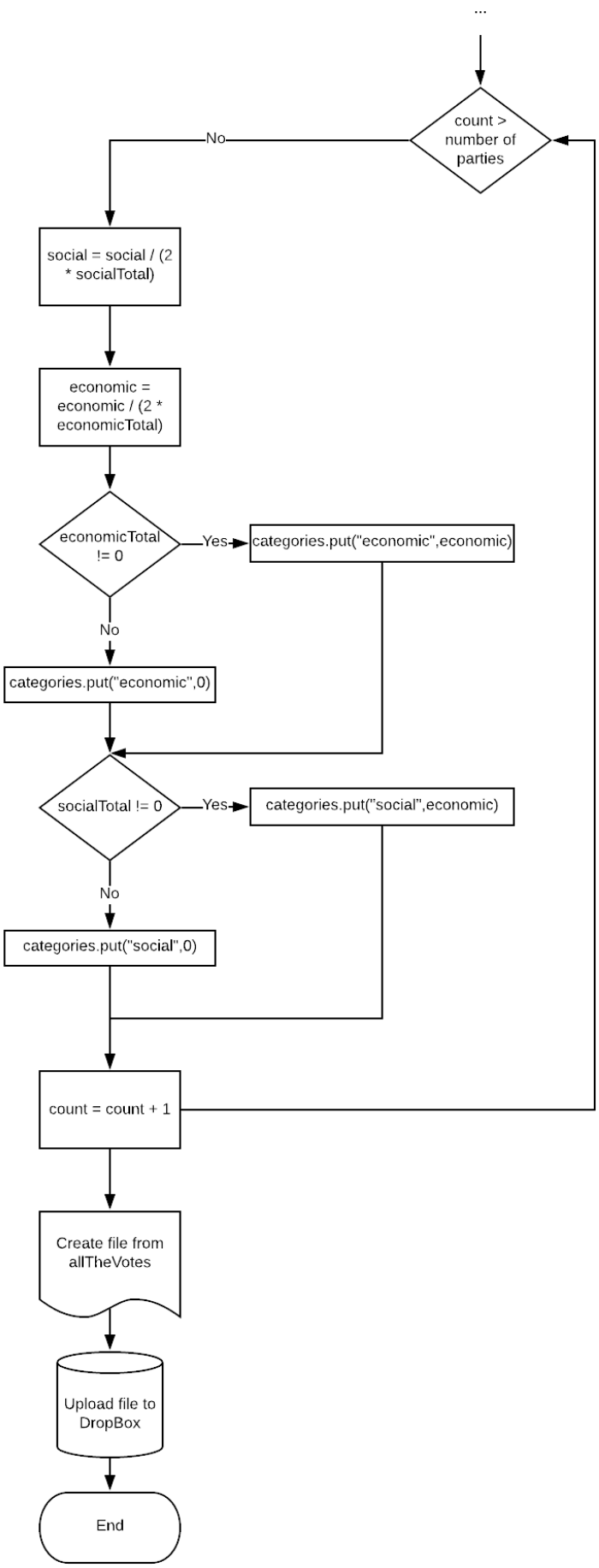
This is in effect how much they've supported the policy subtracted by how much they haven't – giving the net support level, and making the calculation effectively work in the desired vector movements. This is why, in the end, the overall, accumulated social and economic values of the MP is divided by twice the count of policies they've voted on, rather than just the count itself. If this subtraction isn't carried out, then this would inflate the MPs' pinpointed coordinates and be less accurate. In addition, the reason why the net policy support level (why is effectively net % support divided by 10) of the policy as a whole is averaged, rather than all the individual policy votes (which practically discards the policies when working this out) is to reduce the impact of outliers.

Party-specific generalisation calculation



By-party file build-up




...
count > number of parties
No
social = social / (2 * socialTotal)
economic = economic / (2 * economicTotal)
economicTotal != 0
Yes
categories.put("economic",economic)
No
categories.put("economic",0)
socialTotal != 0
Yes
categories.put("social",economic)
No
categories.put("social",0)
count = count + 1
Create file from allTheVotes
Upload file to DropBox
End

Heap Sort

This sort algorithm will be used for sorting the data (be it the list of objects that hold the details of MPs, like their name, party, seat, etc; or objects that hold MPs' voting record, by category and policy). Has time complexity of Onlogn in all 3 cases (average, worst, best) – like mergesort, making it fast and consistent. Quick sort has the same time complexity in average and best cases, but it goes down to On^2 in worst case scenarios. Upon assessing, heap sort is the superior option for the purposes of sorting the 650 MPs in various manners (alphabetically by first name, last name, party, seat, numerically by economic and social positions – in multiple, stacked criteria e.g. if same last name, then check first name etc). This is due to its use of both recursion and tree traversal.

Sort options for MP list:

First name	Last name	Party	Constituency	Economic	Social

Tree diagram:

Exemplar snapshot data:

To sort by last name, the criteria will be to compare last name **then** first name **then** seat (no need to go further as the seats are guaranteed to be distinct), making the following strings:

0. CorbynJeremyIslington North
1. JenkinBernardHarwich
2. DaveyEdKingston
3. JenkinPeterDundee
4. KeeganGillianChichester

For simplicities sake to make representation easier, we can discard the constituencies in this case, as none of them have the same name.

0
Jeremy
Corbyn
Islington North
Labour
-3.5
-2.3

1
Bernard
Jenkin
Harwich
Conservative
1.5
0.2

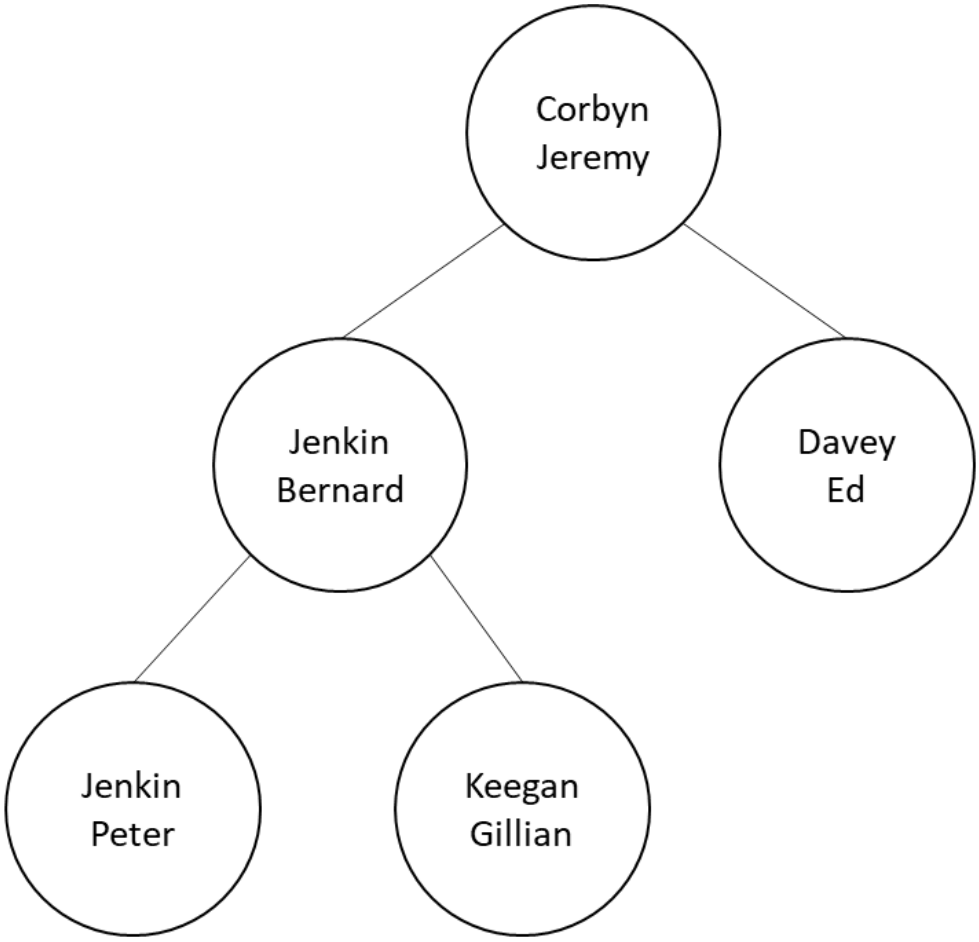
2
Ed
Davey
Kingston
Liberal Democrat
-1.3
-2.4

3
Peter
Jenkin
Dundee
SNP
-0.1
-1.3

4
Gillian
Keegan
Chichester
Conservative
0.3
-1.3

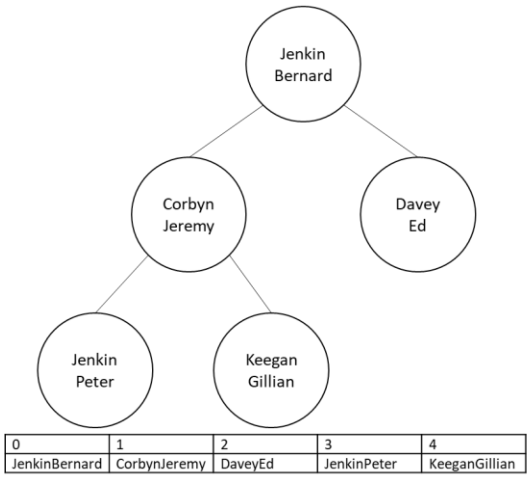
This recursive process encompasses, after the initialisation of a heap based on the initial ordering of the list, the generation of successively smaller max heaps (where parent node always greater than or equal to child nodes), until only one node remains.

Initial heap:

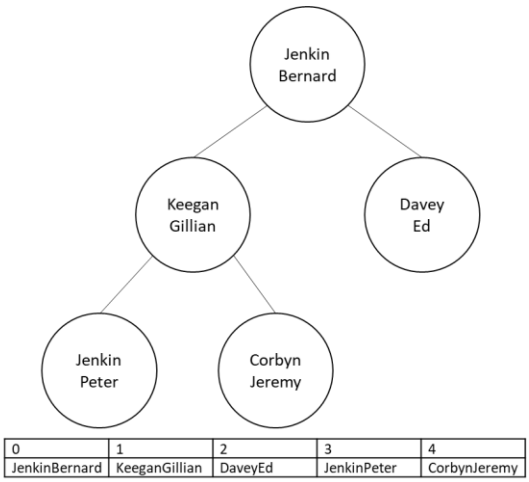


0	1	2	3	4
CorbynJeremy	JenkinBernard	DaveyEd	JenkinPeter	KeeganGillian

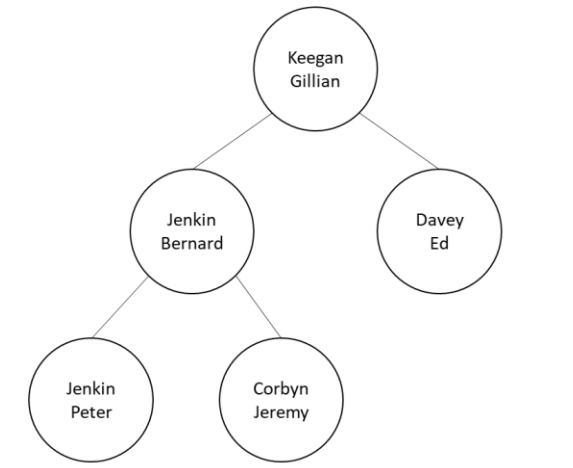
Process 1: Heap turned into max heap by swapping nodes:



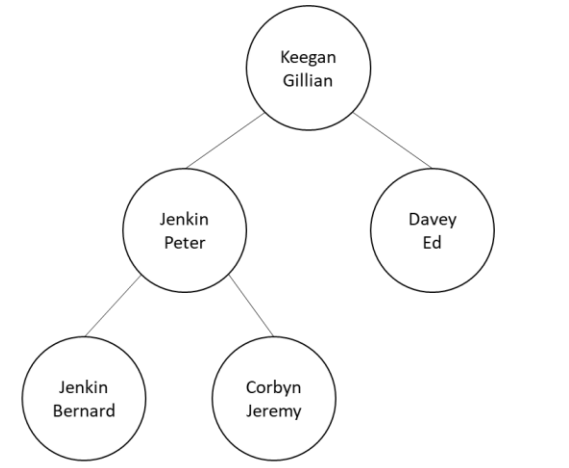
0	1	2	3	4
JenkinBernard	CorbynJeremy	DaveyEd	JenkinPeter	KeeganGillian



0	1	2	3	4
JenkinBernard	KeeganGillian	DaveyEd	JenkinPeter	CorbynJeremy

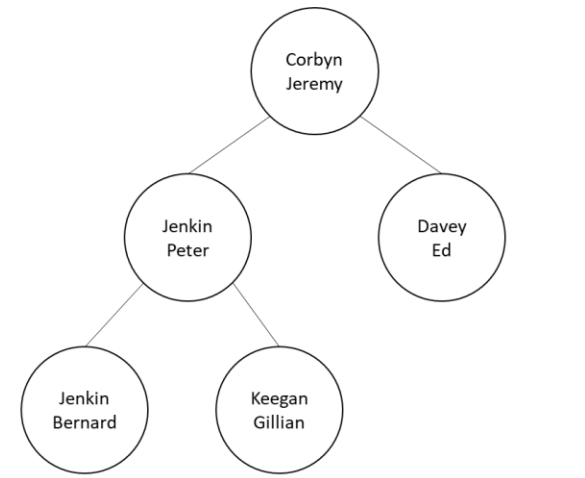


0	1	2	3	4
KeeganGillian	JenkinBernard	DaveyEd	JenkinPeter	CorbynJeremy

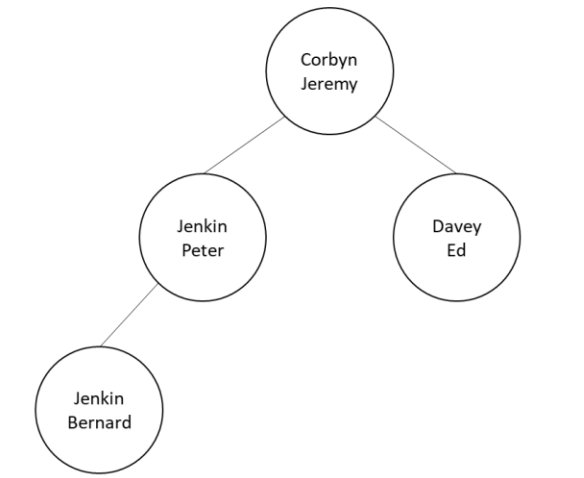


0	1	2	3	4
KeeganGillian	JenkinPeter	DaveyEd	JenkinBernard	CorbynJeremy

Process 2: First and last nodes swapped, and last node deleted from heap:

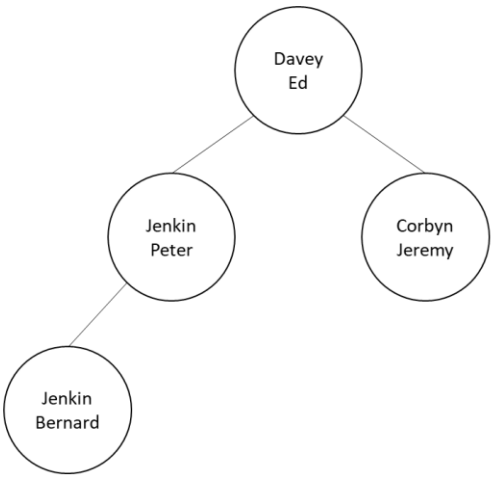


0	1	2	3	4
CorbynJeremy	JenkinPeter	DaveyEd	JenkinBernard	KeeganGillian

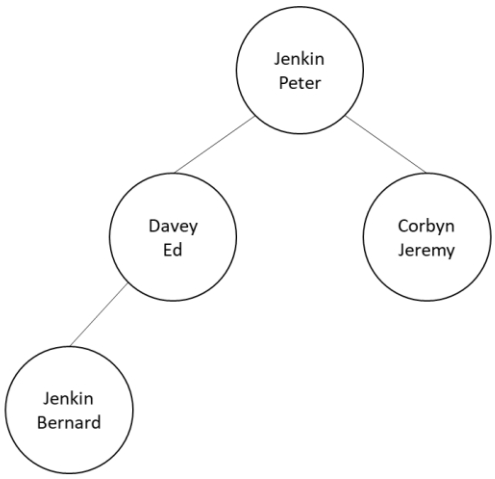


0	1	2	3	4
CorbynJeremy	JenkinPeter	DaveyEd	JenkinBernard	KeeganGillian

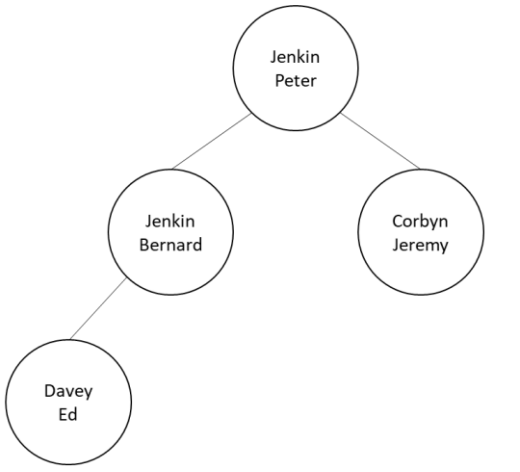
1 again:



0	1	2	3	4
DaveyEd	JenkinPeter	CorbynJeremy	JenkinBernard	KeeganGillian

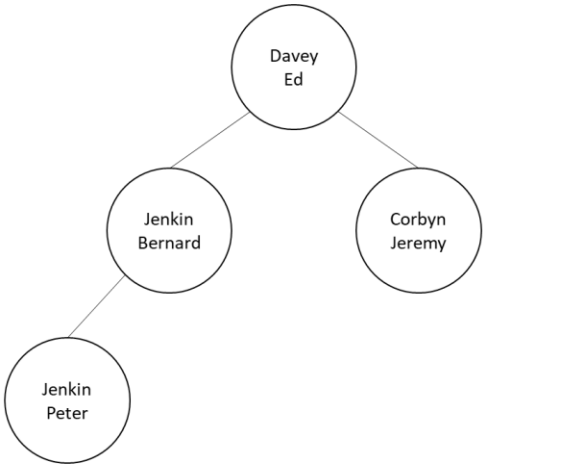


0	1	2	3	4
JenkinPeter	DaveyEd	CorbynJeremy	JenkinBernard	KeeganGillian

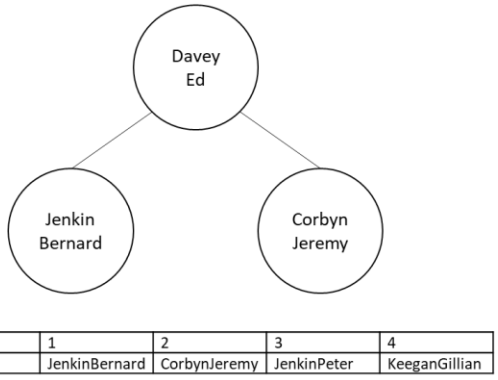


0	1	2	3	4
JenkinPeter	JenkinBernard	CorbynJeremy	DaveyEd	KeeganGillian

2 again:

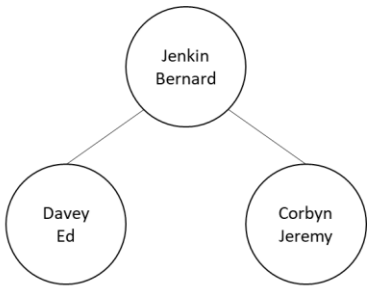


0	1	2	3	4
DaveyEd	JenkinBernard	CorbynJeremy	JenkinPeter	KeeganGillian



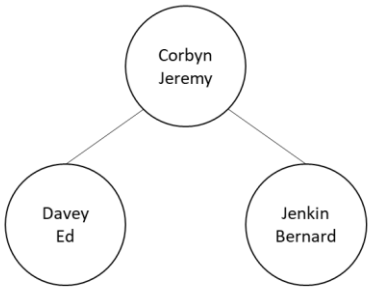
0	1	2	3	4
DaveyEd	JenkinBernard	CorbynJeremy	JenkinPeter	KeeganGillian

1 again:

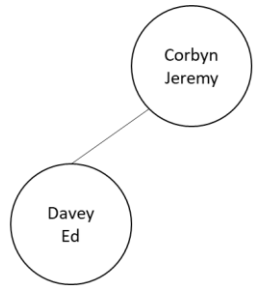


0	1	2	3	4
JenkinBernard	DaveyEd	CorbynJeremy	JenkinPeter	KeeganGillian

2 again (sorted at this point in this example, but subsequent steps are procedure):

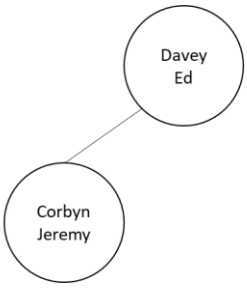


0	1	2	3	4
CorbynJeremy	DaveyEd	JenkinBernard	JenkinPeter	KeeganGillian



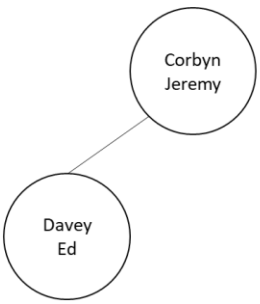
0	1	2	3	4
CorbynJeremy	DaveyEd	JenkinBernard	JenkinPeter	KeeganGillian

1 again:

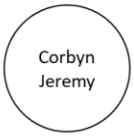


0	1	2	3	4
DaveyEd	CorbynJeremy	JenkinBernard	JenkinPeter	KeeganGillian

2 again:



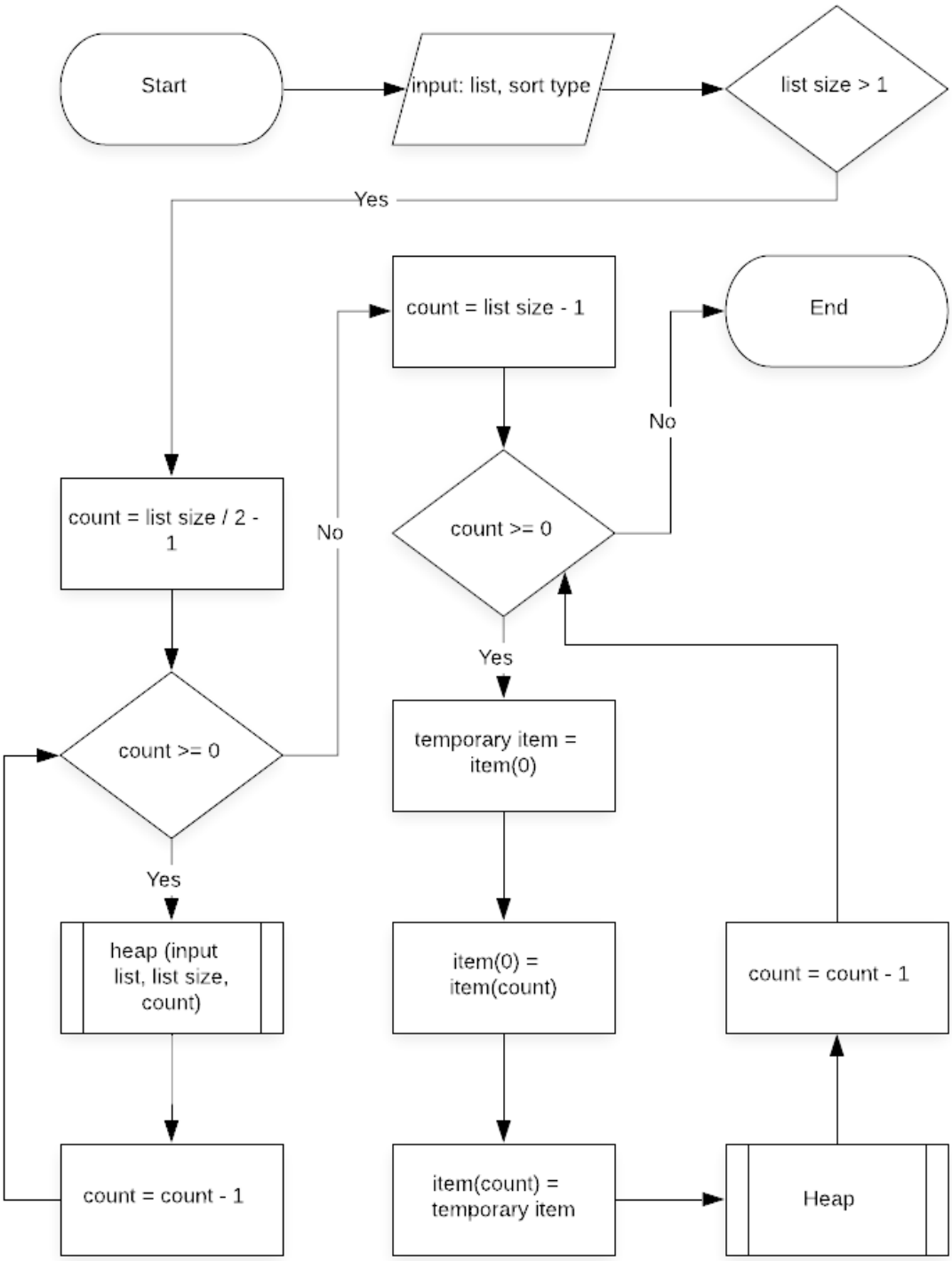
0	1	2	3	4
CorbynJeremy	DaveyEd	JenkinBernard	JenkinPeter	KeeganGillian

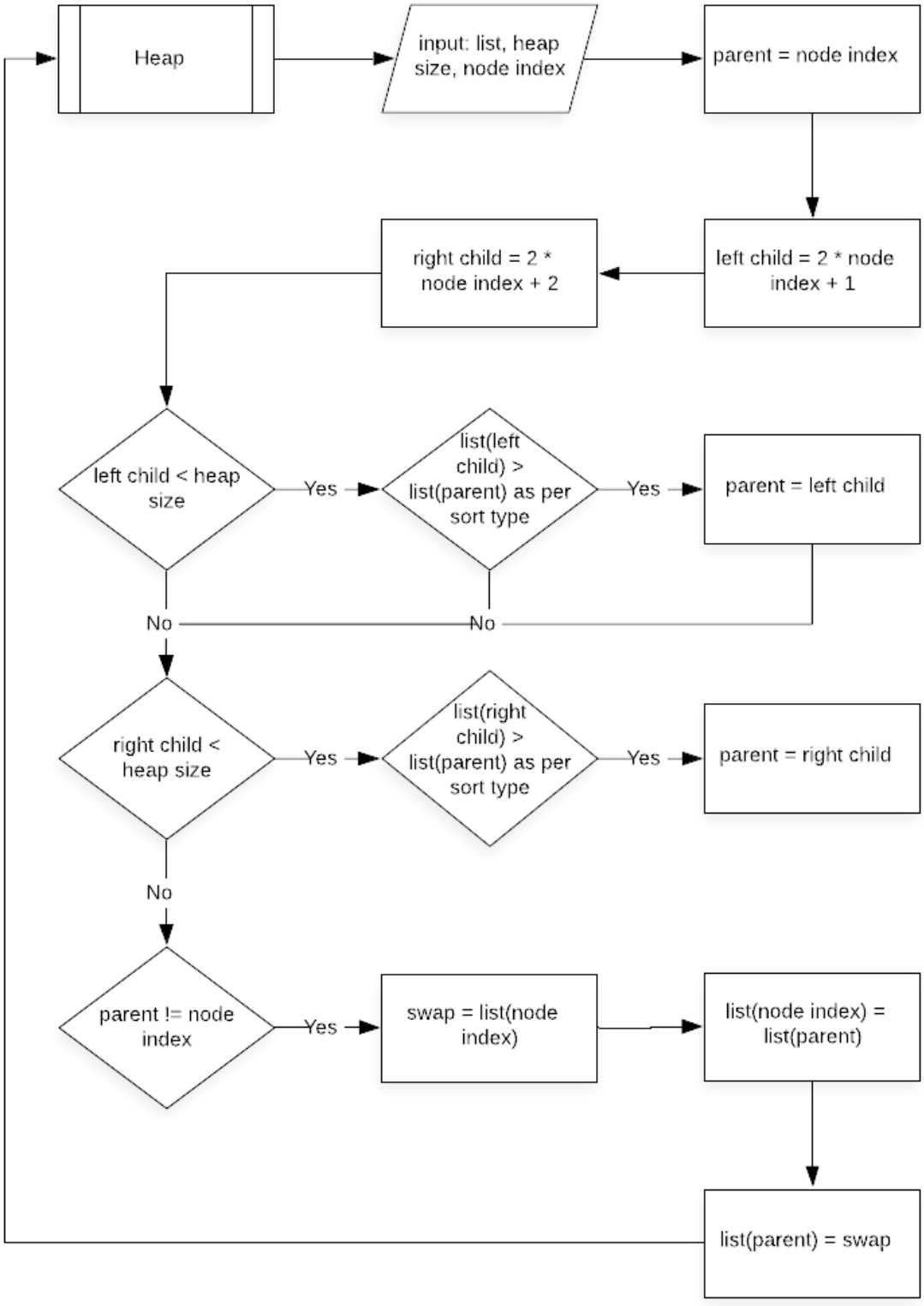


0	1	2	3	4
CorbynJeremy	DaveyEd	JenkinBernard	JenkinPeter	KeeganGillian

With just one node left in the heap, the heap sort is complete.

