

NKSA

PU-ARBEIT

DATEN UND INFORMATIONEN

---

# Magic2Brain

---

*Autoren:*

Roman HIMMEL G3E

Berke ATES G3E

Fabian HALLER G3E

*Betreungsperson:*

Claude GITTELSON

26. Januar 2017

# Abstract

Unser Projekt entstand aus dem Kartenspiel „Magic: The Gathering“. In diesem spielen im Normalfall immer zwei Spieler gegeneinander. Jeder hat sein eigenes Kartendeck aus den aktuellen Karten. Es kommen jeweils alle 2-3 Monate neue Spielkarten auf den Markt und in der Regel wird nur mit den Neusten gespielt. Um einen guten Spielfluss zu erreichen, müssen die Karten bereits bekannt sein. Es ist aber mühsam, alle diese Karten zu lernen. Genau um dieses Problem zu lösen, haben wir eine App geschrieben, von Quizlet und Memorise inspiriert, die genau diese Karten beibringen soll. Auch wenn die Produktion nicht optimal verlaufen ist und wir nahezu das ganze Projekt, als es schon in einem sehr fortgeschrittenen Stadium war, neu schreiben mussten, sind wir doch schlussendlich zu unserem Ziel gelangt und konnten die App fertigstellen.

## Vorwort

Das Thema unserer Projektarbeit musste im sich im Bereich der „Daten und Informationen“ befinden. Wir entschieden uns etwas zu programmieren, weil das eine unserer Leidenschaften ist. Durch verschiedene Projekte konnten wir schon eine gewisse Menge an Erfahrung sammeln. Wir hatten die Idee eine App zu entwickeln um uns auf Neuland zu wagen und um unseren Horizont zu erweitern. Da zwei von drei von uns ein Gerät mit dem Betriebssystem Android besitzten und wir schon eine Android-Entwickler-Lizenz besaßen, entschieden wir uns die App für Android zu schreiben. Die App sollte sowohl funktional als auch visuell ansprechend sein. Wir sind zudem ambitionierte „Magic: The Gathering“-Spieler. Bei „Magic: The Gathering“ (kurz: MTG) werden mehrmals im Jahr neue Kartensets veröffentlicht. Die darin enthaltenen Karten haben neue Namen, Bilder, Fähigkeiten und Attribute. Man muss diese aber erst kennenlernen, bevor man mit ihnen richtig spielen kann. Wir haben keine Software zum Lernen der Karten gefunden. Dies ist allerdings nötig, um ein schnelles und flüssiges Spielen zu ermöglichen. Wir liessen uns von Quizlet und Memorize inspirieren, welche Fremdwörter abfragen. Dadurch kamen wir auf die Idee eine App zu entwickeln, welche die aktuellen Karten dem Benutzer beibringt. Damit kann man viel Zeit eingespart werden, in der die Karten nicht gesucht werden müssen. Folglich haben wir ein technisch anspruchvolles Projekt, welches wir auch privat nutzen können.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
<b>2</b>	<b>Theoretische Grundlagen</b>	<b>7</b>
2.1	Einführung . . . . .	7
2.1.1	Die IDE . . . . .	7
2.1.2	Der Compiler . . . . .	8
2.1.3	Der Debugger . . . . .	8
2.1.4	Die Programmiersprache . . . . .	9
2.2	Vom Binärcode zum Bild . . . . .	9
2.2.1	Vom Binärcode zur Zahl . . . . .	10
2.2.2	Grundoperatoren bei binären Zahlen . . . . .	11
2.2.3	Von der Zahl zur Text . . . . .	12
2.2.4	Von der Zahl zur Farbe . . . . .	13
2.2.5	Von der Farbe zum Bild . . . . .	14
2.3	Grundlagen von Java . . . . .	15
2.3.1	Die Main-Methode . . . . .	16
2.3.2	Datentypen . . . . .	17
2.3.3	Rechnen mit Datentypen . . . . .	19
2.3.4	Die if-Abfrage . . . . .	21
2.3.5	Schleifen . . . . .	22
2.3.6	Methoden . . . . .	25
2.3.7	Klassen . . . . .	27
2.4	Android Studio . . . . .	29
2.4.1	Programmieren mit dem Android SDK . . . . .	30
2.4.2	Die Bedeutung von XML Dateien für Android . . . . .	31
<b>3</b>	<b>Methode</b>	<b>35</b>
3.1	Die Arbeit mit Phonegap . . . . .	35
3.2	Der Aufbau der App . . . . .	35
3.3	Der Wechsel zu „Android Studio“ . . . . .	35
<b>4</b>	<b>Darstellung der Ergebnisse</b>	<b>37</b>
4.1	Home . . . . .	37

4.2	Search . . . . .	38
4.3	Set Browser . . . . .	38
4.4	Quick Learn . . . . .	39
4.5	Favorites . . . . .	39
4.6	Recently Learned . . . . .	39
4.7	Share . . . . .	39
<b>5</b>	<b>Diskussion der Ergebnisse</b>	<b>40</b>
5.1	Diskussion der Teilaspekte . . . . .	40
5.2	Diskussion des Endergebisses . . . . .	41
5.3	Diskussion des methodischen Vorgehens . . . . .	42
<b>6</b>	<b>Zusammenfassung</b>	<b>43</b>
<b>7</b>	<b>Anhang</b>	<b>46</b>

# 1 Einleitung

Das Thema „Daten und Informationen“ wird heutzutage immer wichtiger. Aber was beinhaltet Daten und Informationen alles? In der Steinzeit gab es die erste Form von Datenspeicherung in der Form von Höhlenmalereien. Bereits diese Bilder konnten „Erfahrungen mit Jagdwild, Jagdtechniken oder Wanderrouen von Tieren festhalten“ [1]. Wenn man nun die jüngere Vergangenheit betrachtet, erkennt man etwas revolutionäres in der Ägyptischen Kultur. Jedem Ägyptischen Zeichen wurde eine Bedeutung gegeben. Somit hatte man nun eine (teilweise) willkürliche Zuordnung von einem Zeichen für ein Objekt. Man musste also nun nicht mehr mit Bildern Informationen weitergeben, welche sehr verschieden interpretiert werden konnten, sondern man hatte eine Schrift, die viel weniger Interpretationsspielraum bot und dadurch auch viel präziser war. Genau diese Schrift hat uns dann, wenn auch mit einigen Modifikationen und Abstraktionen, bis in die Neuzeit als effizienteste Datenspeicherung gedient. Natürlich gab es auch in dieser Zeit noch revolutionäre Erfindungen wie die Druckerpresse von Johannes Gutenberg. Aber der erste Schritt zur heutigen Datenspeicherung wurde erst vor 75 Jahren gemacht, als im Mai der „Zuse Z3“(Abbildung 1) fertiggestellt wurde, den man heute als den „ersten funktionstüchtigen Computer der Geschichte“ bezeichnet [2]. Jedoch

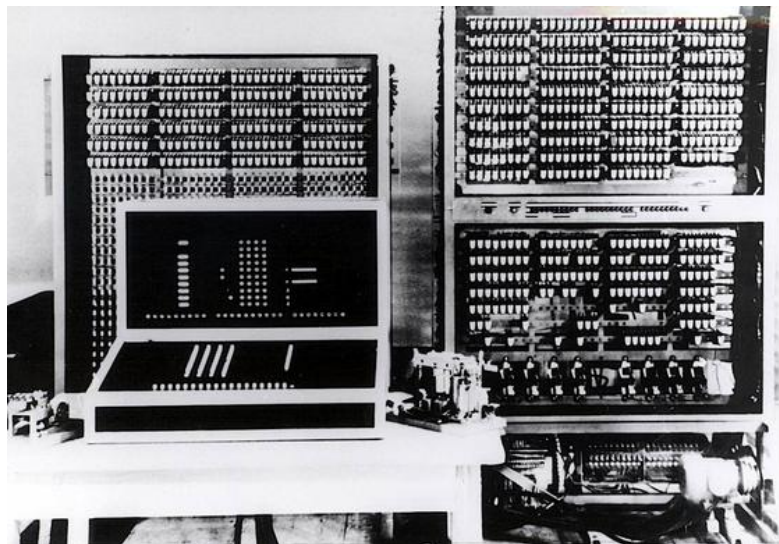


Abbildung 1: Zuse Z3 [3]

konnte dieser erste Computer, wie der Name bereits verrät (von eng. to compute =

rechnen), die Daten nicht hauptsächlich speichern sondern verarbeiten. Und genau diese Datenverarbeitung ist es, die uns in den letzten Jahren so stark geprägt hat. Viele einfache Dinge, die früher ein Mensch manuell praktizieren musste, werden heute von einem Computer übernommen, der das ganze dann meistens auch noch viel schneller kann. Und genau diese Datenverarbeitung in Verbindung mit der Benutzerinteraktion steht bei unser Projektarbeit im Mittelpunkt. Das Ziel unseres Projekt ist es, dem Benutzer etwas beizubringen und das auf einem möglichst ertragreichem Weg. Das beizubringende wird auf Karten des Spiels „Magic: The Gathering“ (Abbildung 2) beschränkt. Neben der Funktion mussten wir uns auch

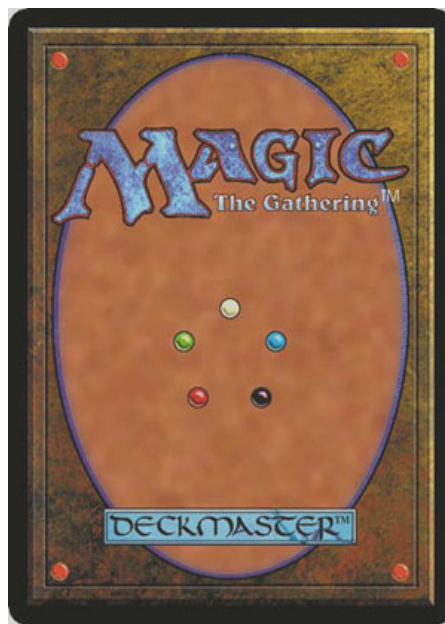


Abbildung 2: Magickarte [4]

für eine Plattform entscheiden, auf welcher unsere App verfügbar sein sollte. Da man jeder Zeit die Karten lernen können sollte, haben wir uns für eine mobile Plattform entschieden. Der Benutzer kann über ein Menu entscheiden, welche Karten er lernen will. Dann kann er über eine Lernfunktion die Karten lernen. Wie oben schon angesprochen, steht die Datenverarbeitung im Mittelpunkt. Deswegen speichern wir, ob der Benutzer die Karte memorisieren konnte. Falls das der Fall sein sollte, erscheint die Karte in nächster Zukunft nicht noch einmal, andernfalls, wird sie in naher Zukunft wiedererscheinen. Insofern kann man den maximalen Lernerfolg erreichen, da gezielt die Karten abgefragt werden, die noch nicht richtig

sitzen. Ausserdem werden Daten auf zwei verschiedene Arten gespeichert: Auf der App und im Gedächtniss des Benutzers. Natürlich sollte die App nicht nur funktionieren sondern auch noch ansprechend Aussehen und benutzerfreundlich sein. Und genau das ist es, was unsere App ausmacht.



## 2 Theoretische Grundlagen

In diesem Teil werden alle theoretischen Informationen gegeben, die man für das Verständnis des ganzen Projektes braucht. Nach dieser Einführung sollte man in der Lage sein, den Quellcode im Anhang zu verstehen. Da es jedoch sehr viel Übung erfordert, einen Quellcode zu lesen und zu verstehen, ist es naheliegend, dass der Quellcode nicht verstanden wird. In der Einführung wird grob angeschaut, wie man überhaupt vom Quellcode zur App kommt, im zweiten Teil wird erklärt, wie Daten jeglicher Art als Zahlen in einem Computer abgespeichert werden, im dritten Teil gibt es eine Einführung in die Programmiersprache Java und im letzten Teil werden noch ein paar spezifische Informationen in Bezug auf Java gegeben, die für die Erstellung einer App gebraucht werden und die Software „Android Studio“ wird erklärt.

### 2.1 Einführung

#### 2.1.1 Die IDE

Die IDE (eng. integrated development environment) oder auf Deutsch die Entwicklungsumgebung ist das Programm, womit die meisten Programmierer arbeiten. Sie bietet alle wichtigen Werkzeuge, die man zum Entwickeln von Software benötigt (Editor mit Color-Highlighting, Compiler, Debugger, Dateibrowser etc.). Auf einige Begriffe wird später noch genauer eingegangen. In unserem Falle heisst die Entwicklungsumgebung Android Studio. In der Entwicklungsumgebung findet die ganze Entwicklung einer Software statt. Der wichtigste Bereich davon ist der Editor. Dort wird der ganze Quellcode hingeschrieben und Dank Color-Highlighting werden die wichtigen Komponenten (Kontrollstrukturen, Variablen, Kommentare etc. siehe Abschnitt 2.3 Grundlagen von Java) mit Farbe hervorgehoben (Abbildung 3).

Das ist sehr wichtig, da man ansonsten schnell den Überblick verliert. Natürlich gibt es nicht nur eine IDE sondern ganz viele. Welche man davon benutzt ist jedem selbst überlassen. Es gibt auch Entwickler, die es bevorzugen, ohne eine IDE zu arbeiten. Zwar lässt sich alles so gestalten wie es einem beliebt, jedoch wird dadurch alles viel komplizierter und besonders für neu beginnende Programmierer nicht empfehlenswert.

```

#include <iostream>
using namespace std;
int main(){
    int a;
    cin >> a;
    for(int i = 1; i<= a; i++){
        if(i%3==0 && i%5==0){
            cout << "FizzBuzz";
        }
        else if(i%3==0){
            cout << "Fizz";
        }
        else if(i%5==0){
            cout << "Buzz";
        }
        else{
            cout << i;
        }
        cout << endl;
    }
}
#include <iostream>
using namespace std;
int main(){
    int a;
    cin >> a;
    for(int i = 1; i<= a; i++){
        if(i%3==0 && i%5==0){
            cout << "FizzBuzz";
        }
        else if(i%3==0){
            cout << "Fizz";
        }
        else if(i%5==0){
            cout << "Buzz";
        }
        else{
            cout << i;
        }
        cout << endl;
    }
}

```

Abbildung 3: Beispiel mit und ohne Farbhervorhebung [5]

### 2.1.2 Der Compiler

Leider versteht der Computer nichts von dem, was wir in den Quellcode schreiben. Alles was er versteht besteht aus Nullen und Einsen (mehr darüber im Abschnitt 2.2). Deshalb muss der für uns verständliche Quellcode in Maschinencode übersetzt werden. Dies geschieht mit dem sogenannten „Compiler“ (eng. to compile = zusammentragen). Meistens ist dieser bereits in der IDE enthalten und auf Knopfdruck abrufbar.

### 2.1.3 Der Debugger

Der Debugger ist sehr eng mit dem Compiler verbunden. Meistens schafft man es nicht, auf Anhieb einen fehlerlosen Code zu schreiben. Es passiert sehr schnell, dass irgendwo ein „;“ oder eine Kontrollstruktur falsch geschrieben wird (mehr dazu im Abschnitt 2.3). Deshalb ist es sehr wichtig, dass man den Fehler findet. Wenn man aber Quellcode von mehreren hundert Zeilen geschrieben hat, ist es sehr mühsam, wenn man nur weiss, dass man einen Fehler hat aber den Ort nicht genau bestimmen kann. Genau dafür ist der Debugger zuständig. Er ist aus dem englischen Wort „Bug“ entstanden, was so viel wie Käfer heisst, und im Programmieren als Synonym zu Fehler verwendet wird. Dementsprechend könnte man den Debugger als „Entfehlerer“ oder verdeutscht als „Fehlersuchprogramm“ bezeichnen. Meistens wird er aber einfach Debugger genannt. Offenbar ist der Debugger dazu in der Lage, die Programmierfehler zu finden. Er findet aber leider nur Syntaxfehler und keine, die den gewünschten Programm Output betreffen. Man kann sich das so vorstellen: In Microsoft Word werden auch Rechtschreibfehler angezeigt. Trotzdem kann man Sätze bilden die entweder keinen Sinn ergeben oder etwas anderes Aussagen, als gewünscht. Aber der Programmier Alltag ohne Debugger wäre fast unvorstellbar, da man die meisten Fehler nicht so einfach sieht wie ein falsch geschriebenes Wort. Es gibt verschiedene Arten von Debuggern. Die Meisten zeigen einem die Fehler erst an, wenn man den Quellcode zu kompilieren versucht. Es gibt sogar solche, die das in Echtzeit machen, so wie Microsoft Word Rechtschreibfehler anzeigt.

### 2.1.4 Die Programmiersprache

Eine Programmiersprache kann man sich am einfachsten wie eine richtige Sprache vorstellen. Sie bildet den Grundbaustein des Programmierens. Bevor man dem Computer etwas beibringen kann, muss man eine lernen. Jede Programmiersprache hat seine eigene Syntax. Trotzdem sind sie meistens ähnlich aufgebaut. Sobald man eine Programmiersprache gelernt hat, fällt es einem leicht, eine weitere zu lernen. Man muss sich das so vorstellen: Nachdem man gelernt hat, wie Windows XP funktioniert, hat man nicht mehr so grosse Schwierigkeiten, zu lernen wie Windows 7 oder Windows 8 funktioniert. Meistens haben die Programmiersprachen verschiedene Anwendungsbereiche: Wird zum Beispiel JavaScript und PHP meist nur für Webanwendungen verwendet und C++ wegen seiner Geschwindigkeit in Systemanwendungen gebraucht. Java hingegen wird für Geräte wie Drucker oder in unserem Fall für eine Android-Applikation benutzt. Ausserdem ist das berühmte Computerspiel „Minecraft“ in Java geschrieben.

## 2.2 Vom Binärcode zum Bild

In diesem Teil geht es darum zu verstehen, wie ein Computer komplexe Informationen wie Bilder speichern kann. Dazu geht man zuerst von Binärcode (z.B. 01110110101001) aus und arbeitet sich dann hoch. Zwar ist dieses Thema nicht unbedingt notwendig, um unser Projekt nachvollziehen zu können, aber es hilft dabei, sich die Datenspeicherung besser vorzustellen, was auf jeden Fall sinnvoll ist, da die Datenspeicherung eines unserer Kernthemen ist. Ausserdem kann man auch die Datentypen in Java besser verstehen (Abschnitt 2.3).

### 2.2.1 Vom Binärcode zur Zahl

Die kleinste Speichereinheit eines Computers ist das Bit. Man sollte das Bit auf keinen Fall mit dem Byte verwechseln. Das Byte beinhaltet nämlich acht Bits und ist dadurch viel grösser als das Bit. Man kann sich das Bit als Schalter vorstellen: Entweder ist es an oder ist aus. Es gibt genau diese 2 Zustände. Mit der 1 bezeichnet man den eingeschalteten Zustand und mit 0 den ausgeschalteten. Wenn man zwei Schalter nimmt, dann gibt es bereits 4 verschiedene Schaltzustände: 00, 01, 10 und 11. Wenn wir nun  $N$  Schalter aneinander reihen, haben wir dementsprechend

auch  $2^N$  verschiedene Schaltzustände. Wenn wir jeden dieser Schaltzustände einer Zahl zuordnen, kann man je nach Schalteranzahl beliebig grosse Zahlen abspeichern. Jedoch wurden Zahlen nicht willkürlich irgendeiner Schalterkombination zugeordnet sondern es gibt ein gewisses System dahinter, damit die Zahlen nachher auch miteinander verrechnet werden können. Dieses System nennt man das binäre Zahlensystem. Es ist nicht wie unser dezimales Zahlensystem auf zehn ausgerichtet sondern auf zwei. Um die Zahlen schreiben können, muss zuerst definiert werden, wie viele Bits gross eine Zahl ist. Diese Definition wird hier der Einfachheit halber auf 4 Bits gesetzt. Wie auch in unserem Zahlensystem wird die Zahl 0 als 0 dargestellt. Weil aber 4 Bits zur Verfügung stehen und nicht nur eines, müssen wir auch den anderen 3 einen Zustand geben, also auch 0. Dann sieht die Zahl 0 also binär dargestellt so aus: 0000. Auch die Zahl 1 ist noch einfach darzustellen: 0001. Wenn aber die Zahl 2 geschrieben werden soll, wird es ein wenig komplizierter. Wir können nämlich den ersten Schalter nicht auf 2 Stellen. Die Antwort ist eigentlich einfach: Wie in unserem dezimalen Zahlensystem nach der Zahl 9 eine neue Stelle gebraucht wird, so wird es binär genau gleich nach der Zahl 1 gemacht. Deshalb wird die Zahl 2 also so geschrieben: 0010. In diesem System geht es weiter:

Dezimal	0	1	2	3	4	5	6	7	8	...
Binär	0000	0001	0010	0011	0100	0101	0110	0111	1000	...

Dieses System gut zu kennen kann sehr nützlich sein. Wenn man die Zahlen nämlich noch ein bisschen genauer analysiert, stellt man fest, dass die hinterste Stelle der binären Zahl immer angibt, ob der Zahl eine 1 addiert werden muss, die zweit hinterste dasselbe mit 2, der dritt hintersten mit 4. Bei jeder weiteren Stelle nach vorne nimmt die zu addierende Zahl um den Faktor 2 zu. Wenn man dieses System erkannt hat, kann man auch grosse binäre Zahlen in einer nützlichen Frist in eine dezimale übersetzen und umgekehrt. Hier ein Beispiel mit der achtstelligen Binärzahl 10100101:

$$\begin{array}{cccccccc}
 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 128 & + & 0 & + & 32 & + & 0 & + & 0 & + & 4 & + & 0 & + & 1 & = & 165
 \end{array}$$

Das hier vorgestellte binäre Zahlensystem ist jedoch stark vereinfacht, da sich in diesem nur Zahlen  $\in \mathbb{N}$  darstellen lassen. Negative Zahlen und Gleitkommazahlen würden zu stark vom eigentlichen Thema abweichen. Wichtig ist nur, dass

der Computer nur so genannte Maschinenzahlen abspeichern kann. Das heisst die Zahlen müssen endlich gross sein und deshalb können periodische und irrationale Zahlen nicht genau abgespeichert werden.