Flowchart:


Start
input: list, sort type
list size > 1
Yes
No
End
count = list size - 1
count = list size / 2 - 1
count >= 0
No
Yes
temporary item = item(0)
count >= 0
Yes
heap (input list, list size, count)
item(0) = item(count)
count = count - 1
count = count - 1
item(count) = temporary item
Heap


Heap
input: list, heap size, node index
parent = node index
left child = 2 * node index + 1
right child = 2 * node index + 2
left child < heap size
Yes
list(left child) > list(parent) as per sort type
Yes
parent = left child
No
No
right child < heap size
Yes
list(right child) > list(parent) as per sort type
Yes
parent = right child
No
parent != node index
Yes
swap = list(node index)
list(node index) = list(parent)
list(parent) = swap

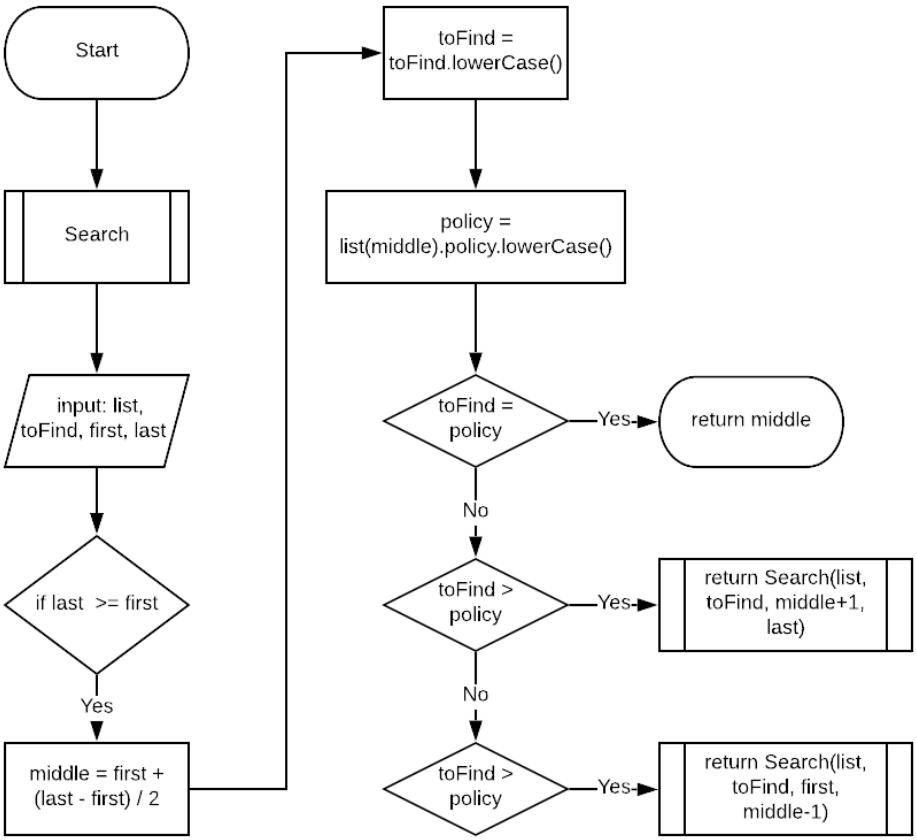
Binary search

The built-in algorithms for finding the indexes of and verifying the existence of the MP/votes-property-holding objects in lists by their properties isn't possible using built in methods. This necessitates a custom search algorithm.

The three main options before me are linear search, binary search and Fibonacci search. Linear search is simply sequential checking of the list items (by the property it's being identified by), making it the slowest, with its time complexity of O(n). Binary search, meanwhile, via its divide (into halves) and conquer strategy, is much faster, with average and worst-case time complexities of O(logn) and a best of O(n). The similar (but diving into Fibonacci sequential-sized parts) Fibonacci search also has time complexity of O(logn). However, with the amount of data I'm working with, Binary is the more optimal method, as Fibonacci would break it into ever-increasing chunks recursively. I will be implementing a recursive, rather than iterative, method.

In addition, when comparing via Heap Sort, all the characters of the string must be of the same case (lower case in this case), as Java will do a Unicode codepoint comparison, which treats uppercase letters differently (lexicographically greater) than lower case ones. So subsequent references to lexicographical/alphabetic comparison will be in terms of lower case. However, it must be emphasised that the strings and values themselves won't be changed to lower case. Rather, their full lower case variants will be compared in if/else statements.

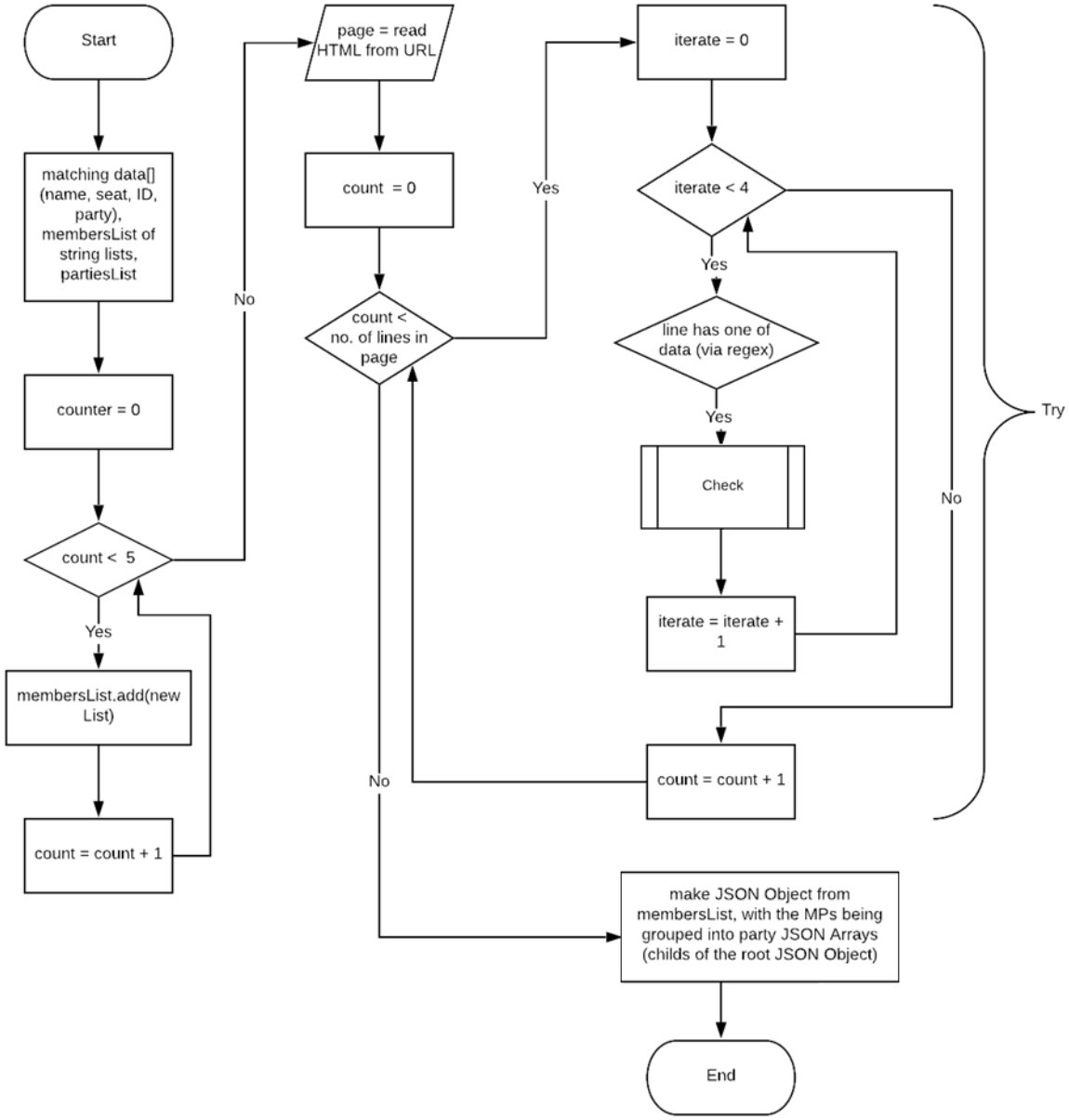
Flowchart:

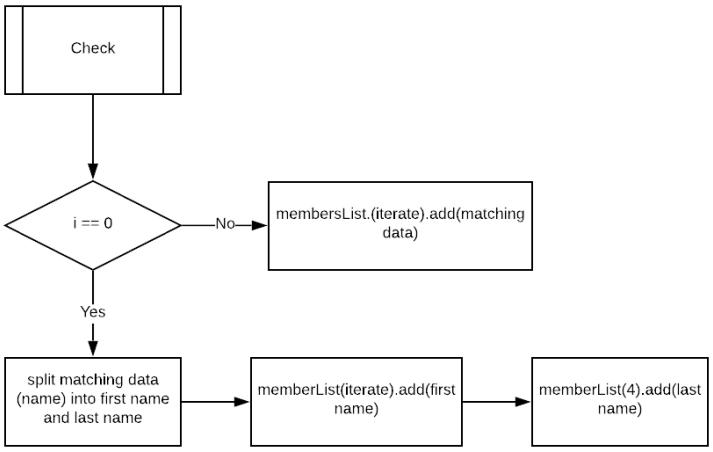


If there are multiple items with the same property (‘policy’ in the previous diagram, ‘searchedProperty’ in this example) that the BinarySearch was performed against (occurring with (multi-criteria) policies that can’t always be unique even – MPs’ economic or social positions), then the fact that the sorted list means other items of the same value for that property are immediately and consecutively either side of the found-but-not wanted item can be used to find the position of the object-to-find:

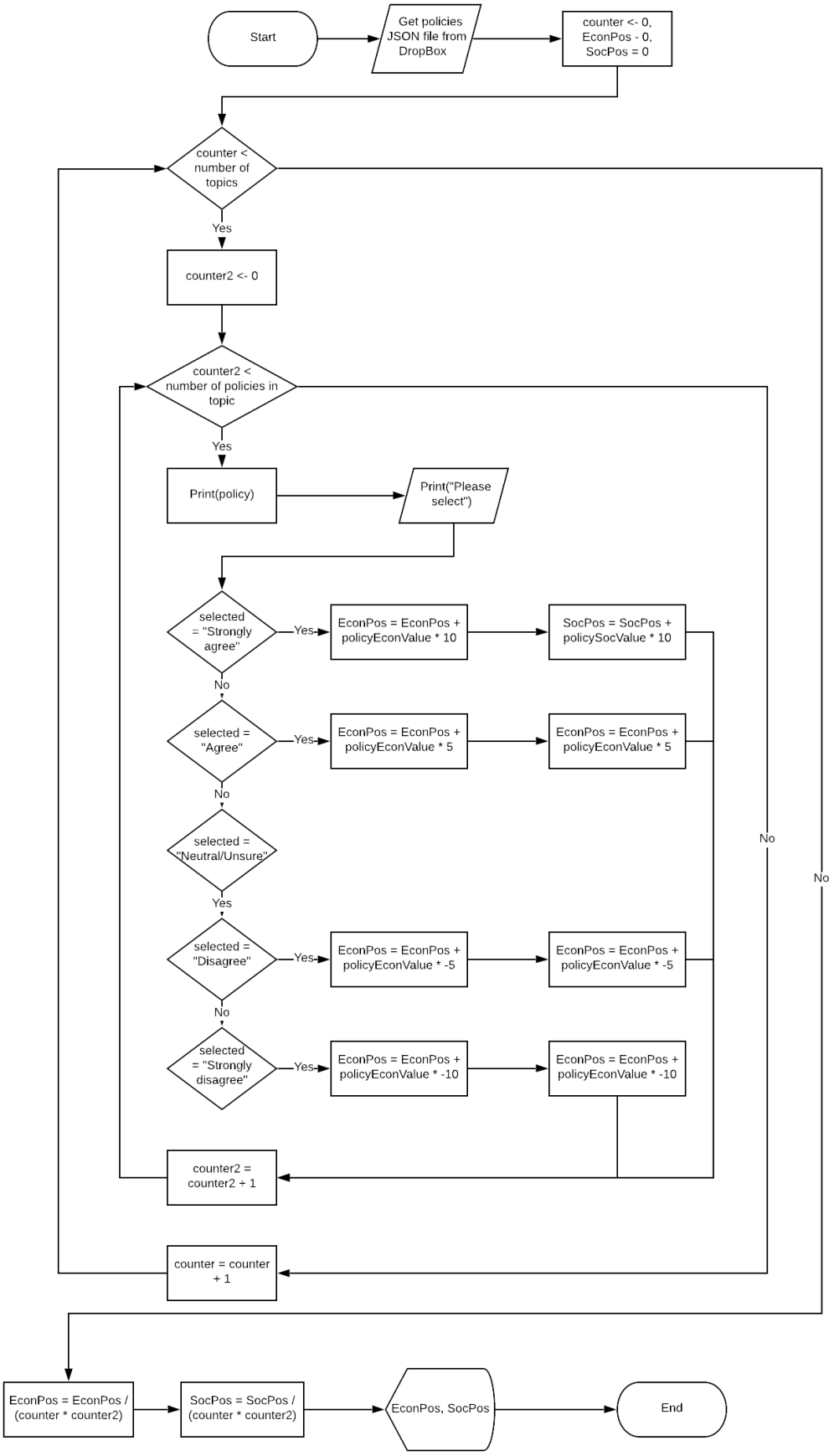
```
MPs <- list of all MP objects
position <- BinarySearch(MPs, MPToFind, 0, MPs.size() - 1)
IF MPs.get(position).getUniqueProperty() !=
    MPToFind.getUniqueProperty() THEN
      positionDown <- TraverseDown(position, MPToFind)
      IF positionDown = -1 THEN
            position <- TraverseUp(position, MPToFind)
      ELSE position <- positionDown
      ENDIF
ENDIF

FUNCT TraverseDown(position, MPTOFind)
      FOR iterator <- position TO 0 STEP -1:
            MPAtPos <- MPs.get(iterator)
            IF MPAtPos.getSearchedProperty() =
               MPToFind.getSearchedProperty() THEN
                  IF MPAtPos.getUniqueProperty() =
                     MPToFind.getUniqueProperty() THEN
                     RETURN iterator
                  ENDIF
            ELSE BREAK;
            ENDIF
      ENDFOR
ENDFUNCT

FUNCT TraverseUp(position, MPTOFind)
      FOR iterator <- 0 TO MPs.size() - 1 STEP 1:
            MPAtPos <- MPs.get(iterator)
            IF MPAtPos.getSearchedProperty() =
               MPToFind.getSearchedProperty() THEN
                  IF MPAtPos.getUniqueProperty() =
                     MPToFind.getUniqueProperty() THEN
                     RETURN iterator
                  ENDIF
            ELSE BREAK;
            ENDIF
      ENDFOR
ENDFUNCT
```

Making JSONObject of MPs from online HTML





User-position-determining quiz



Explained

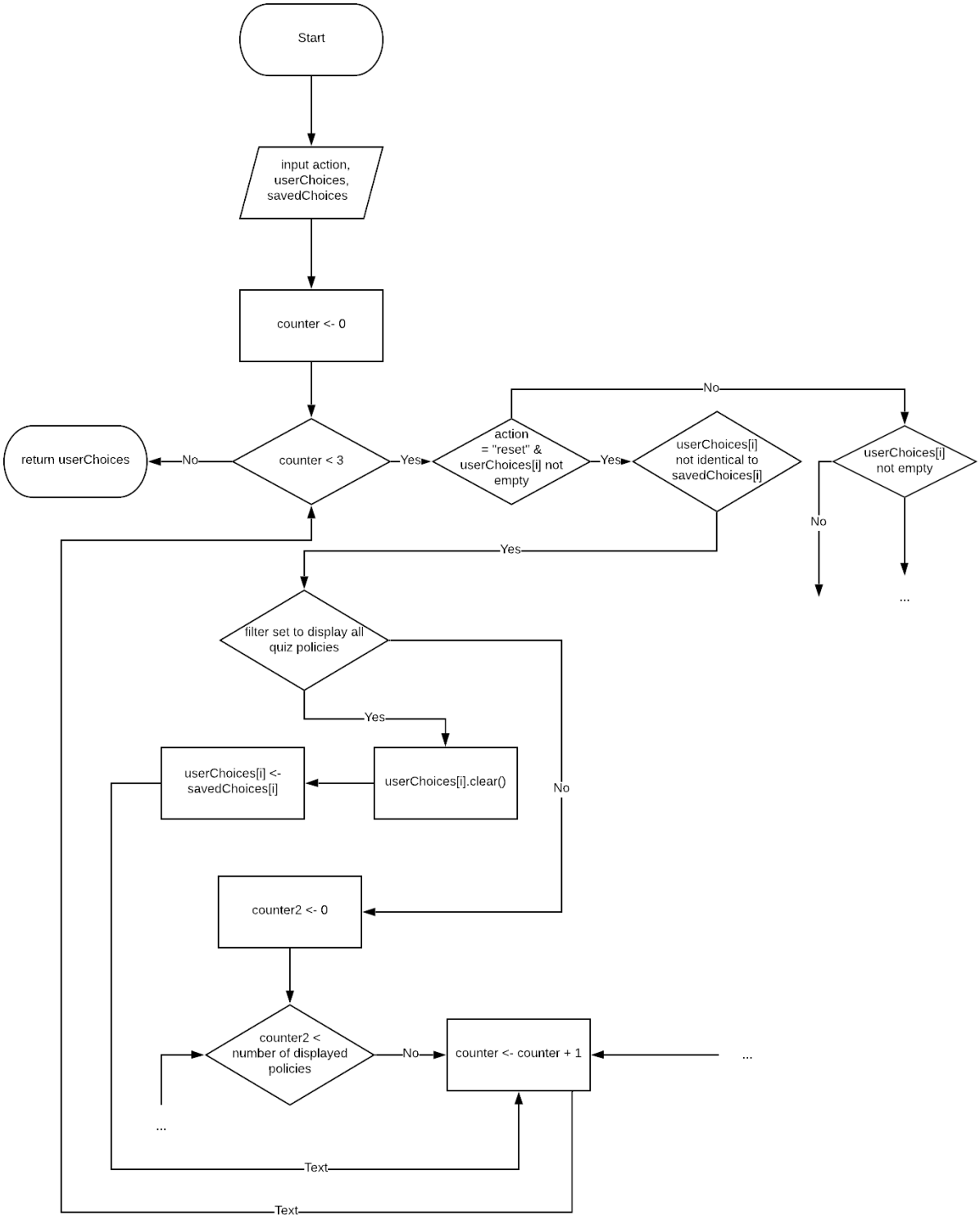
"Agree" results in the policy's economic/social valuation (1, 0 or -1) being multiplied by 5, because of valuation is multiplied by 0.75 (represents agreement), and 0.25 * valuation (represents times where the case is the opposite) is subtracted from that; the result is multiplied by 10 to facilitate the -10 to 10 axis. Likewise with disagree, where 0.25 and 0.75 are reversed.

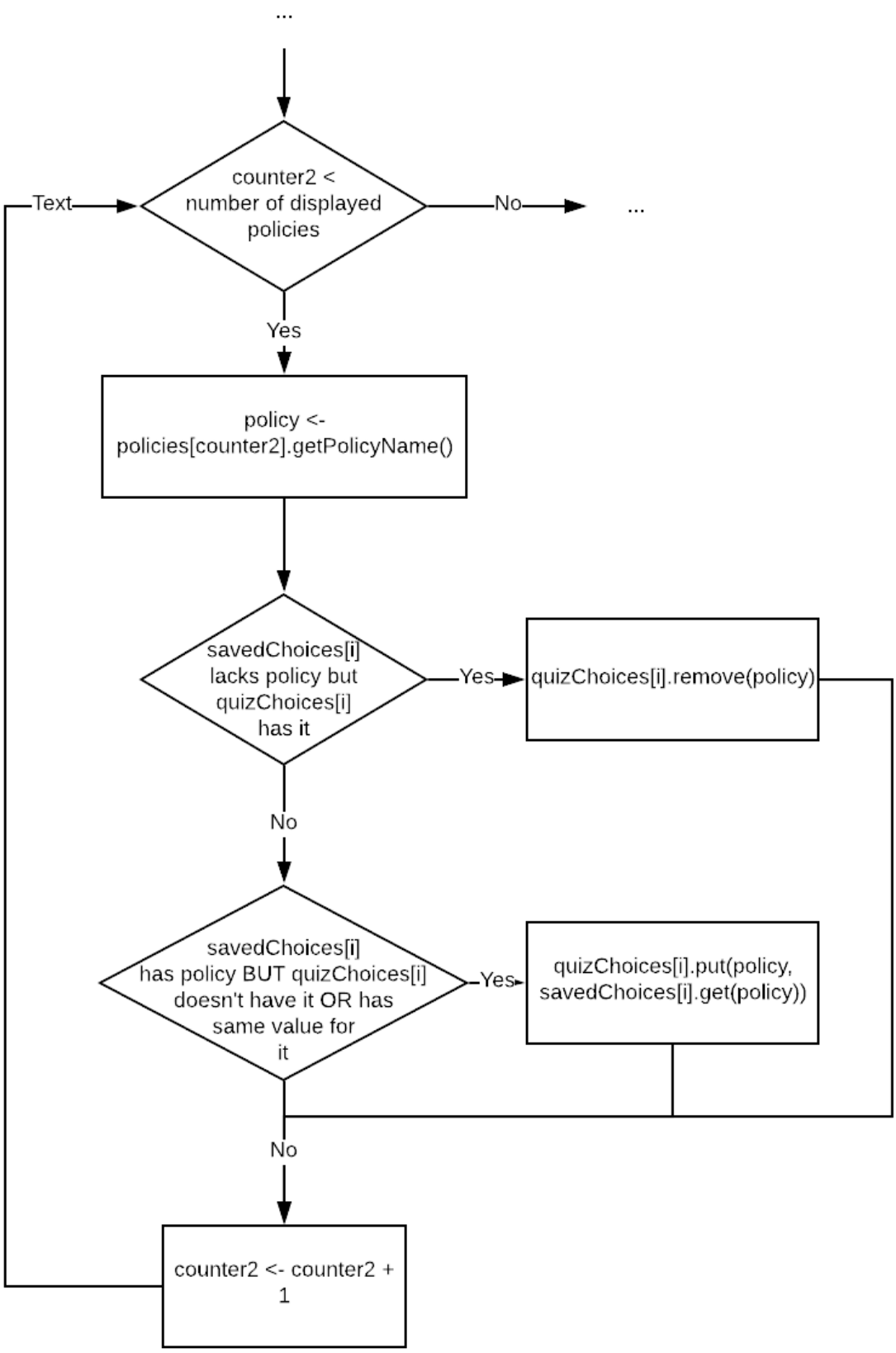
However, in order to implement this in an app where the user will be able to go back and forth between questions, and the preferences won't be saved as they are made, but rather saved in-bulk when the user chooses to, this will have to be done differently:

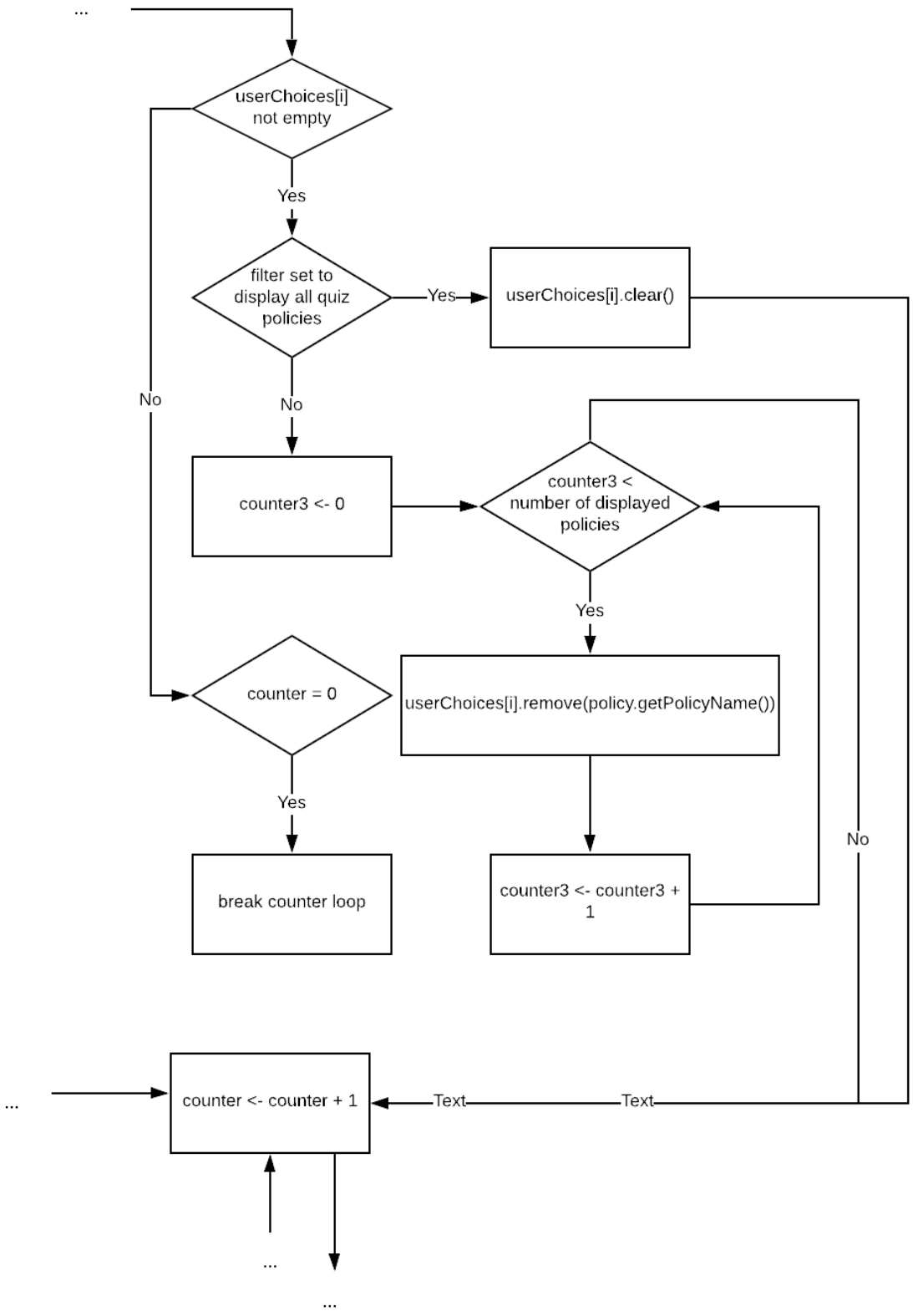
```
userChoices <- array[3] of dictionaries (key: string, value:
integer)
#key for all usersChoices' dictionaries' items is a policy
usersChoices[0] values <- userSelection (option chosen)
userChoices[1] values <- policy economic values calculated
using userSelection
userChoices[2] values <- policy social values calculated using
userSelection

…

FUNCT onOptionChosen(policy, userSelection):
      IF quizDictionaries[0] doesn't contain policy OR
         quizDictionaries[0].get(policy) != userSelection THEN
            economic <- policy.economicValue,
            social <- policy.socialValue
            weight <- [10, 5, 0, -5, -10]
            FOR counter <- 0 TO weight.length:
                  IF userSelection == counter THEN
                        IF economic != 0 THEN
                        userChoices[1].put(policy, economic
                        * weight[counter] ENDIF
                        IF social != 0 THEN
                        userChoices[2].put(policy, social
                        * weight[counter] ENDIF
                        BREAK LOOP
                  ENDIF
            ENDFOR
      ENDIF
ENDFUNCT

FUNCT calculatePosition()
      userEconomic <- 0
      userSocial <- 0

      FOREACH value in userChoices[1]:
            userEconomic <- userEconomic + value
      ENDFOREACH
```

```
      FOREACH value in userChoices[2]:
            userSocial <- userSocial + value
      ENDFOREACH

      IF userChoices[1].size() != 1 THEN userEconPos =
         userEconPos / userChoices[1].size() ENDIF

      IF userChoices[1].size() != 1 THEN userSocPos =
         userSocPos / userChoices[1].size() ENDIF
      OUTPUT userEconPos, userSocPos
ENDFUNCT
```

In order to clear or reset the quiz as efficiently as possible:

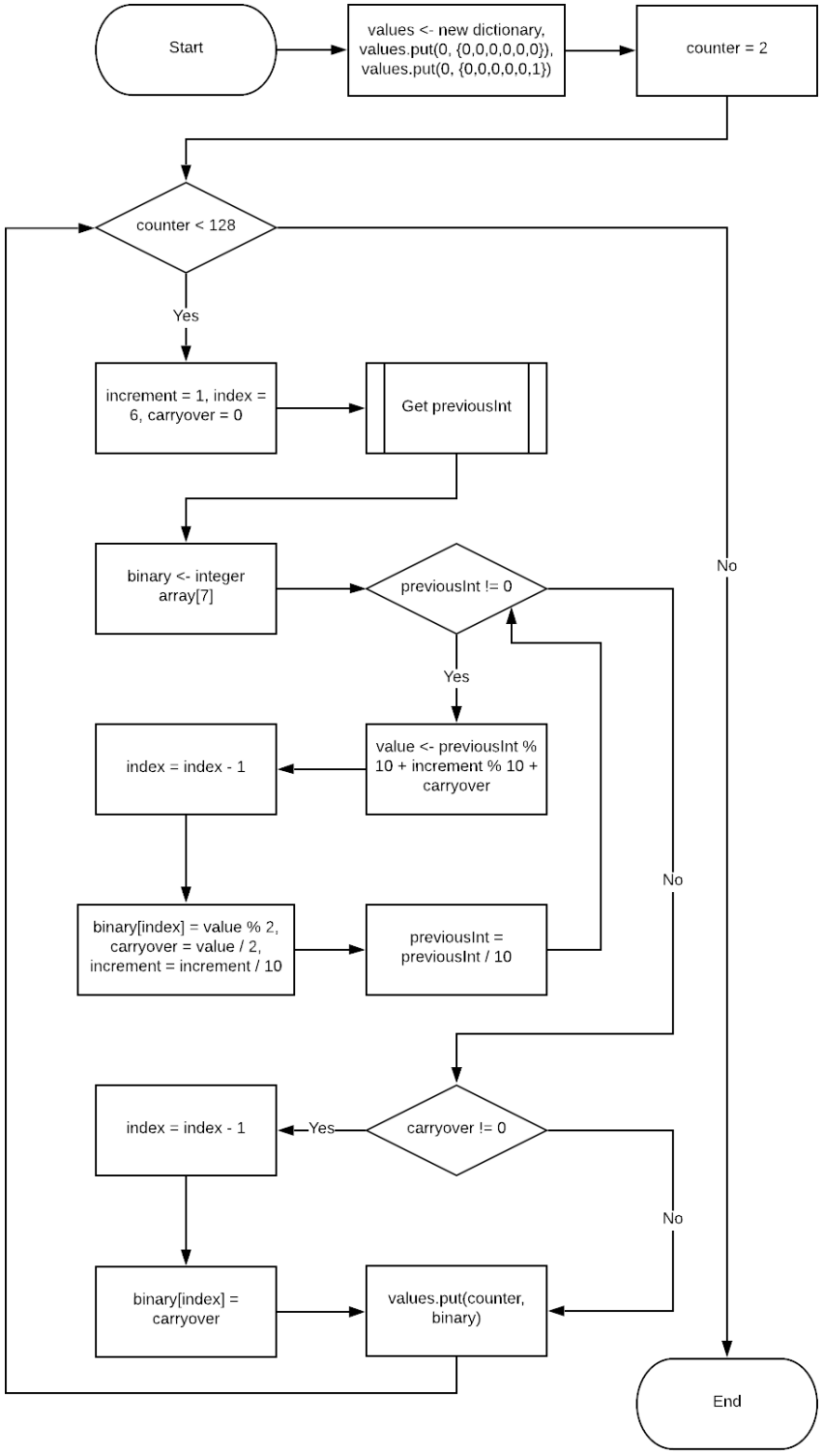



...
counter2 < number of displayed policies
Text
No
...
Yes
policy <- policies[counter2].getPolicyName()
savedChoices[i] lacks policy but quizChoices[i] has it
Yes
quizChoices[i].remove(policy)
No
savedChoices[i] has policy BUT quizChoices[i] doesn't have it OR has same value for it
Yes
quizChoices[i].put(policy, savedChoices[i].get(policy))
No
counter2 <- counter2 + 1

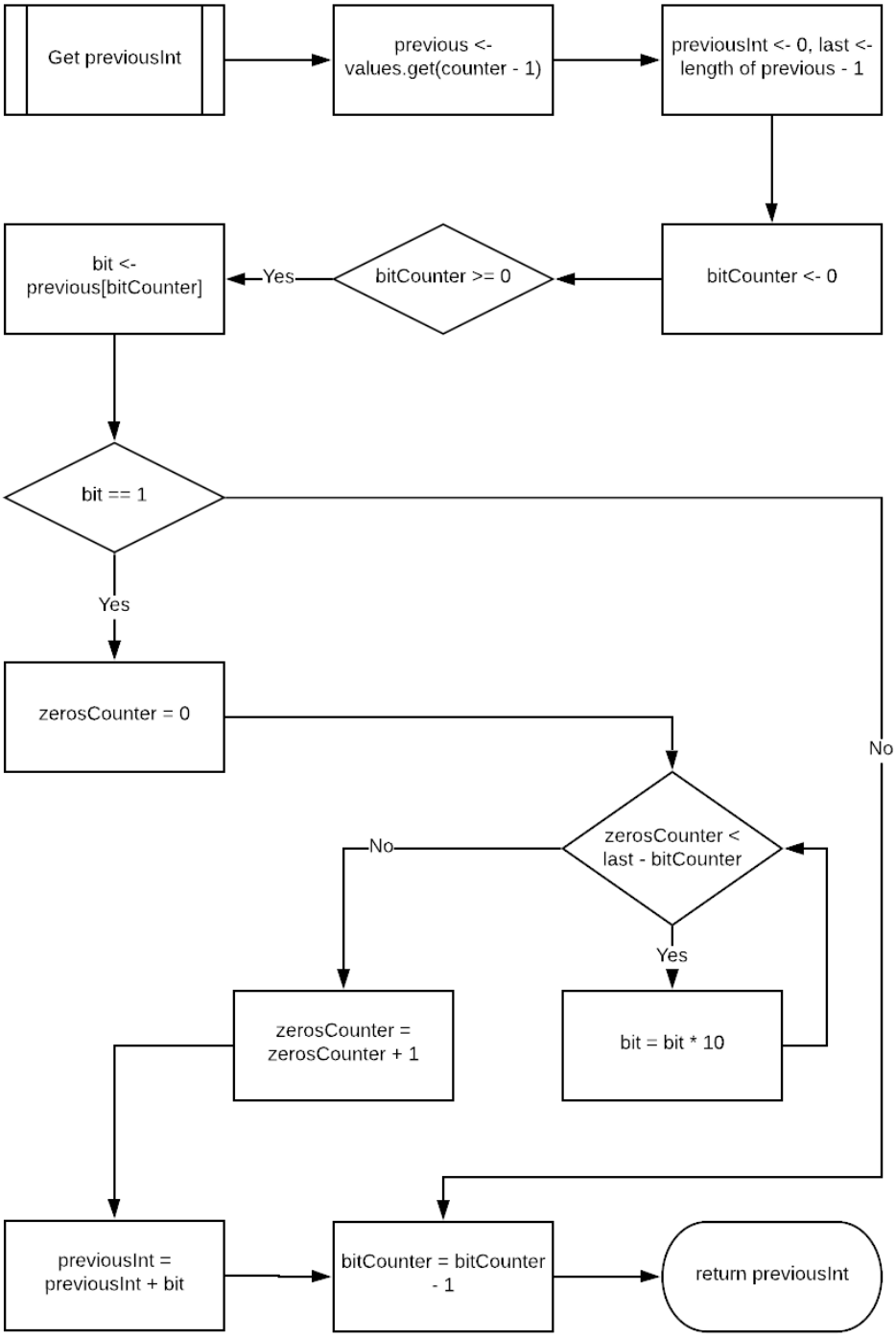

...
userChoices[i] not empty
Yes
filter set to display all quiz policies
Yes
userChoices[i].clear()
No
No
counter3 <- 0
counter3 < number of displayed policies
Yes
counter = 0
userChoices[i].remove(policy.getPolicyName())
Yes
No
break counter loop
counter3 <- counter3 + 1
...
counter <- counter + 1
Text
Text
...
...

Dropbox code encryption algorithm(s)

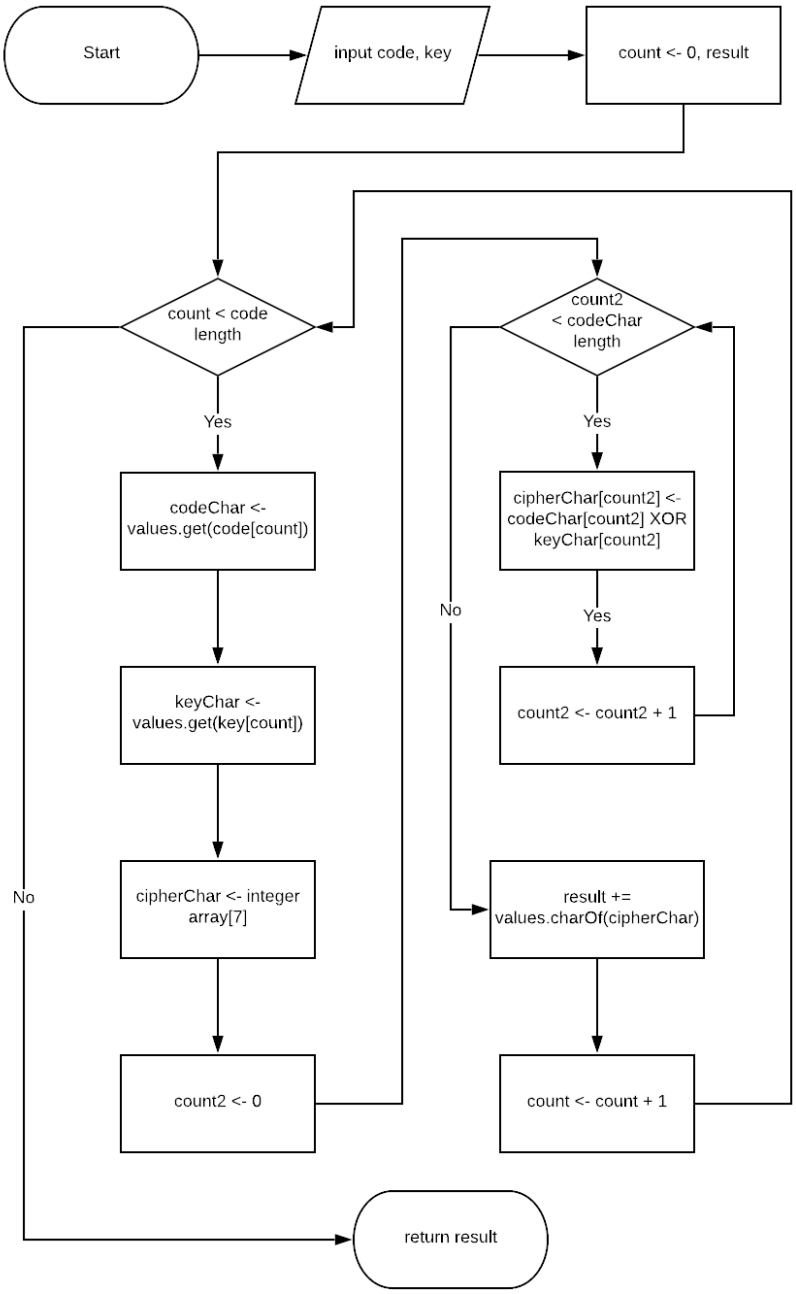
For security purposes, the Dropbox URL access token is encrypted (and then subsequently decrypted via the same method, the next time the app is started) using the **Vernham Cipher** with XOR (between the bits of the key and cipher/plaintext) functionality, with a new randomised key and ciphertext being saved and made each time the app is opened. In order to use this, the 7-bit binary values of the ASCII characters must be found:



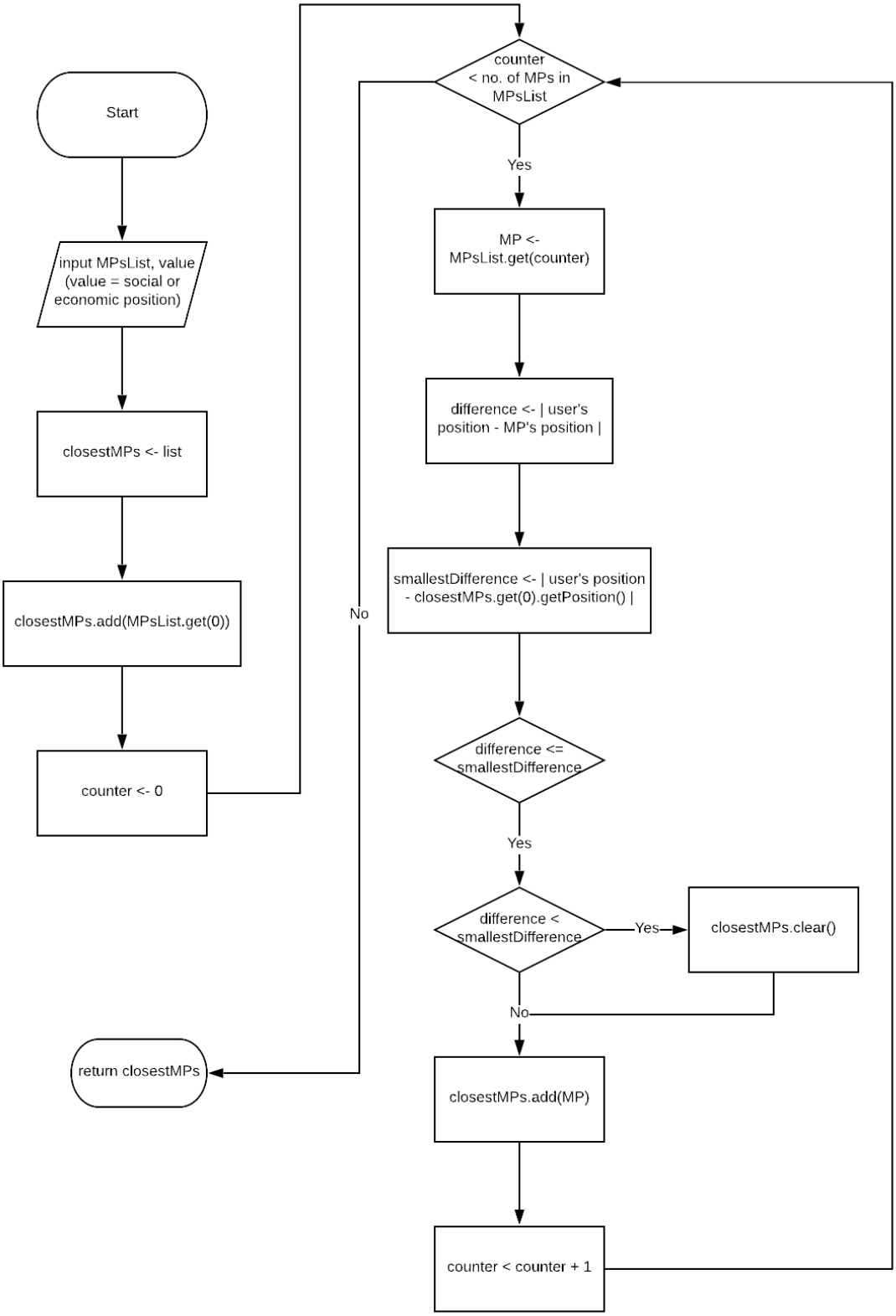
The following converts the binary array into integer form:



The following is the code for the Vernam Cipher algorithm, which encrypts and decrypts the plaintext/ciphertext (code) using the key:



Algorithm to find closest MPs to user

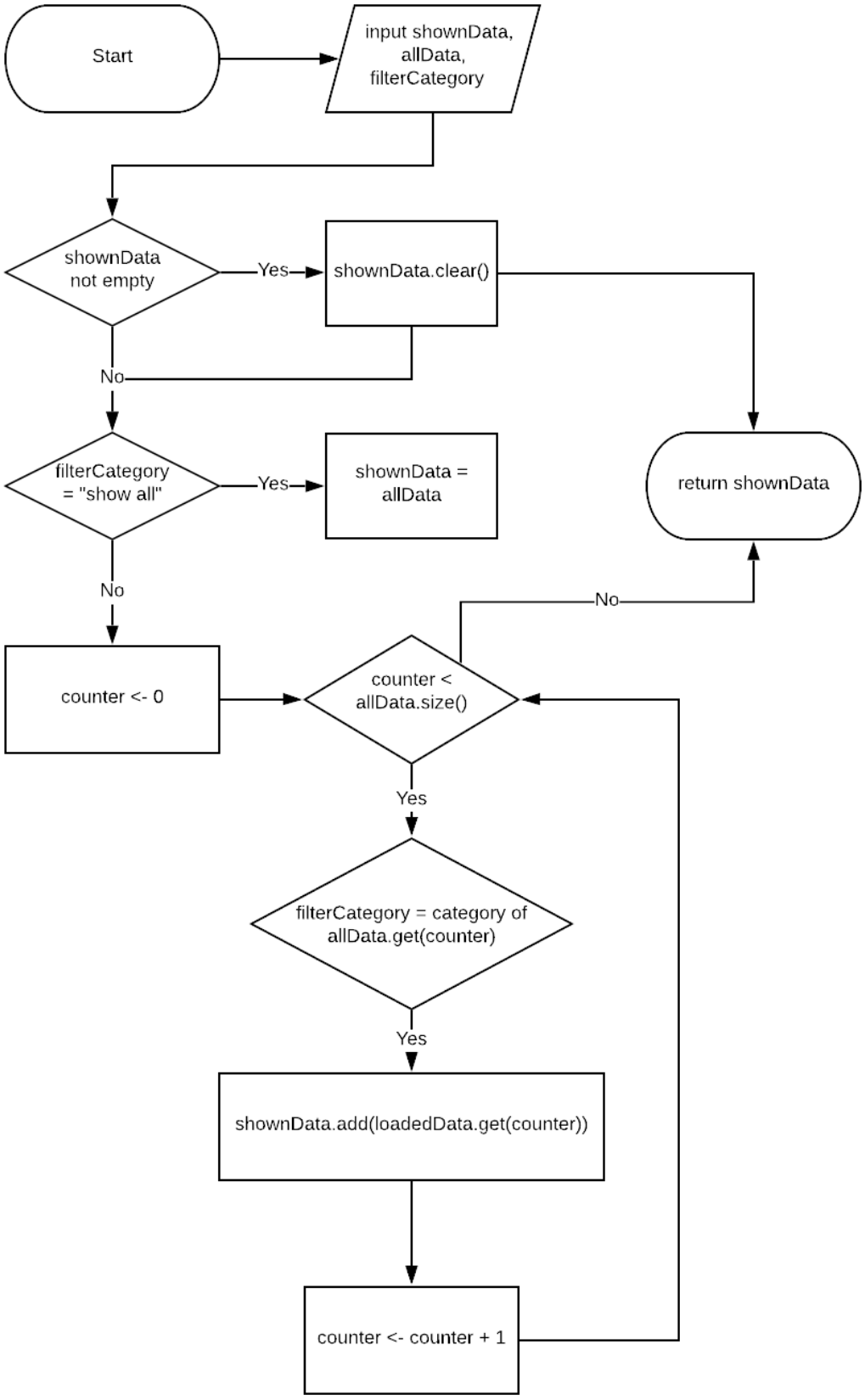


Retrieve JSON data from Dropbox to Android

```
data <- list
TRY
    json <- getJSONData(fileName)
    partiesVotes <- IF showing an MP's voting record AND MP is
in a party THEN
       partiesVotes.get(party)
      ENDIF

    FOR counter <- 0 to json.size STEP 1:
        category <- json[counter]
        array <- json.get(category);
        FOR counter2 <- 0 to array.length STEP 1:
                    obj <- array[counter2]
            IF on quiz THEN
                        #category = policy topic
                economic <- obj.get("economic")
                social <- obj.get("social")
                IF economic != 0 && social != 0 THEN
                    data.add(policy(obj.get("policy"),

category,

economic,

social))
                        ENDIF
            ELSEIF showing list of MPs THEN

                        #category = MP's party
                data.add(MP(obj.get("first"),

      obj.get("surname"),
                                            category,
                                            obj.get("seat"),
                                            obj.get("id"),

      obj.get("economic"),
                                            obj.get("social"))
            ELSEIF showing an MP's voting record
                        #category = policy topic
                policy <- obj.get("policy")
                percent <- obj.get("percent")
                data.add(policy(policy,
                                                  category,
                                                  percent +
(IF MP in a party THEN
                                                  "\nParty: "
+ partiesVotes.get(category).get(policy) ENDIF))
            ENDIF
```

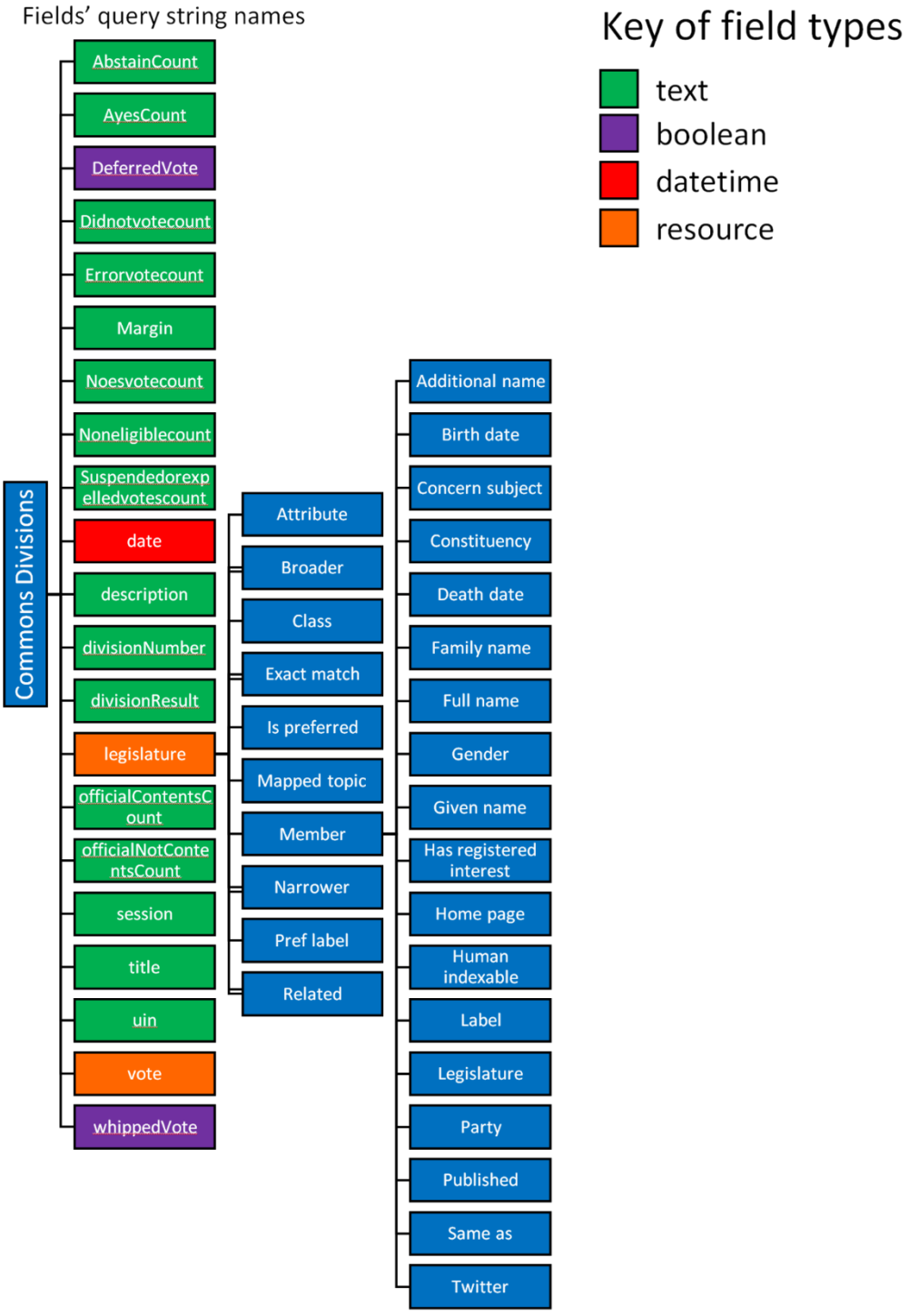
```
        ENDFOR
    ENDFOR
    HeapSort(data)
    IF mode = 1 THEN
        misc <- list
            FOREACH data item: IF item is miscellaneous THEN
misc.add(m) ENDIF ENDFOREACH
                    FOREACH misc item: data.remove(item)
ENDFOREACH
                    FOREACH misc item: data.add(item)
ENDFOREACH
        ENDIF
 CATCH and HANDLE exception
```

Filter data


Start
input shownData, allData, filterCategory
shownData not empty
Yes
shownData.clear()
No
filterCategory = "show all"
Yes
shownData = allData
No
return shownData
counter <- 0
counter < allData.size()
No
Yes
filterCategory = category of allData.get(counter)
Yes
shownData.add(loadedData.get(counter))
counter <- counter + 1

Grabbing the data

Initially, I intended to use the official House of Commons Divisions API:



This would provide information on votes, parties, votes topics and types, suspension/expulsion, party whip adherence and comparison of MPs. However, the lack of detail in its documentation proved a barrier, and the lack of any policy categorisation of any of the divisions made it unworkable in terms of fulfilling the objectives. I subsequently decided to instead scrape the voting data off of TheyWorkForYou, using a HTML parser library called JSoup:

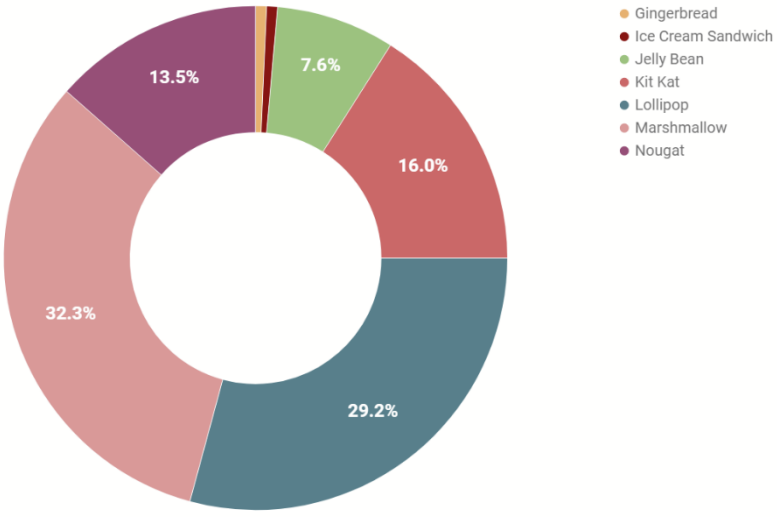
```
Document doc =
Jsoup.connect("https://www.theyworkforyou.com/mp/number/name/constituency".get();
Elements votes = doc.getElementsByClass("vote-descriptions").select("li");
for (Element vote : votes) textView.setText(votes.text());
```

However, I later realised that I could instead simply create my own mini-parser using regex, thus eliminating the need for this 3rd party library. However, as detailed in the analysis, TheyWorkForYou's data was slow to update and lacking in comparison to PublicWhip, though its categories are somewhat better. In order to access the greater range of data, I decided to use the then 86MB JSON source file on the raw data section of PubilicWhip, which was plentiful in the amount of data it provided. But with the file too large for Android to establish a connection to, this required a program to break it down into each individual MP.

Updating the data

Initially I sought to use NodesJS in order to host the breakdown program, with automated breakdowns in order to ensure the JSON source data is up-to-date. However, Heroku proved superior.

Compatibility

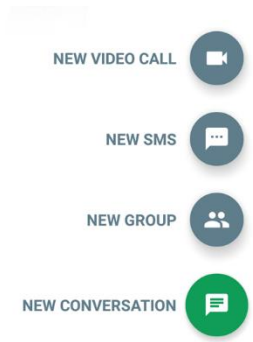


When developing an Android application, a target and minimum SDK level must be selected. The default in IntelliJ IDEA's Android plugin is API level 29 aka Android Q SDK, which replaces the previous design library with a new Android X library that brings in features that were previously left to third party libraries. However, the device that I will be testing the application on is on Android P (API level 28), and the design library is more than enough to fulfil my requirements. Material design, which is the user interface and experience (UI/UX) framework which this application is built on, was only supported from Android Lollipop (5.0). Given that Lollipop, Marshmallow and Nougat have a supermajority of the Android userbase at 75%, as shown above

(and this isn't even counting Android P and the newer but so far Android Q/10), it is a no-brainer to target devices on Lollipop and above.

User Interface and Experience

As mentioned before, the app will be rooted in material design, a type of flat design that is used by Google across all of its products, from Android apps to Google Search itself. The following are its most common UX patterns:



Floating Action Button: I have no firm intention on using this but am open to integrating it, given its versatility of applications. Another option regarding these is them functioning as a collapsible menu of other floating buttons

App bar: this facilitates the app's navigation and configuration (via a 3-dot menu that contains a link to the settings if defined). I will additionally include a tab switcher in this, so that users can move between the MPs, their own profile, and policies.

Navigation drawer: this links to different 'activities' (pages) of the app, making it more accessible. This isn't a priority, but it could be useful for displaying favourited pages.

APIs and libraries

The app itself will by nature have to make use of packages in the built-in Android support library for the UI aspects, but also the Jackson JSON Java library in order to facilitate the parsing of the online JSON data. The reason Jackson was chosen over JSON.org is because it offers the option of stream the JSON data instead of bulk downloading it, and thus incurring far less memory utilisation. The file breakdown program meanwhile makes us of the Dropbox API in order to host the broken-down files, which the app accesses. In order to use the Dropbox API, I will generate an app-specific access code and hardcore that into both the program (which uploads to it) and the app (which reads from it). This will eliminate the need for a manual sign-in to Dropbox each time the API is called, which would have severely impaired the functionality of the project.

Using Jackson

The dependencies of artifact ids `jackson-core`, `jackson-bind` and `jackson-annotations` and group id `com.fasterxml.jackson.core` are added to the pom.xml of the Maven JSON breadown java program.

In Android, the following is added to the build.gradle app module:

```
implementation 'com.fasterxml.jackson.core:Jackson-core:2.10.2'
implementation 'com.fasterxml.jackson.core:jackson-
annotations:2.10.2'
implementation 'com.fasterxml.jackson.core:jackson-databind:2.10.2'
```

To use objects of the class `ArrayNode`, `JsonNode` and `ObjectNode`, to read and write JSON, an `ObjectMapper` is needed:

```
ObjectMapper mapper = new ObjectMapper()
```

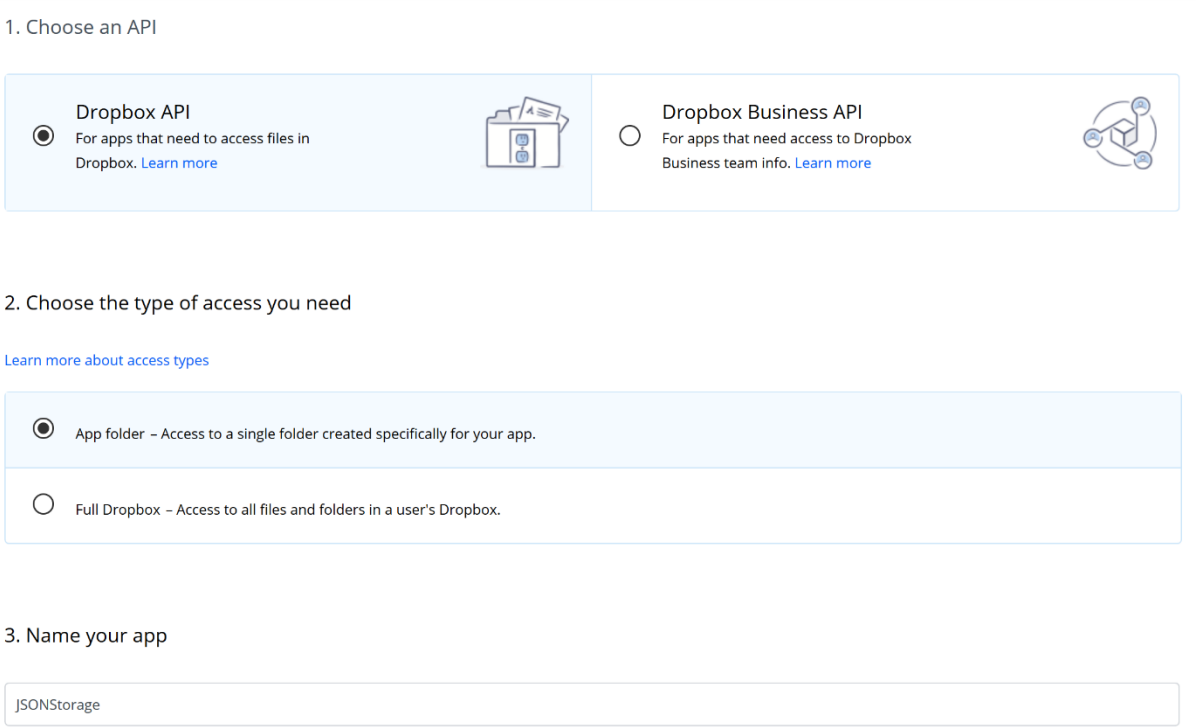
Using Dropbox

To use in Android, the following is added to the build.gradle app module:

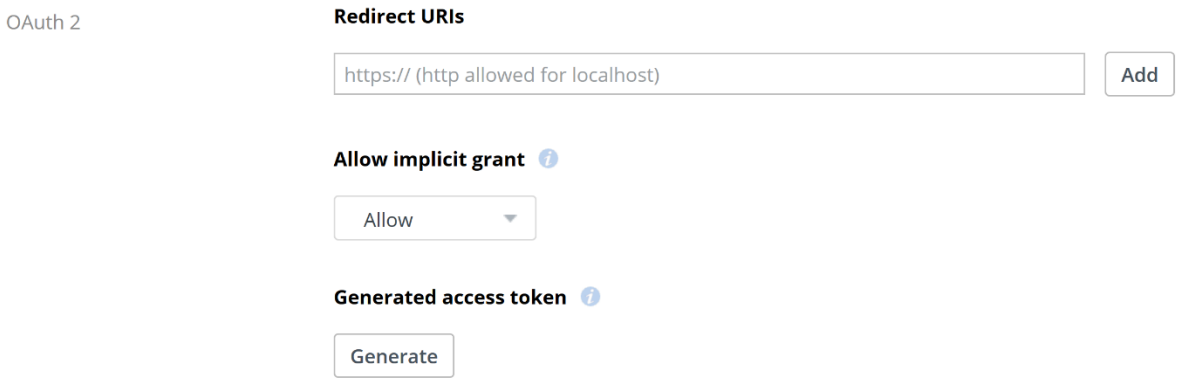
```
implementation 'com.dropbox.core:dropbox-core-sdk:3.1.1'
```

To use in the JSON Breakdown and Creation Program, the dependency on artifact id `dropbox-core-sdk` and group id `com.dropbox.core` is added to the pom.xml file.

On the Dropbox developer platform app console:



In the app settings, the access token is created:



The following Java code is necessary to actually use the API:

```
DbxRequestConfig config =
DbxRequestConfig.newBuilder(clientIdentifier).withUserLocale(L
ocale.getDefault().toString()).build();
client = new DbxClientV2(config, accessToken);
```

GraphView

In order to visually represent the MPs' positions – to create the political compass itself, the GraphView library will be used (https://github.com/jjoe64/GraphView). To use:

Build.gradle:

```
implementation 'com.jjoe64:graphview:4.2.2'
```

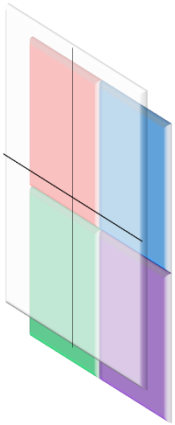
To the XML:

```
<com.jjoe64.graphview.GraphView
      android:id="@+id/graphView"
      android:layout_height="100dp"
      android:layout_width="100dp">
</com.jjoe64.graphview.GraphView>
```

Java code for initialisation and setting axis

```
GraphView graphView = findViewById(R.id.graphView);

graphView.getViewport().setXAxisBoundsManual(true);
        gView.getViewport().setMinX(-10);
        gView.getViewport().setMaxX(10);
graphView.getViewport().setYAxisBoundsManual(true);
        gView.getViewport().setMinY(-10);
        gView.getViewport().setMaxY(10);
…
```

Given only simple points will be plotted on this GraphView, **`PointsGraphSeries`** will be the type of the points series to be added onto the view.



In order to achieve the effect of the four quadrants having distinct colours, the GraphView will need to be 1) transparent and 2) stacked within a RelativeLayout, on top of a Relative Layout of matching size that consists of 4 different-coloured views of half the width and length that are adjacent to one and other in the following way:

Heroku

In order to ensure that the JSON data used by the application is up-to-date, Heroku will schedule the JSON file breakdown and creation program to run at set intervals.

To facilitate this, the breakdown program needs to be made as a Maven project, and then the Maven app assembler (groupId: `org.codehaus.mojo`; artifactId: `appassembler-maven-plugin`) and compiler (groupId: `org.apache.maven.plugins`; artifactId: `maven-compiler-plugin`) plugins need to be added to its pom.xml file.

Deploy to Heroku

The following code will be run in command-line, at the project's local location:

To push and commit to Git version control:

```
git init (if on first run)
git add .
git commit -m "Commit name"
```

To push and deploy to Heroku:

```
heroku create
git push heroku master
```

To manually run:

```
git push heroku run "sh target/bin/main"
```

To schedule the app, Heroku Advanced Scheduler will be used.

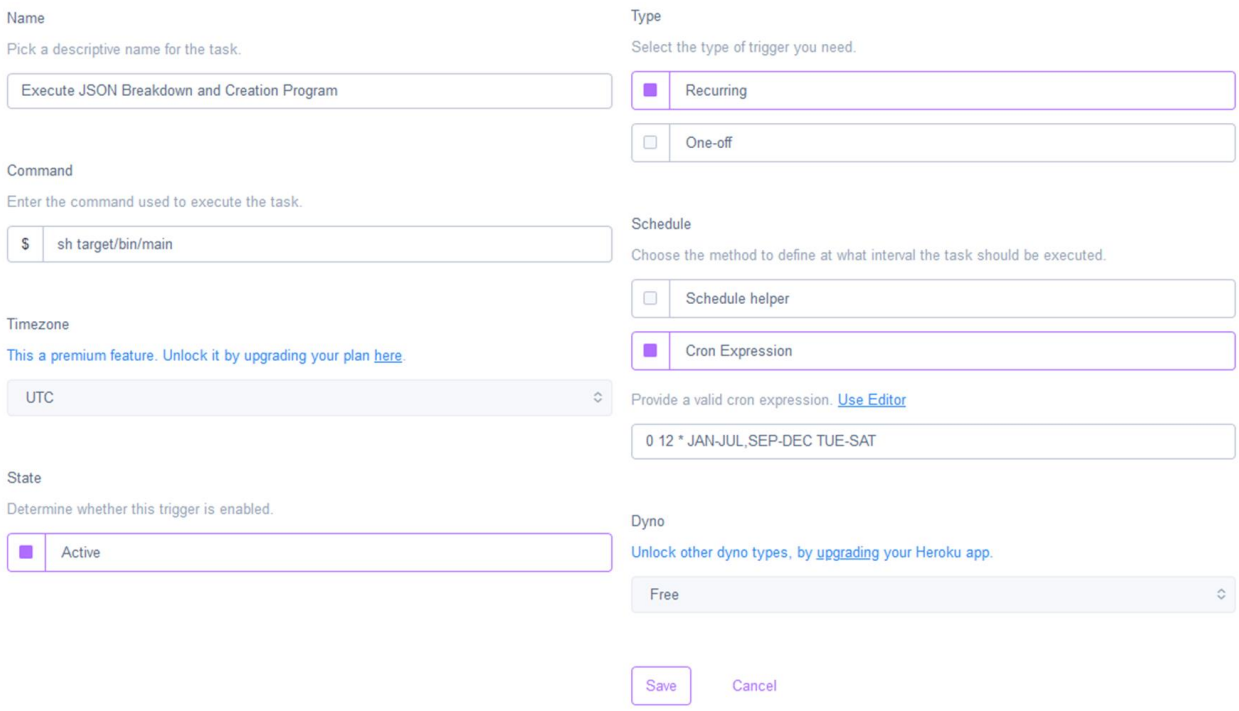
To add to the Heroku app, and subsequently to access the plugin dashboard:

```
heroku addons:create advanced-scheduler
heroku addons:open advanced-scheduler
```

On PublicWhip's FAQ, they state that "New divisions usually appear in Public Whip the next morning, but sometimes take a day or two longer". As parliament sits from Monday to Friday, tho less frequently on the latter day, and the only month where it doesn't sit at all is August the following Cron expression will be apt:

```
0 12 * JAN-JUL,SEP-DEC TUE-SAT
```

This sets a schedule for the Heroku app to run at 12:00 from Tuesday to Saturday. This is set as follows in the dashboard:



Stacks

A stack will be used for the purpose of app navigation in the Android app. This last in in first out data structure will store 4-item arrays containing the app mode (quiz/-1, listing MPs/0 or an MP's record/1, sort mode, filter mode and position of the data item in view/clicked). Whenever the user changes from one mode to another (except when returning to mode 0, as that is the home mode), these configurations are stored so that when the back-navigation is pressed (or when a quiz is saved and exited, which gives the same effect), the user can return to them.

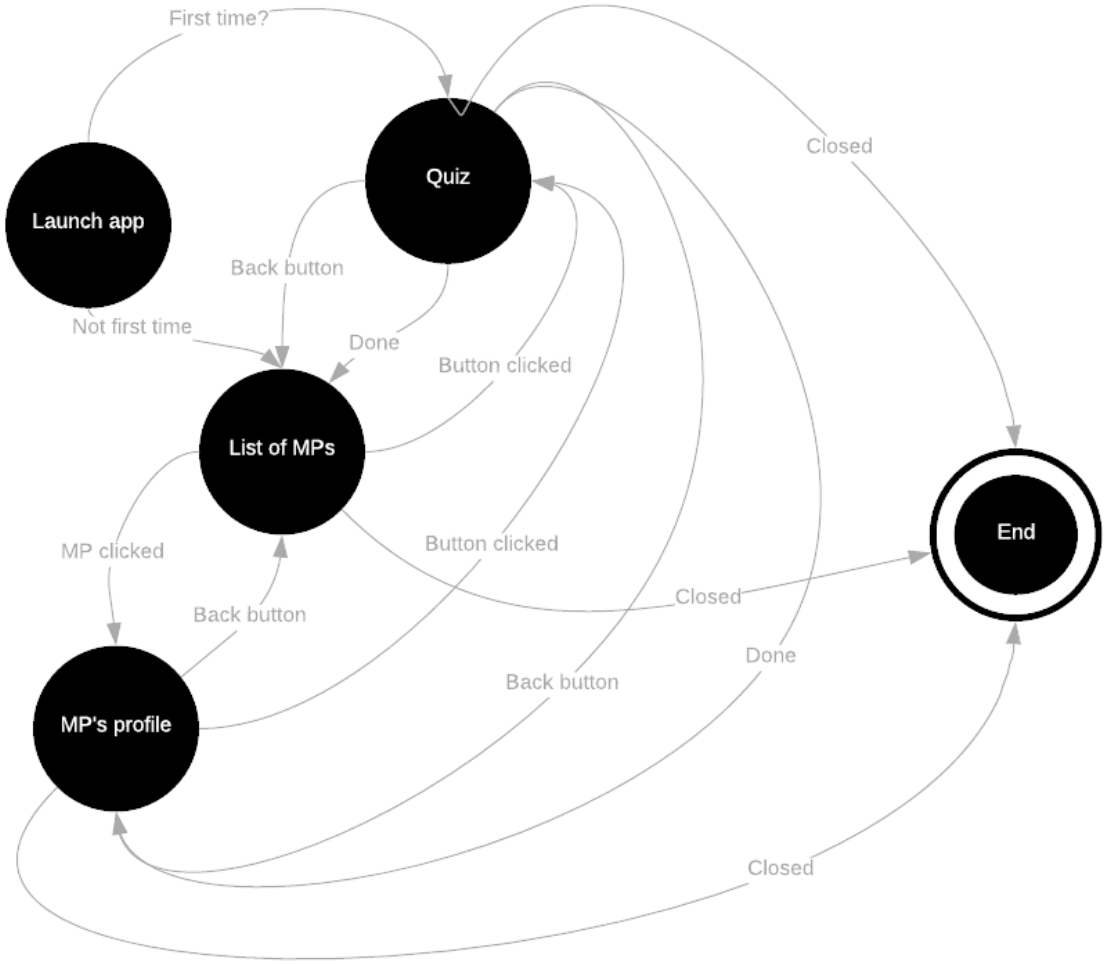
Stack item	mode	sort	filter	position
0	0	5	3	475
1	1	1	1	23

A maximum of 2 items will be on the stack, because only the following movements between 1 mode to another mode are possible:

- mode 0 -> mode 1 (1 item added)
    - mode 1 -> mode 0 (1 item removed)
- mode 0 -> mode -1 (1 item added)
    - mode -1 -> mode 0 (if quiz has been done before: 1 item removed; else no action, as stack empty)
- mode 1 -> mode -1 (1 item added)
    - mode -1 -> mode 1 (1 item removed)

As no new stacks can be accumulated from moving to another mode from mode -1 (as you can only go back to a previous mode from there), the maximum stack accumulation is: mode 0 -> mode 1 -> mode -1 (2 items added).

App processes



Error handling

For much of the functionality, and definitely the first time of usage pre-caching, the app will require a WiFi connection. Thus, an exception check will be needed to ensure it doesn't crash. In addition, measures must be put in place to deal with MPs who have no voting record, so as to prevent any parsing errors that may arise. The error handling must also be clear to the user, in the form of messages (like Android 'toasts') briefly describing why the app is not functioning as expected The primary choices before me would be try/catch statements and throw statements. Try/catch will be employed whereby the app can still function if the tried code fails, while throw will be employed where the app would have to crash either way.