### 2.2.2 Grundoperationen bei binären Zahlen

Wenn man erst einmal verstanden hat, wie das binäre Zahlensystem funktioniert, dann sind die Grundoperationen sehr einfach. Sie lassen sich ganz einfach mit den schriftlichen Operationen berechnen. Als Beispiel werden 5 Bit grosse Zahlen genommen: Die Zahl 6 (00110) und die Zahl 3(00011).

$$\begin{array}{r} \text{Addition} \quad \quad \quad \begin{array}{r} 0 \ 0 \ 1 \ 1 \ 0 \\ + \ 0 \ 0 \ 0 \ 1 \ 1 \\ \hline 0 \ 1 \ 0 \ 0 \ 1 \end{array} \end{array}$$

$$\begin{array}{r} \text{Subtraktion} \quad \quad \begin{array}{r} 0 \ 0 \ 1 \ 1 \ 0 \\ - \ 0 \ 0 \ 0 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 1 \ 1 \end{array} \end{array}$$

$$\begin{array}{r} \text{Multiplikation} \quad \quad \begin{array}{r} 0 \ 0 \ 1 \ 1 \ 0 \cdot 0 \ 0 \ 0 \ 1 \ 1 \\ \hline 0 \ 1 \ 1 \ 0 \ 0 \\ 0 \ 0 \ 1 \ 1 \ 0 \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \end{array} \end{array}$$

$$\begin{array}{r} \text{Division} \quad \quad \begin{array}{r} 0 \ 0 \ 1 \ 1 \ 0 : 0 \ 0 \ 0 \ 1 \ 1 = 0 \ 0 \ 0 \ 1 \ 0 \\ \hline 1 \ 1 \\ \hline 0 \ 0 \ 0 \end{array} \end{array}$$

### 2.2.3 Von der Zahl zur Text

Mit dem Wissen, wie Zahlen binär gespeichert werden, kann man dann sehr einfach verstehen, wie Texte abgespeichert werden. Im Grundsatz ist es nur eine willkürliche Zuordnung von Zeichen zu einer Zahl. Es gibt verschiedene Zuordnungen, die wichtigsten hierbei sind ASCII (American Standard Code for Information Interchange) und UTF-8 (Universal Character Set Transformation For-

mat). ASCII hierbei speichert jegliche Zeichen in 7 Bits ab und kann nur 128 verschiedene Zeichen speichern. Zum Beispiel würde man das Wort Haus so kodieren: 72 97 117 115. Wenn man diese Zahlen auch noch in Bits übersetzt, hat man die exakte Kodierung, wie sie auch auf dem Computer abgespeichert würde: 1001000110000111101011110011(siehe Abbildung 4)

The ASCII code  
American Standard Code for Information Interchange

www.theasciicode.com.ar

ASCII control characters			ASCII printable characters			Extended ASCII characters											
DEC	HEX	Simbolo ASCII	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
00	00h	NULL (carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	á
01	01h	SOH (inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	â
02	02h	STX (inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó
03	03h	ETX (fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	ä	163	A3h	û
04	04h	EOT (fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	å	164	A4h	ü
05	05h	ENQ (enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	ä	165	A5h	ñ
06	06h	ACK (acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	å	166	A6h	ä
07	07h	BEL (timbre)	39	27h	'	71	47h	G	103	67h	g	135	87h	ç	167	A7h	ö
08	08h	BS (retroceso)	40	28h	(	72	48h	H	104	68h	h	136	88h	è	168	A8h	ç
09	09h	HT (tab horizontal)	41	29h	)	73	49h	I	105	69h	i	137	89h	é	169	A9h	©
10	0Ah	LF (salto de línea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	¬
11	0Bh	VT (tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ï	171	ABh	¼
12	0Ch	FF (form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	ï	172	ACH	½
13	0Dh	CR (retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ï	173	ADh	¾
14	0Eh	SO (shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	Ä	174	AEh	«
15	0Fh	SI (shift In)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	Å	175	AFh	»
16	10h	DLE (data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	É	176	B0h	≡
17	11h	DC1 (device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	≡
18	12h	DC2 (device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	≡
19	13h	DC3 (device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ø	179	B3h	≡
20	14h	DC4 (device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ö	180	B4h	≡
21	15h	NAK (negative acknowle.)	53	35h	5	85	55h	U	117	75h	u	149	95h	ö	181	B5h	≡
22	16h	SYN (synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	ü	182	B6h	≡
23	17h	ETB (end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ü	183	B7h	≡
24	18h	CAN (cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ý	184	B8h	≡
25	19h	EIM (end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	Ö	185	B9h	≡
26	1Ah	SUB (substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	Ü	186	BAh	≡
27	1Bh	ESC (escape)	59	3Bh	;	91	5Bh	[	123	7Bh	{	155	9Bh	ø	187	BBh	≡
28	1Ch	FS (file separator)	60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	ø	188	BCh	≡
29	1Dh	GS (group separator)	61	3Dh	=	93	5Dh	]	125	7Dh	}	157	9Dh	ø	189	BDh	≡
30	1Eh	RS (record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	x	190	BEh	≡
31	1Fh	US (unit separator)	63	3Fh	?	95	5Fh	_				159	9Fh	f	191	BFh	≡
127	20h	DEL (delete)															

Abbildung 4: ASCII Tabelle [7]

Dadurch werden viele sprachspezifischen Zeichen wie die deutschen Umlaute nicht abspeicherbar. Deshalb wird ASCII heutzutage auch meistens nicht mehr gebraucht. UTF-8 ist hierbei viel nützlicher, aber auch komplizierter. Es kann jegliche Spezialzeichen jeder Sprache abspeichern. UTF-8 speichert die einfachen ASCII Zeichen in 8 Bits ab, das erst hinzugekommene Bit speichert ab, ob es sich bei dem Zeichen um ein ASCII Zeichen handelt oder nicht. Falls ja, können die restlichen sieben Bits wie ein ASCII Zeichen interpretiert werden. Ansonsten folgen nicht nur 7 Bits sondern wieder abhängig von der nächsten Zeichenfolge bis zu 31 weitere Bits (im ganzen dann 4 Bytes). Die deutschen Umlaute brauchen zum Beispiel 2 Bytes Speicherplatz. UTF-8 ist heutzutage der meist verbreitete Zeichensatz im Internet (im November 2016 benutzen rund 88% aller Webseiten UTF-8). [6] Ein weiterer heutzutage sehr verbreiteter Zeichensatz ist Unicode, auf

diesen wird hier aber nicht weiter eingegangen.

#### 2.2.4 Von der Zahl zur Farbe

Um Bilder abspeichern zu können, muss man zuerst einmal wissen, wie Farben abgespeichert werden. Die meist verbreitete Methode für das Abspeichern von Farben ist RGB (eng. **R**ed **G**reen **B**lue). Der Name spricht für sich. Nacheinander wird die Menge der Farbe Rot, Grün und Blau in einer Skala von 0 bis 255 angegeben. Aber auch die Farben werden in einem unüblichen Zahlensystem dargestellt: im Hexadezimalen. Dieses basiert nicht wie das binäre auf 2 und das dezimale auf 10, sondern auf 16. Da in unserem Zahlensystem keine Ziffern für 10-15 existieren, hat man die ersten Buchstaben des Alphabets genommen. 10 wäre also „A“ und 15 wäre „F“ hexadezimal geschrieben. **Achtung: Wegen diesem System ist die hexadezimale 10 nicht dasselbe wie die dezimale 10 sondern 16.** 255 wäre dementsprechend „FF“ hexadezimal geschrieben und somit die grösste hexadezimale Zahl, die mit zwei Ziffern geschrieben werden kann. Das ist natürlich kein Zufall sondern extra so gewählt. Um eine hexadezimale Zahl zu kennzeichnen, benutzt man in der Regel den Präfix 0x. Ein RGB wert besteht also aus drei zweistelligen hexadezimalen Zahlen. Meistens wird bei RGB-Farben ein # als Präfix verwendet anstatt des hexadezimal üblichen 0x. Auch noch wichtig anzumerken ist, dass das RGB-System ein additiver Farbraum ist. Das heisst also wenn man die volle Farbstärke aller Farben Rot, Grün und Blau verwendet, erhält man Weiss. Hier nun eine Tabelle mit den wichtigsten Farben und dem dazugehörigen hexadezimalen Wert.

Rot	Grün	Blau	Weiss
#FF0000	#00FF00	#0000FF	#FFFFFF
Schwarz	Gelb	Magenta	Cyan
#000000	#FFFF00	#FF00FF	#00FFFF

Übrigens ist es auch kein Zufall, dass die Farben hexadezimal dargestellt werden und nicht dezimal. Ein System, welches auf 10 basieren würde, hätte immer ein gewissen Datenverlust, da man um 10 darstellen zu können 4 Bits braucht. Mit 4 Bits lassen sich aber Zahlen bis 16 darstellen. Um also mit demselben Speicherbedarf die maximale Speichernutzung zu erreichen, hat man sich für ein hexadezimalen



System entschieden. Es gibt auch hier nicht nur ein System. Ein weiteres sehr wichtiges Farbsystem ist das „CMYK“, welches vor allem für Drucker verwendet wird. Dieses System ist im Gegensatz zum RGB-Farbsystem subtraktiv. Der Name steht für **C**yan **M**agenta **Y**ellow **K**ey. Wie im RGB-Farbsystem wird mit Cyan, Magenta und Gelb die Menge der Farbe dargestellt. Das CMYK-Farbsystem besitzt aber noch einen vierten Wert: Den Schwarzanteil (hier mit K für Key angegeben um nicht mit B für Black abkürzen zu müssen, damit eine Verwechslung mit Blue vermieden wird).

### 2.2.5 Von der Farbe zum Bild

Nun sollten alle Grundlagen vorhanden sein, um zu verstehen, wie ein Bild gespeichert werden kann. Hier wird zuerst von einem total unkomprimierten Bild ausgegangen, das mit RGB Farben ausgestattet ist. Digitale Bilder können in zwei Gruppen unterteilt werden: Rastergrafiken und Vektorgrafiken. Zuerst wird ersteres angeschaut. Ein Computerbildschirm ist bekanntlicherweise aus einzelnen Pixeln aufgebaut. Jedes dieser Pixel funktioniert eigenständig und kann eine beliebige Farbe anzeigen. Heutzutage ist das wohl am weitesten verbreitete Bildschirmformat das Full-HD mit 1920x1080 Pixeln. In einem völlig unkomprimierten Bild wird also für jedes Bild seine eigene Farbe gespeichert und schon hat man sein Bild abgespeichert. Wie im letzten Kapitel bekannt wurde, werden die Farben üblicherweise mit RGB gespeichert. Wenn man nun bedenkt, wie diese Zahlen binär aufgebaut werden, kommt man für jede hexadezimale Stelle auf 4 Bits, die gebraucht werden. Weil eine normale RGB-Farbe aus 6 hexadezimalen Zahlen bestehen, heisst das wiederum, dass eine RGB-Farbe mit 24 Bits geschrieben werden kann, was das Gleiche wie 3 Bytes ist. Um also ein Bild abzuspeichern, werden  $1920 \cdot 1080 \cdot 3 = 6220800$  Bytes  $\approx 6.5$  Megabytes. Das ist bereits eine sehr grosse Datenmenge. Für qualitativ hochwertige Bilder ist die Auflösung oftmals grösser und die Farbe genauer, wodurch der Speicherplatzbedarf nochmals enorm wachsen würde. Wenn man sogar ein Video mit 60FPS (eng. Frames Per Second = Bilder pro Sekunde) mit der Datengrösse des berechneten Bildes mit 6.5MB abspeichert, würde jede Sekunde 390MB verbrauchen. Dadurch wäre also ein Film mit einer durchschnittlichen Länge von 1.5h  $\approx 2.106$ TB gross, also grösser als die meisten Speicher eines Computers. Die Realität sieht aber ganz anders aus: Um Speicher-

platz zu optimieren werden erstens bei Bildern Komprimierungen angewendet, welche über komplexe Algorithmen die Bilder so abspeichern, dass diese viel weniger Speicherplatz einnehmen. Ausserdem speichern Videos nicht jedes Bild neu sondern nur dessen Veränderung. So braucht man etwa 1000 mal weniger Speicher. Bei Bildkompression unterscheidet man übrigens auch zwischen verlustbehafteten Kompressionen und verlustfreien. Bekannte verlustfreie Kompressionen sind zum Beispiel TIFF und PNG und verlustbehaftete JPEG und GIF [8]. Jedes Dateiformat hat aber seine eigenen Vorteile und Nachteile und je nach Zweck sollte man sich überlegen, welches davon am meisten Sinn macht. Wie Anfangs erwähnt gibt es auch noch die Vektorgrafik. Diese speichert keine Informationen über einzelne Pixel sondern speichert zum Beispiel eine Linie oder ein Kreis in Form einer mathematischen Gleichung. Der Vorteil einer Vektorgrafik ist, dass man diese frei skalieren kann, ohne dass man eine Qualitätseinbusse hat. Meistens sind Firmenlogos oder Ähnliches mit einer Vektorgrafik gezeichnet.

## 2.3 Grundlagen von Java

Jetzt da alles Nötige erarbeitet wurde, um sich etwa vorstellen zu können, wie ein Computer Informationen speichern kann, wird es einfach werden, die Datentypen von einer Programmiersprache wie Java zu verstehen. Im Abschnitt „Grundlagen von Java“ wird die Programmiersprache schnell überflogen, so dass jemand, der bereits eine andere Programmiersprache kennt, Java schnell begreifen sollte und den Code im Anhang verstehen können sollte. Da jedoch keine praktischen Aufgaben gestellt werden und auch nicht alles im Detail angeschaut wird, sind diese Grundlagen nicht dazu geeignet, nach diesen entwickeln zu können. Für jene, die noch über keine Grundlagen über eine Programmiersprache verfügen, könnte dieser Einstieg zu schwierig sein wie auch für jene, die über eine umfassende und vertiefte Erklärung wünschen, empfehlen wir eine andere Wissensquelle. Die Fachhochschule Südwestfalen hat zum Beispiel ein schönes Skript zur Einführung in Java geschrieben <https://www4.fh-swf.de/media/java.pdf>. Wer lieber ein Buch zur Hand nimmt, dem können wir O'Reilly's „Java von Kopf bis Fuss“ oder „Java ist auch eine Insel“ empfehlen, die ein solides Grundwissen vermitteln.

### 2.3.1 Die Main-Methode

In Java liegt der Einstiegspunkt in das Programm immer in der Main-Methode. Diese befindet sich in einer Klasse, die wiederum in einem Package liegt, das sich im Projektordner befindet. Fürs erste steht aber nur die Klasse im Fokus. Die Packages dienen hierbei einzig und allein der Struktur, wie Ordner auf dem Betriebssystem. Sobald der Compiler die Main-Methode entdeckt (unten aufgeführt) erkennt er diese als Einstiegspunkt an. Im Normalfall wird auch die Klasse „main“ genannt, um den Ort der Main Methode sofort ersichtlich zu machen. In eben dieser Main Methode können jetzt beliebige Befehle geschrieben werden. Im Code unten wird zum Beispiel eine Konsolenausgabe gemacht. Innerhalb der Anführungs- und Schlusszeichen kann ein beliebiger Text geschrieben werden, der dann vom Programm in der Konsole ausgegeben wird.

```
1
2
3 public class main { //erstellt eine neue Klasse mit dem Namen main
4
5
6     public static void main(String[] args) { //Beginn der Main Methode
7
8
9         System.out.println("Hello World!"); /*erzeugt eine
10        Konsolenausgabe mit dem Text Hello World*/
11
12     } //Ende der Main Methode
13
14
15 }
```

Konsole:

```
Hello World!
```

### 2.3.2 Datentypen

**Einfache Datentypen** Im Kapitel 2.2 „Vom Binärcode zum Bild“ wurde bereits ausführlich beschrieben, wie ein Computer die Daten abspeichert. Um einer Programmiersprache zu sagen, dass sie etwas abspeichern soll, muss eine Variable definiert werden. Es gibt hierbei verschiedene Variablen, die unterschiedlich gross sind und auch unterschiedliche Datentypen speichern können.

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         boolean b; //Speichert einen Wahrheitswert ab, also true oder
10        false. Anders ausgedrueckt 0 oder 1
11
12        int i; //Speichert eine ganze Zahl ab
13
14
15        float f; //Speichert eine gleitkomma Zahl ab
16
17
18        char c; //Speichert einen einzelnen Buchstaben ab
19
20
21        String s; //Speichert einen Text ab
22
23
24    }
25
26
27 }
```

Der obige Codeabschnitt reserviert aber nur den Speicher. Um etwas darin abspeichern zu können, muss zuerst noch ein Wert hinzugefügt werden

```
1
2
```

```

3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         boolean b = true;
10
11
12         int i = 14;
13
14
15         float f = 3.14159265;
16
17
18         char c = 'l';
19
20
21         String s = "Hello World";
22
23
24     }
25
26
27 }

```

Um also eine Information abspeichern zu können, braucht es zuerst einen Bezeichner, der angibt, was abgespeichert werden soll und dem Computer sagt, wie viel Speicher er reservieren muss. Als zweites wird diesem Speicher einen Namen gegeben, um diesen später wieder abrufen zu können. Es ist aber nicht erlaubt, zwei dieser Speicher mit demselben Namen zu versehen. als Letztes kann optional noch ein Wert zugewiesen werden, der in diesen Speicher passt, der vom Bezeichner deklariert wurde.

**Arrays** Es gibt aber nicht nur diese einfachen Datentypen (auch primitiv genannt) sondern es gibt auch sogenannte Arrays (Listen), die z.B. aus primitiven Datentypen bestehen. Man kann also zum Beispiel ein Array aus Integers mit der Länge 5 generieren. Dieses beinhaltet 5 Integers, auf die man über das Array zugreifen kann. Jedoch muss bei den Standard Arrays, auf welche hier eingegangen

wird, immer schon am Anfang die Grösse festgelegt werden. Im Anschluss kann das Array gefüllt werden und die einzelnen Elemente können wieder abgerufen werden. Um ein Array zu deklarieren, muss zuerst der Datentyp, aus welchem das Array besteht, definiert werden und gleich im Anschluss zwei eckige Klammern. Dann muss, wie bei primitiven Datentypen, der Bezeichner hingeschrieben werden. Um die Grösse zu bestimmen muss schliesslich hinter einem Gleichheitszeichen ein new und noch einmal der Datentyp mit den eckigen Klammern hingeschrieben werden. Innerhalb der eckigen Klammern muss die Grösse des Arrays geschrieben sein. Um auf ein beliebiges Element des Arrays zuzugreifen, schreibt man ganz normal den Variablennamen und innerhalb der eckigen Klammern die Position des Elementes. Die Positionszahl muss sich aber zwischen 0 und n-1 befinden, wobei n die Grösse des Arrays darstellt. Da üblicherweise auch 0 ein Element in der Informatik darstellt, bildet bereits n-1 das letzte Element. Nun ein Beispiel dazu:

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         int[] array = new int[5];
10
11
12         array[0] = 0;
13
14
15         array[1] = 2;
16
17
18         array[2] = 4;
19
20
21         array[3] = 6;
22
23
24         array[4] = 5;
25
```

```

26
27     System.out.println("Das dritte Element des Arrays: " + array[3]);
28
29
30     System.out.println("Das nullte Element des Arrays: " + array[0]);
31
32
33 }
34
35
36 }

```

Konsole:

```
Das dritte Element des Arrays: 6
```

```
Das nullte Element des Arrays: 0
```

### 2.3.3 Rechnen mit Datentypen

**Rechnen mit Boolean** Da Boolean nur zwei Zustände kennen, nämlich true und false haltet sich auch die Menge der möglichen Operationen in Grenzen.

```

1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         boolean a = false;
10
11
12         boolean b = false;
13
14
15         boolean c = true;
16

```

```

17
18     boolean d = true;
19
20
21     //Nur wenn beide Werte bereits true beinhalten, wird true
    zurueckgegeben (AND)
22
23
24     boolean e = a && b;
25
26
27     System.out.println("false && false = " +e);
28
29
30     e = a && c;
31
32
33     System.out.println("false && true = " +e);
34
35
36     e = c && d;
37
38
39     System.out.println("true && true = " +e);
40
41
42     //Wenn einer der beiden Werte true besitzt, wird true
    zurueckgegeben (OR)
43
44
45     e = a || b;
46
47
48     System.out.println("false || false = " +e);
49
50
51     e = a || c;
52
53
54     System.out.println("false || true = " +e);
55

```



```
56
57     e = c || d;
58
59
60     System.out.println("true || true = " +e);
61
62
63     //Der aktuelle Wert wird invertiert
64
65
66     e = !a;
67
68
69     System.out.println("!a = " +e);
70
71
72     e = !c;
73
74
75     System.out.println("!c = " +e);
76
77
78     //Zwei Werte werden verglichen
79
80
81     e = a == b;
82
83
84     System.out.println("false == false = " +e);
85
86
87     e = a == c;
88
89
90     System.out.println("false == true = " +e);
91
92
93     e = c == d;
94
95
96     System.out.println("true == true = " +e);
```

```
97
98
99 }
100
101
102 }
```

Konsole:

```
false && false = false
```

```
false && true = false
```

```
true && true = true
```

```
false || false = false
```

```
false || true = true
```

```
true || true = true
```

```
!a = true
```

```
!c = false
```

```
false == false = true
```

```
false == true = false
```

```
true == true = true
```

**Rechnen mit Zahlen** Bei Zahlen gibt es zwar wesentlich mehr Operatoren, diese funktionieren aber genau gleich wie in der Mathematik. Sobald eine ganze Zahl (Datentyp `int`) dividiert wird und die Division nicht aufgeht, wird die Zahl auf die nächst kleinere ganze Zahl abgerundet.

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         int a = 5;
10
11
12         int b = 3;
13
14
15         int c = a/b;
16
17
18         //Ganzzahlen generieren auch einen Ganzzahligen Output
19
20
21         System.out.println("5/3 = " +c);
22
23
24         int c = a%b;
25
26
27         //Der Modulooperator gibt den Rest aus
28
29
30         System.out.println("5%3 = " +c);
31
32
33         float d = 5;
34
35
36         float e = 3;
```

```

37
38
39     float f = d/e;
40
41
42     //Gleitkommazahlen berechnen auch den Wert hinter dem Komma.
43
44
45     System.out.println("5/3 = " +f);
46
47
48 }
49
50
51 }

```

Konsole:

```

5/3 = 1

5%3 = 2

5/3 = 1.6666666

```

Es gibt natürlich auch noch andere Operatoren als nur  $+$ ,  $-$ ,  $*$  und  $/$ . Im Programmieren auch sehr wichtig ist der Modulo Operator  $\%$ . Dieser gibt immer den Rest einer Division zurück (siehe oben). Die anderen Datentypen besitzen natürlich ebenfalls Operatoren, auf diese wird hier aber nicht genauer eingegangen.

### 2.3.4 Die if-Abfrage

Bei einer if-Abfrage können Bedingungen geprüft werden, die entweder true oder false sind. Ist der Wert true, wird der Inhalt der if-Abfrage ausgeführt.