Technical solution

Techniques used in the code

- **Graph/Tree Traversal** (in the heapsort)
- **List operations**
- **Stack operations** (in app navigation – priorStates stack)
- **Recursive algorithms** (in heapsort and binary search)
- **Complex user-defined algorithms**
- **Heapsort**
- **Dynamic generation of objects based on complex user-defined use of OOP model**
- **Using OOP concepts** such as **classes**, **encapsulation** (with respect to the Item and Socioeconomic objects, in the app and the breakdown program, respectively), **interfaces** (in RecyclerAdapter), **inheritance** and **composition** (between RecyclerAdapter and ItemHolder)
- **Calling and using Dropbox API** and **parsing and making JSON** to service a **client-server relationship** between the Android app and the program being run on the Heroku cloud platform
- **Binary search**
- **Reading and writing from files** (JSON files, policies text file, online HTML)
- Using **dictionaries** (to store quiz data) and **multi-dimensional arrays**

Folder and files structure layout

Only showing important files and folders

Android app:

root

- app
  - src
    - main
      - java
        - com.manasrawat.bigtent
          - activity
          - dataloader
          - dataretriever
          - heapsort
          - item
          - recycleradapter
      - res
        - color
          - changing.xml
        - drawable
          - compassbackdrop.png
          - ic_launcher_background.xml
        - menu
          - menu.xml
        - values
          - colors.xml
          - strings.xml
          - styles.xml
        - androidmanifest.xml
  - build.gradle
- gradle
  - build.gradle

JSON Breakdown and Creation Program:

Root

- src
  - main
    - java
      - Main
      - Socioeconomic
    - Policies.txt
- pom.xml

Android application

Activity.java

```
package com.manasrawat.bigtent;

import android.app.AlertDialog;
import android.content.*;
import android.graphics.Typeface;
import android.net.ConnectivityManager;
import android.net.Network;
import android.net.NetworkCapabilities;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.LoaderManager;
import android.support.v4.content.ContextCompat;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.text.SpannableString;
import android.text.style.ForegroundColorSpan;
import android.util.Log;
import android.view.*;
import android.widget.*;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.PointsGraphSeries;

import java.math.BigDecimal;
import java.math.MathContext;
import java.util.*;

public class Activity extends AppCompatActivity implements
LoaderManager.LoaderCallbacks<List<Item>>, SwipeRefreshLayout.OnRefreshListener,
AdapterView.OnItemSelectedListener, RecyclerAdapter.ItemClickListener {

    private LoaderManager loaderManager; //manages loading and reloading of web-
retrieved content from DataRetrieval
    private Context context; //interface enabling access to app-specific resources

    private boolean permsToLoad; //used to ensure unauthorised data (re)loading
doesn't occur

    /*Declarations (and in some cases, initialisations) of global variables*/

    //Toolbar views declarations
    private Spinner spinner; //used to sort the list of data presented in the
RecyclerView
    private TextView title; //if on quiz or listing MPs, not visible; name of MP
if showing one's voting record
    private Button button1, quizButton, resetButton; //used to go to quiz
    // (or graphically display quiz result if already on quiz)
    // and to display MP's political position alongside user's quiz derived one
```

```
and MP's party's one, also graphically

    //Other main activity views declared
    private SwipeRefreshLayout refresher; //Used to refresh the recyclerView and
its data by pulling down
    private View loadingIndicator; //loading circle, displayed when app started up
and moving between different modes
    // (and thus datasets)
    private Snackbar snack = null; //used to make internet connection error
message if no content loaded

    //Graph views
    private View graphLayout, closestMPsLayout; //the UI for the
    private RelativeLayout coordinates; //List of TextView displaying the
(economic, social) positions of the MP,
    // their party and the User. MP and Party ones only shown when on an
individual MP's page.
    // User one shown on both that and the quiz
    private GraphView graphView; //the user interface for the political compass
graph

    private LinearLayout closestMPsBase; //base layout for dialog showing
politically-closest MPs to user
    private LinearLayout[] closestMPsSections = new LinearLayout[2]; //one for
economically closest, other for socially

    //Accessories
    public static SharedPreferences sharedPreferences; //used to store and save
data even after app closed
    public static List<String> categories = new ArrayList<>(); //to be accumulated
with the distinct parties
    // represented in parliament
    private LinearLayoutManager layoutManager; //manages recyclerView
    private Menu optionsMenu; //shows different options to filter the current data
    private RecyclerAdapter adapter; //adapts the recyclerView
    private ArrayAdapter<String> sortAdapter; //for the filter options menu
    private List<Item> loadedData = new ArrayList<>(), //data loaded from DropBox
for use in mode,
            displayData = new ArrayList<>(); //data being displayed (depends on
sort and filter)

    //Numerical
    public static int selected = 0, //filter option
            sortBy = 0, //sort option
            mode, //quiz = -1; list of MPs = 0; MP's voting record = 1
            black = android.R.color.black;
    private int gotoPos = 0, //position of data item that user clicked/was on when
they changed mode
            accent = R.color.colorAccent;
    public static double partyEcon = Double.NaN, partySoc = Double.NaN;
//socioeconomic position of the party
    //of the mode-1 MP whose page the user is on
    public static double memberEcon, memberSoc; //socioeconomic position the mode-
1 MP whose page the user is on

    //0=item position,1=econ,2=soc; to be stored in SharedPreferences
    public static HashMap<String, Integer>[] quizChoices = new HashMap[3];
    private Stack<int[]> priorStates = new Stack(); //accumulates the filter,
sort, data item index and mode
```

```
    // the user was last on prior to mode changes; so that they can be
sequentially restored via back button press
    private HashMap<Integer, int[]> values; //dictionary of ASCII characters (key)
and their corresponding 7-bit
    // binary values

    //Misc
    public static String url, party, code; //url=ID of current mode-1 MP; party =
their party;
    // code is to access the Dropbox folder hosting the JSON data needed for all
of the different modes
    public static ObjectMapper objectMapper = new ObjectMapper(); //
    // Jackson JSON API class used to read, write, serialise and traverse JSON
data
    private AlertDialog dialog; //for displaying closest MPs to user politically
(mode 0)
    // or MP's political positions (mode 1)
    private List<String> sortOptions = new ArrayList<>(Arrays.asList("Sort by
surname",
            "Sort by first name", "Sort by constituency",
            "Sort by economic", "Sort by social", "Sort by party"));

    private boolean showClosestWithoutPrompt = false; //on whether to show closest
MPs after quiz completion

    private Toast toaster; //stores latest toast, to dismiss it later if needs be

    private LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT); //height and width for
dynamically generated views

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); //set XML UI
        context = getApplicationContext();
        sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);
        establishViews();
        createASCIIBinaries();

        if (firstTime()) { //app first run
            //code's value
            code = "CODE GOES HERE";
            mode = -1; //quiz mode
        } else {
            String key = sharedPreferences.getString("key", ""), //get the key
                    cipher = sharedPreferences.getString("cipher", ""); //get the
encrypted code
            code = vernamCipher(cipher, key); //decode the code using the key
            mode = 0; //list of MPs
        }
        //generate new key and encrypted code (ciphertext), and save to access
next time app opened
        String key = newKey(),
                cipher = vernamCipher(code, key);
        sharedPreferences.edit().putString("key", key).putString("cipher",
cipher).apply();
```

```
        new DataRetriever(); //connect to DropBox
        loaderManager = getSupportLoaderManager();
        loadNewContent();
    }

    private boolean firstTime() {
        return sharedPreferences.getBoolean("firstTime", true);
    }

    private HashMap<String, Integer> savedChoices(int i) {
        try {
            //retrieve the respective dictionaries' values from their JSON-saved
string form in sharedPreferences
            // - if they exist
            return objectMapper.readValue(sharedPreferences.getString("pos" + i,
""),
                    HashMap.class);
        } catch (JsonProcessingException e) { //mapping or processing error with
object mapper
            Log.i("Error retrieving hash " + i, e.getMessage());
            return new HashMap<>();
        }
    }

    //checks if connected to a network (WiFi/Mobile Data), and if that network has
a working internet connection
    // (so networks requiring a web sign-in that aren't signed-in-to aren't
falsely equated to an internet connection)
    private boolean internetConnected() {
        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        Network activeNetwork = connectivityManager.getActiveNetwork();
        NetworkCapabilities capabilities =
connectivityManager.getNetworkCapabilities(activeNetwork);

        //true if a network is connected to and it is validated (web-signed-in if
needs be)
        return capabilities != null &&
capabilities.hasCapability(NetworkCapabilities.NET_CAPABILITY_VALIDATED);
    }

    private AlertDialog.Builder graphDialogBuilder, closestMPsDialogBuilder;
    private ActionBar actionBar;
    private ScrollView closestMPsScroll;

    //Initialise views (UI elements) and their adapters
    private void establishViews() {
        Toolbar toolbar = findViewById(R.id.toolbar); //XML initialisation of
toolbar
        setSupportActionBar(toolbar); //set as app's main toolbar
        actionBar = getSupportActionBar();
        actionBar.setTitle(null); //remove default title to enable title TextView
and spinner to take the place
        toolbar.setNavigationOnClickListener(this::goBack); //set back button
functionality

        params.setMargins(0, 5, 0, 0);

        title = findViewById(R.id.title);
```

```
        graphLayout = getLayoutInflater().inflate(R.layout.graph, null); //set to
the XML file
        coordinates = graphLayout.findViewById(R.id.pinpoint); //value is a view
from the secondary layout

        closestMPsLayout = getLayoutInflater().inflate(R.layout.closest, null);
//similarly set to an XML file
        closestMPsBase = closestMPsLayout.findViewById(R.id.base); //again, value
= view from a secondary layout
        closestMPsScroll = closestMPsLayout.findViewById(R.id.closestScroll);
//scroll view for closest MPs list,
        //which becomes necessary in case of too many MPs being equally
economically/socially close to the user

        //set value of child linear layouts to hold the economically and socially
closest MPs.
        // 2 multiplied by the iterator + 1 used to skip non-linear layout child
views
        for (int i = 0; i < 2; i++) closestMPsSections[i] = (LinearLayout)
closestMPsBase.getChildAt(2 * (i + 1));

        //initialise and set core properties of both dialog builders
        graphDialogBuilder = new
AlertDialog.Builder(Activity.this).setView(graphLayout);
        closestMPsDialogBuilder = new
AlertDialog.Builder(this).setView(closestMPsLayout)
                .setPositiveButton("OK", null);

        loadingIndicator = findViewById(R.id.loading_indicator);

        //Spinners and buttons
        spinner = findViewById(R.id.sortBySpinner);

        button1 = findViewById(R.id.button1);
        button1.setOnClickListener(this::button1Click); //set on click
functionality, which varies between the modes

        quizButton = findViewById(R.id.quizButton);
        quizButton.setOnClickListener(this::quizButtonClick); //likewise, click
functionality set

        resetButton = findViewById(R.id.resetButton);
        resetButton.setOnClickListener(this::resetButtonClick);

        //RecyclerView
        //used to display the current mode's data; recyclerView status enables the
        //recycler view item holders to be recycled, thus saving space and making
the processing faster
        RecyclerView recyclerView = findViewById(R.id.recycler_view);
        recyclerView.setHasFixedSize(true); //prevents scroll disruption
        layoutManager = new LinearLayoutManager(this); //manages RecyclerView
layout
        recyclerView.setLayoutManager(layoutManager);
        adapter = new RecyclerAdapter(this, new ArrayList<>()); //manages
RecyclerView list items
        recyclerView.setAdapter(adapter);

        //spinner
        sortAdapter = new ArrayAdapter<>(this, R.layout.sort_spinner_layout,
```

```
sortOptions);
        spinner.setAdapter(sortAdapter); //manages Spinner items
        spinner.setOnItemSelectedListener(this); //activity implements spinner
selection listener already

        //refresher
        refresher = findViewById(R.id.refresh);
        refresher.setOnRefreshListener(this); //activity implements refresh
listener already

        //graph
        graphView = graphLayout.findViewById(R.id.gv); //initialise as per XML
        drawGraph();
    }

    //Set and define properties of application toolbar
    //set on-click functionality of the toolbar's back button
    private void goBack(View view) {
        int[] prior = priorStates.pop(); //access and remove from stack the last-
in item

        mode = prior[0]; //restore mode value to prior mode
        sortBy = prior[1]; //restore sort mode to prior one
        selected = prior[2]; //restore filter to prior one
        gotoPos = prior[3]; //to go to last position user was on when they had
exited the mode they're now returning to

        loadNewContent(); //to prepare app for the brief intermission
        // before a new mode and its respective dataset is loaded
    }

    private void drawGraph() {
        graphView.bringToFront(); //so backdrop doesn't block the graph

        //set X-axis range
        graphView.getViewport().setXAxisBoundsManual(true); //fix size
        //cap at limits
        graphView.getViewport().setMinX(-10);
        graphView.getViewport().setMaxX(10);

        //set Y-axis range
        graphView.getViewport().setYAxisBoundsManual(true); //fix size
        //cap at limits
        graphView.getViewport().setMinY(-10);
        graphView.getViewport().setMaxY(10);

        //set the X and Y spacing (21 as also accounting for the origin)
        graphView.getGridLabelRenderer().setNumHorizontalLabels(21);
        graphView.getGridLabelRenderer().setNumVerticalLabels(21);

        //hide labelling to maintain the principles of the Political Compass
        graphView.getGridLabelRenderer().setVerticalLabelsVisible(false);
        graphView.getGridLabelRenderer().setHorizontalLabelsVisible(false);

        //transparent in order to enable the backdrop 4-colour, 4-quadrant compass
background

graphView.setBackgroundColor(getResources().getColor(android.R.color.transparent))
;
```

```
    }

    private void createASCIIBinaries() {
        values = new HashMap<>();
        //set binary values of the first 2 ASCII values (starting with only
0000000 causes the algorithm to fail)
        values.put(0, new int[]{0, 0, 0, 0, 0, 0, 0});
        values.put(1, new int[]{0, 0, 0, 0, 0, 0, 1});

        for (int DEC = 2; DEC < 128; DEC++) { //iterate through remaining ASCII
characters
            int increment = 1, i = 6, carryover = 0;
            //increment = how much prior ASCII value to be increased by (1 binary
valuation)
            //i=6=index of last bit of 7-binary-digit number (for later backwards-
iteration)
            int[] prev = values.get(DEC - 1); //get binary value of previous (1-
less) character
            assert prev != null;
            int prevInt = 0, last = prev.length - 1;
            for (int j = last; j >= 0; j--) { //start at the last bit, iterating
up the digits
                int bit = prev[j]; //get current bit
                if (bit == 1) {
                    for (int k = 0; k < last - j; k++) bit *= 10; //turn bit into
multiple of 10 as per  digit location
                    prevInt += bit; //add to overall bit
                }
            }

            int[] binary = new int[7];
            for (; prevInt != 0; prevInt /= 10) { //iterate (in backwards) with a
step of diving the integer by 10
                // each time
                //set binary digit
                int val = prevInt % 10 + increment % 10 + carryover;
                binary[i--] = val % 2;
                carryover = val / 2; //for 1+1 and 1+1+1 addition
                increment /= 10; //move along binary digits
            }
            if (carryover != 0) binary[i--] = carryover; //carrying over to first
digit
            values.put(DEC, binary); //add character's binary value to dictionary
        }
    }

    //create new randomised key
    private String newKey() {
        Random random = new Random();
        StringBuilder key = new StringBuilder();
        //iterate code to build up randomised key of same length
        //draws from same pool of indexes (0 to 127) corresponding to all the
ASCII characters
        for (int i = 0; i < code.length(); i++) key.append((char)
random.nextInt(128));
        return key.toString();
    }

    private String vernamCipher(String code, String key) {
```

```
        StringBuilder result = new StringBuilder();

        //to deal with problem whereby the code or key's last character being a
space resulting in 4 additional
        // space characters
        if (code.length() > key.length()) code = code.substring(0, key.length());
        else if (key.length() > code.length()) key = key.substring(0,
code.length());

        //iterate through code's characters (alongside key's)
        for (int i = 0; i < code.length(); i++) {
            int[] codeChar = values.get((int) code.charAt(i)), //get binary value
of i-th character of code
                    keyChar = values.get((int) key.charAt(i)), //get binary value
of i-th character of key
                    cipherChar = new int[7]; //array to hold the binary value of
the resultant cipher's i-th character
            assert codeChar != null;
            for (int j = 0; j < codeChar.length; j++) { //iterate through code's
i-th character's binary value's
                // 7 digits
                assert keyChar != null;
                //set cipher digit to 1 if XOR conditions met; else 0
                if (codeChar[j] == 1 && keyChar[j] == 0 || codeChar[j] == 0 &&
keyChar[j] == 1) cipherChar[j] = 1;
                else cipherChar[j] = 0;
            }

            //iterate through ASCII character dictionary
            for (HashMap.Entry<Integer, int[]> entry : values.entrySet()) {
                //find the cipher's i-th ASCII character from its corresponding
binary value
                if (Arrays.equals(cipherChar, entry.getValue())) {
                    int oneChar = entry.getKey(); //get the decimal value
                    result.append((char) oneChar); //add to StringBuilder after
converting from decimal to char form
                    break; //halt to avoid unnecessary further iterations when
objective achieved
                }
            }
        }
        return result.toString();
    }

    @Override
    public android.support.v4.content.Loader<List<Item>> onCreateLoader(int id,
Bundle args) {
        return new DataLoader(this); //set loader functionality
    }

    //on completion of loader
    @Override
    public void onLoadFinished(@NonNull
android.support.v4.content.Loader<List<Item>> loader, List<Item> data) {
        if (permsToLoad) { //if authorised
            permsToLoad = false; //revoke
            if (!refresher.isEnabled()) refresher.setEnabled(true);
            refresher.setRefreshing(false); //stop the refresher
            setButtonsState(true); //restore the buttons' functions
```

```
            loadingIndicator.setVisibility(View.GONE); //indicates that loading
done

            loadedData = data; //set new data
            if (!loadedData.isEmpty()) { //ensure there's content to worth with
                populateFilter(); //create filters
                filterDisplayedData(); //filter data
                updateAdapter(); //restore prior filter modes and RecyclerView
index position, if recorded
                graphView.removeAllSeries(); //remove any old coordinates that
were on the graph
                set(optionsMenu.getItem(selected), accent); //highlight current
filter
                if (mode == 0) showClosestMPs();
                else makeGraph();
            } else if (!internetConnected()) noWiFi();
            else makeToast("Could not load data");
        }
    }

    //filter data
    private void filterDisplayedData() {
        //if showing all data, regardless of category
        if (!displayData.isEmpty()) displayData.clear();
        if (selected == 0) displayData.addAll(loadedData);
            //if showing specific category's data,
            //for each item in the dataset, add to category data list if it comes
under that category
        else for (Item item : loadedData)
            if (categories.get(selected - 1).equals(item.getCategory()))
displayData.add(item);
    }

    //called when a previously created loader is being reset, and thus making its
data unavailable.
    @Override
    public void onLoaderReset(@NonNull
android.support.v4.content.Loader<List<Item>> loader) {
        adapter.clear();
        Log.i("RESET", "Loader reset");
    }

    //when new data loaded, to find, visually construct and show the closest MPs
to the user
    private void showClosestMPs() {
        //Show user's position, for comparision
        ((TextView) closestMPsBase.getChildAt(0)).setText("Most similar to you ("
+
                round(toDouble("economic")) + "," + round(toDouble("social")) +
")");
        findClosestMPs();
        //reset layout of previous configurations
        if (closestMPsLayout.getParent() != null)
            ((ViewGroup)
closestMPsLayout.getParent()).removeView(closestMPsLayout);
        dialog = closestMPsDialogBuilder.create(); //define dialog's current use
as being to show the closest MPs
        if (showClosestWithoutPrompt) { //when exiting the quiz into mode 0 (list
of MPs)
```

```
            showClosestWithoutPrompt = false; //so this isn't repeated anytime
mode 0 is loaded
            dialog.show();
        }
    }

    private List<Item>[] closestSetsOfMPs;

    //traverses through data
    private void findClosestMPs() {
        closestMPsScroll.scrollTo(0, 0);
        closestSetsOfMPs = (ArrayList<Item>[]) new ArrayList[2]; //0=economically,
1=socially
        double[] userVals = {toDouble("economic"), toDouble("social")}; //get
user' position into an array

        for (int i = 0; i < 2; i++) { //for both the economic and social lists
            List<Item> closestMPs = new ArrayList<>();
            closestMPs.add(displayData.get(0)); //add the first MP
            closestSetsOfMPs[i] = closestMPs; //set the value of the
economic/social list
        }

        for (int i = 1; i < displayData.size(); i++) { //iterate through the rest
of the MPs
            Item member = displayData.get(i); //get the current MP
            if (!member.getCategory().equals("Sinn Féin")) { //ensure they aren't
abstentionist
                for (int j = 0; j < closestSetsOfMPs.length; j++) { //look from an
economic lens first,
                    // then from a social one
                    List<Item> closestOne = closestSetsOfMPs[j]; //get list

                    //find the difference between the user's position and the
MP's,
                    //and the small difference between the user's position and
MP(s) iterated past so far
                    //ternary operator used in accordance with iterator to ensure
the difference is found with the right
                    //property (economic value if iterator j = 0, else social
value)
                    double diff = userVals[j] - (j == 0 ? member.getEcon() :
member.getSoc()),
                            smallestDiff = userVals[j] - (j == 0 ?
closestOne.get(0).getEcon() :
                                    closestOne.get(0).getSoc());
                    if (diff < 0) diff *= -1; //get difference as modulus/absolute
value
                    if (smallestDiff < 0) smallestDiff *= -1; //likewise
                    if (diff <= smallestDiff) { //if MP is closer to the user's
position than
                        // the previously closest MP(s) or is as close, add them
to the economic/social list
                        if (diff < smallestDiff) closestOne.clear(); //remove all
previously-closest MPs if newly
                        //closest MP is closer than them
                        closestOne.add(member); //add closest MP
                    }
                }
```

```
            }
        }
        for (int i = 0; i < 2; i++) populateDialogOfClosestMPs(i); //for loop for
both economic and social sets
    }

    //0=econ;1=soc; visually updates
    private void populateDialogOfClosestMPs(int i) {
        closestMPsSections[i].removeAllViews(); //clear
        for (Item MP : closestSetsOfMPs[i]) { //iterate through closest MPs
            TextView MPView = new TextView(this);

            //MP first name + MP last name + MP economic position + MP social
position set as text
            MPView.setText(MP.getHeadingMain() + " " + MP.getHeadingExtra() +
                    " (" + round(MP.getEcon()) + "," + round(MP.getSoc()) + ")");

            //make text bold, and make it change colour when pressed and clicked
            MPView.setTypeface(Typeface.DEFAULT_BOLD);

MPView.setTextColor(getResources().getColorStateList(R.color.changing));

            MPView.setOnClickListener(v -> { //on click
                int position;
                if (sortBy == 3 || sortBy == 4) { //if sorting economically or
socially
                    //find position of MP by binary search
                    position = search(MP, 0, displayData.size() - 1, 0);

                    //if the unique constituencies (details) don't match with
actual MP at that position, this means
                    //that there are multiple MPs with the same economic/social
value
                    if
(!displayData.get(position).getDetail().equals(MP.getDetail())) {
                        //look below the found MP in the list
                        int findingPosDown = traverse(position, MP, true);
                        //if not found below, look above
                        if (findingPosDown == -1) position = traverse(position,
MP, false);
                        else position = findingPosDown; //else if found below, set
as such
                    }
                } else position = search(MP, 0, displayData.size() - 1, 1); //if
sorting by string value
                //i.e. by first name/last name/party/seat (all of which are multi-
criteria, except by unique-seat/
                // constituency)
                //check if not already in position
                if (layoutManager.findFirstCompletelyVisibleItemPosition() !=
position)
                    layoutManager.scrollToPositionWithOffset(position, 0);
//scroll to position
            });
            MPView.setLayoutParams(params); //add margins, height and width to
text view
            closestMPsSections[i].addView(MPView); //add text view to the
economic/social layout
        }
```

```
    }

    //Binary Search
    private int search(Item MPToFind, int first, int last, int type) {
        if (last >= first) { //ensure in range
            int centre = first + (last - first) / 2; //get middle value
            if (type == 0) { //numerical binary search
                //get relevant property of currently-in-the-middle-MP and the MP
to find
                double toFind = sortBy == 3 ? MPToFind.getEcon() :
MPToFind.getSoc();
                double comparator = sortBy == 3 ?
displayData.get(centre).getEcon() : displayData.get(centre).getSoc();

                if (toFind == comparator) return centre; //found MP
                    //MP above centre; perform recursion in upper half of section
already searched through
                else if (toFind > comparator) return search(MPToFind, centre + 1,
last, type);
                    //MP below centre; perform recursion in lower half of section
already searched through
                else if (toFind < comparator) return search(MPToFind, first,
centre - 1, type);
            } else { //string-based binary search
                //same principles as numerical one
                String toFind = MPToFind.getSorter(sortBy).toLowerCase();
                String comparator =
displayData.get(centre).getSorter(sortBy).toLowerCase();

                if (toFind.equals(comparator)) return centre;
                else if (toFind.compareTo(comparator) > 0) return search(MPToFind,
centre + 1, last, type);
                else if (toFind.compareTo(comparator) < 0) return search(MPToFind,
first, centre - 1, type);
            }
        }
        return -1;
    }

    //to find the MP if there are multiple MPs with same value as them
    //as the BinarySearch was done on sorted data, the MPs of the same value are
consecutively located
    //so traversal need only be immediately either side of the found-but-not-
wanted MP
    private int traverse(int j, Item MPToFind, boolean moveBelow) {
        double comparator = sortBy == 3 ? MPToFind.getEcon() : MPToFind.getSoc();
//relevant property
        while (moveBelow ? j >= 0 : j < displayData.size()) { //respective
conditions for whether traversing below
            //or above the first-instance-MP
            Item currentMP = displayData.get(j); //get the currently-iterated MP
(current MP)
            //if value matches the current MP's
            if ((sortBy == 3 ? currentMP.getEcon() : currentMP.getSoc()) ==
comparator) {
                //if unique constituencies of the current MP and the MP to find
match, the index position has been found
                if (currentMP.getDetail().equals(MPToFind.getDetail())) return j;
            } else break; //means MP to find not above/below first instance MP,
```

```
but below/above respectively
            //iterator changed according to whether the traversal is below or
above the first instance MP
            if (moveBelow) j--;
            else j++;
        }
        return -1; //not found
    }

    //convert into double from saved long state
    private double toDouble(String id) {
        return Double.longBitsToDouble(sharedPreferences.getLong(id,
Double.doubleToLongBits(0.0)));
    }

    //round the double to 3 significant figures
    public static String round(double val) {
        return new BigDecimal(val).round(new MathContext(3))
                .stripTrailingZeros().toPlainString();
    }

    //when refreshing
    @Override
    public void onRefresh() {
        setButtonsState(false);
        if (internetConnected()) {
            refresher.setRefreshing(true); //refresh throbber visible and circling
            permsToLoad = true; //authorise load
            loaderManager.restartLoader(0, null, this);
            Log.i("Refresh", "Refreshed");
        } else {
            refresher.setRefreshing(false); //stop the refresher
            setButtonsState(true);
            noWiFi();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        optionsMenu = menu; //global variable defined
        getMenuInflater().inflate(R.menu.menu, menu); //XML layout attached
        return super.onCreateOptionsMenu(menu);
    }

    //visually update data in UI
    private void updateAdapter() {
        if (adapter.getItemCount() != 0) adapter.clear(); //clear recycler view
        //check if position in data range, in case of MP resignation update on the
DropBox-hosted end
        layoutManager.scrollToPositionWithOffset(
                gotoPos < displayData.size() ? gotoPos : displayData.size() - 1,
0);
        gotoPos = 0; //reset
        adapter.addAll(displayData); //visually updated
    }

    //fill options menu with categories
    private void populateFilter() {
        List<Item> localData = new ArrayList<>(loadedData);
```

```
        optionsMenu.clear(); //reset
        optionsMenu.add(0, Menu.FIRST, Menu.NONE, "Show all"); //first item,
default; shows items of
        //all categories

        if (mode != 1 && sortBy != 5) new HeapSort(localData, 5); //categorically
sort the dataset
        //sortMisc(toFilter); //move misc data to end
        if (!categories.isEmpty()) categories.clear(); //reset categories list
        int counter = 0; //for assigning ID to categories

        for (Item member : localData) { //iterate through dataset
            String category = member.getCategory();
            if (!categories.contains(category)) { //if not included in category
list already
                categories.add(category);
                counter++; //increment category ID counter
                optionsMenu.add(0, Menu.FIRST + counter, Menu.NONE, category);
//add category to filter
                // menu
            }
        }
    }

    //when filtering
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int pos = item.getItemId() - 1; //subtraction so it doesn't count show all
        if (pos != selected) { //ensure the current filter mode isn't being
clicked, to avoid unnecessarily filtering
            set(optionsMenu.getItem(selected), black); //unselect previous filter
text
            selected = pos; //universally set new filter mode
            set(optionsMenu.getItem(selected), accent); //select newly-selected
filter text
            if (loadedData.size() != 1) filterDisplayedData(); //filter data (no
need to if only 1 data item)
            updateAdapter(); //filter dataset itself
            if (mode == 0) { //when listing MPs
                findClosestMPs();
                //if filter mode not to show all (i.e. user is filtering by
party), remove the party sort from
                //the sorter. if filter mode changed to show all, restore party
sort
                if (sortAdapter.getCount() == 6 && selected != 0)
sortAdapter.remove("Sort by party");
                else if (sortAdapter.getCount() == 5 && selected == 0)
sortAdapter.add("Sort by party");
            } else if (mode == -1) { //quiz
                if (selected == 0) {
                    button1.setText("CLEAR ALL"); //to clear all policies' options
                    resetButton.setText("RESET ALL");
                } else {
                    button1.setText("CLEAR"); //to clear the options of the
policies of the shown topic
                    resetButton.setText("RESET");
                }
            }
        }
```

```
        return super.onOptionsItemSelected(item);
    }

    //to set text colour of an options menu item
    private void set(MenuItem item, int color) {
        SpannableString ss = new SpannableString(item.getTitle().toString());
//get text in markup-compatible form
        //reflect colour onto spannable string
        ss.setSpan(new ForegroundColorSpan(ContextCompat.getColor(context,
color)), 0, ss.length(), 0);
        item.setTitle(ss); //change text colour of item itself by setting the
label to it
    }

    //when sorting (mode 0)
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {
        if (sortBy != position) { //don't execute code unnecessarily if already on
selected sort
            sortBy = position; //universally set new sort mode
            new HeapSort(loadedData, sortBy); //sort
            if (displayData.size() > 1) { //ensure it's actually worthwhile to
sort (1 data item needs no sorting)
                filterDisplayedData();
                updateAdapter(); //get sorted data

                //to sort list of closest MPs
                for (int i = 0; i < 2; i++) { //first economically, then socially
                    //no point resorting if only one MP (already accounted for in
HeapSort code too)
                    new HeapSort(closestSetsOfMPs[i], sortBy); //sort list of
economically/socially closest MPs
                    if (closestSetsOfMPs[i].size() != 1)
populateDialogOfClosestMPs(i); //reflect in dialog UI
                }
            }
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }

    //when clicking on item to launch selected MP's record
    @Override
    public void onItemClick(View view, int position) {

        if (!refresher.isRefreshing()) { //to prevent app from crashing

            if (displayData.get(position).isActive()) { //ensure MP has voted
                //push current configurations to stack
                priorStates.push(new int[]{mode, sortBy, selected, position});
                mode = 1;
                sortBy = 5;
                selected = 0;
                gotoPos = 0;

                //set toolbar title to MP's name
```

```
                title.setText(
                        displayData.get(position).getHeadingMain() + " " +
displayData.get(position).getHeadingExtra());
                //set cross-class party and ID of MP
                party = displayData.get(position).getCategory();
                url = displayData.get(position).getUrl();

                loadNewContent(); //prepare for new data to be loaded into
activity
            } else makeToast("Haven't voted");

        }
    }

    //prep for the intermission before the new data is loaded
    private void loadNewContent() {
        permsToLoad = true; //authorise load
        if (!displayData.isEmpty()) displayData.clear(); //to prevent app crash
from changing sort during load

        if (loaderManager.hasRunningLoaders()) loaderManager.destroyLoader(0);
//if another mode is being loaded,
        // stop it to prevent the loader from collapsing in on itself

        if (refresher.isRefreshing()) refresher.setRefreshing(false); //prevent
data from being loaded twice when the
        //load completes, which would lead to the recyclerView being populated
with the dataset twice
        refresher.setEnabled(false); //also to prevent the loader from collapsing
in on itself
        setButtonsState(false);
        //hide dialog box's User/Party/MP position TextViews via iteration
        for (int i = 0; i < coordinates.getChildCount(); i++)
            coordinates.getChildAt(i).setVisibility(View.GONE);

        //bar title only needed to show the name of the MP whose record is on
display
        title.setVisibility(mode == 1 ? View.VISIBLE : View.GONE);

        //clear and reset the following views to avoid the previous mode's
configurations from being shown
        // during the brief loading-transition period to the new mode (also to
prevent errors via clicking)
        if (optionsMenu != null) {
            optionsMenu.clear();
            optionsMenu.add(0, Menu.FIRST, Menu.NONE, "");
            optionsMenu.getItem(0).setEnabled(false); //can't be clicked
        }

        loadingIndicator.setVisibility(View.VISIBLE); //show the data is being
loaded
        adapter.clear(); //clear current data

        if (mode == -1) { //quiz
            button1.setText("CLEAR ALL");
            quizButton.setText("FIND");
            resetButton.setVisibility(View.VISIBLE);

            //if (!appPaused) {
```

```
            for (int i = 0; i < 3; i++) {
                quizChoices[i] = new HashMap<>(); //reset displayed choices data
                //if quiz done before
                if (sharedPreferences.contains("pos" + i))
quizChoices[i].putAll(savedChoices(i));
            }
            //}
        } else { //list of MPs and MP page
            button1.setText("•");
            quizButton.setText("QUIZ");
            resetButton.setVisibility(View.GONE);
        }

        //when on mode 0, or app opened for first time
        if (priorStates.isEmpty()) {
            //hide back button navigation
            actionBar.setDisplayHomeAsUpEnabled(false);
            actionBar.setDisplayShowHomeEnabled(false);
        } else {
            //show back navigation
            actionBar.setDisplayHomeAsUpEnabled(true);
            actionBar.setDisplayShowHomeEnabled(true);
        }

        if (mode == 0) { //list of MPs
            //restore necessary functions
            adapter.setClickListener(this); //so clicking recycler view item
launches MP's page
            spinner.setVisibility(View.VISIBLE); //restore sort
        } else {
            //disable recycler view items from being clicked to prevent error
(only clickable on mode 0, to open
            // selected MP's record)
            adapter.setClickListener(null);
            spinner.setVisibility(View.GONE); //spinner only relevant in mode 0/MP
list
        }
        loaderManager.restartLoader(0, null, this); //reload the data
    }

    //prevent interference or hindrance when data being (re)loaded
    private void setButtonsState(boolean state) {
        //to prevent the operation from being carried out before the required data
is loaded,
        //which would lead to the wrong data being used or the app from crashing,
disable button click function
        //and in case button in pressed state before, reset to prevent discrepancy
with new status
        if (button1.isPressed()) button1.setPressed(false);
        button1.setClickable(state);
        if (quizButton.isPressed()) quizButton.setPressed(false);
        quizButton.setClickable(state);
        //button only relevant and shown in quiz mode
        if (mode == -1) {
            if (resetButton.isPressed()) resetButton.setPressed(false);
            resetButton.setClickable(state);
        }

        if (toaster != null) toaster.cancel(); //hide any toasts that may be
```

```
visible
        if (snack != null && snack.isShown()) snack.dismiss(); //hide any
snackbars that are visible
        if (dialog != null && dialog.isShowing()) dialog.dismiss(); //hide any
open dialogs
    }

    //to plot (economic, social) points on graph
    private void constructCompass(int color, double econ, double soc) {
        //series holding the point value coordinate
        PointsGraphSeries<DataPoint> series = new PointsGraphSeries<>(new
DataPoint[]{new DataPoint(econ, soc)});
        series.setColor(getResources().getColor(color)); //uses colour passed in
as argument
        series.setSize(15); //coordinate point size
        graphView.addSeries(series); //added to graph
    }

    //create graph dialog
    private void makeGraph() {
        //get user's positions from sharedPreferences
        double userEcon = toDouble("economic"), userSoc = toDouble("social");
        //store labels, colours, and coordinates for the positions of the MP,
their party and the user
        // (tho MP and party ones irrelevant in mode -1/quiz)
        String[] types = {"MP", "Party", "User"};
        int[] colours = {android.R.color.holo_red_dark,
android.R.color.holo_orange_dark, black};
        double[][] points = {{memberEcon, memberSoc}, {partyEcon, partySoc},
{userEcon, userSoc}};

        //if on quiz (-1), not MP page (1), start with 2 so that only the User
position is set and visible
        for (int i = mode == -1 ? 2 : 0; i < coordinates.getChildCount(); i++) {
            if (!Double.isNaN(points[i][0])) { //for mode 1 in case of speaker,
independents and 1-MP parties
                TextView current = (TextView) coordinates.getChildAt(i); //get
relevant field (MP/party/user)
                //set field coordinate point in text
                current.setText(types[i] + ": (" + round(points[i][0]) + "," +
round(points[i][1]) + ")");
                //show field
                current.setVisibility(View.VISIBLE);
                //set field coordinate point in graph
                constructCompass(colours[i], points[i][0], points[i][1]);
            }
        }
        //prevent removeView error
        if (graphLayout.getParent() != null) ((ViewGroup)
graphLayout.getParent()).removeView(graphLayout);
        //if on quiz, save quiz progress and exit; else (if on MP page), just
dismiss dialog (null)
        graphDialogBuilder.setPositiveButton(mode == -1 ? "DONE" : "OK", mode ==
-1 ?
                (DialogInterface.OnClickListener) (dialog, which) ->
saveAndEndQuiz() : null);
        //only show redo button (simply dismisses dialog) in quiz
        if (mode == -1) {
            graphDialogBuilder.setNeutralButton("CHANGE", null);
```

```
            graphDialogBuilder.setNegativeButton("SAVE", (dialog, which) ->
saveQuiz());
        }
        dialog = graphDialogBuilder.create(); //create graph-UI dialog
    }

    private void button1Click(View view) {
        if (!loadedData.isEmpty()) { //if there's data to worth with
            if (mode == 0 &&
optionsMenu.getItem(selected).getTitle().toString().equals("Sinn Féin"))
                makeToast("N/A"); //Sinn Fein have never sworn in as MPs, so have
never turned up to even not vote
            else if (mode == -1) clearOrResetQuiz(false); //if on quiz mode
            else {
                if (mode == 0) closestMPsScroll.scrollTo(0, 0); //go back to top
of list
                dialog.show(); //rest of the time on MP list mode 0, and mode 1
specific MP page
            }
        } else if (!internetConnected()) noWiFi();
        else makeToast("Could not load data"); //other error, can be caused when
dropbox-hosted json data is
        //being updated
    }

    private void resetButtonClick(View view) {
        if (!loadedData.isEmpty()) clearOrResetQuiz(true); //there's actually data
to revert
        else if (!internetConnected()) noWiFi();
        else makeToast("Could not load data"); //other error, can be caused when
DropBox-hosted json data is
        //being updated
    }

    double userEconPos, userSocPos; //user's position

    //to enter quiz or find quiz result
    private void quizButtonClick(View view) {
        if (mode == -1) { //if already on quiz (finding result)
            if (!loadedData.isEmpty()) { //if quiz data has loaded
                graphView.removeAllSeries(); //remove prior position

                userEconPos = 0;
                userSocPos = 0;

                //foreach loop finds total for economic and social questions
answered
                for (double econPos : quizChoices[1].values()) userEconPos +=
econPos;
                for (double socPos : quizChoices[2].values()) userSocPos +=
socPos;

                //after making sure at least a question has been answered (to
prevent 0/0 which is nullity),
                //divide by number of questions respectively answered to get
average, which equals user position
                //between 0 and 10 (inclusive) both economically and socially
                if (!quizChoices[1].isEmpty()) userEconPos /=
quizChoices[1].size();
```

```
                if (!quizChoices[2].isEmpty()) userSocPos /=
quizChoices[2].size();

                //display the economic and social positions in text
                ((TextView) coordinates.getChildAt(2)).setText("(" +
round(userEconPos) + "," + round(userSocPos) + ")");
                //graphically display said positions
                constructCompass(black, userEconPos, userSocPos);
                //show result of quiz
                dialog.show();
            } else if (!internetConnected()) noWiFi(); //no internet available
            else makeToast("Could not load data");
        } else { //if trying to launch quiz
            //index of recycler view item in focus (fully appears first below
toolbar)
            int firstPos = layoutManager.findFirstCompletelyVisibleItemPosition();
            //put current configurations in the last mode stack
            priorStates.push(new int[]{mode, sortBy, selected, firstPos});

            //prepare for new mode with new configurations
            mode = -1;
            sortBy = 0;
            selected = 0;
            gotoPos = 0;
            loadNewContent();
        }
    }

    //to clear the questions answered in the quiz or to restore the quiz to its
last saved state
    //if reset=true: restoring; else if false then clearing
    private void clearOrResetQuiz(boolean reset) {
        boolean adapterClear = false; //don't visually clear
        for (int i = 0; i < 3; i++) { //iterate through the 3 HashMaps (for
question/policy position, economic
            //value and social value )
            //if restoring quiz to last saved state, if user has done the quiz
and/or completed economic or social
            //questions, and the user has actually answered any of the questions
with an option
            if (reset && sharedPreferences.contains("pos" + i) &&
!savedChoices(i).isEmpty()) {
                if (!quizChoices[i].equals(savedChoices(i))) { //if the displayed
user=answered qui\ data
                    // differs from the saved one
                    if (selected == 0) { //when all policies on display
                        //clear all, if not already empty
                        if (!quizChoices[i].isEmpty()) quizChoices[i].clear();
                        //restore last-saved data to displayed data
                        quizChoices[i].putAll(savedChoices(i));
                    } else {
                        //iterate policies on display (of a specific category)
                        for (Item m : displayData) {
                            String policy = m.getHeadingMain(); //get policy to
identify hash map items
                            //remove answered question if it hasn't been saved
                            //but if answered question has been saved but it isn't
displayed as such in the current
                            //selection by the user, or if the user has answered
```

```
it differently in the current
                            //unsaved display, add it visually
                            if (!savedChoices(i).containsKey(policy) &&
quizChoices[i].containsKey(policy))
                                quizChoices[i].remove(policy);
                            else if (savedChoices(i).containsKey(policy) &&
(!quizChoices[i].containsKey(policy)
                                    ||
!quizChoices[i].get(policy).equals(savedChoices(i).get(policy))))
                                quizChoices[i].put(policy,
savedChoices(i).get(policy));
                        }
                    }
                    if (!adapterClear) adapterClear = true; //indicate that
something has changed
                }
            } else {
                if (!quizChoices[i].isEmpty()) { //check if any of the quiz has
actually been done
                    if (selected == 0) { //when all policies on display
                        //clear all and save as such to shared preferences
                        quizChoices[i].clear();
                    } else {
                        //iterate policies on display and remove them from the
HashMap
                        for (Item m : displayData)
quizChoices[i].remove(m.getHeadingMain());
                    }
                    if (!adapterClear) adapterClear = true; //indicate that
something has changed
                } else if (i == 0) break; //if first hash map (holds positions of
answered policies/questions)
                //is empty, then the rest are too, so no point iterating through
them; break loop
            }
        }
        //only update the layout if something has been changed, else executing the
code is redundant
        if (adapterClear) {
            adapter.clear(); //remove all items
            adapter.addAll(displayData); //add updated items dataset
        }
    }

    //save quiz
    private void saveQuiz() {
        //if first time app opened (and quiz being done), let the app know that
the first time is done
        //so next time app opened, it opens with list of MPs (mode 0), not mode -1
quiz
        if (firstTime()) sharedPreferences.edit().putBoolean("firstTime",
false).apply();

        //save position to app in bit form
        sharedPreferences.edit().putLong("economic",
Double.doubleToRawLongBits(userEconPos))
                .putLong("social", Double.doubleToRawLongBits(userSocPos))
                .apply();
```

```
        //iterate through all 3 hash maps
        for (int i = 0; i < 3; i++) {
            //if quiz hasn't been done before (or questions that are specifically
economic or social anyway)
            //or if the saved data doesn't match the unsaved data on display as
done by the user
            if (!sharedPreferences.contains("pos" + i) ||
!quizChoices[i].equals(savedChoices(i))) {
                try {
                    //update saved data as the user-done HashMap
                    sharedPreferences.edit()
                            .putString("pos" + i,
objectMapper.writeValueAsString(quizChoices[i]))
                            .apply();
                } catch (JsonProcessingException e) { //object mapper error
                    //write to app error has occurred
                    Log.i("Error setting hash " + i, e.getMessage());
                }
            }
        }
    }

    //when exiting quiz
    private void saveAndEndQuiz() {
        saveQuiz(); //auto-save progress
        if (priorStates.isEmpty()) { //when app first opened (user not having been
on prior modes)
            mode = 0; //defaults to list of MPs
            selected = 0; //default filter
        } else {
            //retrieve values of last stack item
            int[] prev = priorStates.pop();
            mode = prev[0];
            sortBy = prev[1];
            selected = prev[2];
            gotoPos = prev[3];
        }

        if (mode == 0) showClosestWithoutPrompt = true; //automatically show user
the closest MPs to them once
        //data in new mode has finished loading
        loadNewContent(); //prepare views and trigger mode change
    }

    //displays message for short period of time
    private void noWiFi() {
        if (loadedData.isEmpty()) { //if nothing to show user
            //snackbar error message that won't be disappear until dismissed by
user clicking
            snack = Snackbar.make(findViewById(android.R.id.content),
                    "No internet connection", Snackbar.LENGTH_INDEFINITE)
                    .setAction("REFRESH", view -> onRefresh()) //user clicking
refreshes page

.setActionTextColor(getResources().getColor(R.color.colorPrimary)); //text colour
            snack.show();
        } else makeToast("No internet connection"); //auto-disappears
    }
```

```
    //toast error message
    private void makeToast(String text) {
        toaster = Toast.makeText(this, text, Toast.LENGTH_SHORT);
        toaster.show();
    }
}
```