```

1
2
3 public class main {
4

```

```

5
6 public static void main(String[] args) {
7
8
9     boolean IMTrue = true;
10
11
12     boolean IMFalse = false;
13
14
15     if(IMTrue){ //Wert wird hier ueberprueft
16
17
18         /*Ab hier bis zur geschlossenen geschweiften Klammer wird der
19         Code nur ausgefuehrt, falls die Bedingung wahr ist.*/
20
21         System.out.println("IMTrue-Abfrage positiv");
22
23     }
24
25
26
27     if(IMFalse){ //ein neuer Wert wird ueberprueft
28
29
30         System.out.println("IMFalse-Abfrage positiv");
31
32     }
33
34
35
36 }
37
38
39 }

```

```
IMTrue-Abfrage positiv
```

Auch innerhalb der Bedingungsklammern der if-Abfrage können Berechnungen

durchgeführt werden. Ausserdem kann es auch vorkommen, dass irgendein Befehl ausgeführt werden soll, wenn die Abfrage korrekt war und ein anderer, wenn die Abfrage falsch war. Für das gibt es dann die else-Abfrage. Wenn man sogar mehrere Abfragen von derselben Variable machen will, gibt es die else if-Abfrage.

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         boolean IMFalse = false;
10
11
12         if(IMFalse){
13
14
15             System.out.println("IMFalse besitzt den Wert true");
16
17
18         }
19
20
21         else{
22
23
24             System.out.println("IMFalse besitzt den Wert false");
25
26
27         }
28
29
30         int a = 2;
31
32
33         if(a==1){
34
35
36             System.out.println("a besitzt den Wert 1");
```

```

37
38
39     }
40
41
42     else if (a==2){
43
44
45         System.out.println("a besitzt den Wert 2");
46
47     }
48
49
50
51     else{
52
53
54         System.out.println("a besitzt weder den Wert 1 noch den Wert 2"
55     );
56
57     }
58
59
60 }
61
62
63 }

```

IMFalse besitzt den Wert false

a besitzt den Wert 2

Übrigens lassen sich auch Variablen in einer if-Abfrage definieren, jedoch kann man ausserhalb der if-Abfrage nicht mehr auf diese zugreifen.

### 2.3.5 Schleifen

Eine Schleife ist eine Konstruktion wie die if-Abfrage. Die Aufgabe der Schleife liegt aber nicht in der Abfrage eines Wertes sondern in der mehrmaligen Ausführung deren Inhaltes.

**Die For-Schleife** Die For-Schleife lässt sich aus 3 Komponenten aufbauen: Im ersten wird eine Variable deklariert und einem Wert zugewiesen. Im zweiten Teil muss eine Bedingung wie in einer if-Abfrage geschrieben werden. Falls die Bedingung wahr ist, wird die Schleife ein weiteres Mal ausgeführt, ansonsten wird sie abgebrochen. Im letzten Teil muss die Variable irgendwie verändert werden, damit diese irgendwann abgebrochen wird. Wenn sie nicht abgebrochen werden würde, würde sie endlos andauern und das Programm würde dann stecken bleiben. Deswegen ist es sehr wichtig, dass sie immer irgendwann abgebrochen wird.

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         for(int i = 0; i < 10; i++){ //Der ++Operator erhoeht die
Variable i um 1
10
11
12         System.out.println("i besitzt im Moment den Wert " +i);
13
14
15     }
16
17
18 }
19
20
21 }
```



```
i besitzt im Moment den Wert 0

i besitzt im Moment den Wert 1

i besitzt im Moment den Wert 2

i besitzt im Moment den Wert 3

i besitzt im Moment den Wert 4

i besitzt im Moment den Wert 5

i besitzt im Moment den Wert 6

i besitzt im Moment den Wert 7

i besitzt im Moment den Wert 8

i besitzt im Moment den Wert 9
```

Die obige For-Schleife zeigt den normalen Aufbau: einer Variablen wird der Wert 0 zugewiesen und die Schleife wird ausgeführt bis ein gewisser Wert (hier im Beispiel 10) erreicht wurde. Dadurch ist schon im voraus klar, wie oft der Code ausgeführt wird.

**Die While-Schleife** Eine While-Schleife ist eigentlich dasselbe wie eine For-Schleife, der einzige Unterschied liegt darin, dass die erste und die letzte Komponente der For-Schleife fehlen. Dadurch wird nur noch überprüft, ob ein Wert noch immer wahr ist. Die Aufgabe liegt jetzt beim Programmierer, wann er die Schleife abbrechen will bzw. wann die Abfrage falsch werden wird.

```

1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         boolean b = true;
10
11
12         int a = 8;
13
14
15         while(b){
16
17
18             a--; //Reduzuert den Wert a um 1
19
20
21             if(a<0){
22
23
24                 /*Sobald der Wert von a kleiner als 0 wurde, wird b auf false
25                 gesetzt und somit die while-Schleife abgeborchen.*/
26
27                 b = false;
28
29
30             }
31
32
33             System.out.println("Der Wert von a betraegt im Moment " +a);
34
35
36         }
37
38
39     }
40

```

```
41  
42 }
```

Der Wert von a betraegt im Moment 7

Der Wert von a betraegt im Moment 6

Der Wert von a betraegt im Moment 5

Der Wert von a betraegt im Moment 4

Der Wert von a betraegt im Moment 3

Der Wert von a betraegt im Moment 2

Der Wert von a betraegt im Moment 1

Der Wert von a betraegt im Moment 0

Der Wert von a betraegt im Moment -1

Es gibt auch noch zwei andere wichtige Befehle in der Schleife: `break` und `continue`. Sobald ein `continue` ausgeführt wird, springt man wieder zum Anfang und die Schleife wird wieder von neuem ausgeführt, wobei auch die Bedingung neu überprüft wird und die Variable verändert wird. `Break` dagegen bricht die Schleife komplett ab.

```
1  
2  
3 public class main {  
4  
5
```

```

6 public static void main(String[] args) {
7
8
9     for(int i = 0; i < 1000; i++){
10
11
12         if(i%3 == 0){
13
14
15             /*Falls i ein Vielfaches von drei ist , wird diese Zahl
16             uebersprungen*/
17
18             continue;
19
20         }
21
22
23
24         else if(i==10){
25
26
27             /*Falls i = 10 ist wird die Schleife abgebrochen*/
28
29
30             break;
31
32         }
33
34
35
36         else if(i%2 == 0){
37
38
39             /*Falls i eine gerade Zahl ist , wird diese Ausgegeben und
40             dann wird die Schleife wieder neu gestartet*/
41
42
43             System.out.println("CONTINUE " +i);
44

```

```

45         continue;
46
47
48     }
49
50
51     /*Ansonsten wird die Zahl ausgegeben*/
52
53
54     System.out.println("Keine der Bedingungen erfuehlt: " +i);
55
56
57 }
58
59
60 }
61
62
63 }

```

Keine der Bedingungen erfuehlt: 1

CONTINUE 2

CONTINUE 4

Keine der Bedingungen erfuehlt: 5

Keine der Bedingungen erfuehlt: 7

CONTINUE 8

### 2.3.6 Methoden

Mit der Einführung der Methoden wird erstmals die Main-Methode verlassen. Methoden sind eine der besten Möglichkeiten, ein Projekt gut zu strukturieren. Eine Methode übernimmt im Normalfall ein kleines Sub-Problem des Programmes. Eine Methode könnte man also als ein separates Programm anschauen, das irgendetwas für das Hauptprogramm berechnet. Deshalb hat jede Methode mehrere Inputwerte (Variablen) und genau einen Outputwert (auch eine Variable). Man könnte also zum Beispiel eine Methode schreiben, die 2 Zahlen zusammenzählen kann. Jede Methode braucht unter anderem einen Namen, um diese von einem anderen Ort her benutzen zu können. Die Struktur zur Deklaration einer Methode sieht also wie folgt aus: Als erstes der Bezeichner `static`, als zweites der Datentyp des Outputwertes, als drittes der Name der Methode und dann in Klammern die verschiedenen Inputwerte, die durch ein Komma voneinander getrennt sind und für jeder dieser Inputwerte braucht es eine Variablendeklaration (also Datentyp und Name). Die Inputvariablen können dann innerhalb der Methode benutzt werden und beliebig miteinander verrechnet werden. Als letztes wird dann der Outputwert mit `return` zurückgegeben. Der Datentyp der Variable muss logischerweise mit dem am Anfang deklarierten Datentyp der Methode übereinstimmen. Hier ein Beispiel, in dem eine Methode zwei Werte zusammenzählt:

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         int a = 2;
10
11
12         int b = 5;
13
14
15         int c;
16
17
```

```

18     c = addition(a, b);
19
20
21     System.out.println("The calculated result: " +c);
22
23
24 }
25
26
27 static int addition(int f, int d){
28
29
30     int e = f+d;
31
32
33     return e;
34
35
36 }
37
38
39 }

```

```
The calculated result: 7
```

Falls kein Wert von der Methode zurückgegeben werden sollte, kann anstelle eines Datentyps auch void geschrieben werden. Falls keine Input-Daten gebraucht werden, darf die Klammer einfach leer gelassen werden, muss aber trotzdem noch existent sein. Von einer Funktion kann auch eine weitere Funktion aufgerufen werden oder sie kann sich sogar selbst aufrufen (das zweite wird auch Rekursive genannt).

```

1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7

```

```

8
9     start();
10
11
12 }
13
14
15 static void start(){
16
17
18     System.out.println("Thats the starting Method");
19
20
21     rek(5);
22
23
24 }
25
26
27 static void rek(int i) {
28
29
30     if(i > 0){
31
32
33         rek(i-1);
34
35
36     }
37
38
39     System.out.println("Thats the Rek Method and i has the value " +i
40 );
41
42 }
43
44
45 }

```



```
Thats the starting Method
```

```
Thats the Rek Method and i has the value 0
```

```
Thats the Rek Method and i has the value 1
```

```
Thats the Rek Method and i has the value 2
```

```
Thats the Rek Method and i has the value 3
```

```
Thats the Rek Method and i has the value 4
```

```
Thats the Rek Method and i has the value 5
```

### 2.3.7 Klassen

Die grösste Struktur zur Übersicht in Java wird Klasse genannt. Es ist auch der Grund, weshalb Java „objektorientiert“ heisst. Bisher hat sich der ganze obere Abschnitt auf die Main-Klasse bezogen, in diesem Teil wird diese verlassen. Mit dem Prinzip des objektorientierten Programmierens wird eine grosse Menge neuer Möglichkeiten eröffnet. Eine Klasse könnte man ganz einfach als eigener Datentyp anschauen, der das kann, was man selbst definiert hat. Gleich wie die Main-Klasse, die immer in der Main-Methode beginnt, hat auch ein Objekt eine solche. Diese Main-Methode einer neuen Klasse wird aber Konstruktor genannt. Der Konstruktor muss immer denselben Namen tragen wie der der Klasse lautet. Wenn man also ein neues Objekt erzeugt (wird so genannt, wenn man eine Klasse aufruft), wird zuallererst dieser Konstruktor aufgerufen und dessen Inhalt ausgeführt. Im vorherigen Kapitel wurden die Methoden durchgenommen. Eine Methode muss nicht zwingendermassen vom Konstruktor aufgerufen werden, um ausgeführt zu werden. Sie kann, wenn man ein solches Objekt erzeugt hat, auch von dem Erzeugungsort aufgerufen werden. Eine Klasse wird in eine separate Datei geschrieben.

In der Main-Klasse wird ein Objekt dieser Klasse generiert. Das Erzeugen eines Objektes passiert beinahe gleich wie bei einem Array. Anstatt des Datentyps mit den eckigen Klammern muss aber der Klassenname aufgeführt werden und am Ende müssen die Argumente des Konstruktors der Klasse in runden Klammern eingefüllt werden. Hier ein Beispiel, in welchem ein Objekt der Klasse test erzeugt wird, welche einen Parameter des Datentyps Integer verlangt.

```
1
2
3 public class main {
4
5
6     public static void main(String [] args) {
7
8
9         Test t1 = new Test(5);
10
11     }
12
13
14
15 }
```

Die Klasse test könnte so aussehen:

### **Test.java**

```
1
2
3 public class Test {
4
5
6     /* Eine globale Variable wird erstellt , auf welche alle Methoden
7        zugreifen koennen*/
8
9     int ga;
10
11
12     public Test(int a) {
13
```

```

14
15     /*Der dem neu generierten Objekt mitgegebenen Parameter wird vom
16     Konstruktor auf eine globale Variable kopiert, damit auch von den
17     Methoden aus darauf zugegriffen werden kann*/
18
19     ga = a;
20
21 }
22
23
24
25
26
27 public int square(int n){
28
29
30     //In dieser Methode wird die n-te Potenz der Zahl ga ausgegeben
31
32
33     int result = 1;
34
35
36     for(int i = 0; i < n; i++){
37
38
39         result = result*ga;
40
41
42     }
43
44
45     return result;
46
47 }
48
49
50
51 }

```

Und die Main-Klasse etwa so:

### main.java

```
1
2
3 public class main {
4
5
6     public static void main(String[] args) {
7
8
9         /*Zwei Objekte werden generiert mit den Parametern 5 und 3*/
10
11
12         Test t1 = new Test(5);
13
14
15         Test t2 = new Test(3);
16
17
18         /*Es wird von der in der Klasse beinhaltenden Methode square
19         gebrauch gemacht*/
20
21         System.out.println("5^4 = " + t1.square(4));
22
23
24         System.out.println("5^9 = " + t1.square(9));
25
26
27         System.out.println("3^0 = " + t2.square(0));
28
29
30         System.out.println("3^5 = " + t2.square(5));
31
32
33     }
34
35
36 }
```

## Konsole

```
5^4 = 625
```

```
5^9 = 1953125
```

```
3^0 = 1
```

```
3^5 = 243
```

## 2.4 Android Studio

Android Studio ist eine Entwicklungsumgebung für Android auf der Basis von IntelliJ. IntelliJ ist eine bereits bestehende Entwicklungsumgebung für Java und Android. Im Grunde wurde Android Studio mit der Hinsicht auf Applikationsentwicklung für das Android-Betriebssystem erstellt. Es beinhaltet von Haus aus notwendige, aber auch arbeitserleichternde Komponenten, wie einen Gradle-Compiler (notwendig) oder einen Android-Emulator (erleichtert das Testen der App auf einem Computer).

Gradle ist ein Programm, das beim Kompilieren als erstes ausgeführt wird. Gradle steuert und übernimmt teilweise Aufgaben von den Java Compilern. Gradle ist hauptsächlich ein Zusammenträger von zusätzlichen Paketen und Dateien, die für das Kompilieren erforderlich sind. Dabei werden Dateien und Bibliotheken aus dem Internet heruntergeladen, die man zu der Quellenliste hinzugefügt hat. Dies erleichtert das Hinzufügen von Bibliotheken wie z.B. einer Bibliothek zum Herunterladen und Anzeigen von Bildern, da man nur die Download Adresse angeben muss und sich nicht mehr um das Einbinden kümmern muss.

Die grafische Benutzeroberfläche ist das Wichtigste an einer IDE. Sie erleichtert dem Entwickler die Navigation durch seinen Code in Form von Color-Highlighting (rechts) und Dateibrowser (links). Die Toolbar von Android Studio (oben) ist mit wichtigen Shortcuts, wie z.B. Kompilierung starten oder Emulator starten gefüllt.

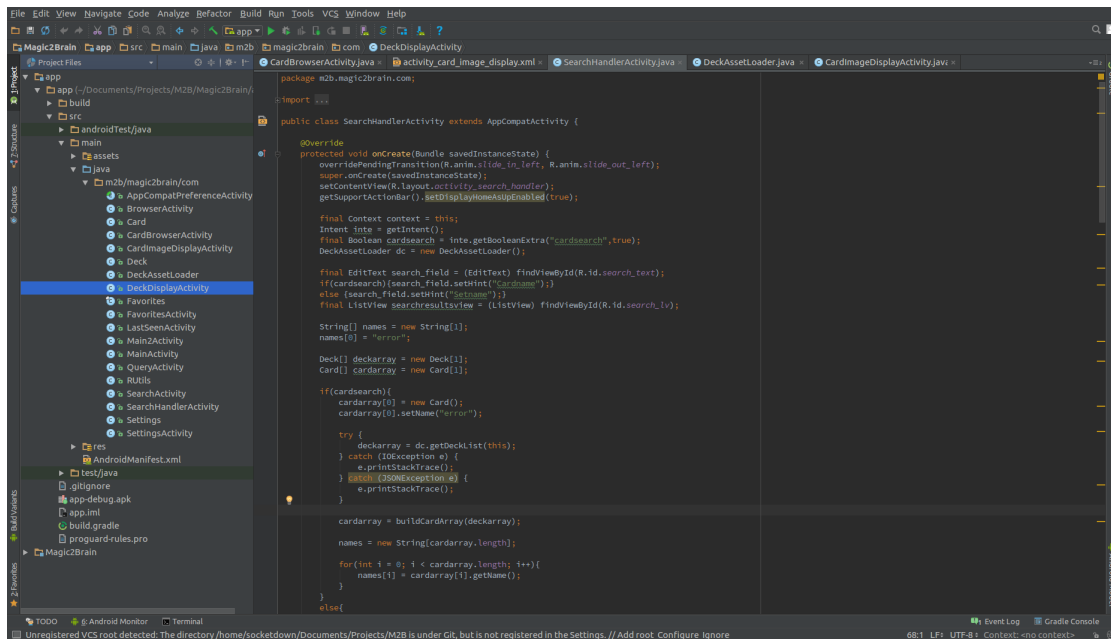


Abbildung 5: Android Studio Benutzeroberfläche [?]

Der Editor von Android Studio erlaubt eine automatische Vervollständigung mit Alt+Enter. Mitunter sind auch schon fertige Vorlagen für Activities in Android Studio verfügbar, die auf Knopfdruck in das Projekt kopiert werden und gleichzeitig für die Kompilierung registriert werden. Activities sind Java-Klassen, die der Benutzer letztendlich braucht. Der Dateibrowser erlaubt es zwischen Quellcode-Dateien zu wechseln und diese zur Bearbeitung zu öffnen. Zusätzlich sind Funktionen zum importieren von Dateien oder Assets vorhanden. Mit Assets sind hier nicht Immobilien oder Vermögen gemeint, sondern Dateien, die von der App benötigt werden um richtig zu funktionieren. Dazu gehören Bilder, Musikdateien und generell alles was kein Quellcode ist. Der IDE kann man auch Plugins hinzufügen. Plugins sind kleine Programmiererweiterungen, welche dazugeladen werden können. Bei der Entwicklung von Magic2Brain ist das Plugin "Ässer Studio" besonders nützlich gewesen, da man mit ihm Icons nicht nur zu den Assets kopieren konnte, sondern diese ganz einfach in allen auf Android gängigen Auflösungen abspeichern konnte. Normalerweise müsste das von Hand gemacht werden, was bei unseren ca. 20 Icons sehr zeitaufwändig gewesen wäre.

### 2.4.1 Programmieren mit dem Android SDK

Die Android SDK ist eines der wichtigsten Elemente in Android Studio. Ein SDK (Software Development Kit) ist eine Ansammlung von Dateien und Bibliotheken. Das eigentliche Entwickeln basiert auf bereits vorgegebenen Klassen, welche man über das Klassenattribut **extends** mit seinem eigenen Code erweitert. Zum Beispiel nimmt man die Klasse `Rechnen.java`

```
1 public class Rechnen{
2     public Rechnen(){
3
4     }
5
6     public int addieren(int a, int b){
7         return a+b;
8     }
9 }
```

Diese kann man auch mit einer Subtraktion erweitern. `Erweitert.java` würde dann so aussehen:

```
1 public class Erweitert extends Rechnen{
2     public Erweitert(){
3
4     }
5
6     public int subtrahieren(int a, int b){
7         addieren(a, -b); //in einer erweiterten Klasse koennen
8         funktionen aus der Elternklasse benutzt werden.
9     }
10 }
```

Selbstverständlich gehen dabei die herkömmlichen Programmiermöglichkeiten von Java nicht verloren, da es einem immer noch möglich ist eigene Klassen und Bibliotheken zu erstellen. Eine Besonderheit von Android ist die enge Verbindung zwischen den XML-Dateien, welche Layouts, Strings und Vektorgrafiken der einzelnen Activities enthalten. Diese Verbindung wird einzig und allein von Gradle aufrecht erhalten. Eine aus Variablen bestehende Liste wird somit bei jedem Gradle-Build resp. bei jeder Veränderung in der Ordnerstruktur von neuem erstellt.

### 2.4.2 Die Bedeutung von XML Dateien für Android

Das Auslagern von bestimmten Informationen in XML-Dateien ist für den erfahrenen Programmierer eine sehr schnelle, platzsparende aber auch nützliche Angelegenheit. Somit kann er sich einige Zeilen Code sparen um ein Element einer Activity hinzuzufügen. Als Beispiel dafür wird das Erstellen eines Texteingabefeldes in einer Activity benutzt.

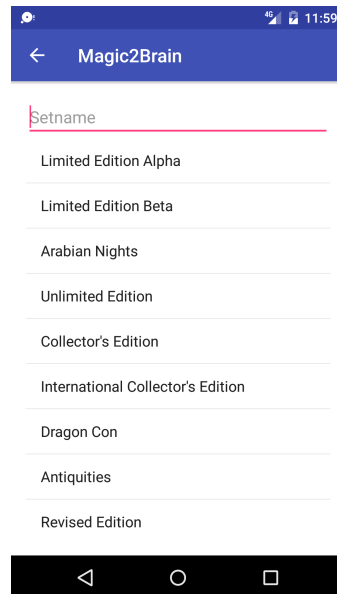


Abbildung 6: Search Activity

Dieses Resultat kann entweder mit XML bewerkstelligt werden oder mit Java selber. Im Grunde ist einem diese Entscheidung selber überlassen, da das Wichtigste beim Programmieren persönliche Präferenzen sind.

#### XML

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:id="@+id/activity_search_handler"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
```



```

7   android:paddingBottom="@dimen/activity_vertical_margin"
8   android:paddingLeft="@dimen/activity_horizontal_margin"
9   android:paddingRight="@dimen/activity_horizontal_margin"
10  android:paddingTop="@dimen/activity_vertical_margin"
11  tools:context="m2b.magic2brain.com.SearchHandlerActivity">
12
13  <EditText
14      android:layout_width="wrap_content"
15      android:layout_height="wrap_content"
16      android:inputType="textPersonName"
17      android:ems="10"
18      android:id="@+id/search_text"
19      android:layout_alignParentTop="true"
20      android:layout_alignParentStart="true"
21      android:layout_alignParentEnd="true"
22      android:contentDescription="Name" />
23 </RelativeLayout>

```

Das Element „EditText“ ist hier das Texteingabefeld. Im wird mit `android:id=` eine eindeutige ID zugewiesen, mit der später vom Code aus auf das Element zugegriffen werden kann. In Java wird das XML folgendermassen eingebunden und auf das Element zugegriffen:

```

1 //Wichtige imports werden ausgelassen
2 import m2b.magic2brain.com.magic2brain.R;
3
4 //Das Element wird mit der Funktion findViewById(int ID)
5 //in der indexierten liste gesucht und gefunden
6
7 EditText mTextField = (EditText) findViewById(R.id.search_text);
8
9 //nun kann mit der Variabel mTextField wie gewohnt gearbeitet werden

```

## Java

Auch bei einer puren Umsetzung in Java ist es nicht ganz möglich dem XML zu entrinnen. Um ein Objekt an ein Layout anzufügen, muss das zuerst verfügbar gemacht werden. Folglich sieht das XML deutlich kürzer aus, muss aber vorhanden sein. Es muss ausserdem darauf geachtet werden, dem Layout eine ID zu geben, sonst ist es nicht möglich vom Code aus auf das Layout zuzugreifen.

```

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

```

```

2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:id="@+id/activity_search_handler"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:paddingBottom="@dimen/activity_vertical_margin"
8   android:paddingLeft="@dimen/activity_horizontal_margin"
9   android:paddingRight="@dimen/activity_horizontal_margin"
10  android:paddingTop="@dimen/activity_vertical_margin"
11  tools:context="m2b.magic2brain.com.SearchHandlerActivity">
12 </RelativeLayout>

```

Wie bereits erwähnt besteht dieses XML nur noch aus dem rohen Layout-block. `RelativeLayout` ist eine besondere Art von Layout, die sich hauptsächlich für Apps eignet, die später auf Endgeräten mit verschiedenen Bildschirmgrößen vertrieben wird. Das Einbinden in den Code geschieht wie gewohnt über `findViewById(R.id.activity_search_handler);`

```

1 //Imports werden hier ausgelassen
2
3 //Das Layout muss f"ur die nachfolgenden Schritte gefunden werden
4 RelativeLayout lyt = (RelativeLayout) findViewById(R.id.
   activity_search_handler);
5
6 //Ein neues, ungeordnetes Textfeld wird erstellt
7 EditText mTextField = new EditText(this);
8
9 //Nun wird das Textfeld zum Layout hinzugef"ugt
10 lyt.addView(mTextField);

```

Man sollte XML nur für statische Dinge benutzen, währenddem man Java für dynamische, interaktive Dinge, wie z.B. Bilder oder Textfelder, welche per Knopfdruck hinzugefügt werden können, benutzen sollte.

## 3 Methode

Es gibt mehrere Möglichkeiten eine Android App zu programmieren. Man kann ein Framework (Eine Art Bibliothek, die als Gerüst funktioniert) (z.B. „LibGDX“) nutzen um mit einer vertrauten Programmiersprache und IDE eine App zu entwickeln. Google selbst bietet eine eigene IDE namens „Android Studio“. Wir entschieden uns aber anfangs mit dem Framework Phonegap zu arbeiten.

### 3.1 Die Arbeit mit Phonegap

Einer der Vorteile von Phonegap besteht darin, dass es für das Design HTML nutzt. Wir kennen HTML bereits und können damit schnell relativ ansehnliche Benutzeroberflächen erstellen. Als Programmiersprache nutzt Phonegap Javascript. Auch diese kennen wir, weil wir ein Semester lang damit gearbeitet haben. Wir konnten also direkt loslegen. Phonegap bietet zudem die Möglichkeit die App direkt auf dem Smartphone zu betrachten. Diese Funktion ist sehr praktisch, wenn man etwas betrachten will.

### 3.2 Der Aufbau der App

Wir wollten die Arbeit an der App gleichmässig aufteilen. Wenn mehrere Personen an einem Projekt arbeiten, muss man dafür sorgen, dass keine Konflikte entstehen. Solche können entstehen, wenn z.B. mehrere Personen die gleiche Datei bearbeiten. Um dies zu verhindern haben wir jede Activity der App in eine eigene Datei ausgelagert. Es sollte ein Menü geben, von wo aus man auf jede Seite zugreifen konnte. Dieses Menü musste anfangs von einer Person erstellt werden. Daraufhin konnten wir die einzelnen Seiten aufteilen und gleichzeitig arbeiten, ohne dass Konflikte entstehen. Jede Activity sollte eine eigene Funktion haben. z.B. gab es eine Activity für die Abfragen, eine mit Favoriten etc. Mit einem Zurück-Knopf kommt man von jeder Activity zum Menü zurück.

### 3.3 Der Wechsel zu „Android Studio“

Wir haben viel Zeit investiert um mit Phonegap eine schöne Benutzeroberfläche zu erstellen. Als wir dann dazu kamen die Funktionen zu programmieren, fiel

uns auf was der Nachteil von Phonegap ist. Javascript ist asynchron. Das kann behilflich sein, wenn man Animationen erstellen will, aber wenn ein Befehl auf dem Resultat eines anderen Befehls aufbaut, kann es zu Problemen führen. Man kann das Problem mit Workarounds lösen. Zum einen besteht keine Garantie, dass diese Workarounds auf allen Geräten funktionieren, zum anderen ist es äusserst aufwendig einen zu schreiben. Also haben wir den Entschluss gefasst nochmals neu anzufangen, aber diesmal mit „Android Studio“. Android Studio nutzt Java als Programmiersprache. Auch mit dieser Programmiersprache waren wir geübt. Dies vereinfachte die Entwicklung der Funktionalität um einiges. Android Studio nutzt für die Benutzeroberfläche XML, welches syntaktisch identisch mit HTML ist. Also mussten wir nicht gross umdenken, um auch XML nutzen zu können. Bei XML gibt es andere Elemente, welche wir lernen mussten. Etwas gewöhnungsbedürftig war die Verknüpfung zwischen XML und Java. Wir mussten lernen, wie man mit Java auf Elemente vom XML zugreift und diese verändert. Hinzu kam, dass wir uns mit dem Lebenszyklus einer App auseinandersetzen mussten. Bei Phonegap wurde das automatisch erledigt. Nach dem Lernen dieser Kleinigkeiten ging die Entwicklung viel schneller voran als bei Phonegap. Somit hat sich der Umstieg definitiv gelohnt.

## 4 Darstellung der Ergebnisse

Das Logo der App dient als Splashscreen (Abbildung 7). Nach dem Laden wird man von der Home-Seite begrüßt. Streift man mit dem Finger von Links nach Rechts oder klickt auf das Menü-Sandwich, öffnet sich ein Side-Menü. Das Side-Menü hat oben wieder das Logo und darunter zwei Texte („Magic 2 Brain“ und „Your MTG learning companion“). Unter dem befinden sich 7 Schaltflächen mit den Texten: „Home“, „Search“, „Set Browser“, „Quick Learn“, „Favorites“, „Recently Learned“ und „Share“ (Abbildung 7).

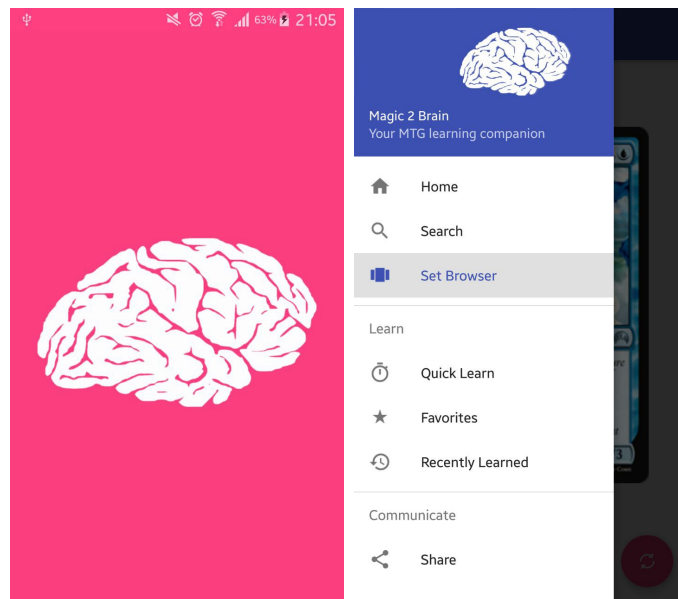


Abbildung 7: Links: Der Splash-Screen, Rechts: Das Menü

### 4.1 Home

Der Homescreen ist simpel aufgebaut. Ganz oben ist „Magic2Brain“ geschrieben und direkt unter dem „Random Card“. Mitten auf dem Bildschirm ist eine zufällige Karte abgebildet. Mit einem Klick auf die Karte lässt sich diese vergrößern. Ganz unten rechts ist ein runder Knopf. Dieser ersetzt bei einem Klick die Karte mit einer anderen zufälligen Karte. (Abbildung 8).



Abbildung 8: Der Home-Screen [4]

## 4.2 Search

Wird auf „Search“ gedrückt, kann ausgewählt werden, ob nach Karten oder Sets gesucht wird. Wählt man eine der beiden Optionen aus, wird man zu einer Ansicht mit einem Textfeld über einer Liste weitergeleitet. Tippt man etwas in das Textfeld ein, ändert sich die Liste und zeigt nur die Suchergebnisse an.

## 4.3 Set Browser

Der Set-Browser zeigt alle Sets in einer Liste an (Abbildung 9). Drückt man auf ein Set, erscheint eine weitere Ansicht mit einer Liste. Diese Liste beinhaltet alle Karten vom ausgewählten Set. Navigiert man auf eine Karte, sieht man die Karte in einer Detailansicht. Bild, Text, Manakosten und der Name der Karte werden angezeigt (Abbildung 9). Zudem hat man einen Knopf mit einem Herzen drauf. Betätigt man diesen, wird die Karte zu den Favoriten hinzugefügt.

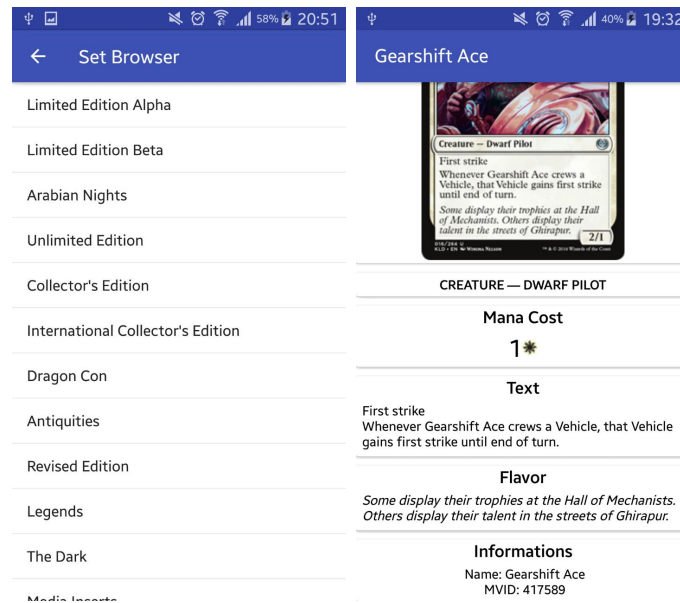


Abbildung 9: Links: Der Set-Browser, Rechts: Die Karten-Ansicht [4]

## 4.4 Quick Learn

„Quick Learn“ startet eine Abfrage mit dem aktuellsten Set. In diesem Fall ist es Kaladesh. In der Mitte wird eine Karte vom Set gezeigt. Dabei ist der Text der Karte abgedeckt. Unterhalb der Karte befinden sich vier Schaltflächen mit verschiedenen Texten. Einer der Knöpfe enthält den Text der Karte. Drückt man die richtige Schaltfläche, wird kurz ein Haken auf die Karte projiziert, ansonsten ein Kreuz. Nach dieser Animation wird das Bild der Karte durch die nächste Karte im Set ersetzt. Die Schaltflächen ändern zudem ihren Text. Oben rechts befindet sich ein Menü-Sandwich. Bei dessen Betätigung sich ein Menü mit 3 Optionen öffnet: „Query Lands“, „Reverse query“ und „Restart“. Mit „Query Lands“ kann man auswählen, ob die Länder auch abgefragt werden sollen. „Reverse query“ ändert den Abfragemodus. Im alternativen Modus ist der Name der Karte verdeckt. Unter der Karte befindet sich ein Textfeld, worin man den Namen der Karte eingeben soll. „Restart“ startet die ganze Abfrage neu (Abbildung 10).



Abbildung 10: Die Abfrage eines Sets [4]

## 4.5 Favorites

Wechselt man auf den Reiter Favoriten, werden alle Karten, die als Favorit markiert wurden, in einer Liste angezeigt.

## 4.6 Recently Learned

„Recently Learned“ zeigt die letzten 10 gelernten Sets in einer Liste an.

## 4.7 Share

Drückt man auf diese Schaltfläche, öffnet sich ein Menü. In diesem Menü kann man auswählen über welche App man den Link zu dieser App versenden möchte.



## 5 Diskussion der Ergebnisse

### 5.1 Diskussion der Teilaspekte

Unsere Teilaspekte umfassen vier Punkte:

1. **Der Benutzer muss intuitiv mit einer anschaulichen Benutzeroberfläche zurechtkommen.**

Dieser Punkt wurde recht gut erfüllt, da die Struktur sehr logisch aufgebaut ist und auch anderen ähnlichen Applikationen wie z.B. Quizlet, das für das Lernen von Vokabeln dient, gleicht. Auch haben mehrere Testpersonen die App als einfach und intuitiv zu handhaben bezeichnet. Auch grafisch und im Aufbau entspricht die App den heutigen Standards.

2. **Der Benutzer kann sich selber aussuchen, welche Karten er lernen will und welche nicht.**

Auch dieser Punkt wurde zweifelsohne erfüllt. In einem Menü kann der Benutzer das Set, das er lernen will, zuerst in einer Suchleiste eingeben und dann über einen einfachen Knopfdruck lernen.

3. **Dem Benutzer stehen eine Android-OS App und eine Webseite zum Lernen bereit.**

Dieser Punkt wurde leider nur teilweise erfüllt: Zwar steht dem Benutzer die versprochene Android-App zur Verfügung aber leider kann er auf der Webseite die Karten nicht lernen. Dieser Fakt ist aus dem Wechsel des Frameworks herzuleiten. Wenn wir das Projekt mit dem anfangs gewählten Phonegap abgeschlossen hätten, hätten wir automatisch zur Applikation auch eine funktionierende Webseite erhalten, da Phonegap beides erzeugen kann. Wegen dem Wechsel auf Android Studio wurde uns dieser Punkt leider verwehrt. Wegen mangelnder Zeit konnten wir nicht auch noch eine Webseite entwickeln. Das heisst aber auf keinen Fall, dass die Webseite nicht existiert. Diese hat nun die einfache Funktion einer Homepage, auf der man die Applikation

herunterladen kann. Sie ist auf [www.magic2brain.com](http://www.magic2brain.com) im Internet auffindbar.

4. **Das Backend, bzw. das Programm wird weitgehend autonom (wartungsfrei) arbeiten. Dabei ist die Problematik zu beachten, dass ständig neue Spielkarten veröffentlicht werden, welche ins System eingetragen werden müssen.**

Auch dieser Punkt musste aufgrund mangelnder Zeit vernachlässigt werden, da die Priorität auf der Funktionsfähigkeit der Applikation lag. Das Problem mit den ständig neu erscheinenden Spielkarten wird jetzt ganz einfach durch regelmässige Updates der Applikation gelöst, damit der Benutzer trotzdem immer die neusten Karten lernen kann. Wenn die App einen guten Zulauf geniessen wird, ist es sehr wahrscheinlich, dass wir auch nach der Projektabgabe diesen Punkt noch beheben werden.

## 5.2 Diskussion des Endergebnisses

Nur Dank unserer bereits reichlich gesammelten Erfahrung war es möglich, eine solche Applikation umzusetzen, da wir uns nicht mehr stundenlang mit den Grundlagen auseinandersetzen mussten, sondern direkt zu entwickeln beginnen konnten. Trotzdem haben wir uns auf Neuland gewagt und unseren Horizont im Bereich der Entwicklung einer Android-Applikation erweitern können. Wir haben es tatsächlich geschafft, eine ansehnliche Applikation zu erstellen, welche zumindest im grafischen Bereich und in der Benutzerfreundlichkeit dem heutigen Standard entspricht. Bei den Optionen, über die der Benutzer verfügt, mussten wir aber sparen. Unser Entwicklungsteam bestand nur aus 3 Personen und eine Applikation mit reichhaltiger Funktionalität war deshalb in diesem Zeitraum schlichtweg nicht möglich. Die Zeit, die uns für die Projektarbeit zur Verfügung stand war zudem ziemlich knapp bemessen, da wir zum einen auch den Schulalltag bewältigen mussten und zum anderen auch diese Arbeit schreiben mussten. Wenn man aber bedenkt, dass die meisten erfolgreichen Applikationen von einem professionellen Team entwickelt wurden, ist unser Produkt einer Projektarbeit definitiv würdig. Deshalb sehen wir unser Projekt als gelungen an.

### 5.3 Diskussion des methodischen Vorgehens

Was man jedoch als grossen Kritik- und Verbesserungspunkt ansehen muss, ist die Erörterung eines geeigneten Frameworks. Während mehreren Wochen haben wir an einem Projekt mit Phonegap gearbeitet, in das wir viel Zeit und Arbeit investiert haben, nur um dann zu merken, dass Phonegap für unsere Art von Projekt überhaupt nicht geeignet ist. In diesem Fall haben wir im Voraus viel zu wenig Nachforschung betrieben, um das geeignete Framework zu finden. Von dieser Erfahrung haben wir definitiv profitiert. Die Arbeit an sich verlief sonst konfliktfrei mitunter dank der Organisationssoftware GitHub und der guten Aufteilung der Arbeit. GitHub hat uns hierbei nicht nur bei der Applikation sondern auch bei der schriftlichen Arbeit sehr geholfen. Dank der Include-Funktion von LaTeX war es auch möglich, die Projektarbeit in einzelne Teile aufzuteilen wodurch ein paralleles Arbeiten an den Teilen problemlos vonstatten ging und trotzdem jeder von uns immer die aktuellste Version besessen hat. Des Weiteren hat uns auch unsere Erfahrung von Projektarbeiten vom Lehrgang Infcom.ch geholfen, dieses richtig zu organisieren. Auch während des Infcom.ch-Unterrichts mussten wir mehrmals eine technische Arbeit organisieren und konnten dadurch auf die bereits gesammelte Erfahrung zurückgreifen.

## 6 Zusammenfassung

Unsere Projektidee entstand mit dem Hinblick auf neue und bestehende Turnierspieler der Kartenspiels Magic: The Gathering (*kurz: MTG*). Ungefähr alle 2-3 Monate kommt ein neues Kartenpaket heraus. Dieses wird Set genannt. Die Karten in jedem Set haben andere Eigenschaften in der Angriffsstärke, der Verteidigung oder haben sogar besondere oder einzigartige Eigenschaften. Um nicht ständig diese Daten von den Karten zu lesen wollten wir eine App machen, die genau diese Daten deren Nutzern beibringt. Ein Spiel hat nämlich auch eine zeitliche Begrenzung, die in einem Unentschieden endet. Daher ist es für beide Spieler von Vorteil wenn sie schnell spielen.

Um dies zu bewerkstelligen haben wir zahlreiche APIs (*Application Programming Interface*) verglichen. Zu beachten gab es 3 grundlegende Komponenten: Bilder der Karte, Text mit Eigenschaften der Karte und Eine Möglichkeit den Fortschritt zu speichern. Die Bilder haben wir direkt von dem Kartenbrowser von Wizards of the Coast genommen. Als API für die Karteninformationen hat MTGJSON gesiegt, wobei dies keine API ist, sondern eine Downloadquelle für Karten im JSON Format. JSON ist eine schnelle und einfach zu begreifende Alternative zu XML. Somit kann die App auch offline verwendet werden. Bilder, welche einmal heruntergeladen wurden, werden bis zum Deinstallieren der App oder bis zum manuellen Löschen auf dem Gerät des Benutzers gespeichert. Informationen über den Fortschritt werden in den *SharedPreferences* von Android gespeichert. Dies kann man sich wie ein Kartesystem vorstellen. Jede App erhält ihr eigenes Fach und kann dort Informationen ablegen.

Ursprünglich wollten wir unsere App in HTML, CSS und JavaScript schreiben, welche wir dann mit Cordova zu einer Android und IOS App kompiliert hätten. Jedoch wurde uns durch die Natur von JavaScript verwehrt einige essentielle Funktionen zu implementieren. Dies lag vorwiegend daran, dass JavaScript asynchron arbeitet und man dies nicht mehr wie damals ausschalten kann. Asynchron beschreibt die Arbeitsweise von einer Programmiersprache. In diesem Fall werden mehrere Funktionen gleichzeitig ausgeführt, was ein gut durchdachtes und stabiles Programm erfordert. Ausserdem müssen in einigen Fällen "*Promises*" zurückgegeben werden. Das sind Verweise auf einen Wert, der in der Zukunft existieren wird, aber noch nicht bekannt ist wann es so weit ist. Als Beispiel nehme

man eine ladende Webseite. Niemand weiss wann sie fertig geladen ist, da das von der Verbindungsgeschwindigkeit abhängt, aber der Fakt, dass sie einmal geladen sein wird ist unbestritten. Promises kann man auch nicht ohne weiters abwarten bzw. *auflösen*, weil das in sehr vielen Fällen kritische Auswirkungen auf die Geschwindigkeit des Programms hat. Synchrone Sprachen hingegen arbeiten den Programmcode wie eine Art Stapel ab. Zeile für Zeile liest der Compiler den Code durch und wandelt ihn in Maschinensprache um. Das lässt dem Programmierer wenig Spielraum für Fehler übrig. Weil die Zeit schon relativ fortgeschritten und die App gröstenteil fertig war entschieden wir uns nur schweren Herzens dazu unser Projekt aufzugeben und die App in Java zu schreiben. In Java schritten wir jedoch viel schneller voran als in JavaScript.

## Quellenverzeichnis

- [1] Wikipedia, the free encyclopedia. <https://de.wikipedia.org/wiki/Höhlenmalerei>  
[Online, zugegriffen 4. Januar 2017]
- [2] Wikipedia, the free encyclopedia. <https://de.wikipedia.org/wiki/Computer>  
[Online, zugegriffen 4. Januar 2017]
- [3] <http://www.ingenieur.de/> Bild der Zuse Z3 [Online, zugegriffen 4. Januar 2017]
- [4] Magic the Gathering - Wizards of the Coast <http://magic.wizards.com/de>  
[Online, zugegriffen 4. Januar 2017]
- [5] Screenshots aus Codeblocks <http://www.codeblocks.org/> und aus texmaker <http://www.xmlmath.net/texmaker/> [online 5.1.2017]
- [6] Wikipedia, the free encyclopedia. <https://de.wikipedia.org/wiki/UTF-8>  
[online 6.1.2017]
- [7] Bild einer ASCII-Tabelle <http://www.theasciicode.com.ar/> [online 6.1.2017]
- [8] [http://www.hbksaar.de/uploads/media/Dateiformate\\_Bilddateien.pdf](http://www.hbksaar.de/uploads/media/Dateiformate_Bilddateien.pdf)  
[online 7.1.2017]
- [9] Android Studio - Graphical User Interface (GUI) <https://www.jetbrains.com/idea/chooseYourEdition> [Online, zugegriffen 19. Januar 2017]

## 7 Anhang

### MainActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Intent;
4 import android.content.SharedPreferences;
5 import android.graphics.Color;
6 import android.os.Bundle;
7 import android.preference.PreferenceManager;
8 import android.support.design.widget.FloatingActionButton;
9 import android.support.design.widget.NavigationView;
10 import android.support.v4.view.GravityCompat;
11 import android.support.v4.widget.DrawerLayout;
12 import android.support.v7.app.ActionBarDrawerToggle;
13 import android.support.v7.app.AppCompatActivity;
14 import android.support.v7.widget.Toolbar;
15 import android.view.Gravity;
16 import android.view.MenuItem;
17 import android.view.View;
18 import android.view.Window;
19 import android.view.WindowManager;
20 import android.widget.ImageView;
21 import android.widget.RelativeLayout;
22 import android.widget.TextView;
23 import com.google.gson.Gson;
24 import com.google.gson.reflect.TypeToken;
25 import com.squareup.picasso.Picasso;
26 import java.lang.reflect.Type;
27 import java.util.ArrayList;
28 import java.util.Arrays;
29 import m2b.magic2brain.com.magic2brain.R;
30
31 /*
32 This is where our app starts. It has a menu, shows a random card and
33 loads some needed data.
34 */
35 public class MainActivity extends AppCompatActivity implements
    NavigationView.OnNavigationItemSelectedListener {
```

```

36
37     public static Card[] cardarray; // We will store all cards in
this array. This is important for the SearchActivity. It makes the
whole process alot faster.
38     private ImageView imgv; // We will show the random card with this
ImageView
39
40     protected void onCreate(Bundle savedInstanceState) { // This is
the very first function the app will execute at the start of the
app.
41         if (!checkCardArray()) { // We try to load our card-list.
42             saveCardArray(); // If we don't find the card-list, we
have to generate it and save it.
43         }
44         hideStatusBar(); // Does what is says. It hides the Statusbar
45         setTheme(R.style.AppTheme.NoActionBar); // We want to hide
the ActionBar too
46         super.onCreate(savedInstanceState); // This does some intern
stuff. We don't need to worry about that. It's just needed.
47         setContentView(R.layout.activity_main); // This adds an View
to our Activity. We defined at "/res/layout/activity_main.xml" how
our activity should look like.
48         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); //
With this we get the Toolbar of the View.
49         setSupportActionBar(toolbar); // We add the Toolbar as a
SupportActionBar. If we click something on the Toolbar it will
call onNavigationItemSelected();
50
51         FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab); // We get the "new random card"-button
52         fab.setOnClickListener(new View.OnClickListener() {
53             public void onClick(View view) {
54                 setListener(setRandomCard()); // If we click the
button it should generate a new random card.
55             }
56         });
57         // The following lines generate the drawer (Sidemenu)
58         DrawerLayout drawer = (DrawerLayout) findViewById(R.id.
drawer_layout);
59         ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
60             this, drawer, toolbar, R.string.