DataLoader.java

```
package com.manasrawat.bigtent;

import android.content.Context;
import android.support.v4.content.AsyncTaskLoader;

import java.util.List;

//loads data from DataRetriever
public class DataLoader extends AsyncTaskLoader<List<Item>> {

    //Constructor, passes passed application context to parent
    public DataLoader(Context context) {
        super(context);
    }

    @Override
    protected void onStartLoading() {
        forceLoad();
    }

    @Override
    public List<Item> loadInBackground() {
        return DataRetriever.getData(); //load data from DropBox
    }
}
```

DataRetriever.java

```
package com.manasrawat.bigtent;

import android.util.Log;
import com.dropbox.core.DbxException;
import com.dropbox.core.DbxRequestConfig;
import com.dropbox.core.v2.DbxClientV2;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.node.ArrayNode;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;

import static com.manasrawat.bigtent.Activity.*;

public class DataRetriever {
    //global variables declarations
```

```
    private static DbxClientV2 client;

    //constructor, called by Activity when class instantiated
    public DataRetriever() {
        //to connect to the DropBox folder
        DbxRequestConfig config = DbxRequestConfig.newBuilder("JSONBigTent/2.0")
                .withUserLocale(Locale.getDefault().toString()).build();
        client = new DbxClientV2(config, code); //connect to DropBox
    }

    //retrieves data to fill RecyclerView with
    public static List<Item> getData() {
        List<Item> data = new ArrayList<>();
        try {
            JsonNode json = getJSONData(mode == -1 ? "/thepolicies.json" //list of
policies file
                                        : (mode == 0 ? "/MPs.json" //list of MPs file
                                        : "/" + url + ".json")), //individual MP's
file
                    partiesRecords = objectMapper.createObjectNode();
//initialise new json object
            if (mode == 1) { //individual MP's record mode
                //get the MP's economic and social position
                memberEcon = json.get("economic").doubleValue();
                memberSoc = json.get("social").doubleValue();
                json = json.get("votes"); //get MP's votes

                partiesRecords = getJSONData("/partyish.json"); //the parties'
general voting records
                if (partiesRecords.has(party)) { //if party in parties file
                    partiesRecords = partiesRecords.get(party); //get specific
party's record
                    //get party's economic and social position
                    partyEcon = partiesRecords.get("economic").doubleValue();
                    partySoc = partiesRecords.get("social").doubleValue();
                } else { //if independent, 1-MP party, Sinn Fein or speaker
                    //set null
                    partyEcon = Double.NaN;
                    partySoc = Double.NaN;
                }
            }

            Iterator<String> itr = json.fieldNames(); //to iterate through the
JSON's fields
            while (itr.hasNext()) { //while more field to iterate through
                String next = itr.next(); //get current field
                ArrayNode array = json.withArray(next); //get json array by
iterated field name
                for (JsonNode obj : array) { //for each object in the array
                    if (mode == -1) { //quiz mode
                        String policy = obj.get("policy").textValue();
                        int economic = obj.get("economic").intValue(),
                                social = obj.get("social").intValue();
                        if (!(economic == 0 && social == 0)) //if policy has some
value
                            //add object to list
                            data.add(new Item(policy,
                                    null,
                                    next, //topic
```

```
                                    null,
                                    null,
                                    economic,
                                    social,
                                    false)); //make it unclickable
                    } else if (mode == 0) { //listing MPs
                        data.add(new Item(obj.get("first").textValue(), //first
name
                                obj.get("surname").textValue(),
                                next, //party
                                obj.get("seat").textValue(),
                                obj.get("id").textValue(),
                                //if MP doesn't have an economic/social position
calculated (if they haven't voted):
                                //if of Sinn Fein, set the position to 11
                                // (so they appear at the bottom of the list of
MPs),
                                //else set to 0
                                obj.has("economic") ?
obj.get("economic").doubleValue() :
                                        (next.equals("Sinn Féin") ? 11.0 : 0.0),
                                obj.has("social") ?
obj.get("social").doubleValue() :
                                        (next.equals("Sinn Féin") ? 11.0 : 0.0),
                                //i.e. only clickable if the MP has voted:
                                obj.has("economic") && obj.has("social")));
                    } else { //individual MP's voting record
                        //Get policy and the MP's % support for it
                        String policy = obj.get("policy").textValue(),
                                percent = obj.get("percent").textValue();

                        String partyPercent = "";
                        //i.e. if in an eligible party, get their party's policy
support %
                        if (!Double.isNaN(partyEcon))
                            partyPercent = "\nParty: " +
partiesRecords.get(next).get(policy).textValue();
                        data.add(new Item(policy,
                                null,
                                next, //policy's topic
                                percent + partyPercent, //MP support % + Party
support %
                                null,
                                Double.NaN,
                                Double.NaN,
                                false)); //unclickable
                    }
                }
            }
            new HeapSort(data, sortBy); //sort generated data
            if (mode == 1) { //only mode with a miscellaneous category (on
policies)
                List<Item> misc = new ArrayList<>();
                for (Item m : data) if (m.getCategory().equals("Misc"))
misc.add(m); //accumulate all misc policies
                data.removeAll(misc); //remove from data
                data.addAll(misc); //add to end of data. Now all misc policies are
at the end of the list
            }
```

```
        } catch (DbxException | IOException | NullPointerException e) {
            Log.w("Err on the side of caution", e.getMessage());
            //when internet not connected (handled in Activity on UI-side) or
going back and forth very quickly
            //between different loaded modes (causes a NullPointerException)
        }
        return data;
    }

    //retrieve from DropBox by identifier location
    public static JsonNode getJSONData(String location) throws DbxException,
IOException {
        //download the requested file and get an ordered stream of bytes
        InputStream fileInputStream =
client.files().download(location).getInputStream();
        //return as a JSON tree
        return objectMapper.readTree(fileInputStream);
    }
}
```

HeapSort.java

```
package com.manasrawat.bigtent;

import java.util.List;

public class HeapSort {

    private int sort; //global sort

    //constructor
    public HeapSort(List<Item> data, int sort) {
        int n = data.size(); //number of data items to sort
        if (n > 1) { //check, to avoid unnecessarily executing code
            this.sort = sort; //set global variable value

            //recursively construct heap by iterating downwards from midpoint
            for (int i = n / 2 - 1; i >= 0; i--) heap(data, n, i);

            //move down heap tree, to reverse
            for (int i = n - 1; i >= 0; i--) {
                // Move current root to end
                Item temp = data.get(0); //initial item, temporarily store
                data.set(0, data.get(i)); //set to current item iterated
                data.set(i, temp); //set iterated item to what initial was (swap)

                heap(data, i, 0); //get max reduced heap
            }
        }
    }

    //traverse tree
    private void heap(List<Item> data, int n, int i) {
        int largest = i; //root
        int left = 2 * i + 1; //left child
        int right = 2 * i + 2; //right child

        // If left/right child is larger than largest so far
        for (int k = 0; k < 2; k++) { //iterate to check both left and right child
```

```
            int comparator = k == 0 ? left : right; //get respective one as per
iteration

            if (comparator < n //if child in range of data being traversed through
                //if sorting by economic value, child's is greater than the
current largest's
                && ((sort == 3 ? data.get(comparator).getEcon() >
data.get(largest).getEcon()
                //if sorting by social value, child's is greater than the current
largest's
                : (sort == 4 ? data.get(comparator).getSoc() >
data.get(largest).getSoc()
                //if sorting is text-based, child is lexicographically greater
than the current largest
                : data.get(comparator).getSorter(sort).toLowerCase()
                  .compareTo(data.get(largest).getSorter(sort).toLowerCase()) >
0))
                //if sorting by economic value, child's equals current largest's
                || (((sort == 3 && data.get(comparator).getEcon() ==
data.get(largest).getEcon())
                //if sorting by social value, child's equals current largest's
                || (sort == 4 && data.get(comparator).getSoc() ==
data.get(largest).getSoc()))
                //for fallback criteria: child's case 0 text sort (surname-first)
is greater than largest's
                && data.get(comparator).getSorter(0).toLowerCase()
                   .compareTo(data.get(largest).getSorter(0).toLowerCase()) > 0)))
                largest = comparator; //new largest set if the conditions above
are met
        }

        if (largest != i) {         //if not root
            Item swap = data.get(i); //set to iterated item, holding temporarily
            data.set(i, data.get(largest)); //set iterated as largest
            data.set(largest, swap); //set largest as the temporarily-held data
(what iterated was)

            heap(data, n, largest); //recursively heap subsequent sub-tree
        }
    }
}
```

Item.java

```
package com.manasrawat.bigtent;

import static com.manasrawat.bigtent.Activity.mode;

//object holding item
public class Item {
    private String headingMain, headingExtra, category, detail, url;
    private double econ, soc;
    private boolean active;

    public Item(String headingMain, String headingExtra, String category, String
detail, String url, double econ, double soc, boolean active) {
        this.headingMain = headingMain; //firstName or policy
        this.headingExtra = headingExtra; //surname
        this.category = category; //party or policy topic
```

```
        this.detail = detail; //constituency or MPs' policy support % + their
party's support %
        this.url = url; //MP ID
        this.econ = econ; //MP or policy's economic position
        this.soc = soc; //MP or policy's social position
        this.active = active; //whether item can be clicked
    }

    //series of getters and setters

    public String getHeadingMain() {
        return headingMain; //to retrieve property
    }

    public void setHeadingMain(String headingMain) { //to set property
        this.headingMain = headingMain; //updates property globally
    }

    public String getHeadingExtra() {
        return headingExtra;
    }

    public void setHeadingExtra(String headingExtra) {
        this.headingMain = headingExtra;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public String getDetail() {
        return detail;
    }

    public void setDetail(String detail) {
        this.detail = detail;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public double getEcon() {
        return econ;
    }

    public void setEcon(double econ) {
        this.econ = econ;
    }

    public double getSoc() {
```

```
        return soc;
    }

    public void setSoc(double soc) {
        this.soc = soc;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    //when dataset of items is being sorted as per multiple criteria (with
'fallbacks') to ensure no data matches
    //in order to prevent incorrect sorting
    public String getSorter(int type) { //type = sort mode
        switch (type) {
            //by surname (first name and seat are fallbacks); only used in mode 0
            default: case 0: return headingExtra + headingMain + detail;
            case 1: return headingMain + headingExtra + detail; //by first name or
policy; only used in mode 0
            case 2: return detail; //by seat
            //by party or policy topic, with fallbacks of surname + seat (if mode
0, MPs list)
            //or fallback of just MP and their party's policy support (mode 0)
            case 5: return category + (mode == 0 ? headingExtra : "") +
headingMain + (mode == -1 ? "" : detail);
        }
    }

}
```