```



```

navigation_drawer_open , R.string.navigation_drawer_close);
61     drawer.setDrawerListener(toggle);
62     toggle.syncState();
63     NavigationView navigationView = (NavigationView) findViewById
(R.id.nav_view);
64     navigationView.setNavigationItemSelectedListener(this);
65
66     Favorites.init(); // Initiate Favorites
67
68     // The following code loads the Favorites from the memory
69     SharedPreferences appSharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
70     Gson gson = new Gson();
71     String json = appSharedPrefs.getString("favobj", null);
72     Type type = new TypeToken<ArrayList<Card>>() {
73     }.getType();
74     ArrayList<Card> favs = gson.fromJson(json, type);
75     Favorites.init();
76     if (favs != null) {
77         Favorites.favorites_mvid = favs;
78     }
79     buildNewsFeed(); // This builds the frontpage
80 }
81
82 private void hideStatusBar() { // This method simply removes the
Statusbar
83     requestWindowFeature(Window.FEATURE_NO_TITLE); // remove the
title
84     getWindow().setFlags(WindowManager.LayoutParams.
FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN); //
set it to fullscreen
85 }
86
87 protected void onPause() {
88     super.onPause();
89     // If the user leaves our app in any way, we save all his
favorites.
90     SharedPreferences appSharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
91     SharedPreferences.Editor prefsEditor = appSharedPrefs.edit();
92     Gson gson = new Gson();

```

```

93     String json = gson.toJson(Favorites.favorites_mvid);
94     prefsEditor.putString("favobj", json);
95     prefsEditor.commit();
96 }
97
98     public void onBackPressed() { // Closes the Drawer if we press
99         // the Back-Button on the phone
100         DrawerLayout drawer = (DrawerLayout) findViewById(R.id.
101             drawer_layout);
102         if (drawer.isDrawerOpen(GravityCompat.START)) {
103             drawer.closeDrawer(GravityCompat.START);
104         } else {
105             super.onBackPressed();
106         }
107     }
108
109     public boolean onNavigationItemSelected(MenuItem item) {
110         // Handle navigation view item clicks here.
111         int id = item.getItemId();
112         if (id == R.id.nav_search) {
113             // Starts the SearchActivity if we click on Search
114             Intent intent = new Intent(this, SearchActivity.class);
115             startActivity(intent);
116
117         } else if (id == R.id.nav_favorite) {
118             // Starts the FavoritesActivity if we click on Favorites
119             Intent intent = new Intent(this, FavoritesActivity.class);
120             startActivity(intent);
121
122         } else if (id == R.id.nav_browser) {
123             // Starts the BrowserActivity if we click on the
124             Cardbrowser
125             Intent intent = new Intent(this, BrowserActivity.class);
126             startActivity(intent);
127
128         } else if (id == R.id.nav_quick_learn) {
129             // Starts the QueryActivity and passes the newest set (
130             Kaladesh), if we click Quick learn
131             Deck d = new Deck();
132             Card[] c = DeckAssetLoader.getDeck("KLD.json", this);

```

```

129         d.setSet(c);
130         d.setName("Kaladesh");
131         d.setCode("KLD");
132         Intent i = new Intent(this, QueryActivity.class);
133         i.putExtra("Set", d);
134         startActivity(i);
135
136     } else if (id == R.id.nav_history) {
137         // Starts the LastSeenActivity if we click recently
138         learned
139         Intent intent = new Intent(this, LastSeenActivity.class);
140         startActivity(intent);
141
142     } else if (id == R.id.nav_share) {
143         // Starts an Activity, which opens all possibilities to
144         send the link to our app.
145         Intent sendIntent = new Intent();
146         sendIntent.setAction(Intent.ACTION_SEND);
147         sendIntent.putExtra(Intent.EXTRA_TEXT, getString(R.string
148         .play_store_link));
149         sendIntent.setType("text/plain");
150         startActivity(Intent.createChooser(sendIntent,
151         getResources().getText(R.string.send_to)));
152     }
153
154     DrawerLayout drawer = (DrawerLayout) findViewById(R.id.
155     drawer_layout); // We get the drawer
156     drawer.closeDrawer(GravityCompat.START); // Th drawer should
157     start closed
158     return true;
159 }
160
161 public void buildNewsFeed() { // This Function generates a text
162     and a Imageview on the screen
163     int scrWidth = getWindowManager().getDefaultDisplay().
164     getWidth(); // Get the width of the screen
165     int scrHeight = getWindowManager().getDefaultDisplay().
166     getHeight(); // Get the Height of the screen
167     RelativeLayout lyt = (RelativeLayout) findViewById(R.id.
168     main_absolute); // Get the View of the XML
169     RelativeLayout.LayoutParams params; // The parameters we want

```

```

160         to add our TextView and ImageView
161         TextView score = new TextView(this); // Create new Textview
162         score.setTextColor(Color.BLACK);
163         score.setGravity(Gravity.CENTER);
164         score.setText("Random Card");
165         score.setTextSize(26);
166         params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
167         (int) (0.1 * scrHeight)/*Height*/);
168         params.leftMargin = 0; // X-Position
169         params.topMargin = (int) (0.1 * scrHeight); // Y-Position
170         lyt.addView(score, params); // add it to the View
171
172         imgv = new ImageView(this); // Create new Imageview
173         params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
174         (int) (0.6 * scrHeight))/*Height*/);
175         params.leftMargin = 0; // X-Position
176         params.topMargin = (int) (0.2 * scrHeight); // Y-Position
177         lyt.addView(imgv, params); // add it to the View
178         setListener(setRandomCard()); // Opens a random card
179     }
180
181     public void setListener(int MultiID2) { // This function sets an
182     listener to the image, so if we click it, it shows a big image of
183     the card.
184         final int MultiID = MultiID2;
185         imgv.setOnClickListener(new View.OnClickListener() {
186             public void onClick(View view) {
187                 Intent intent = new Intent(MainActivity.this,
188                 CardImageDisplayActivity.class);
189                 intent.putExtra("pic", MultiID);
190                 startActivity(intent);
191             }
192         });
193     }
194
195     public int setRandomCard() { // This gets a random card from the
196     card-array and loads the image into the ImageView
197         int MultiID = 1;
198         Card c = cardarray[(int) (Math.random() * cardarray.length)];
199         if (c != null) {

```

```

194         MultiID = c.getMultiverseid();
195     }
196     Picasso.with(this)
197         .load(getString(R.string.image_link_1) + MultiID +
198             getString(R.string.image_link_2)) // This tries to load an image
199             from a link.
200         .placeholder(R.drawable.loading_image) // We want to
201             show a image while its loading. We load our image from the "/res/
202             drawable" folder
203         .error(R.drawable.image_not_found) // If it fails to
204             load image we show an error-image.
205         .into(imgv); // This places the image into our
206             ImageView.
207     return MultiID;
208 }
209
210 public boolean checkCardArray() { // This method checks if there
211     is a saved version of our card array. If that's the case it tries
212     to load it into the card-array.
213     SharedPreferences sharedPrefs = PreferenceManager.
214     getDefaultSharedPreferences(this);
215     Gson gson = new Gson();
216     String json = sharedPrefs.getString("cardArray", null);
217     Type type = new TypeToken<Card[]>() {
218     }.getType();
219     Card[] aL = gson.fromJson(json, type);
220     if (aL == null) {return false;}
221     cardarray = aL;
222     return true;
223 }
224
225 public void saveCardArray() { // This function saves the card-
226     array into the memory.
227     Card[] c = buildCardArray();
228     SharedPreferences sharedPrefs = PreferenceManager.
229     getDefaultSharedPreferences(this);
230     SharedPreferences.Editor editor = sharedPrefs.edit();
231     Gson gson = new Gson();
232     String json = gson.toJson(c);
233     editor.putString("cardArray", json);
234     editor.commit();

```

```

224         cardarray = c;
225     }
226
227     private Card[] buildCardArray() { // This method generates the
card-array by opening all sets and loading the cards from it into
out array.
228         ArrayList<Card> list = new ArrayList<>();
229         Deck[] deckarray = DeckAssetLoader.getDeckList(this);
230         for (int i = 0; i < deckarray.length; i++) {
231             //load current deck and append it to list
232             Card[] c = DeckAssetLoader.getDeck(deckarray[i].getCode()
+ ".json", this);
233             if (c[0].getName().equals("error")) {
234                 System.err.println("error occurred at " + deckarray[i
].getCode() + ", please update your database");
235             } else {
236                 list.addAll(Arrays.asList(c));
237             }
238         }
239         return list.toArray(new Card[list.size()]);
240     }
241 }

```

## SearchActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.support.v7.app.AppCompatActivity;
7 import android.view.MenuItem;
8 import android.view.View;
9 import android.view.View.OnClickListener;
10 import android.widget.Button;
11 import m2b.magic2brain.com.magic2brain.R;
12
13 /*
14 This Activity asks the user if he wants to search for cards or set
15 and opens a SearchHandlerActivity if the answer.
16 */
17 public class SearchActivity extends AppCompatActivity {
18
19     protected void onCreate(Bundle savedInstanceState) {
20         overridePendingTransition(R.anim.slide_in_left, R.anim.
21         slide_out_left); // This adds an fancy slide animation, when this
22         activity starts.
23         super.onCreate(savedInstanceState); // This does some intern
24         stuff. We don't need to worry about that. It's just needed.
25         setContentView(R.layout.activity_search); // This adds an
26         View to our Activity. We defined at "/res/layout/activity_search.
27         xml" how our activity should look like.
28         getSupportActionBar().setDisplayHomeAsUpEnabled(true); //
29         With this line we add an "back"-Button to the Toolbar. If we press
30         it it calls onOptionsItemSelected();
31         final Context context = this; // This doesn't actually do
32         anything, but it's needed if we want to refer to "this" from an
33         inner class.
34         setTitle(getString(R.string.SearchActivity_title)); // We set
35         a title to our View. We defined the title in "/res/values/strings
36         .xml". We just load it from there.
37         Button cards = (Button) findViewById(R.id.c_search_cards); //
38         We get the cards-button
```

```

27     Button decks = (Button) findViewById(R.id.c_search_decks); //
    We get the sets-button
28
29     cards.setOnClickListener(new OnClickListener() {
30         public void onClick(View view) {
31             // If the user presses the cards-button, we start
    SearchHandlerActivity and pass to it, that the user pressed "cards
    "
32             Intent intent = new Intent(context,
    SearchHandlerActivity.class);
33             intent.putExtra("cardsearch", true);
34             startActivity(intent);
35         }
36     });
37
38     decks.setOnClickListener(new OnClickListener() {
39         public void onClick(View view) {
40             // If the user presses the sets-button, we start
    SearchHandlerActivity and pass to it, that the user pressed "sets"
41             Intent intent = new Intent(context,
    SearchHandlerActivity.class);
42             intent.putExtra("cardsearch", false);
43             startActivity(intent);
44         }
45     });
46 }
47
48 public boolean onOptionsItemSelected(MenuItem item) {
49     switch (item.getItemId()) {
50         // Respond to the action bar's Up/Home button
51         case android.R.id.home:
52             onBackPressed();
53     }
54     return true;
55 }
56
57 public void onBackPressed() {
58     finish(); // This closes our Activity
59     overridePendingTransition(R.anim.slide_in_right, R.anim.
    slide_out_right); // We want to close it with an fancy animation.
60 }

```





## SearchHandlerActivity.java

```

1 package m2b. magic2brain. com;
2
3 import android. content. Context;
4 import android. content. Intent;
5 import android. os. Bundle;
6 import android. support. v7. app. AppCompatActivity;
7 import android. text. Editable;
8 import android. text. TextWatcher;
9 import android. view. MenuItem;
10 import android. view. View;
11 import android. widget. AdapterView;
12 import android. widget. ArrayAdapter;
13 import android. widget. EditText;
14 import android. widget. ListView;
15 import m2b. magic2brain. com. magic2brain. R;
16
17 /*
18 This Activity is the one that actually searches for cards/sets.
19 */
20
21 public class SearchHandlerActivity extends AppCompatActivity {
22     Card[] cardarray; // This is the array, where all cards are
        stored
23
24     protected void onCreate(Bundle savedInstanceState) {
25         overridePendingTransition(R. anim. slide_in_ left , R. anim.
        slide_out_ left); // This adds an fancy slide animation, when this
        activity starts.
26         super. onCreate(savedInstanceState); // This does some intern
        stuff. We don't need to worry about that. It's just needed.
27         setContentView(R. layout. activity_ search_ handler); // This
        adds an View to our Activity. We defined at "/res/layout/
        activity_ search_ handler. xml" how our activity should look like.
28         getSupportActionBar().setDisplayHomeAsUpEnabled( true ); //
        With this line we add an "back"—Button to the Toolbar. If we press
        it it calls onOptionsItemSelected();
29
30         final Context context = this; // This doesn't actually do
        anything, but it's needed if we want to refer to "this" from an

```

```

inner class.
31     Intent inte = getIntent(); // We want to access any data
that is passed.
32     final Boolean cardsearch = inte.getBooleanExtra("cardsearch",
true); // We look if the user wants to search for a card or set
33
34     final EditText search_field = (EditText) findViewById(R.id.
search_text); // We get the search-field
35     if (cardsearch) {
36         search_field.setHint(R.string.SearchHandler_hint_1); //
if he wants to search a card, we set the hint to "Cardname"
37     } else {
38         search_field.setHint(R.string.SearchHandler_hint_2); //
if he wants to search a set, we set the hint to "Setname"
39     }
40     final ListView searchresultsview = (ListView) findViewById(R.
id.search_lv); // We get the ListView so we can show the results
41
42     String[] names = new String[1]; // We only need the names so
we create a names-array
43     names[0] = getString(R.string.error); // This is to ensure
that the array isn't empty
44     Deck[] deckarray = DeckAssetLoader.getDeckList(this); // We
get all decks
45
46
47     if (cardsearch) {
48         cardarray = MainActivity.cardarray; // We load our card-
array
49         names = new String[cardarray.length];
50
51         for (int i = 0; i < cardarray.length; i++) {
52             names[i] = cardarray[i].getName(); // And add all
names to the string-array
53         }
54     } else {
55         names = new String[deckarray.length]; // The same like
cards but with sets
56
57         for (int i = 0; i < deckarray.length; i++) {
58             names[i] = deckarray[i].getName(); // The same like

```

```

cards but with sets
59         }
60     }
61
62     //finalizing variables for further use
63     final Deck[] cdeckarray = deckarray;
64     final Card[] ccardarray = cardarray;
65
66
67     final ArrayAdapter adapter = new ArrayAdapter(context ,
android.R.layout.simple_list_item_1 , names); // We turn the string
-array into a list
68     search_field.addTextChangedListener(new TextWatcher() { //
and add a listener , so it updates when we write something
69         public void beforeTextChanged(CharSequence charSequence ,
int i , int i1 , int i2) {}
70         public void onTextChanged(CharSequence charSequence , int
i , int i1 , int i2) {}
71         public void afterTextChanged(Editable editable) {
72             adapter.getFilter().filter(editable.toString().
toLowerCase());
73         }
74     });
75
76     searchresultsview.setAdapter(adapter); // We show the list
77     searchresultsview.setOnItemClickListener(new AdapterView.
OnItemClickListener() {
78         public void onItemClick(AdapterView<?> parent , View view ,
int position , long id) {
79             if (cardsearch) {
80                 // If it's a card-search , we open
CardBrowserActivity with the clicked card.
81                 String cardname = adapter.getItem(position).
toString();
82                 Intent intent = new Intent(context ,
CardBrowserActivity.class);
83                 intent.putExtra("currentCard" , searchforcard(
cardname , ccardarray));
84                 startActivity(intent);
85             } else {
86                 // If it's a set-search , we open

```

### DeckDisplayActivity with the clicked Set

```
87         String deckname = adapter.getItem(position).
toString();
88         Intent intent = new Intent(context,
DeckDisplayActivity.class);
89         intent.putExtra("code", searchfordeck(deckname,
cdeckarray));
90         intent.putExtra("name", deckname);
91         startActivity(intent);
92     }
93 }
94 });
95
96 }
97
98 private Card searchforcard(String name, Card[] cardarray) { //
This function searchs for a card and returns it
99     Card rc = new Card();
100
101     for (int i = 0; i < cardarray.length; i++) {
102         if (cardarray[i].getName().equals(name)) {
103             rc = cardarray[i];
104             break;
105         }
106     }
107     return rc;
108 }
109
110 private String searchfordeck(String name, Deck[] darray) { //
This function searchs for a set and returns it
111     String code = getString(R.string.error);
112
113     for (int i = 0; i < darray.length; i++) {
114         String dname = darray[i].getName();
115
116         if (dname.equals(name)) {
117             code = darray[i].getCode();
118             break;
119         }
120     }
121 }
```

```

122         return code;
123     }
124
125     public boolean onOptionsItemSelected(MenuItem item) {
126         int id = item.getItemId();
127         switch (id) {
128             // Respond to the action bar's Up/Home button
129             case android.R.id.home:
130                 onBackPressed();
131         }
132         return true;
133     }
134
135     public void onBackPressed() {
136         finish(); // This closes our Activity
137         overridePendingTransition(R.anim.slide_in_right, R.anim.
138             slide_out_right); // We want to close it with an fancy animation.
139     }

```

## BrowserActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.MenuItem;
9 import android.view.View;
10 import android.widget.AdapterView;
11 import android.widget.AdapterView.OnItemClickListener;
12 import android.widget.ArrayAdapter;
13 import android.widget.ListView;
14 import m2b.magic2brain.com.magic2brain.R;
15
16 /*
17 This Activity should show a List with all sets and open
18 DeckDisplayActivity with the selected set on a click.
19 */
20 public class BrowserActivity extends AppCompatActivity {
21
22     protected void onCreate(Bundle savedInstanceState) {
23         overridePendingTransition(R.anim.slide_in_left, R.anim.
24             slide_out_left); // This adds an fancy slide animation, when this
25             activity starts.
26         super.onCreate(savedInstanceState); // This does some intern
27             stuff. We don't need to worry about that. It's just needed.
28         setContentView(R.layout.activity_browser); // This adds an
29             View to our Activity. We defined at "/res/layout/activity_browser.
30             xml" how our activity should look like.
31         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); //
32             With this we get the Toolbar of the View.
33         setSupportActionBar(toolbar); // We add the Toolbar as a
34             SupportActionBar. This is needed for the next line.
35         getSupportActionBar().setDisplayHomeAsUpEnabled(true); //
36             With this line we add an "back"-Button to the Toolbar. If we press
37             it it calls onOptionsItemSelected();
38         setTitle(getString(R.string.BrowserActivity_title)); // We
```

```

set a title to our View. We defined the title in "/res/values/
strings.xml". We just load it from there.
30     final Context currentContext = this; // This doesn't actually
        do anything, but it's needed if we want to refer to "this" from
        an inner class.
31
32     Deck d[] = DeckAssetLoader.getDeckList(this); // We load all
        Sets with our own function into an Deck-Array.
33     String[] it = new String[d.length]; // We just need the names
        of the sets. So we create an String-Array for the names.
34     for (int i = 0; i < d.length; i++) {
35         it[i] = d[i].getName(); // This for-loop adds the names
        from every Deck from the Deck-Array to the String-Array.
36     }
37
38     final String[] listItems = it; // We need to finalize the
        String-Array for the ArrayAdapter.
39     final Deck[] finalD = d; // We need to finalize the Deck-
        Array, because we want to access it from an inner class.
40     final ArrayAdapter adapter = new ArrayAdapter(this, android.R
        .layout.simple_list_item_1, listItems); // This is just a helper-
        class. It transforms our String-Array into an clickable List.
41     ListView lv = (ListView) findViewById(R.id.deckbrowser_list);
        // We want to add our list to the ListView. So we get the
        ListView from the View
42     lv.setAdapter(adapter); // We add our list into it.
43
44
45     lv.setOnItemClickListener(new OnItemClickListener() { // This
        performs an action when we click on an item from the list.
46         public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
47             // The following code starts DeckDisplayActivity and
            passes the Deck-Code and the Deck-Name of the item we clicked.
48             Intent intent = new Intent(currentContext,
                DeckDisplayActivity.class);
49             intent.putExtra("code", finalD[position].getCode());
50             intent.putExtra("name", finalD[position].getName());
51             startActivity(intent);
52         }
53     });