RecyclerAdapter.java

```
package com.manasrawat.bigtent;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.*;

import java.util.*;

import static com.manasrawat.bigtent.Activity.*;

public class RecyclerAdapter extends
RecyclerView.Adapter<RecyclerAdapter.ItemHolder> {
    private List<Item> dataset; //Recycler View data
    private Context context; //interface to global information about app
```

```
environment (passed from the activity)
    private ItemClickListener itemClickListener; //for class-global storage

    //Class constructor
    public RecyclerAdapter(Context context, List<Item> dataset) {
        //initialise global variables
        this.context = context;
        this.dataset = dataset;
    }

    //set each recycler view list item's layout
    @NonNull
    @Override
    public ItemHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        @SuppressLint("InflateParams") View layoutView =
LayoutInflater.from(viewGroup.getContext())
                                                        .inflate(R.layout.item,
null);
        return new ItemHolder(layoutView);
    }

    //set the content and functionality of each list item
    @Override
    public void onBindViewHolder(@NonNull ItemHolder itemHolder, int i) {
        int pos = itemHolder.getAdapterPosition(); //get item index
        Item currentItem = dataset.get(pos); //get item object from index
        String headingMain = currentItem.getHeadingMain(); //get item main heading
(MP first name/policy name)
        double economic = currentItem.getEcon(), social = currentItem.getSoc();
//get item's economic and social values

        //when showing list of MPs
        if (mode == 0) {
            //selectable/clickable animation enabled
            TypedArray typedArray = context.obtainStyledAttributes(new int[]{
R.attr.selectableItemBackground });
            Drawable drawable = typedArray.getDrawable(0);
            typedArray.recycle();
            itemHolder.item.setForeground(drawable);

            //if not Sinn Fein (who never turn up to vote), set MP's economic and
social positions to visually show;
            //else, for Sinn Fein, show that they don't vote
            itemHolder.detail2.setText(currentItem.getEcon() == 11.0 ? "DON'T
VOTE" :
                    "(" + round(economic) + "," + round(social) + ")");
            itemHolder.detail2.setVisibility(View.VISIBLE); //show position
        } else {
            itemHolder.item.setForeground(null); //no on-select animation
            itemHolder.detail2.setVisibility(View.GONE); //not needed
        }

        //visually set the item's main title: if item is an MP, add extra heading
(surname), else (when item is a
        // policy) don't
        itemHolder.heading.setText(
                headingMain + (currentItem.getHeadingExtra() != null ? " " +
currentItem.getHeadingExtra() : ""));
```

```
        itemHolder.category.setText(currentItem.getCategory()); //visually set
item category (party/policy)
        //if not on show-all-categories filter (i.e. are filtering by a specific
category), hide the category
        //as redundant to show, given that all items will be of the same, filtered
category
        itemHolder.category.setVisibility(selected == 0 ? View.VISIBLE :
View.GONE);

        if (currentItem.getDetail() != null) { //MP constituency or MP's support +
their party's support for a policy
            //in % (not null when mode = 0 or 1)
            itemHolder.detail1.setText(currentItem.getDetail()); //visually set
            itemHolder.detail1.setVisibility(View.VISIBLE); //make visible if it
wasn't previously
        } else itemHolder.detail1.setVisibility(View.GONE); //not needed; not just
hide, but prevent from taking any
        //space up as well

        if (mode == -1) { //if in quiz
            itemHolder.choice.setVisibility(View.VISIBLE); //show policy option
selections - only if in the quiz
            //Establish the default (Please select) + the answer options
            String[] toAdd = {
                    "Please select", "Strongly agree", "Agree", "Neutral/Unsure",
"Disagree", "Strongly disagree"
            };
            //establish option selection functionality
            //create array adapter from the toAdd array
            ArrayAdapter<String> sortAdapter = new ArrayAdapter<String>(context,
R.layout.sort_spinner_layout,
                    new ArrayList<>(Arrays.asList(toAdd))) {
                @Override
                public boolean isEnabled(int position) {
                    return position != 0; //don't count default as an option;
disable it
                }

                //establish selection menu layout
                @Override
                public View getDropDownView(int position, View convertView,
ViewGroup parent) {
                    View view = super.getDropDownView(position, convertView,
parent);
                    TextView tv = (TextView) view; //get options item
                    //if default, option being disabled is reflected through text
colour
                    if (position == 0) tv.setTextColor(Color.GRAY);
                    else tv.setTextColor(Color.BLACK);
                    return view;
                }
            };
            itemHolder.choice.setAdapter(sortAdapter); //bind adapter to view

            if (quizChoices[0].containsKey(headingMain))
itemHolder.choice.setSelection(quizChoices[0].get(headingMain));

            final boolean[] isTouched = {false};
            //if touched, option is selected; boolean to tell whether this has
```

```
occurred, for selection listener
            itemHolder.choice.setOnTouchListener((v, event) -> {
                isTouched[0] = true;
                return false;
            });

            //what to do on selection of an option
            itemHolder.choice.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
                @Override
                public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                    //if policy question not been answered before, add it and the
position of the option/answer
                    //chosen to the first hash map
                    //check if policy question being answered, and if the selected
answer
                    // doesn't match the previously selected one, to avoid
executing the code unnecessarily
                    if (isTouched[0] &&
                       (!quizChoices[0].containsKey(headingMain) ||
quizChoices[0].get(headingMain) != position)) {
                        quizChoices[0].put(headingMain, position);
                        //each value in the array goes from strongly agree (10) to
strongly disagree (-10)
                        int[] weight = {10, 5, 0, -5, -10};
                        //start at 1 to not count the unselected default, and
iterate through the rest of
                        //the user-selectable options
                        for (int i = 1; i < sortAdapter.getCount(); i++) {
                            if (position == i) { //if iterator matches selected
position
                                //check policy for both social and economic value
as some policies are both
                                //if it has an economic/social value, put that
multiplied by the selected option weight
                                //(1 subtracted from iterator to, again, not count
the default non-selection)
                                //into the respective hash map
                                if (economic != 0.0)
quizChoices[1].put(headingMain, (int) economic * weight[i - 1]);
                                if (social != 0.0) quizChoices[2].put(headingMain,
(int) social * weight[i - 1]);
                                break; //have found match to position, so no need
to iterate through the rest
                            }
                        }
                        isTouched[0] = false; //relieve selection
                    }
                }

                @Override
                public void onNothingSelected(AdapterView<?> parent) {
                }

            });
        } else itemHolder.choice.setVisibility(View.GONE); //if not in quiz, hide
policy agreement selection options
    }
```

```
    @Override
    public int getItemCount() {
        return dataset.size();
    }

    public void clear() {
        //if not already empty, make it so, and then notify the recycler view to
update visually
        if (!dataset.isEmpty()) {
            dataset.clear();
            notifyDataSetChanged();
        }
    }

    //add data items in bulk, and notify the recycler view
    public void addAll(List<Item> data) {
        dataset.addAll(data);
        notifyDataSetChanged();
    }

    //catches click events
    void setClickListener(ItemClickListener itemClickListener) {
        this.itemClickListener = itemClickListener;
    }

    //implemented by parent activity to respond to click events
    public interface ItemClickListener {
        void onItemClick(View view, int position);
    }

    //object holder for each data item of the recycler view
    public class ItemHolder extends RecyclerView.ViewHolder implements
View.OnClickListener,
            AdapterView.OnItemSelectedListener {

        public RelativeLayout item; //base layout
        public TextView heading, category, detail1, detail2;
        public Spinner choice; //for quiz only

        public ItemHolder(View item) {
            super(item);
            item.setTag(this); //bind item
            item.setOnClickListener(this); //set on-click functionality to that
method implemented by this class
            //set global variables to their XML counterparts
            this.item = item.findViewById(R.id.item);
            heading = item.findViewById(R.id.heading);
            category = item.findViewById(R.id.category);
            detail1 = item.findViewById(R.id.detail1);
            detail2 = item.findViewById(R.id.detail2);
            choice = item.findViewById(R.id.choice);
        }

        @Override
        public void onClick(View view) {
            //set click for defining in activity
            if (itemClickListener != null) itemClickListener.onItemClick(view,
getAdapterPosition());
```

```
        }

        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) { }

        @Override
        public void onNothingSelected(AdapterView<?> parent) { }
    }

}
```

changing.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:color="@color/colorPrimaryDark"/> <!--on press-->
    <item android:state_focused="true"
          android:color="@color/colorPrimaryDark"/> <!-- on focus-->
    <item android:color="@color/colorPrimary"/>  <!--default-->
</selector>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                tools:context=".Activity">

    <android.support.design.widget.AppBarLayout
            android:id="@+id/appBar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

        <!--for custom app bar-->
        <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary">

            <RelativeLayout android:layout_width="match_parent"
                            android:layout_height="match_parent">

                <!--to sort data (only for mode 0)-->
                <android.widget.Spinner
                        android:layout_centerVertical="true"
                        android:layout_alignParentLeft="true"
                        android:id="@+id/sortBySpinner"
                        android:gravity="left"
                        android:title="Sort by"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"/>

                <!--MP name (only for mode 1)-->
                <TextView
```

```
                        android:layout_centerVertical="true"
                        android:layout_alignParentLeft="true"
                        android:textColor="@android:color/black"
                        android:id="@+id/title"
                        android:textSize="15sp"
                        android:gravity="left"
                        android:visibility="gone"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"/>

                <!--reset quiz to last saved state (only for mode -1)-->
                <Button android:layout_centerVertical="true"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:id="@+id/resetButton"
                        android:text="RESET ALL"
                        android:minWidth="0dp"
                        android:layout_toLeftOf="@id/button1"/>

                <!--to show closest MPs (mode 0), to clear quiz (mode -1) or to
                show MP's position compared to their party and the user (mode 1)--
>
                <Button android:layout_centerVertical="true"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:id="@+id/button1"
                        android:text="•"
                        android:minWidth="0dp"
                        android:layout_toLeftOf="@id/quizButton"/>

                <!--to launch quiz (modes 0 or 1) or to calculate user position
(mode -1/quiz)-->
                <Button android:layout_centerVertical="true"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:id="@+id/quizButton"
                        android:text="QUIZ"
                        android:layout_alignParentRight="true"
                        android:minWidth="0dp"/>
            </RelativeLayout>

        </android.support.v7.widget.Toolbar>

    </android.support.design.widget.AppBarLayout>

    <!--to enable user to manually reload data-->
    <android.support.v4.widget.SwipeRefreshLayout
            android:layout_below="@id/appBar"
            android:id="@+id/refresh"
            android:layout_height="match_parent"
            android:layout_width="match_parent">

        <!--holds data, recycles item views within frame to save space and fasten
loading-->
        <android.support.v7.widget.RecyclerView
                android:id="@+id/recycler_view"
                android:scrollbars="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent"/>
```

```
    </android.support.v4.widget.SwipeRefreshLayout>

    <!--when loading data of a different mode to the current mode-->
    <ProgressBar
            android:id="@+id/loading_indicator"
            style="@style/Widget.AppCompat.ProgressBar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"/>

</RelativeLayout>
```

closest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--scrollable if too many MPs with equal closeness to fit dialog-->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/closestScroll">

    <!--base layout from which other views are dynamically referenced-->
    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:paddingTop="20dp"
            android:paddingLeft="20dp"
            android:paddingRight="20dp"
            android:paddingBottom="10dp"
            android:id="@+id/base">

        <!--To show user's position-->
        <TextView android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:textColor="@android:color/black"
                  android:textSize="20dp"
                  android:textStyle="bold"/>

        <!--economically closest MPs title-->
        <TextView android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:paddingTop="20dp"
                  android:textSize="15dp"
                  android:textStyle="italic"
                  android:text="Economically"/>

        <!--to show economically closest MP(s)-->
        <LinearLayout android:layout_width="match_parent"
                      android:layout_height="wrap_content"
                      android:orientation="vertical">
        </LinearLayout>

        <!--socially closest MPs title-->
        <TextView android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:paddingTop="20dp"
                  android:textSize="15dp"
```

```
                android:textStyle="italic"
                android:text="Socially"/>

        <!--to show socially closest MP(s)-->
        <LinearLayout android:layout_width="match_parent"
                      android:layout_height="wrap_content"
                      android:orientation="vertical">
        </LinearLayout>

    </LinearLayout>
</ScrollView>
```

graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <!--holds various positions (coordinates)-->
    <RelativeLayout
            android:id="@+id/pinpoint"
            android:layout_centerHorizontal="true"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="15dp">

        <!--MP position-->
        <TextView android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:textColor="@android:color/holo_red_dark"
                  android:id="@+id/MP"
                  android:layout_marginBottom="0.5dp"/>

        <!--Party position-->
        <TextView android:textColor="@android:color/holo_orange_dark"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:id="@+id/party"
                  android:layout_below="@id/MP"
                  android:layout_marginBottom="0.5dp"/>

        <!--User position-->
        <TextView android:textColor="@android:color/black"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:id="@+id/User"
                  android:layout_below="@id/party"/>
    </RelativeLayout>

    <!--For graph and its backdrop-->
    <RelativeLayout
            android:layout_width="350dp"
            android:layout_height="350dp"
            android:layout_below="@id/pinpoint">

        <!--Graph; transparent background-->
        <com.jjoe64.graphview.GraphView
```

```
                android:layout_centerInParent="true"
                android:id="@+id/gv"
                android:layout_height="350dp"
                android:layout_width="350dp">
        </com.jjoe64.graphview.GraphView>

        <!--4-quadrant-colour background-->
        <ImageView android:layout_centerInParent="true"
                   android:scaleType="fitXY"
                android:layout_width="350dp"
                   android:layout_height="350dp"
                   android:src="@drawable/compassbackdrop"/>
    </RelativeLayout>
</RelativeLayout>
```

item.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--for selectable backdrop animation effect-->
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/item"
        android:background="#BDBDBD"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    <!--to hold the details-->
    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#ffffff"
            android:layout_marginBottom="0.5dp"
            android:padding="15dp"
            android:orientation="vertical">

        <!--MP / Policy name-->
        <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/heading"
                android:layout_marginBottom="5dp"
                android:textStyle="bold"
                android:textColor="@android:color/black"
                android:textSize="20sp"/>

        <!--MP party / policy topic (data category)-->
        <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/category"
                android:textColor="@color/colorPrimary"
                android:textSize="15sp"/>

        <!--MP constituency / policy support %-->
        <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/detail1"
```

```
                android:textColor="#696969"
                android:textSize="15sp"/>

        <!--MP position-->
        <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/detail2"
                android:textColor="#696969"
                android:textSize="15sp"/>

        <!--user policy position selection-->
        <android.widget.Spinner
                android:id="@+id/choice"
                android:gravity="left"
                android:title="Choose"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>

    </LinearLayout>

</RelativeLayout>
```

sort_spinner_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--default; template; used by data spinner (mode 0) and policy options spinner
(mode -1)-->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:padding="16dp"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:textColor="@android:color/black"/>
```

menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/menu">
    <item android:title=""
          android:enabled="false"/>
</menu>
```

JSON breakdown and creation program

Main.java

```
import com.dropbox.core.DbxException;
import com.dropbox.core.DbxRequestConfig;
import com.dropbox.core.v2.DbxClientV2;
import com.dropbox.core.v2.files.ListFolderResult;
import com.dropbox.core.v2.files.Metadata;
import com.dropbox.core.v2.files.WriteMode;
import com.dropbox.core.v2.sharing.CreateSharedLinkWithSettingsErrorException;
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.core.JsonToken;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;

import java.io.*;
import java.math.BigDecimal;
import java.math.MathContext;
import java.net.URL;
import java.net.URLConnection;
import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {

    private static DbxClientV2 client; //to connect to DropBox
    private static ObjectMapper mapper = new ObjectMapper(); //for processing json
    private static ObjectNode topics; //to hold topics and their respective
policies

    public static void main(String[] args) {
        try {
            List<String> parties = new ArrayList<>(); //Make list to store parties
            List<List<String>> members = new ArrayList<>(); //make list to store
different information properties
            //about the MPs
            for (int i = 0; i < 5; i++) members.add(new ArrayList<String>());
//for each different property
            String[] htmlKeys = { //HTML page keys for MP's properties
                    "/mp/", //for their ID
                    "party",
                    "name",
                    "constituency"};

            /*construct lists of MPs' properties*/
            BufferedReader br = null; //to read online data
            try {
                URL url = new URL("https://www.theyworkforyou.com/mps/");
//connect to website
                br = new BufferedReader(new InputStreamReader(url.openStream()));
//read data

                String line;
                while ((line = br.readLine()) != null) { //reading each line of
```

```
the HTML
                    for (int i = 0; i < 4; i++) { //for each property
                        //if line has required currently-iterated property
                        if (line.contains((i == 0 ? "" : "people-list__person__")
+ htmlKeys[i])) {
                            //establish property-matching regex
                            Matcher found = Pattern.compile((i == 0) ? "\\d{5}" :
">.+<").matcher(line);
                            found.find(); //find property
                            String s = found.group(); //get property
                            if (i > 0) { //if getting MP ID
                                s = s.substring(1, s.length() - 1); //remove
unnecessary characters
                                if (i == 2) { //if getting their name
                                    //split to first name and surname
                                    String[] nameSplit = s.split(" ", 2);
                                    s = nameSplit[0];
                                    //add surname to separate child list
                                    members.get(3).add(nameSplit[1]);
                                } else if (i == 1) { //MP party
                                    if (s.equals("Labour/Co-operative")) s =
"Labour"; //remove unnecessary parts
                                    if (!parties.contains(s)) parties.add(s);
//compiling list of unique parties
                                }
                            }
                            members.get(i == 3 ? 4 : i).add(s); //so constituency
list doesn't intrude on the list
                            //of surnames
                        }
                    }
                }
            } finally {
                if (br != null) br.close(); //to prevent data leakage
            }

            if (!members.get(0).isEmpty() && !members.get(4).isEmpty()) { //check
if web page showing the right content
                //(is cleared during election time)

                /*create JSON from MPs object lists*/
                ObjectNode MPs = mapper.createObjectNode(); //make new object
                String[] attribs = {"id", "first", "surname", "seat"}; //array of
attributes
                for (int i = 0; i < members.get(0).size(); i++) { //iterate
through all MPs
                    String party = members.get(1).get(i); //get the currently-
iterated MP's party
                    if (!MPs.has(party)) MPs.putArray(party); //add party array to
object if not already there
                    ArrayNode partyArray = MPs.withArray(party); //get party array
                    ObjectNode member = mapper.createObjectNode(); //new object
for MP
                    //add MP's attributes to object
                    for (int j = 0; j < 4; j++) member.put(attribs[j],
members.get(j == 0 ? 0 : j + 1).get(i));
                    partyArray.add(member); //add MP to array of their respective
party
                }
```

```
                System.out.println("Number of MPs: " + members.get(0).size());
//total number of MPs made

                //read file of policies and their corresponding socioeconomic
values,
                //separating each section (topic) by an astrix
                Scanner scanner = new Scanner(new
File("src/main/policies.txt")).useDelimiter("\\*");
                //array of policy topics
                String[] topicNames = {
                        "Social issues",
                        "Foreign policy",
                        "Welfare",
                        "The Economy",
                        "The Constitution",
                        "Home affairs",
                        "Environment and transport",
                        "Misc"
                };

                int counter = 0;
                topics = mapper.createObjectNode(); //make json object
                while (scanner.hasNext()) { //iterate each section (topic)
                    ArrayNode topicsArray = mapper.createArrayNode(); //make array
for holding topic's policies
                    String topic = scanner.next(); //get topic's policies
                    //separate the policies
                    String[] topicPolicies =
topic.split(System.getProperty("line.separator"));
                    for (String policy : topicPolicies) { //for each loop on the
policies
                        //split policy into properties of its title, economic
value and social value (the latter two
                        // representing either -1, 0 or 1)
                        String[] policyProps = policy.split(";");
                        //add new object to array, comprising its respective
properties
                        topicsArray.add(mapper.createObjectNode()
                                .put("policy", policyProps[0])
                                .put("economic",
Integer.parseInt(policyProps[2]))
                                .put("social",
Integer.parseInt(policyProps[1])));
                    }
                    topics.put(topicNames[counter], topicsArray); //add array of
topic's policies to object of
                    // all topics
                    counter++; //increment for next topic
                }

                //Connect to DropBox, via access code
                DbxRequestConfig config =
DbxRequestConfig.newBuilder("JSONBigTent/1.0")
                        .withUserLocale(Locale.getDefault().toString()).build();
                client = new DbxClientV2(config, "CODE GOES HERE");

                System.out.println("Linked account: " +
client.getClass().getCanonicalName());
```

```
                //show number of unique shareable links
                System.out.println("Links: " +
client.sharing().listSharedLinks().getLinks().size());
                uploadFile(topics, "thepolicies"); //upload policies json

                ObjectNode partiesVotes = mapper.createObjectNode(); //for later
use (parties' records)
                int MPCounter = 0; //to keep count of all MPs (not just within a
specific party)
                for (String party : parties) { //for each loop of each party
                    if (!party.matches("Sinn Féin")) { //ensure not an
abstentionist party
                        ArrayNode partyArray = MPs.withArray(party); //get party
array of MP objects
                        for (JsonNode partyMPJson : partyArray) { //Iterate MPs

                            //Read (stream) JSON file
                        /*Need to reopen connection as member-list-iterated
through,
                        else stream prematurely closed by Heroku*/
                            //establish URL connection with required
authentication properties
                            URL url = new
URL("https://www.publicwhip.org.uk/data/popolo/policies.json");
                            URLConnection connection = url.openConnection();
                            connection.setRequestProperty("User-Agent",
                                    "Mozilla 5.0 (Windows; U; " + "Windows NT 5.1;
en-US; rv:1.8.0.11) ");
                            InputStream connectionStream =
connection.getInputStream(); //get bytes stream
                            JsonFactory jsonFactory = new JsonFactory();
//configurs and constructs json read/write
                            //parses json from byte input stream
                            JsonParser jsonParser =
jsonFactory.createParser(connectionStream);

                            ObjectNode MP = mapper.createObjectNode(), //object to
hold MP's votes + positioning
                                    MPRecord = mapper.createObjectNode(), //object
to hold MP's votes specifically
                                    partyMP = (ObjectNode) partyMPJson; //get MP's
properties (name, seat, etc)
                            String MPId = partyMP.get("id").textValue(); //get
MP's url-identifier
                            double economic = 0, social = 0; //socioeconomic-
position-pinpointing variables
                            //to hold total number of economically and socially
oriented votes
                            int econTotal = 0, socialTotal = 0;

                            /*for (String topicName : topicNames)
                                MPRecord.putArray(topicName); //Add topics to MP's
record*/
                            if (jsonParser.nextToken() != JsonToken.START_ARRAY)
//ensure online json file read properly
                                throw new IllegalStateException("Not an array");
                            //iterate through object; read from start to matching
end of object
                            //go through each policy MP has voted on
```

```
                            while (jsonParser.nextToken() ==
JsonToken.START_OBJECT) {
                                //get currently-streamed tree-structure ObjectNode
                                JsonNode node = mapper.readTree(jsonParser);
                                //get policy name, and remove trailing and leading
spaces
                                String policy =
node.path("title").textValue().trim();
                                //traverse json to get to different motion votes
                                JsonNode aspects = node.path("aspects");

                                //variables for later use in getting percentage
support per policy
                                double level = 0; //total support points
(accumulating points of value 0, 0.5 and 1)
                                double votesIn = 0; //votes of specific policy
that've been voted on by MP

                                String topic = "";
                                for (int i = 0; i < topics.size(); i++) {
//iterate topics
                                    //Binary Search to verify if the list contains
variable 'topic'
                                    if (search(topics.get(topicNames[i]), policy,
0,
                                            topics.get(topicNames[i]).size() - 1)
!= -1) {
                                        topic = topicNames[i]; //policy's topic
found
                                        break; //no need to iterate the rest
                                    }
                                }
                                if (topic.equals(""))
                                    topic = "Misc"; //if not found a topic for it,
miscellaneous policy

                                for (int j = 0; j < aspects.size(); j++)
{//iterate votes (motions) of the policy
                                    JsonNode motion =
aspects.get(j).path("motion"); //get motion json

                                    //how to vote to be in support of policy (in
favour, against or abstaining -
                                    // aye(3), no or absent/both,
                                    // immediately preceded by "tell" if the MP
was a 'teller')
                                    String policy_vote =
motion.path("policy_vote").textValue();
                                    String motionId =
motion.path("id").textValue(); //unique identifier for vote

                                    //remove unnecessary character from way of
voting recorded
                                    if (policy_vote.contains("3"))
                                        policy_vote = policy_vote.substring(0,
policy_vote.length() - 1);
                                    else if (policy_vote.equals("both"))
                                        policy_vote = "absent"; //establish
uniform value
```

```
                                    JsonNode votes =
motion.path("vote_events").get(0).path("votes"); //get votes of
                                    //all MPs for the policy
                                    for (int k = 0; k < votes.size(); k++) {
//iterate all MPs' votes to find
                                        //required MP's vote
                                        JsonNode MPVote = votes.get(k); //get
currently iterated MP's vote
                                        //if MP matches the required MP
                                        if
(MPVote.path("id").textValue().equals("uk.org.publicwhip/person/" + MPId)) {
                                            //Given that MP now found to have
voted on a motion of this topic,
                                            // add said topic to MP's record (only
added as found to avoid blank
                                            // topics)
                                            if (!MPRecord.has(topic))
MPRecord.putArray(topic);
                                            votesIn++; //increase vote count of
required MP on this policy
                                            String option =
MPVote.path("option").textValue(); //get their vote
                                            //if their vote matches the policy
vote (in favour or against the motion),
                                            //increase their support for it by 1
                                            //else if they abstain, increase by
half a point
                                            if (option.matches("(tell|)" +
policy_vote)) level++;
                                            else if (option.equals("absent"))
level += 0.5;

                                            /*Building up the parties' general
voting pattern on motions*/
                                            //if MP isn't without a party with >1
member
                                            if (!party.equals("Independent") &&
partyArray.size() > 1) {
                                                //if the party doesn't exist in
parties object, create such an object
                                                if (!partiesVotes.has(party))
partiesVotes.putObject(party);

                                                //get the party's object
                                                ObjectNode partyObj =
partiesVotes.with(party);
                                                //if the topic doesn't exist in
the party's object,
                                                // create such an object
                                                if (!partyObj.has(topic))
partyObj.putObject(topic);

                                                //get the topic's object
                                                ObjectNode topicObj =
partyObj.with(topic);
                                                //if the policy doesn't exist in
the topic's object,
                                                // create such an object
                                                if (!topicObj.has(policy))
topicObj.putObject(policy);
                                                //get the policy's object
```

```
                                                ObjectNode policyObj =
topicObj.with(policy);
                                                //if the motion doesn't exist in
the policy's object,
                                                // create such an object
                                                if (!policyObj.has(motionId))
policyObj.putObject(motionId);
                                                //get the motion's object
                                                ObjectNode voteObj =
policyObj.with(motionId);
                                                //the different ways a motion can
be voted on
                                                String[] ways = {"for", "absent",
"against"};
                                                //iterate through the ways; if
motion object doesn't have any one of
                                                //these ways present as integers,
add them to it
                                                for (String way : ways) if
(!voteObj.has(way)) voteObj.put(way, 0);
                                                //if currently-iterated
("required") MP voted on the motion in the way
                                                //that is in favour of its policy,
the count of the party's MPs voting
                                                //like that is increased by 1;
likewise for those voting against policy
                                                //or being neutral (by abstaining)
on it
                                                String which =
option.matches("(tell|)" + policy_vote) ? "for"
                                                        : (option.equals("absent")
? option : "against");
                                                voteObj.put(which,
voteObj.get(which).intValue() + 1);
                                            }
                                            break; //MP been found, so no need to
iterate through the rest
                                        }
                                    }
                                }
                                //only execute code if MP has voted/abstained
(i.e. been an MP at the time)
                                // on motions relating to this policy
                                if (votesIn != 0) {
                                    //called and passed values in order to help
calculate MP's position
                                    Socioeconomic socioeconomic =
updateSocioeconomic(
                                            true,
                                            social,
                                            economic,
                                            level,
                                            votesIn,
                                            MPRecord,
                                            topic,
                                            policy,
                                            econTotal,
                                            socialTotal);
```

```
                                    social = socioeconomic.getSoc(); //get MP's
undivided social position
                                    economic = socioeconomic.getEcon(); //get MP's
undivided economic position
                                    socialTotal = socioeconomic.getSocTotal();
//total number of social votes
                                    econTotal = socioeconomic.getEconTotal();
//total number of economic votes
                                }
                            }
                            if (!MPRecord.isEmpty()) { //only execute if MP has
voted/abstained on anything
                                //if MP has voted/abstained on economic/social-
nature policies
                                //divided by twice the totals in order to account
for the
                                // policy % movement in the opposite
                                //direction (from 1 - policy support), which is
necessary as part of the
                                // vector movement
                                if (econTotal != 0) economic = economic / (2 *
econTotal);
                                if (socialTotal != 0) social = social / (2 *
socialTotal);
                                //add to MP object in MPs list
                                partyMP.put("economic", economic)
                                       .put("social", social);
                                //add to MP's json page
                                MP.put("economic", economic)
                                  .put("social", social);
                                MP.put("votes", MPRecord); //add MPs record to
their json
                                uploadFile(MP, MPId); //upload MP's json
                                try { //create shareable url for their json based
on their ID

client.sharing().createSharedLinkWithSettings("/" + MPId + ".json");
                                } catch
(CreateSharedLinkWithSettingsErrorException ignored) { //if already made
                                }
                            }
                            System.out.println(MPCounter); //log count
                            MPCounter++; //increment
                            //close streams to prevent data leakage
                            jsonParser.close();
                            connectionStream.close();
                        }

                        //no point executing this if party has only one MP, as in
that case party position is exactly
                        //equal to the sole MP's
                        if (!party.equals("Independent") && partyArray.size() > 1)
{
                            //establish political position-pinpointing-needed
variables
                            double economic = 0, social = 0;
                            int econTotal = 0, socTotal = 0;
                            //get each parties' topics
                            ObjectNode partyRecord = partiesVotes.with(party);
```

```
                            Iterator<String> topicItr = partyRecord.fieldNames();
                            while (topicItr.hasNext()) { //iterate through topics
                                String topic = topicItr.next(); //get current
topic
                                ObjectNode topicObj = partyRecord.with(topic);
//get topic json, holding policies

                                Iterator<String> policyItr =
topicObj.fieldNames();
                                while (policyItr.hasNext()) { //iterate through
policies
                                    String policy = policyItr.next(); //policy
                                    //variables to use to calculate % support for
policy
                                    double supportLevel = 0;
                                    int votesIn = 0;
                                    ObjectNode policyObj = topicObj.with(policy);
//get json object, holding votes

                                    Iterator<String> votesItr =
policyObj.fieldNames();
                                    while (votesItr.hasNext()) { //iterate through
votes in policy
                                        String vote = votesItr.next(); //get
current vote/motion
                                        ObjectNode voteObj = policyObj.with(vote);
//json object
                                        //increase policy support if most MPs of
the party in favour;
                                        //partially increase if most abstained
                                        if (voteObj.get("for").intValue() >
voteObj.get("against").intValue() &&
                                                voteObj.get("for").intValue() >
voteObj.get("absent").intValue())
                                            supportLevel++;
                                        else if (voteObj.get("absent").intValue()
> voteObj.get("for").intValue() &&
                                                voteObj.get("absent").intValue() >
voteObj.get("for").intValue())
                                            supportLevel += 0.5;
                                        votesIn++; //increment reflects vote
                                    }
                                    if (votesIn != 0) { //if parties MPs have
voted on the policy at all,
                                        //adjust socioeconomic position and
reflect changes in the variables
                                        Socioeconomic socioeconomic =
updateSocioeconomic(
                                                false,
                                                social,
                                                economic,
                                                supportLevel,
                                                votesIn,
                                                topicObj,
                                                topic,
                                                policy,
                                                econTotal,
                                                socTotal);
                                        economic = socioeconomic.getEcon();
```

```
                                    social = socioeconomic.getSoc();
                                    econTotal = socioeconomic.getEconTotal();
                                    socTotal = socioeconomic.getSocTotal();
                                }
                            }
                        }
                        //same reasoning to MP-specific positioning
                        if (econTotal != 0) economic = economic / (2 *
econTotal);
                        if (socTotal != 0) social = social / (2 * socTotal);
                        partyRecord.put("economic", economic)
                                   .put("social", social);
                    }
                }
            }
            //having compiled all the requried data and finished the loop,
upload MP list and party list files
            uploadFile(MPs, "MPs");
            uploadFile(partiesVotes, "partyish");

            ListFolderResult result = client.files().listFolder(""); //get all
files
            while (true) {
                for (Metadata metadata : result.getEntries()) { //iterate
files
                    String filename = metadata.getName().replace(".json", "");
//get file name,
                    //without extension
                    //check if MP file is that of an MP currently in
parliament (relevant when MP resigns or
                    // post-election)
                    if (!filename.matches("thepolicies|MPs|partyish") &&
!members.get(0).contains(filename)) {
                        //if not, delete
                        client.files().deleteV2(metadata.getPathLower());
                        System.out.println("Deleted: " + filename);
                    }
                }

                if (!result.getHasMore()) break; //end loop when no more to
loop through
                //paginate through all files and retrieve updates to the
folder
                result =
client.files().listFolderContinue(result.getCursor());
            }
        } else System.out.println("Cannot execute code"); //web data
unavailable
    } catch (IOException | DbxException e) { //connection or processing error
        e.printStackTrace(); //full details
    }
}

public static void uploadFile(ObjectNode obj, String path) throws IOException,
DbxException {
    String content =
mapper.writerWithDefaultPrettyPrinter().writeValueAsString(obj); //well-readable
format
    System.out.println(content);
```

```
        InputStream file = new
ByteArrayInputStream(content.getBytes(StandardCharsets.UTF_8)); //byte stream
        //upload to DropBox at provided path
        client.files().uploadBuilder("/" + path +
".json").withMode(WriteMode.OVERWRITE).uploadAndFinish(file);
    }

    //to update MP's (economic, social) position after having found their support
level for a particular policy
    public static Socioeconomic updateSocioeconomic(
            boolean MPmode, //true if updating MP's positioning, false if updating
a party's
            double social, //current undivided cumulative social positon
            double economic, //likewise economically
            double level, //policy support level
            double votesIn, //how many votes of the said policy the MP has been in
parliament for
            ObjectNode object, //reference object to hold policy
            String topic, //policy's topic
            String policy,
            int econTotal, //total number of economically-inclined policies
overall
            int socialTotal) { //total number of socially-inclined policies
overall
            double div = level / votesIn; //calculate decimal support
            String agreement = new BigDecimal(div * 100).round(new MathContext(3))
                    .stripTrailingZeros().toPlainString() + "%"; //get % support
to 3 significant figures
            if (MPmode) //add policy and its support level to object of its
respective topic
                object.withArray(topic).add(mapper.createObjectNode()
                        .put("policy", policy)
                        .put("percent", agreement));
            else object.put(policy, agreement); //add straight to party's object

            if (!topic.equals("Misc")) { //ensure policy is accounted for as
topical and thus it can have a
                //socioeconomic value
                //BinarySearch to find policy position within overall by-topic
list column of their economic values
                //in order to retrieve policy's value (0, -1 or 1)
                int econPos = topics.get(topic)
                        .get(search(topics.get(topic), policy, 0,
topics.get(topic).size() - 1))
                        .get("economic").intValue();

                //BinarySearch to find policy position within overall by-topic
list column of their social values
                //in order to retrieve policy's value (0, -1 or 1)
                int socPos = topics.get(topic)
                        .get(search(topics.get(topic), policy, 0,
topics.get(topic).size() - 1))
                        .get("social").intValue();

                //ensure policy causes movement in user position - is actually
inclined economically or socially
                if (econPos != 0) {
                    //multiple % support for policy by -1 or +1 (depending on
policy value); also subtract from this
```