```



```
54     }
55
56     public boolean onOptionsItemSelected(MenuItem item) {
57         switch (item.getItemId()) {
58             // Respond to the action bar's Up/Home button
59             case android.R.id.home:
60                 onBackPressed();
61             }
62         return true;
63     }
64
65     public void onBackPressed() {
66         finish(); // This closes our Activity
67         overridePendingTransition(R.anim.slide_in_right, R.anim.
68         slide_out_right); // We want to close it with an fancy animation.
69     }
70 }
```

## DeckDisplayActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import android.content.DialogInterface;
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.support.design.widget.FloatingActionButton;
8 import android.support.v7.app.AlertDialog;
9 import android.support.v7.app.AppCompatActivity;
10 import android.view.MenuItem;
11 import android.view.View;
12 import android.widget.AdapterView;
13 import android.widget.AdapterView.OnItemClickListener;
14 import android.widget.ArrayAdapter;
15 import android.widget.ListView;
16 import java.util.ArrayList;
17 import java.util.Arrays;
18 import java.util.List;
19 import m2b.magic2brain.com.magic2brain.R;
20
21 /*
22 This class shows all cards of a deck in a list
23 */
24 public class DeckDisplayActivity extends AppCompatActivity {
25
26     protected void onCreate(Bundle savedInstanceState) {
27         overridePendingTransition(R.anim.slide_in_left, R.anim.
28         slide_out_left); // This adds an fancy slide animation, when this
29         activity starts.
30         super.onCreate(savedInstanceState); // This does some intern
31         stuff. We don't need to worry about that. It's just needed.
32         setContentView(R.layout.activity_deck_display); // This adds
33         an View to our Activity. We defined at "/res/layout/
34         activity_deck_display.xml" how our activity should look like.
35         getSupportActionBar().setDisplayHomeAsUpEnabled(true); //
36         With this line we add an "back"-Button to the Toolbar. If we press
37         it it calls onOptionsItemSelected();
38
39         final Context currentContext = this; // This doesn't actually
```

```

do anything, but it's needed if we want to refer to "this" from
an inner class.
33     Intent intent = getIntent(); // We want to access any data
that is passed.
34     final String deckcode = intent.getStringExtra("code"); // we
get the deck-code
35     final String name = intent.getStringExtra("name"); // we get
the deck-name
36
37     setTitle(name); // We set the title of the activity to the
name of the deck
38     Card c[] = DeckAssetLoader.getDeck(deckcode + ".json", this);
// This gets all the cards from the deck
39     final Card[] cCopy = c; // and adds it to an Array
40
41     FloatingActionButton fam = (FloatingActionButton)
findViewById(R.id.fab_setlearn); // This gets the "Learn"-Button
42     fam.setOnClickListener(new View.OnClickListener() { // and
adds a listener to it
43         public void onClick(View view) {
44             // The following code starts the QueryActivity and
passes the deck
45             Intent intent = new Intent(currentContext,
QueryActivity.class);
46             Deck d = new Deck();
47             List<Card> clist = Arrays.asList(cCopy);
48             d.setCode(deckcode);
49             d.setName(name);
50             d.setSet(new ArrayList<Card>(clist));
51             intent.putExtra("Set", d);
52             startActivity(intent);
53         }
54     });
55
56     ListView lv = (ListView) findViewById(R.id.deckdisplay); //
We get the Listview
57
58     if (c[0] == null) { // If there are no cards in the deck, we
show an error message
59         AlertDialog.Builder dlgAlert = new AlertDialog.Builder(
this);

```

```

60         dlgAlert.setMessage("Sadly, this Deck was not Found");
61         dlgAlert.setTitle("Error");
62         dlgAlert.setPositiveButton("I understand, bill me your
server costs",
63             new DialogInterface.OnClickListener() {
64                 public void onClick(DialogInterface dialog,
int which) {
65                     //dismiss the dialog
66                     onBackPressed();
67                 }
68             });
69         dlgAlert.setCancelable(true);
70         dlgAlert.create().show();
71
72         lv.setVisibility(View.GONE);
73         fam.setVisibility(View.GONE);
74     } else { // If there are some cards in the deck, we generate
a card-list like we did in various other Activities
75         final String[] listItems = RUtils.getListified(c);
76         final ArrayAdapter adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, listItems);
77         lv.setAdapter(adapter);
78
79         final Card[] finalC = c;
80
81         lv.setOnItemClickListener(new OnItemClickListener() {
82             public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
83                 Intent intent = new Intent(currentContext,
CardBrowserActivity.class);
84                 intent.putExtra("currentCard", finalC[position]);
85                 startActivity(intent);
86             }
87         });
88     }
89 }
90
91 public boolean onOptionsItemSelected(MenuItem item) {
92     int id = item.getItemId();
93     switch (id) {
94         // Respond to the action bar's Up/Home button

```

```
95         case android.R.id.home:
96             onBackPressed();
97         }
98         return true;
99     }
100
101     public void onBackPressed() {
102         finish(); // This closes our Activity
103         overridePendingTransition(R.anim.slide_in_right, R.anim.
104             slide_out_right); // We want to close it with an fancy animation.
105     }
```

## CardBrowserActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.graphics.Color;
6 import android.os.Bundle;
7 import android.support.design.widget.FloatingActionButton;
8 import android.support.design.widget.Snackbar;
9 import android.support.v7.app.AppCompatActivity;
10 import android.support.v7.widget.Toolbar;
11 import android.util.TypedValue;
12 import android.view.Gravity;
13 import android.view.View;
14 import android.widget.ImageView;
15 import android.widget.LinearLayout;
16 import android.widget.TextView;
17 import com.squareup.picasso.Picasso;
18 import java.util.ArrayList;
19 import java.util.Arrays;
20 import m2b.magic2brain.com.magic2brain.R;
21
22 /*
23 This activity should show one card with all informations
24 */
25 public class CardBrowserActivity extends AppCompatActivity {
26
27     ImageView cImage; // We will store our ImageView in a global
28     // variable, because we want to access it from various methods.
29
30     protected void onCreate(final Bundle savedInstanceState) {
31         // overridePendingTransition(R.anim.slide_in_left, R.anim.
32         // slide_out_left); // This adds a fancy slide animation, when this
33         // activity starts.
34         final Context context = this; // This doesn't actually do
35         // anything, but it's needed if we want to refer to "this" from an
36         // inner class.
37         super.onCreate(savedInstanceState); // This does some intern
38         // stuff. We don't need to worry about that. It's just needed.
```

```

34     Intent mIntent = getIntent(); // We want to access any data
    that is passed.
35     final Card card = (Card) mIntent.getSerializableExtra("
    currentCard"); // We load the card-object from the Intent
36
37     setContentView(R.layout.activity_card_browser); // This adds
    an View to our Activity. We defined at "/res/layout/
    activity_card_browser.xml" how our activity should look like.
38     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); //
    With this we get the Toolbar of the View.
39     toolbar.setTitle(card.getName()); // We set a title to our
    View. The title should be the name of the card.
40     setSupportActionBar(toolbar); // We add the Toolbar as a
    SupportActionBar.
41
42     cImage = (ImageView) findViewById(R.id.cbImage); // We load
    our ImageView from the View
43     cImage.setPadding(0, 10, 0, 10); // We add a padding to it.
44
45     // The next couple lines builds a GUI with the informations
    from the card. We access all Views from our View.
46     TextView tv = (TextView) findViewById(R.id.cbaInfo);
47     String info = "";
48     info += " Name: " + card.getName();
49     info += "\n MVID: " + card.getMultiverseid();
50     tv.setText(info);
51     tv.setTextColor(Color.BLACK);
52     tv.setPadding(0, 10, 0, 10);
53
54     TextView text = (TextView) findViewById(R.id.cbaText);
55     text.setText(card.getText());
56     text.setTextColor(Color.BLACK);
57     text.setPadding(20, 10, 10, 20);
58
59     TextView flavor = (TextView) findViewById(R.id.cbaFlavor);
60     flavor.setText(card.getFlavor());
61     flavor.setTextColor(Color.BLACK);
62     flavor.setPadding(20, 10, 10, 20);
63
64     TextView type = (TextView) findViewById(R.id.cbaType);
65     type.setText(card.getType());

```

```

66         type.setTextColor(Color.BLACK);
67         type.setPadding(0, 10, 0, 10);
68
69         LinearLayout ll = (LinearLayout) findViewById(R.id.
cba_mcost_layout);
70         setManaCost(card.getManaCost(), ll);
71         ll.setGravity(Gravity.CENTER);
72         ll.setPadding(0, 10, 0, 10);
73
74         final int mvid = card.getMultiverseid(); // This gets the
MultiverseID from the card
75         showPic(mvid); // and loads the picture of the card.
76
77         clImage.setOnClickListener(new View.OnClickListener() { //
This performs an action if we click the image
78             public void onClick(View view) {
79                 // The following code starts the
CardImageDisplayActivity and passes the MultiverseID of the Card.
80                 Intent intent = new Intent(context,
CardImageDisplayActivity.class);
81                 intent.putExtra("pic", mvid);
82                 startActivity(intent);
83             }
84         });
85
86         final FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab); // This gets our "add to favorites"—button
from the view
87         // We check if its already added to favorites. If so we mark
it as already added.
88         if (!checkCard(card.getName())) {
89             fab.setImageResource(R.drawable.ic_favorite_border);
90         } else {
91             fab.setImageResource(R.drawable.ic_favorite);
92         }
93
94         fab.setOnClickListener(new View.OnClickListener() { // This
performs an action if we click the "add to favorites"—Button
95             // The following code toggles the Button, so it shows
that the card was added or not, gives a notification and add the
card to the favorites—list (which will be saved).

```



```

96         public void onClick(View view) {
97             if (checkCard(card.getName())) {
98                 removeCard(card.getName());
99                 fab.setImageResource(R.drawable.ic_favorite_border);
100                 Snackbar.make(view, R.string.remove_fav, Snackbar
101                     .LENGTH_LONG)
102                     .setAction("Action", null).show();
103             } else {
104                 Favorites.favorites_mvid.add(card);
105                 fab.setImageResource(R.drawable.ic_favorite);
106                 Snackbar.make(view, R.string.add_fav, Snackbar.
107                     LENGTH_LONG)
108                     .setAction("Action", null).show();
109             }
110         }
111     });
112 }
113
114 private boolean checkCard(String name) { // This function checks
115     if the card was added to the favorites
116     for (Card c : Favorites.favorites_mvid) {
117         if (c.getName().contains(name)) {
118             return true;
119         }
120     }
121     return false;
122 }
123
124 private void removeCard(String name) { // This function removes
125     the card from the favorites-list
126     ArrayList<Card> cards = new ArrayList<>();
127     for (Card c : Favorites.favorites_mvid) {
128         if (c.getName().contains(name)) {
129             cards.add(c);
130         }
131     }
132     for (Card c : cards) {
133         Favorites.favorites_mvid.remove(c);
134     }
135 }

```

```

132
133     private void setManaCost(String manatext, LinearLayout layout) {
134         // This function adds the mana-cost to the View by loading the
135         // images.
136         manatext = manatext.replaceAll("\\{", "");
137         manatext = manatext.replaceAll("\\}", "");
138         String[] items = manatext.split("");
139         items = Arrays.copyOfRange(items, 1, items.length);
140
141         for (int i = 0; i < items.length; i++) {
142             if (RUtils.isInteger(items[i])) {
143                 TextView tv = new TextView(this);
144                 tv.setText(items[i]);
145                 tv.setTextSize(TypedValue.COMPLEX_UNIT_PX, 80);
146                 tv.setTextColor(Color.BLACK);
147                 layout.addView(tv);
148             } else {
149                 ImageView imgv = new ImageView(this);
150                 int resid = 0;
151                 if (items[i].equals("B")) {
152                     resid = R.drawable.b;
153                 } else if (items[i].equals("C")) {
154                     resid = R.drawable.c;
155                 } else if (items[i].equals("G")) {
156                     resid = R.drawable.g;
157                 } else if (items[i].equals("R")) {
158                     resid = R.drawable.r;
159                 } else if (items[i].equals("U")) {
160                     resid = R.drawable.u;
161                 } else if (items[i].equals("W")) {
162                     resid = R.drawable.w;
163                 } else {
164                     resid = R.drawable.ic_action_cancel;
165                 }
166                 imgv.setBackgroundResource(resid);
167                 layout.addView(imgv);
168                 android.view.ViewGroup.LayoutParams layoutParams =
169                 imgv.getLayoutParams();
170                 layoutParams.width = 60;
171                 layoutParams.height = 60;
172                 imgv.setLayoutParams(layoutParams);

```

```

170     }
171 }
172 }
173
174 private void showPic(int MultiID) { // This method loads an
cardimage into clmage with a given MultiverseID.
175     Picasso.with(this)
176         .load(getString(R.string.image_link_1) + MultiID +
getString(R.string.image_link_2)) // This tries to load an image
from a link.
177         .placeholder(R.drawable.loading_image) // We want to
show a image while its loading. We load our image from the "/res/
drawable" folder
178         .error(R.drawable.image_not_found) // If it fails to
load image we show an error-image.
179         .into(clmage); // This places the image into our
ImageView.
180     }
181
182 public void onBackPressed() {
183     finish(); // This closes our Activity
184     overridePendingTransition(R.anim.slide_in_right, R.anim.
slide_out_right); // We want to close it with an fancy animation.
185 }
186 }

```

## CardImageDisplayActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.os.Handler;
6 import android.support.v7.app.ActionBar;
7 import android.support.v7.app.AppCompatActivity;
8 import android.view.View;
9 import android.widget.ImageView;
10 import com.squareup.picasso.Picasso;
11 import m2b.magic2brain.com.magic2brain.R;
12
13 /*
14 This Activity should show a big image of a card
15 */
16
17 public class CardImageDisplayActivity extends AppCompatActivity {
18     private final Handler mHideHandler = new Handler();
19     private View mContentView;
20     private final Runnable mHidePart2Runnable = new Runnable() {
21         public void run() {
22             // Delayed removal of status and navigation bar
23             // Note that some of these constants are new as of API 16
24             // (Jelly Bean)
25             // and API 19 (KitKat). It is safe to use them, as they
26             // are inlined
27             // at compile-time and do nothing on earlier devices.
28             mContentView.setSystemUiVisibility(View.
29 SYSTEM_UI_FLAG_LOW_PROFILE
30 | View.SYSTEM_UI_FLAG_FULLSCREEN
31 | View.SYSTEM_UI_FLAG_LAYOUT_STABLE
32 | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY
33 | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
34 | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
35         }
36     };
37     private View mControlsView;
38     private final Runnable mShowPart2Runnable = new Runnable() {
39         @Override
```

```

37     public void run() {
38         // Delayed display of UI elements
39         ActionBar actionBar = getSupportActionBar();
40         if (actionBar != null) {
41             actionBar.show();
42         }
43         mControlsView.setVisibility(View.VISIBLE);
44     }
45 };
46
47 protected void onCreate(Bundle savedInstanceState) {
48     super.onCreate(savedInstanceState);
49     setContentView(R.layout.activity_card_image_display);
50
51     mControlsView = findViewById(R.id.fullscreen_content_controls
52 );
53     mContentView = findViewById(R.id.fullscreen_content);
54     Intent mIntent = getIntent();
55
56     // Set up the user interaction to manually show or hide the
57     // system UI.
58     mContentView.setOnClickListener(new View.OnClickListener() {
59         public void onClick(View view) {
60             finish();
61         }
62     });
63
64     // Upon interacting with UI controls, delay any scheduled
65     // hide()
66     // operations to prevent the jarring behavior of controls
67     // going away
68     // while interacting with the UI.
69     hide();
70
71     int mvid = mIntent.getIntExtra("pic", 0);
72     showPic(mvid);
73 }
74
75 private void showPic(int MultiID) {
76     ImageView cImage = (ImageView) findViewById(R.id.cida_imgview
77 );

```

```

73         Picasso.with(this)
74             .load(getString(R.string.image_link_1) + MultiID +
getString(R.string.image_link_2))
75             .placeholder(R.drawable.loading_image)
76             .error(R.drawable.image_not_found)
77             .into(cImage);
78     }
79
80     private void hide() {
81         // Hide UI first
82         ActionBar actionBar = getSupportActionBar();
83         if (actionBar != null) {
84             actionBar.hide();
85         }
86         mControlsView.setVisibility(View.GONE);
87         // Schedule a runnable to remove the status and navigation
bar after a delay
88         mHideHandler.removeCallbacks(mShowPart2Runnable);
89         mHidePart2Runnable.run();
90     }
91 }

```

## QueryActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Intent;
4 import android.content.SharedPreferences;
5 import android.graphics.Color;
6 import android.graphics.PorterDuff;
7 import android.os.Bundle;
8 import android.os.Handler;
9 import android.preference.PreferenceManager;
10 import android.support.v7.app.AppCompatActivity;
11 import android.support.v7.widget.Toolbar;
12 import android.view.Gravity;
13 import android.view.KeyEvent;
14 import android.view.Menu;
15 import android.view.MenuItem;
16 import android.view.View;
17 import android.view.WindowManager;
18 import android.view.animation.AccelerateInterpolator;
19 import android.view.animation.Animation;
20 import android.view.animation.AnimationUtils;
21 import android.view.animation.DecelerateInterpolator;
22 import android.widget.Button;
23 import android.widget.EditText;
24 import android.widget.ImageView;
25 import android.widget.RelativeLayout;
26 import android.widget.TextView;
27 import com.google.gson.Gson;
28 import com.google.gson.reflect.TypeToken;
29 import com.squareup.picasso.Picasso;
30 import java.lang.reflect.Type;
31 import java.util.ArrayList;
32 import java.util.Collections;
33 import java.util.Random;
34 import m2b.magic2brain.com.magic2brain.R;
35
36 public class QueryActivity extends AppCompatActivity {
37     private ImageView imgv; // The image of the card gets stored here
38     private ImageView imgCorr; // Is over the image. It's to indicate
39     if the answer was correct or not
```

```

39     private Toolbar hiding; // This is a bar that hides a certain
area
40     private ArrayList<Card> set; //This ArrayList holds all Cards
that need to query. It won't be edited (after loading it)
41     private ArrayList<Card> wrongGuessed;
42     private int indexCard; // Actually not needed (because we remove
cards from WrongGuessed) but may be useful in later edits
43     private boolean firstGuess; //This is to check if he guessed it
at first try. If so we remove the card from the ArrayList. Else it
stays there.
44     private String deckName; //Name of the deck. Only for saving/
loading purpose
45     private TextView score; //this will show the user the current
progress
46     private boolean queryLand = true; // should we really query lands
?
47     private boolean skipped = false; // this is to store if the user
has skipped or not
48     private ArrayList<String> recentlyLearned; // this is to save the
deckname if a new set is learned
49     private ArrayList<String> recentlyLearnedNames; // the name of
the sets
50     private int Mode; // the query mode
51     private ArrayList<Button> choices; // We store the buttons here
52
53     protected void onCreate(Bundle savedInstanceState) {
54         // Standard stuff
55         overridePendingTransition(R.anim.slide_in_left, R.anim.
slide_out_left); // This adds an fancy slide animation, when this
activity starts.
56         super.onCreate(savedInstanceState); // This does some intern
stuff. We don't need to worry about that. It's just needed.
57         setContentView(R.layout.activity_query); // This adds an View
to our Activity. We defined at "/res/layout/activity_query.xml"
how our activity should look like.
58         // Hide the status bar.
59         this.getWindow().setFlags(WindowManager.LayoutParams.
FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
60         // Get some informations
61         Intent i = getIntent();
62         Deck qur = (Deck) i.getSerializableExtra("Set");

```



```

63     deckName = qur.getName(); // Get the name of the Deck
64     String code = qur.getCode(); // Get the Deck-Code
65     if (deckName == null) {
66         deckName = "DEFAULT"; // If deckname isn't defined, we
set it to "DEFAULT"
67     }
68     setTitle(deckName); // we set the title to the deckname
69     //Add set to recent learned
70     if (!loadRecent()) {
71         recentlyLearned = new ArrayList<>();
72         recentlyLearnedNames = new ArrayList<>();
73     }
74     if (!recentlyLearned.contains(code)) {
75         recentlyLearned.add(code);
76         recentlyLearnedNames.add(deckName);
77     }
78     if (recentlyLearned.size() == 10) {
79         recentlyLearned.remove(0);
80         recentlyLearnedNames.remove(0);
81     }
82     saveRecent();
83     //load set
84     set = qur.getSet();
85     if (!loadProgress()) { //First we try to load the progress.
If this fails, we simply start over
86         wrongGuessed = (ArrayList) set.clone(); //Lets assume he
guessed everything wrong and remove the card of this Array when he
guesses it right
87         shuffleWrongs(); //Shuffle it a bit (better learn-effect)
88         indexCard = 0; // We start at card No. 1
89     }
90     //Set Mode
91     Mode = 1;
92     if (set.size() < 4) {
93         Mode = 0;
94     }
95     // Build UI
96     hiding = (Toolbar) findViewById(R.id.toolbar_query);
97     getSupportActionBar().setDisplayHomeAsUpEnabled(true); //
With this line we add an "back"-Button to the Toolbar. If we press
it it calls onOptionsItemSelected();

```

```

98         buildMenu();
99         //Start the query
100        showFirstPic();
101        if (deckName.contains("DEFAULT") || deckName.contains("
Favorites")) {
102            restartAll(); // Because this decks are dynamic
103        }
104    }
105
106    protected void onPause() { // We save the progress when the user
leaves
107        super.onPause();
108        saveProgress();
109    }
110
111    public boolean loadRecent() { // This loads all recently learned
sets
112        SharedPreferences sharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
113        Gson gson = new Gson();
114        String json = sharedPrefs.getString("query_recent4", null);
115        String json2 = sharedPrefs.getString("query_recent_names",
null);
116        Type type = new TypeToken<ArrayList<String>>() {
117        }.getType();
118        ArrayList<String> aL = gson.fromJson(json, type);
119        ArrayList<String> aL2 = gson.fromJson(json2, type);
120        if (aL == null) {
121            return false;
122        }
123        recentlyLearned = aL;
124        recentlyLearnedNames = aL2;
125        return true;
126    }
127
128    public void saveRecent() { // This saves all recently learned
sets
129        SharedPreferences sharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
130        SharedPreferences.Editor editor = sharedPrefs.edit();
131        Gson gson = new Gson();

```

```

132     String json = gson.toJson(recentlyLearned);
133     String json2 = gson.toJson(recentlyLearnedNames);
134     editor.putString("query_recent4", json);
135     editor.putString("query_recent_names", json2);
136     editor.commit();
137 }
138
139 public boolean loadProgress() { // This loads the progress of the
    current deck
140     SharedPreferences sharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
141     Gson gson = new Gson();
142     String json = sharedPrefs.getString("query_list_" + deckName,
null);
143     Type type = new TypeToken<ArrayList<Card>>() {
    }.getType();
144     ArrayList<Card> aL = gson.fromJson(json, type);
145     int loadedIndex = sharedPrefs.getInt("query_index_" +
deckName, -1);
146     if (aL == null) {
147         return false;
148     }
149     wrongGuessed = aL;
150     indexCard = loadedIndex;
151     return true;
152 }
153
154
155 public void saveProgress() { // This saves the progress of the
    current deck
156     SharedPreferences sharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
157     SharedPreferences.Editor editor = sharedPrefs.edit();
158     Gson gson = new Gson();
159     String json = gson.toJson(wrongGuessed);
160     editor.putString("query_list_" + deckName, json);
161     editor.putInt("query_index_" + deckName, indexCard);
162     editor.commit();
163 }
164
165 public boolean onCreateOptionsMenu(Menu menu) {
166     // Inflate the menu; this adds items to the action bar if it

```

```

is present.
167         getMenuInflater().inflate(R.menu.query, menu);
168         return true;
169     }
170
171     public void shuffleWrongs() { // This shuffles the wrongGuessed-
ArrayList
172         Collections.shuffle(wrongGuessed, new Random(System.nanoTime
()));
173     }
174
175     public void showPic(int MultiID) { // This method loads an
cardimage into imgv with a given MultiverseID.
176         Picasso.with(this)
177             .load(getString(R.string.image_link_1) + MultiID +
getString(R.string.image_link_2)) // This tries to load an image
from a link.
178             .placeholder(R.drawable.loading_image) // We want to
show a image while its loading. We load our image from the "/res/
drawable" folder
179             .error(R.drawable.image_not_found) // If it fails to
load image we show an error-image.
180             .into(imgv); // This places the image into our
ImageView.
181     }
182
183     public void showFirstPic() { // We need to call this, if we start
with the first item
184         updateScore(); // This will update the shown score
185         showHiderInstant(); // This will instantly hide a part of the
card
186         firstGuess = true; // The user didn't guess yet, so it's his
first guess
187         showPic(wrongGuessed.get(indexCard).getMultiverseid()); // We
show the current picture
188         hiding.bringToFront(); // We want the hider in the front
189         imgCorr.bringToFront(); // And imgCorr aswell, so the users
sees these
190         if (Mode == 1) {
191             updateChoices(); // If the users uses mode one, we need
to update the text of the buttons

```

```

192     }
193 }
194
195 public void showNextPic() {
196     updateScore(); // This will update the shown score
197     showHider(); // We fade the hider in
198     final Handler handler = new Handler();
199     handler.postDelayed(new Runnable() {
200         public void run() { // We add a little delay of 800
milliseconds, because of the animation
201             firstGuess = true; // The user didn't guess yet, so
it's his first guess
202             showPic(wrongGuessed.get(indexCard).getMultiverseid()
); // We show the current picture
203             hiding.bringToFront(); // We want the hider in the
front
204             imgCorr.bringToFront(); // And imgCorr aswell, so the
users sees these
205             skipped = false; // The user didn't skip the current
card
206         }
207     }, 800);
208     if (Mode == 1) {
209         updateChoices(); // If the users uses mode one, we need
to update the text of the buttons
210     }
211 }
212
213 public void checkAnswer(String txt) { // This method checks the
answer
214     if (txt.replaceAll("\\s+", "").equalsIgnoreCase(wrongGuessed.
get(indexCard).getName().replaceAll("\\s+", ""))) {
215         if (firstGuess) {
216             wrongGuessed.remove(indexCard); //if he guessed it, we
remove it.
217         }
218         else {
219             indexCard++; // else we continue with the next card
220         }
221         if (indexCard == wrongGuessed.size()) { //If this true he
's through the set

```

```

222         final Handler handler = new Handler();
223         handler.postDelayed(new Runnable() { // Add a delay
of 1 second
224             public void run() {
225                 setDone();
226             }
227         }, 1000);
228     } else {
229         if (!skipped) {
230             imgCorr.setImageResource(R.drawable.
correct_answer); // If the user didn't skip we show the "correct"-
symbol
231             showImgCorr();
232         }
233         final Handler handler = new Handler();
234         handler.postDelayed(new Runnable() { // Add a delay
of 800 milliseconds
235             public void run() {
236                 if (!skipped) {
237                     hideImgCorr(); // We hide the symbol
again
238                 }
239                 showNextPic(); // And show the next card
240             }
241             }, 800);
242         }
243     } else {
244         wrongAnswer(); // If the user didn't guess correctly , we
call wrongAnswer()
245     }
246 }
247
248 public void skip() { // Skips the card with an animation
249     if (!skipped) {
250         skipped = true; // The user skipped we set skipped to
true
251         wrongAnswer(); // The user didn't guess correctly
252         final Handler handler = new Handler();
253         handler.postDelayed(new Runnable() {
254             public void run() {
255                 checkAnswer(wrongGuessed.get(indexCard).getName())

```

```

    ); // after a delay of 600 milliseconds we simulate a correct
    answer.
256         }
257     }, 600);
258 }
259 }
260
261 public void wrongAnswer() {
262     firstGuess = false; // The user didn't guess it on the first
    time
263     imgCorr.setImageResource(R.drawable.wrong_answer); // We want
    to show the "Wrong"-image
264     showImgCorr(); // We show the image
265     hideHider(); // We show the correct answer
266     final Handler handler = new Handler();
267     handler.postDelayed(new Runnable() {
268         public void run() {
269             hideImgCorr(); // We hide the image after an delay of
    1 second
270         }
271     }, 1000);
272 }
273
274 public void setDone() { // If the set is done, we show the
    endscreen and reshuffle the wrongGuessed-arraylist
275     buildEndScreen();
276     shuffleWrongs();
277 }
278
279 public void hideHider() { // Shows the name of the card fadingly
280     hiding.animate().alpha(0).setDuration(600).setInterpolator(
    new DecelerateInterpolator()).withEndAction(new Runnable() {
281         public void run() {
282             hiding.animate().alpha(0).setDuration(1000).
    setInterpolator(new AccelerateInterpolator()).start();
283         }
284     }).start();
285 }
286
287 public void showHider() { // Hides the name of the card fadingly
288     hiding.animate().alpha(1).setDuration(600).setInterpolator(

```

```

289     new DecelerateInterpolator()).withEndAction(new Runnable() {
290         public void run() {
291             hiding.animate().alpha(1).setDuration(100).
292             setInterpolator(new AccelerateInterpolator()).start();
293         }
294     }).start();
295
296     public void showHiderInstant() { // Hides the name of the card
297         instantly
298         hiding.animate().alpha(1).setDuration(10).setInterpolator(new
299         DecelerateInterpolator()).withEndAction(new Runnable() {
300             public void run() {
301                 hiding.animate().alpha(1).setDuration(100).
302                 setInterpolator(new AccelerateInterpolator()).start();
303             }
304         }).start();
305     }
306
307     public void hideImgCorr() {
308         Animation myFadeInAnimation = AnimationUtils.loadAnimation(
309         this, R.anim.fadeout);
310         imgCorr.startAnimation(myFadeInAnimation); //Set animation to
311         your ImageView
312     }
313
314     public void showImgCorr() {
315         Animation myFadeInAnimation = AnimationUtils.loadAnimation(
316         this, R.anim.fadein);
317         imgCorr.startAnimation(myFadeInAnimation); //Set animation to
318         your ImageView
319     }
320
321     public void buildMenu() { // This method simply build the UI
322         int scrWidth = getWindowManager().getDefaultDisplay().
323         getWidth(); // Get the width of the screen
324         int scrHeight = getWindowManager().getDefaultDisplay().
325         getHeight(); // Get the height of the screen
326         RelativeLayout lyt = (RelativeLayout) findViewById(R.id.
327         query_absolute); // Get the View of the XML
328         RelativeLayout.LayoutParams params; // The parameters we want

```



```

to add our views
318     lyt.removeAllViews(); //Clear the Board
319
320     imgv = new ImageView(this); // Create new Imageview
321     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.5 * scrHeight))/*Height*/;
322     params.leftMargin = 0; // X-Position
323     params.topMargin = (int) (0.02 * scrHeight); // Y-Position
324     lyt.addView(imgv, params); // add it to the View
325
326     imgCorr = new ImageView(this); // Create new Imageview
327     hideImgCorr(); // Hide this ImageView
328     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.5 * scrHeight))/*Height*/;
329     params.leftMargin = 0; // X-Position
330     params.topMargin = (int) (0.02 * scrHeight); // Y-Position
331     lyt.addView(imgCorr, params); // add it to the View
332
333     switch (Mode) {
334         case 0:
335             buildMode0(); // If we have Mode 0, we continue
building with Mode 0
336             break;
337         case 1:
338             buildMode1(); // else we build Mode 1
339     }
340 }
341
342 public void buildMode0() { // This method builds the UI for Mode
0
343     int scrWidth = getWindowManager().getDefaultDisplay().
getWidth(); // Get the width of the screen
344     int scrHeight = getWindowManager().getDefaultDisplay().
getHeight(); // Get the height of the screen
345     RelativeLayout lyt = (RelativeLayout) findViewById(R.id.
query_absolute); // Get the View of the XML
346     RelativeLayout.LayoutParams params; // The parameters we want
to add our views
347
348     params = new RelativeLayout.LayoutParams((int) (0.55 *
scrWidth) /*Width*/, (int) (0.03 * scrHeight))/*Height*/);

```

```

349     params.leftMargin = (scrWidth / 2 - (int) (0.55 * scrWidth) /
2); // X-Position
350     params.topMargin = (int) (0.045 * scrHeight); // Y-Position
351     lyt.addView(hiding, params); // We add the hider
352
353     // Add EditText like Imageview
354     final EditText inputtxt = new EditText(this);
355     inputtxt.setGravity(Gravity.CENTER);
356     inputtxt.setHint(R.string.query_mode0_hint);
357     params = new RelativeLayout.LayoutParams((int) (0.75 *
scrWidth) /*Width*/, (int) (0.1 * scrHeight)/*Height*/);
358     params.leftMargin = (int) (0.125 * scrWidth);
359     params.topMargin = (int) (0.55 * scrHeight);
360     lyt.addView(inputtxt, params);
361     // Add a Listener to it (so the User can simply press ENTER
on the keyboard)
362     inputtxt.setOnKeyListener(new View.OnKeyListener() {
363         public boolean onKey(View v, int keyCode, KeyEvent event)
{
364             if (event.getAction() == KeyEvent.ACTION.DOWN) {
365                 switch (keyCode) {
366                     case KeyEvent.KEYCODE_DPAD_CENTER:
367                     case KeyEvent.KEYCODE_ENTER:
368                         checkAnswer(inputtxt.getText().toString()
);
369
370                         inputtxt.setText("");
371                         return true;
372                     default:
373                         break;
374                 }
375             }
376             return false;
377         });
378
379     // add answer button
380     Button answer = new Button(this);
381     answer.setText("Answer");
382     answer.setTextColor(Color.WHITE);
383     answer.getBackground().setColorFilter(getResources().getColor
(R.color.colorAccent), PorterDuff.Mode.MULTIPLY);

```

```

384     params = new RelativeLayout.LayoutParams((int) (0.25 *
scrWidth) /*Width*/, (int) (0.1 * scrHeight)/*Height*/);
385     params.leftMargin = (scrWidth / 2) - (int) (0.25 * scrWidth);
386     params.topMargin = (int) (0.50 * scrHeight) + (int) (0.15 *
scrHeight);
387     lyt.addView(answer, params);
388     answer.setOnClickListener(new View.OnClickListener() {
389         public void onClick(View view) {
390             checkAnswer(inputtxt.getText().toString());
391             inputtxt.setText("");
392         }
393     });
394
395     // Add skip button
396     Button skip = new Button(this);
397     skip.setText("Skip");
398     skip.setTextColor(Color.WHITE);
399     skip.getBackground().setColorFilter(getResources().getColor(R
.color.colorPrimary), PorterDuff.Mode.MULTIPLY);
400     params = new RelativeLayout.LayoutParams((int) (0.25 *
scrWidth) /*Width*/, (int) (0.1 * scrHeight)/*Height*/);
401     params.leftMargin = (scrWidth / 2);
402     params.topMargin = (int) (0.50 * scrHeight) + (int) (0.15 *
scrHeight);
403     lyt.addView(skip, params);
404     skip.setOnClickListener(new View.OnClickListener() {
405         public void onClick(View view) {
406             skip();
407         }
408     });
409
410     // show the score
411     score = new TextView(this); // Create new Textview
412     score.setGravity(Gravity.CENTER);
413     score.setTextSize(36);
414     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.1 * scrHeight)/*Height*/);
415     params.leftMargin = 0; // X-Position
416     params.topMargin = (int) (0.78 * scrHeight); // Y-Position
417     lyt.addView(score, params); // add it to the View
418 }

```

```

419
420     public void buildModel1() { // This method builds the UI for Mode
421     1
422         int scrWidth = getWindowManager().getDefaultDisplay().
getWidth(); // Get the width of the screen
423         int scrHeight = getWindowManager().getDefaultDisplay().
getHeight(); // Get the height of the screen
424         RelativeLayout lyt = (RelativeLayout) findViewById(R.id.
query_absolute); // Get the View of the XML
425         RelativeLayout.LayoutParams params; // The parameters we want
to add our views
426
427         params = new RelativeLayout.LayoutParams((int) (0.55 *
scrWidth)/*Width*/, (int) (0.16 * scrHeight)/*Height*/);
428         params.leftMargin = (scrWidth / 2 - (int) (0.55 * scrWidth) /
2); // X-Position
429         params.topMargin = (int) (0.32 * scrHeight); // Y-Position
430         lyt.addView(hiding, params); // Add the hider
431
432         // Add all four buttons
433         Button choice0 = new Button(this);
434         choice0.setText("choice0");
435         choice0.setTextSize(8);
436         choice0.setTextColor(Color.WHITE);
437         choice0.getBackground().setColorFilter(getResources().
getColor(R.color.colorAccent), PorterDuff.Mode.MULTIPLY);
438         params = new RelativeLayout.LayoutParams((int) (0.48 *
scrWidth)/*Width*/, (int) (0.19 * scrHeight)/*Height*/);
439         params.leftMargin = (int) (0.02 * scrWidth);
440         params.topMargin = (int) (0.53 * scrHeight);
441         lyt.addView(choice0, params);
442         choice0.setOnClickListener(new View.OnClickListener() {
443             public void onClick(View view) {
444                 onClickChoice(0);
445             }
446         });
447
448         Button choice1 = new Button(this);
449         choice1.setText("choice1");
450         choice1.setTextSize(8);
451         choice1.setTextColor(Color.WHITE);

```

```

451         choice1.setBackground().setColorFilter(getResources().
getColor(R.color.colorPrimary), PorterDuff.Mode.MULTIPLY);
452         params = new RelativeLayout.LayoutParams((int) (0.48 *
scrWidth)/*Width*/, (int) (0.19 * scrHeight)/*Height*/);
453         params.leftMargin = (int) (0.50 * scrWidth);
454         params.topMargin = (int) (0.53 * scrHeight);
455         lyt.addView(choice1, params);
456         choice1.setOnClickListener(new View.OnClickListener() {
457             public void onClick(View view) {
458                 onClickChoice(1);
459             }
460         });
461
462         Button choice2 = new Button(this);
463         choice2.setText("choice2");
464         choice2.setTextSize(8);
465         choice2.setTextColor(Color.WHITE);
466         choice2.setBackground().setColorFilter(getResources().
getColor(R.color.colorPrimary), PorterDuff.Mode.MULTIPLY);
467         params = new RelativeLayout.LayoutParams((int) (0.48 *
scrWidth)/*Width*/, (int) (0.19 * scrHeight)/*Height*/);
468         params.leftMargin = (int) (0.02 * scrWidth);
469         params.topMargin = (int) (0.71 * scrHeight);
470         lyt.addView(choice2, params);
471         choice2.setOnClickListener(new View.OnClickListener() {
472             public void onClick(View view) {
473                 onClickChoice(2);
474             }
475         });
476
477         Button choice3 = new Button(this);
478         choice3.setText("choice3");
479         choice3.setTextSize(8);
480         choice3.setTextColor(Color.WHITE);
481         choice3.setBackground().setColorFilter(getResources().
getColor(R.color.colorAccent), PorterDuff.Mode.MULTIPLY);
482         params = new RelativeLayout.LayoutParams((int) (0.48 *
scrWidth)/*Width*/, (int) (0.19 * scrHeight)/*Height*/);
483         params.leftMargin = (int) (0.50 * scrWidth);
484         params.topMargin = (int) (0.71 * scrHeight);
485         lyt.addView(choice3, params);

```

```

486         choice3.setOnClickListener(new View.OnClickListener() {
487             public void onClick(View view) {
488                 onClickChoice(3);
489             }
490         });
491
492         // Add the score
493         score = new TextView(this); // Create new Textview
494         score.setGravity(Gravity.CENTER);
495         score.setTextSize(0);
496         params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
497         (int) (0.1 * scrHeight))/*Height*/;
498         params.leftMargin = 0; // X-Position
499         params.topMargin = (int) (0.78 * scrHeight); // Y-Position
500         lyt.addView(score, params); // add it to the View
501
502         choices = new ArrayList<>(); // Fill the arraylist with the
503         buttons
504         choices.add(choice0);
505         choices.add(choice1);
506         choices.add(choice2);
507         choices.add(choice3);
508     }
509
510     public void onClickChoice(int nr) {
511         if (choices.get(nr).getText() == wrongGuessed.get(indexCard).
512         getText()) {
513             if (firstGuess) {
514                 wrongGuessed.remove(indexCard); //if he guessed it, we
515                 remove it.
516             }
517             else {
518                 indexCard++; // else we continue with the next card
519             }
520             if (indexCard == wrongGuessed.size()) { //If this true he
521             's through the set
522                 final Handler handler = new Handler();
523                 handler.postDelayed(new Runnable() { // 1 second
524                     delay
525
526                     public void run() {
527                         setDone(); // finish query

```

```

521         }
522         }, 1000);
523     } else {
524         if (!skipped) {
525             imgCorr.setImageResource(R.drawable.
correct_answer); // set "correct"-image if the user guessed it
right
526             showImgCorr(); // show image
527         }
528         final Handler handler = new Handler();
529         handler.postDelayed(new Runnable() { // 800
milliseconds delay
530             public void run() {
531                 if (!skipped) {
532                     hideImgCorr(); // hide image
533                 }
534                 showNextPic(); // continue query
535             }
536             }, 800);
537     }
538 } else {
539     wrongAnswer(); // If the user guessed it wrong, we call
wrongAnswer()
540 }
541 }
542
543 public void updateChoices() { // This sets the text of one button
to the correct answer and the other ones to texts of random cards
544     int rightOne = (int) (Math.random() * 4);
545     for (int i = 0; i < choices.size(); i++) {
546         if (i == rightOne) {
547             choices.get(i).setText(wrongGuessed.get(indexCard).
getText());
548         } else {
549             choices.get(i).setText(set.get((int) (Math.random() *
set.size())).getText());
550         }
551     }
552
553     for (int i = 0; i < choices.size() - 1; i++) {
554         if (choices.get(i).getText() == choices.get(i + 1).

```

```

555         choices.get(i + 1).setText(set.get((int) (Math.random
556         () * set.size())).getText());
557     }
558 }
559
560 public void updateScore() { // This updates the score text
561     score.setText((set.size() - wrongGuessed.size()) + " / " +
562     indexCard + " / " + (wrongGuessed.size() - indexCard));
563 }
564
565 public void buildEndScreen() { // This builds the end UI with the
566     score and two buttons to restart.
567     int scrWidth = getWindowManager().getDefaultDisplay().
568     getWidth(); // Get screen-width
569     int scrHeight = getWindowManager().getDefaultDisplay().
570     getHeight(); // Get screen-height
571     RelativeLayout lyt = (RelativeLayout) findViewById(R.id.
572     query_absolute); // Get the View of the XML
573     lyt.removeAllViews(); //Clear the board
574     RelativeLayout.LayoutParams params; // the parameters we need
575     for our Views
576
577     // Show the amount of right guessed cards
578     TextView rights = new TextView(this);
579     rights.setText("Right: ");
580     rights.setTextSize(36);
581     rights.setTextColor(getResources().getColor(R.color.
582     colorPrimary));
583     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
584     (int) (0.1 * scrHeight))/*Height*/;
585     params.leftMargin = (int) (0.1 * scrHeight); // X-Position
586     params.topMargin = (int) (0.05 * scrHeight); // Y-Position
587     lyt.addView(rights, params); // add it to the View
588
589     TextView rights2 = new TextView(this);
590     rights2.setText("" + (set.size() - wrongGuessed.size()));
591     rights2.setTextSize(36);
592     rights2.setTextColor(getResources().getColor(R.color.
593     colorPrimary));

```



```

585     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.1 * scrHeight))/*Height*/;
586     params.leftMargin = (int) (0.4 * scrHeight); // X-Position
587     params.topMargin = (int) (0.05 * scrHeight); // Y-Position
588     lyt.addView(rights2, params); // add it to the View
589
590     // Show the amount of wrong guessed cards
591     TextView wrongs = new TextView(this);
592     wrongs.setText("Wrong: ");
593     wrongs.setTextSize(36);
594     wrongs.setTextColor(getResources().getColor(R.color.
colorAccent));
595     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.1 * scrHeight))/*Height*/;
596     params.leftMargin = (int) (0.1 * scrHeight); // X-Position
597     params.topMargin = (int) (0.05 * scrHeight) + (int) (0.1 *
scrHeight); // Y-Position
598     lyt.addView(wrongs, params); // add it to the View
599
600     TextView wrongs2 = new TextView(this);
601     wrongs2.setText("" + (wrongGuessed.size()));
602     wrongs2.setTextSize(36);
603     wrongs2.setTextColor(getResources().getColor(R.color.
colorAccent));
604     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.1 * scrHeight))/*Height*/;
605     params.leftMargin = (int) (0.4 * scrHeight); // X-Position
606     params.topMargin = (int) (0.05 * scrHeight) + (int) (0.1 *
scrHeight); // Y-Position
607     lyt.addView(wrongs2, params); // add it to the View
608
609     // Show the amount of total cards
610     TextView total = new TextView(this);
611     total.setText("Total: ");
612     total.setTextSize(36);
613     total.setTextColor(Color.BLACK);
614     params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
(int) (0.09 * scrHeight))/*Height*/;
615     params.leftMargin = (int) (0.1 * scrHeight); // X-Position
616     params.topMargin = (int) (0.05 * scrHeight) + (int) (0.1 *
scrHeight) + (int) (0.1 * scrHeight); // Y-Position