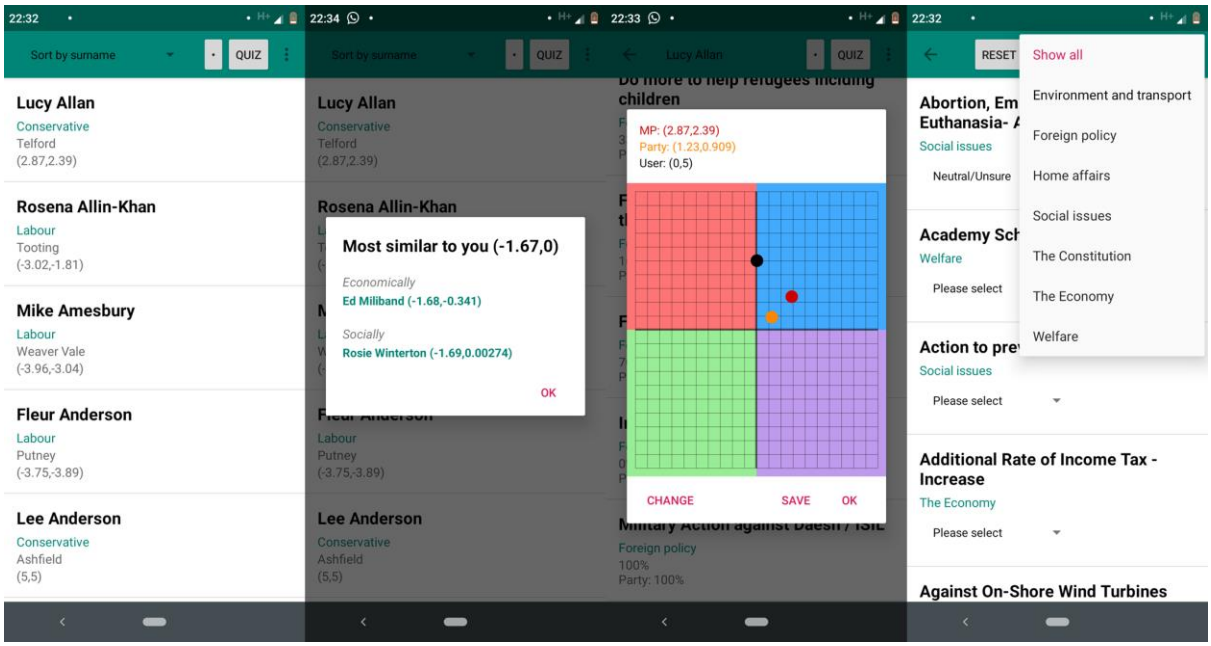
```
                    //the % in opposition to the policy (100% - % support)
multiplied by the negative/opposite of the
                    //assigned -1/+1 value - to represent the times the MP/party
has been in opposition to the policy,
                    //vector-like modelling
                    economic += (div * econPos - (1 - div) * econPos) * 10;
                    econTotal++; //increment number of economically-inclined
policies the MP/party has voted on
                }
                //likewise
                if (socPos != 0) {
                    social += (div * socPos - (1 - div) * socPos) * 10;
                    socialTotal++;
                }
            }
        return new Socioeconomic(social, economic, socialTotal, econTotal);
    }

    //BinarySearch; returns item position
    public static int search(JsonNode list, String toFind, int first, int last) {
        if (last >= first) { //ensure in range
            int centre = first + (last - first) / 2; //midpoint
            toFind = toFind.toLowerCase(); //to prevent ASCII case differentials
leading to erroneous operation
            String policy =
list.get(centre).get("policy").textValue().toLowerCase(); //policy name

            if (toFind.equals(policy)) return centre; //found
            else if (toFind.compareTo(policy) > 0) return search(list, toFind,
centre + 1, last); //in upper half
            //of currently-checked list range
            else if (toFind.compareTo(policy) < 0) return search(list, toFind,
first, centre - 1); //in lower half
        }
        return -1; //not found
    }

}
```

Socioeconomic.java

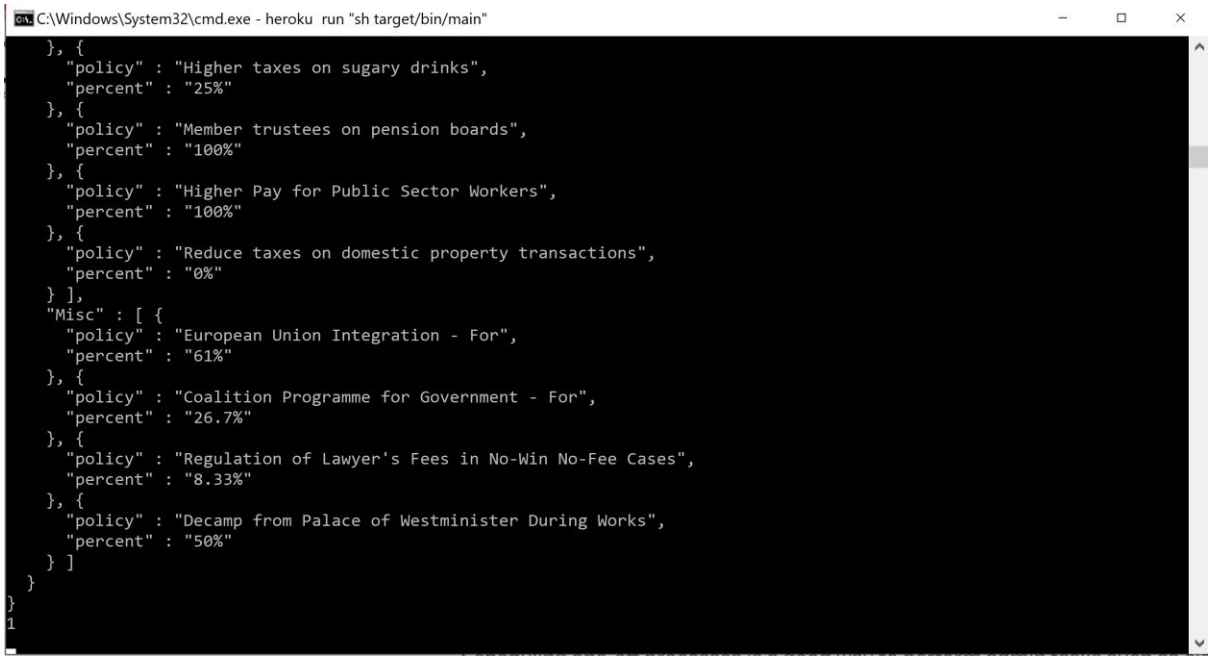
```
public class Socioeconomic {
    private double social, economic;
    private int socialTotal, economicTotal;

    public Socioeconomic(double social, double economic, int socialTotal, int
economicTotal) {
        this.social = social;
        this.economic = economic;
        this.socialTotal = socialTotal;
        this.economicTotal = economicTotal;
    }


    public double getSoc() {
        return social;
    }

    public void setSoc(double social) {
```

```
        this.social = social;
    }

    public double getEcon() {
        return economic;
    }

    public void setEcon(double economic) {
        this.economic = economic;
    }

    public int getSocTotal() {
        return socialTotal;
    }

    public void setSocTotal(int socialTotal) {
        this.socialTotal = socialTotal;
    }

    public int getEconTotal() {
        return economicTotal;
    }

    public void setEconTotal(int economicTotal) {
        this.economicTotal = economicTotal;
    }

}
```

Screenshots



Maven/Heroku Java Breakdown and creation program running:



Evaluation

I am pleased to have met all of my objectives and have developed my initial strategy as I coded.

Appendix

policies.txt

```
*Abortion, Embryology and Euthanasia- Against;+1;0
Action to prevent domestic violence;-1;0
Corporal punishment of children - Against;-1;0
Gambling - Against permissiveness;+1;-1
Homosexuality - Equal rights;-1;0
Human Rights and Equality;-1;0
Same Sex Marriage - for;-1;0
Smoking ban - In favour;+1;-1
Teach children about drugs, sexuality and health;-1;0
Termination of pregnancy - against;+1;0
Transexuality - Against legal recognition;+1;0
*Deployment of UK armed forces in Afghanistan;+1;0
Do more to help refugees inclding children;-1;-1
European Union - For;0;0
For the UK to Remain a Member of the EU;0;0
Free trade;0;+1
Hold a UK referendum on Lisbon EU Treaty;0;0
Iraq 2003 - For the invasion;+1;0
Iraq Investigation - Necessary;-1;0
Military Action against Daesh / ISIL;+1;0
Nuclear power - For;+1;+1
Preserve Environmental Protection on EU Withdrawal;0;0
Referendum on any EU withdrawal arrangements;0;0
Referendum on UK's EU membership -For -Pre 2016;0;0
Right for EU Citizens in the UK to Stay;-1;0
Stronger Military Covenant;0;-1
The UK should not ratify the Lisbon Treaty;0;0
Trident replacement - In favour;0;0
Use of UK Military Forces Overseas;+1;0
*Academy Schools - for;0;+1
Apprenticeships;0;0
Assisted Dying;-1;0
Business and community control of schools: For;0;+1
Cap or Reduce Civil Service Redundancy Payments;0;+1
End support for some 16-18 yr olds in education;0;+1
Excess Bedroom Benefit Reduction - Social Tenants;0;+1
Foundation hospitals - In favour;0;+1
GP Commissioning in the NHS;0;+1
Jobs Guarantee for Long Term Young Unemployed;0;-1
Limit NHS Foundation Trust Private Patient Income;0;-1
Localise Council Tax Support;0;+1
More Emergency Service Workers;0;-1
More funds for social care;0;-1
More Generous Benefits for Ill and Disabled;0;-1
Pension auto-enrolment - For;0;-1
Phase out of Tenancies for Life;+1;-1
Prevent abuse of zero hours contracts;0;-1
Promote Occupational Pensions;0;-1
Reduce Spending on Welfare Benefits;0;+1
Replace Higher Education Grants with Loans;0;+1
Schools - Greater Autonomy;0;0
Tuition fees - Set Upper Limit at Â£9,000 per Year;0;+1
University education fees - Should be free;0;-1
University Tuition Fees - For;0;+1
Welfare benefits ought rise in line with prices;0;-1
Woman's pension age increase - slow transition;0;-1
```

```
*Additional Rate of Income Tax - Increase;0;-1
Balance the Budget Without Borrowing;0;+1
Bankers' Bonus Tax;0;-1
Employee Shareholder Status;0;+1
Encourage and incentivise saving;0;+1
Extend Right to Buy to Housing Associations;0;+1
Higher Pay for Public Sector Workers;0;-1
Higher taxes on alcoholic drinks;0;+1
Higher taxes on banks;0;-1
Higher taxes on sugary drinks;0;+1
Increase Air Passenger Duty;0;+1
Increase the income tax - tax free allowance;0;+1
Increase VAT;0;+1
Inheritance Tax;0;-1
Make High Earners Pay Market Rent for Council Home;0;-1
Mansion Tax;0;-1
Measures to reduce tax avoidance.;0;-1
Member trustees on pension boards;0;-1
Minumum Wage;0;-1
Post office - in favour of Government policy;0;+1
Post office closures - against;0;-1
Privatise Royal Mail;0;+1
Reduce capital gains tax;0;+1
Reduce taxes on domestic property transactions;0;+1
Reduce the rate of Corporation Tax;0;+1
Regulate letting agent fees;0;-1
Require Pub Companies to Offer Rent Only Leases;0;-1
Right to strike;-1;-1
Tax Incentives for Companies Investing in Assets;0;+1
Trade Union Regulation;+1;+1
*Brexit veto for Scotland, Wales and NI;0;0
Delegate more powers to government ministers;+1;0
English Votes on English Laws etc.;0;0
Equal Number of Electors Per Constituency - for;0;0
Fixed Term Parliaments;0;0
Fully Elected House of Lords;+1;0
Further devolution to Northern Ireland;-1;0
Further devolution to Scotland;-1;0
Further devolution to Wales;-1;0
Make it easier to trigger a new election for an MP;0;0
More powers for local councils;-1;0
MPs decide if to approve a withdrawal agreeement;-1;0
No Polls Clash With MP Election System Referendum;0;0
Proportional Representation Voting System - For;0;0
Reduce central funding for local government;0;+1
Reducing the number of MPs - for;0;0
Referendum on Alternative Vote for MP Elections;0;0
Referendums for Directly Elected City Mayors;0;0
Remove Hereditary Peers from the House of Lords;-1;0
Restrict 3rd party campaigners during elections;+1;0
Retain funds from council house sales locally;0;0
Retention of Business Rates by Local Government;0;0
Role of MPs in the House of Commons - Strengthen;-1;0
Transparency of Parliament;-1;0
Voting age - Reduce to 16;-1;0
War - Parliamentary authority not necessary;0;0
*Asylum System - More strict;+1;0
Closed Material Proceedure;+1;0
Control Orders;+1;0
```

```
Freedom of Information Bill 2000 - Strengthen;-1;0
Identity cards - For introduction;+1;0
In Favour of Mass Surveillance;+1;0
Incentivise membership of press regulator;+1;0
Labour's Terrorism laws - For;+1;0
Mass Retention of Communications Data;+1;0
Merge Police and Fire under Police & Crime Cmmr;0;0
Ministers Can Intervene in Coroners' Inquests;+1;0
No detention without charge or trial;-1;0
Openness and Transparency - In Favour;-1;0
Police and Crime Commissioners;0;0
Protesting near Parliament - Unrestricted;-1;0
Recreational drugs - Against legalization;+1;0
Register of Lobbyists;+1;0
Restrict Scope of Legal Aid;0;+1
Tougher on illegal immigration;+1;0
*Against On-Shore Wind Turbines;0;+1
Ban fox hunting;+1;0
Civil aviation pollution - For limiting;0;-1
Crossrail - In favour;0;0
Cull Badgers;0;+1
Energy Prices - More Affordable;0;0
Fox hunting - Ban;+1;0
Heathrow Third Runway - In Favour;0;0
HS2 - In Favour;0;0
Incentivise Low Carbon Electricity Generation;0;-1
Lower taxes on petrol & diesel for motor vehicles;0;-1
Public Ownership of Railways;0;-1
Rail Fares - Lower;0;0
Regulation of Shale Gas Extraction;0;-1
Sell England's Public Forests;0;+1
State control of bus services;0;+1
Stop climate change;0;-1
```

Small example of a new MP's compiled JSON file

```
{
  "economic" : 5.0,
  "social" : 5.0,
  "votes" : {
    "The Constitution" : [ {
      "policy" : "Delegate more powers to government ministers",
      "percent" : "100%"
    }, {
      "policy" : "Role of MPs in the House of Commons - Strengthen",
      "percent" : "0%"
    }, {
      "policy" : "Proportional Representation Voting System - For",
      "percent" : "0%"
    }, {
      "policy" : "Further devolution to Scotland",
      "percent" : "0%"
    } ],
    "Environment and transport" : [ {
      "policy" : "Stop climate change",
      "percent" : "0%"
    } ],
    "Misc" : [ {
      "policy" : "European Union Integration - For",
```

```
      "percent" : "0%"
    } ],
    "Home affairs" : [ {
      "policy" : "Asylum System - More strict",
      "percent" : "100%"
    }, {
      "policy" : "Openness and Transparency - In Favour",
      "percent" : "0%"
    } ],
    "The Economy" : [ {
      "policy" : "Measures to reduce tax avoidance.",
      "percent" : "0%"
    }, {
      "policy" : "Minumum Wage",
      "percent" : "0%"
    } ],
    "Foreign policy" : [ {
      "policy" : "Right for EU Citizens in the UK to Stay",
      "percent" : "0%"
    } ],
    "Welfare" : [ {
      "policy" : "More funds for social care",
      "percent" : "0%"
    } ]
  }
}
```

Small snapshot of a small, newly-into-parliament party object from the parties file:

```
"Social Democratic and Labour Party" : {
    "The Constitution" : {
      "Delegate more powers to government ministers" : "0%",
      "Role of MPs in the House of Commons - Strengthen" : "83.3%",
      "Proportional Representation Voting System - For" : "50%",
      "Further devolution to Scotland" : "50%"
    },
    "Environment and transport" : {
      "Stop climate change" : "50%"
    },
    "Misc" : {
      "European Union Integration - For" : "83.3%"
    },
    "Home affairs" : {
      "Asylum System - More strict" : "0%",
      "Openness and Transparency - In Favour" : "50%"
    },
    "The Economy" : {
      "Measures to reduce tax avoidance." : "25%",
      "Minumum Wage" : "50%"
    },
    "Foreign policy" : {
      "Right for EU Citizens in the UK to Stay" : "100%"
    },
    "Welfare" : {
      "More funds for social care" : "50%"
    },
    "economic" : 0.625,
    "social" : -3.055555555555556
  }
```

Snapshot of the MPs list file:


```
{
  "Labour" : [ {
    "id" : "10001",
    "first" : "Diane",
    "surname" : "Abbott",
    "seat" : "Hackney North and Stoke Newington",
    "economic" : -2.2372759654564525,
    "social" : -1.2112311736484196
  }, {
    "id" : "25034",
    "first" : "Debbie",
    "surname" : "Abrahams",
    "seat" : "Oldham East and Saddleworth",
    "economic" : -2.544186780613748,
    "social" : -1.181042245524282
  }, {
    "id" : "24958",
    "first" : "Rushanara",
    "surname" : "Ali",
    "seat" : "Bethnal Green and Bow",
    "economic" : -2.432199275609205,
    "social" : -1.3747231055932794
  }, {
    "id" : "25888",
    "first" : "Tahir",
    "surname" : "Ali",
    "seat" : "Birmingham, Hall Green",
    "economic" : -4.375,
    "social" : -2.1527777777777777
  }, {
    "id" : "25579",
    "first" : "Rosena",
    "surname" : "Allin-Khan",
    "seat" : "Tooting",
    "economic" : -3.024553571428571,
    "social" : -1.8119265316440238
  }, {
    "id" : "25702",
    "first" : "Mike",
    "surname" : "Amesbury",
    "seat" : "Weaver Vale",
    "economic" : -3.958333333333333,
    "social" : -3.0371586134453783
  }
…
```