```

```

617         lyt.addView(total , params); // add it to the View
618
619         TextView total2 = new TextView(this);
620         total2.setText("" + (set.size()));
621         total2.setTextSize(36);
622         total2.setTextColor(Color.BLACK);
623         params = new RelativeLayout.LayoutParams(scrWidth /*Width*/,
624 (int) (0.09 * scrHeight))/*Height*/;
625         params.leftMargin = (int) (0.4 * scrHeight); // X-Position
626         params.topMargin = (int) (0.05 * scrHeight) + (int) (0.1 *
627 scrHeight) + (int) (0.1 * scrHeight); // Y-Position
628         lyt.addView(total2 , params); // add it to the View
629
630         // Show "repeat wrong guessed"-button if there are cards left
631         if (wrongGuessed.size() > 0) {
632             Button repWrong = new Button(this);
633             repWrong.setText("Repeat wrong guessed");
634             repWrong.setTextColor(Color.WHITE);
635             repWrong.getBackground().setColorFilter(getResources().
636 getColor(R.color.colorAccent), PorterDuff.Mode.MULTIPLY);
637             params = new RelativeLayout.LayoutParams((int) (0.90 *
638 scrWidth)/*Width*/, (int) (0.1 * scrHeight)/*Height*/);
639             params.leftMargin = (int) (0.05 * scrWidth);
640             params.topMargin = (int) (0.4 * scrHeight) + (int) (0.1 *
641 scrHeight) + (int) (0.1 * scrHeight) + (int) (0.1 * scrHeight);
642             lyt.addView(repWrong , params);
643             repWrong.setOnClickListener(new View.OnClickListener() {
644                 public void onClick(View view) {
645                     buildMenu();
646                     indexCard = 0;
647                     showFirstPic();
648                 }
649             });
650         }
651
652         // Show "Repeat all"-button
653         Button repAll = new Button(this);
654         repAll.setText("Repeat all");
655         repAll.setTextColor(Color.WHITE);
656         repAll.getBackground().setColorFilter(getResources().getColor
657 (R.color.colorPrimary), PorterDuff.Mode.MULTIPLY);

```

```

652     params = new RelativeLayout.LayoutParams((int) (0.90 *
scrWidth)/*Width*/, (int) (0.1 * scrHeight)/*Height*/);
653     params.leftMargin = (int) (0.05 * scrWidth);
654     params.topMargin = (int) (0.5 * scrHeight) + (int) (0.1 *
scrHeight) + (int) (0.1 * scrHeight) + (int) (0.1 * scrHeight);
655     lyt.addView(repAll, params);
656     repAll.setOnClickListener(new View.OnClickListener() {
657         public void onClick(View view) {
658             restartAll();
659         }
660     });
661
662 }
663
664 public void restartAll() { // This restarts all. A fresh start
wrongGuessed = (ArrayList) set.clone();
665     if (!queryLand) {
666         removeLands();
667     }
668     shuffleWrongs();
669     buildMenu();
670     indexCard = 0;
671     showFirstPic();
672 }
673
674 public void removeLands() { // This removes all Lands of
wrongGuessed
675     ArrayList<Card> remove = new ArrayList<>();
676     for (Card c : wrongGuessed) {
677         if (c.getType().contains("Land")) {
678             remove.add(c);
679         }
680     }
681     for (Card c : remove) {
682         wrongGuessed.remove(c);
683     }
684 }
685
686 }
687
688 public boolean onOptionsItemSelected(MenuItem item) {
689     int id = item.getItemId();

```

```

690         switch (id) {
691             case android.R.id.home:
692                 onBackPressed(); // If the user presses the "back"-
button we close the activity
693                 break;
694             case R.id.restart_all:
695                 restartAll(); // restart the query if the user
presses "restart"
696                 break;
697             case R.id.query_land: // removes/adds lands
698                 queryLand = !queryLand;
699                 item.setChecked(queryLand);
700                 if (queryLand) {
701                     restartAll();
702                 } else {
703                     removeLands();
704                     updateScore();
705                 }
706                 break;
707             case R.id.query_revers: // Change modes
708                 if (set.size() < 4) {
709                     break;
710                 }
711                 if (Mode == 1) {
712                     Mode = 0;
713                 } else {
714                     Mode = 1;
715                 }
716                 restartAll();
717                 break;
718         }
719         return true;
720     }
721
722     public void onBackPressed() {
723         finish(); // This closes our Activity
724         overridePendingTransition(R.anim.slide_in_right, R.anim.
slide_out_right); // We want to close it with an fancy animation.
725     }
726 }

```

## FavoritesActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.support.design.widget.FloatingActionButton;
7 import android.support.design.widget.Snackbar;
8 import android.support.v7.app.AppCompatActivity;
9 import android.view.MenuItem;
10 import android.view.View;
11 import android.widget.AdapterView;
12 import android.widget.ArrayAdapter;
13 import android.widget.ListView;
14 import java.util.ArrayList;
15 import m2b.magic2brain.com.magic2brain.R;
16
17 /*
18 This class shows all Favorites in a list
19 */
20
21 public class FavoritesActivity extends AppCompatActivity {
22
23     protected void onCreate(Bundle savedInstanceState) {
24         overridePendingTransition(R.anim.slide_in_left, R.anim.
25         slide_out_left); // This adds an fancy slide animation, when this
26         activity starts.
27         super.onCreate(savedInstanceState); // This does some intern
28         stuff. We don't need to worry about that. It's just needed.
29         setTitle(getString(R.string.FavoritesActivity_title)); // We
30         set a title to our View. We defined the title in "/res/values/
31         strings.xml". We just load it from there.
32         setContentView(R.layout.activity_favorites); // This adds an
33         View to our Activity. We defined at "/res/layout/
34         activity_favorites.xml" how our activity should look like.
35         getSupportActionBar().setDisplayHomeAsUpEnabled(true); //
36         With this line we add an "back"-Button to the Toolbar. If we press
37         it it calls onOptionsItemSelected();
38         buildMenu(); // This simply builds the menu
39     }
40 }
```

```

31
32     protected void onResume() { // If we get back to this activity ,
we build the menu again , because its possible that the user
removed a card .
33         super.onResume();
34         buildMenu();
35     }
36
37     private void buildMenu() {
38         final Context currentContext = this; // This doesn't actually
do anything , but it's needed if we want to refer to "this" from
an inner class .
39
40         final ArrayList<Card> alist_favs = Favorites.favorites_mvid ;
// This loads all the favorites
41         final Card[] cards = new Card[alist_favs.size()]; // We want
to turn it into a card-array , so its finite
42         final String[] favs = new String[alist_favs.size()]; // For
the list we need just the names
43         for (int i = 0; i < alist_favs.size(); i++) {
44             favs[i] = alist_favs.get(i).getName(); // We add all
names into the string-array
45             cards[i] = alist_favs.get(i); // We add all cards into
the card-array
46         }
47
48         final ArrayAdapter adapter = new ArrayAdapter(this , android.R
.layout.simple_list_item_1 , favs); // This is just a helper-class .
It transforms our String-Array into an clickable List .
49         final ListView lv = (ListView) findViewById(R.id.favList); //
We want to add our list to the ListView . So we get the ListView
from the View
50         lv.setAdapter(adapter); // We add our list into it .
51
52         lv.setOnItemClickListener(new AdapterView.OnItemClickListener
() { // This performs an action when we click on an item from the
list .
53             public void onItemClick(AdapterView<?> parent , View view ,
int position , long id) {
54                 // The following code starts CardBrowserActivity and
passes the card of the item we clicked .

```

```

55         Intent intent = new Intent(currentContext ,
CardBrowserActivity.class);
56         intent.putExtra("currentCard", cards[position]);
57         startActivity(intent);
58     }
59 });
60
61
62     FloatingActionButton fam = (FloatingActionButton)
findViewById(R.id.fab_addlearn); // This gets the "start"-button
from the view
63     fam.setOnClickListener(new View.OnClickListener() { // when
we press the button an action should be performed
64         public void onClick(View view) {
65             // The following code checks the amount of the
favorites. If it's zero, the user gets a notification which says,
that there are no cards to learn.
66             // If there are cards in favorites, it constructs a
deck-object and fills it with all the cards. After that it starts
the query with that deck.
67             if (alist_favs.size() > 0) {
68                 Intent intent = new Intent(currentContext ,
QueryActivity.class);
69                 Deck d = new Deck();
70                 d.setCode("FAVS");
71                 d.setName("Favorites");
72                 d.setSet(alist_favs);
73                 intent.putExtra("Set", d);
74                 startActivity(intent);
75             } else {
76                 Snackbar.make(view, R.string.No_Favs_to_learn ,
Snackbar.LENGTH_LONG).setAction("Action", null).show();
77             }
78         }
79     });
80
81 }
82
83     public boolean onOptionsItemSelected(MenuItem item) {
84         switch (item.getItemId()) {
85             // Respond to the action bar's Up/Home button

```

```
86         case android.R.id.home:
87             onBackPressed();
88         }
89         return true;
90     }
91
92     public void onBackPressed() {
93         finish(); // This closes our Activity
94         overridePendingTransition(R.anim.slide_in_right, R.anim.
95         slide_out_right); // We want to close it with an fancy animation.
96     }
```



## LastSeenActivity.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.content.SharedPreferences;
6 import android.os.Bundle;
7 import android.preference.PreferenceManager;
8 import android.support.v7.app.AppCompatActivity;
9 import android.view.MenuItem;
10 import android.view.View;
11 import android.widget.AdapterView;
12 import android.widget.AdapterView.OnItemClickListener;
13 import android.widget.ArrayAdapter;
14 import android.widget.ListView;
15 import com.google.gson.Gson;
16 import com.google.gson.reflect.TypeToken;
17 import java.lang.reflect.Type;
18 import java.util.ArrayList;
19 import m2b.magic2brain.com.magic2brain.R;
20
21 /*
22 This class should show all recently learned sets.
23 */
24 public class LastSeenActivity extends AppCompatActivity {
25     private ArrayList<String> recentlyLearned; // loadRecent() will
26     load all deck-codes in this arraylist
27     private ArrayList<String> recentlyLearnedNames; // loadRecent()
28     will load all deck-names in this arraylist
29     private String[] names; // this will contain all the names of the
30     decks
31     private String[] reclearn; // this will contain all the deck-
32     codes of the decks
33
34     protected void onCreate(Bundle savedInstanceState) {
35         overridePendingTransition(R.anim.slide_in_left, R.anim.
36         slide_out_left); // This adds an fancy slide animation, when this
37         activity starts.
38         super.onCreate(savedInstanceState); // This does some intern
39         stuff. We don't need to worry about that. It's just needed.
```

```

33      setContentView(R.layout.activity_last_seen); // This adds an
View to our Activity. We defined at "/res/layout/
activity_last_seen.xml" how our activity should look like.
34      setTitle(getString(R.string.LastSeenActivity_title)); // We
set a title to our View. We defined the title in "/res/values/
strings.xml". We just load it from there.
35      getSupportActionBar().setDisplayHomeAsUpEnabled(true); //
With this line we add an "back"-Button to the Toolbar. If we press
it it calls onOptionsItemSelected();
36      final Context currentContext = this; // This doesn't
actually do anything, but it's needed if we want to refer to "this
" from an inner class.

37
38      if (!loadRecent()) { // if it fails to load all the decks, we
simply replace recentlyLearned with an empty arraylist
39          recentlyLearned = new ArrayList<>();
40      }
41      names = new String[recentlyLearned.size()]; // we initiate
names with the size of recentlyLearned
42      reclearn = new String[recentlyLearned.size()]; // we initiate
reclearn with the size of recentlyLearned
43      for (int i = 0; i < recentlyLearned.size(); i++) {
44          reclearn[i] = recentlyLearned.get(i); // we fill reclearn
with all the deck-codes
45          names[i] = recentlyLearnedNames.get(i); // we fill names
with all the names
46      }
47
48      final ArrayAdapter adapter = new ArrayAdapter(this, android.R
.layout.simple_list_item_1, names); // This is just a helper-class
. It transforms our String-Array into an clickable List.
49      ListView lv = (ListView) findViewById(R.id.mListView); // We
want to add our list to the ListView. So we get the ListView from
the View
50      lv.setAdapter(adapter); // We add our list into it.
51
52      lv.setOnItemClickListener(new OnItemClickListener() { // This
performs an action when we click on an item from the list.
53          public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
54              // The following code starts DeckDisplayActivity and

```

```

55         passes the Deck-Code and the Deck-Name of the item we clicked.
56         String code = reclearn[position];
57         String name = names[position];
58         Intent intent = new Intent(currentContext,
DeckDisplayActivity.class);
59         intent.putExtra("code", code);
60         intent.putExtra("name", name);
61         startActivity(intent);
62     }
63 }
64
65
66 private boolean loadRecent() { // this function loads the
recently learned deck from the memory. If it fails it returns
false.
67     SharedPreferences sharedPrefs = PreferenceManager.
getDefaultSharedPreferences(this);
68     Gson gson = new Gson();
69     String json = sharedPrefs.getString("query-recent4", null);
70     String json2 = sharedPrefs.getString("query-recent-names",
null);
71     Type type = new TypeToken<ArrayList<String>>() {
72     }.getType();
73     ArrayList<String> aL = gson.fromJson(json, type);
74     ArrayList<String> aL2 = gson.fromJson(json2, type);
75     if (aL == null) {
76         return false;
77     }
78     recentlyLearned = aL;
79     recentlyLearnedNames = aL2;
80     return true;
81 }
82
83 public boolean onOptionsItemSelected(MenuItem item) {
84     switch (item.getItemId()) {
85         // Respond to the action bar's Up/Home button
86         case android.R.id.home:
87             onBackPressed();
88     }
89     return true;

```

```
90     }
91
92     public void onBackPressed() {
93         finish(); // This closes our Activity
94         overridePendingTransition(R.anim.slide_in_right , R.anim.
slide_out_right); // We want to close it with an fancy animation.
95     }
96 }
```

## DeckAssetLoader.java

```
1 package m2b.magic2brain.com;
2
3 import android.content.Context;
4 import org.json.JSONArray;
5 import org.json.JSONObject;
6 import java.io.BufferedReader;
7 import java.io.InputStream;
8 import java.io.InputStreamReader;
9 import java.io.Reader;
10 import java.io.StringWriter;
11 import java.io.Writer;
12
13 /*
14 This is an helper-class. All functions all static so we don't need to
15 create an object. This class basicly loads all informations of an
16 deck or card from an JSON-file from our assets-folder.
17 There's alot of low-level actions going on there.
18 */
19 public class DeckAssetLoader {
20
21     public DeckAssetLoader() {}
22
23     public static Card[] getDeck(String deckname, Context context) {
24         Card[] c = null;
25         try {
26             InputStream is = context.getAssets().open(deckname);
27             Writer writer = new StringWriter();
28             char[] buffer = new char[1024];
29             Reader reader = new BufferedReader(new InputStreamReader(
30 is, "UTF-8"));
31             int n;
32             while ((n = reader.read(buffer)) != -1) {
33                 writer.write(buffer, 0, n);
34             }
35             is.close();
36             String jsonString= writer.toString();
37             c = parseCardJSON(jsonString);
38         } catch (Exception e){}
39         return c;
40     }
41 }
```

```

37     }
38
39     public static Deck[] getDeckList(Context context) {
40         Deck[] darray = null;
41         try {
42             InputStream is = context.getAssets().open("
SetList.json");
43             Writer writer = new StringWriter();
44             char[] buffer = new char[1024];
45             try {
46                 Reader reader = new BufferedReader(new
InputStreamReader(is, "UTF-8"));
47                 int n;
48                 while ((n = reader.read(buffer)) != -1) {
49                     writer.write(buffer, 0, n);
50                 }
51             } finally {
52                 is.close();
53             }
54             String jsonString = writer.toString();
55             darray = parseDeckJSON(jsonString);
56         } catch (Exception e){}
57         return darray;
58     }
59
60     private static Deck[] parseDeckJSON(String json) {
61         Deck[] list = null;
62         try {
63             JSONArray file = new JSONArray(json);
64
65             list = new Deck[file.length()];
66
67             for (int i = 0; i < file.length(); i++) {
68                 JSONObject card = file.getJSONObject(i);
69                 Deck c = new Deck();
70
71                 String deck_name = card.getString("name");
72                 String deck_code = card.getString("code");
73                 String deck_releaseDate = card.getString("releaseDate
");
74

```

```

75         c.setName(deck_name);
76         c.setCode(deck_code);
77         c.setReleaseDate(deck_releaseDate);
78
79         list[i] = c;
80     }
81 } catch (Exception e){}
82 return list;
83 }
84
85 private static Card[] parseCardJSON(String json) {
86     Card[] carray = null;
87     try {
88         JSONObject deck = new JSONObject(json);
89         JSONArray cards = deck.getJSONArray("cards");
90         carray = new Card[cards.length()];
91
92         for (int i = 0; i < cards.length(); i++) {
93             JSONObject card = cards.getJSONObject(i);
94             Card c = new Card();
95             String mvid_as_string = card.getString("multiverseid"
96 );
97
98             String card_name = card.getString("name");
99
100             String card_flavor = "";
101             String card_text = "";
102             String card_type = "";
103             String card_cost = "";
104
105             if (card.has("flavor")) {
106                 card_flavor = card.getString("flavor");
107             }
108
109             if (card.has("text")) {
110                 card_text = card.getString("text");
111             }
112
113             if (card.has("type")) {
114                 card_type = card.getString("type");
115             }
116
117             if (card.has("manaCost")) {

```

```
115         card_cost = card.getString("manaCost");
116     }
117
118     c.setName(card_name);
119     c.setMultiverseid(Integer.parseInt(mvid_as_string));
120     c.setText(card_text);
121     c.setFlavor(card_flavor);
122     c.setType(card_type);
123     c.setManaCost(card_cost);
124
125     carray[i] = c;
126 }
127 } catch (Exception e){}
128 return carray;
129 }
130 }
```



## Card.java

```
1 package m2b.magic2brain.com;
2
3 import java.io.Serializable;
4
5 /*
6  This is a class to store Cards. It stores all crucial informations of
7  a card and has the Getters and Setters for it. There are also
8  three Constructors.
9 */
10 public class Card implements Serializable {
11     private int multiverseid;
12     private String name;
13     private String flavor;
14     private String text;
15     private String type;
16     private String manacost;
17
18     public Card(int multiverseid, String name) {
19         setMultiverseid(multiverseid);
20         setName(name);
21         // We fill the undefined variables with something to avoid
22         // NullPointerExceptions.
23         type = "UNKNOWN";
24         flavor = "UNKNOWN";
25         text = "UNKNOWN";
26         manacost = "NA";
27     }
28
29     public Card(String name, String flavor, String text, String type,
30     String manacost) {
31         this.type = type;
32         this.name = name;
33         this.flavor = flavor;
34         this.text = text;
35         this.manacost = manacost;
36     }
37
38     public Card() {}
39 }
```

```
36     public int getMultiverseid() {
37         return multiverseid;
38     }
39
40     public void setMultiverseid(int multiverseid) {
41         this.multiverseid = multiverseid;
42     }
43
44     public String getName() {
45         return this.name;
46     }
47
48     public void setName(String name) {
49         this.name = name;
50     }
51
52     public String getFlavor() {
53         return this.flavor;
54     }
55
56     public void setFlavor(String flavor) {
57         this.flavor = flavor;
58     }
59
60     public String getText() {
61         return this.text;
62     }
63
64     public void setText(String text) {
65         this.text = text;
66     }
67
68     public String getType() {
69         return type;
70     }
71
72     public void setType(String type) {
73         this.type = type;
74     }
75
76     public String getManaCost() {
```

```
77         return manacost;
78     }
79
80     public void setManaCost(String type) {
81         this.manacost = type;
82     }
83 }
```

## Deck.java

```
1 package m2b.magic2brain.com;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5 import java.util.Arrays;
6
7 /*
8  This class stores a deck with all its informations. It has the
9  Getters and Setters for the informations. There are also two
10 Constructors.
11 */
12 public class Deck implements Serializable /* We need to do this, so
13 we can pass Decks with Intents */ {
14
15     private ArrayList<Card> set;
16     private String name;
17     private String code;
18     private String releaseDate;
19     private String icon;
20
21     public Deck() {
22         set = new ArrayList<>();
23     }
24
25     public Deck(String name, String code, String release_date, String
26 iconUri) {
27         this.name = name;
28         this.code = code;
29         this.releaseDate = release_date;
30         this.icon = iconUri;
31         set = new ArrayList<>();
32     }
33
34     public String getIcon() {
35         return icon;
36     }
37
38     public void setIcon(String icon) {
39         this.icon = icon;
40     }
41 }
```

```

36     }
37
38     public String getName() {
39         return name;
40     }
41
42     public void setName(String name) {
43         this.name = name;
44     }
45
46     public String getCode() {
47         return code;
48     }
49
50     public void setCode(String code) {
51         this.code = code;
52     }
53
54     public String getReleaseDate() {
55         return releaseDate;
56     }
57
58     public void setReleaseDate(String releaseDate) {
59         this.releaseDate = releaseDate;
60     }
61
62     public int getSize() {
63         return set.size();
64     }
65
66     public ArrayList<Card> getSet() {
67         return set;
68     }
69
70     public void setSet(ArrayList<Card> al) {
71         set = al;
72     }
73
74     public void setSet(Card[] c){set = new ArrayList<>(Arrays.asList(
75     c));}

```

## Favorites.java

```
1 package m2b.magic2brain.com;
2
3 import java.util.ArrayList;
4
5 /*
6 This is very simple class. It's just here to temporary store all
7 favorites and pass them to any other class.
8 */
9 public final class Favorites {
10     public static ArrayList<Card> favorites_mvid;
11     public static void init() {
12         favorites_mvid = new ArrayList<>();
13     }
14 }
```

## RUtils.java

```
1 package m2b.magic2brain.com;
2
3 /*
4  This class is a helper-class. All functions are static so we don't
5  need any objects.
6  */
7 public class RUtils {
8
9     public static String[] getListified(Card[] cards) { // This turns
10        a Card-Array into a String-Array
11        String[] list = new String[cards.length];
12        for (int i = 0; i < cards.length; i++) {
13            list[i] = cards[i].getName();
14        }
15        return list;
16    }
17
18    public static boolean isInteger(String s) {
19        return isInteger(s, 10);
20    }
21
22    public static boolean isInteger(String s, int radix) {
23        if (s.isEmpty()) return false;
24        for (int i = 0; i < s.length(); i++) {
25            if (i == 0 && s.charAt(i) == '-') {
26                if (s.length() == 1) return false;
27                else continue;
28            }
29            if (Character.digit(s.charAt(i), radix) < 0) return false
30        }
31        return true;
32    }
33 }
```

## anim

### fadein.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <alpha xmlns:android="http://schemas.android.com/apk/res/android"
3     android:interpolator="@android:anim/accelerate_interpolator"
4     android:fromAlpha="0.0" android:toAlpha="1.0"
5     android:duration="500" />
```

### fadeout.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <alpha xmlns:android="http://schemas.android.com/apk/res/android"
3     android:interpolator="@android:anim/accelerate_interpolator"
4     android:fromAlpha="1.0" android:toAlpha="0.0"
5     android:fillAfter="true"
6     android:duration="500" />
```

### slide\_in\_left.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <set xmlns:android="http://schemas.android.com/apk/res/android">
3     <translate android:fromXDelta="100%" android:toXDelta="0"
4         android:duration="500" />
5 </set>
```

### slide\_in\_right.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <set xmlns:android="http://schemas.android.com/apk/res/android">
3     <translate android:fromXDelta="-100%" android:toXDelta="0"
4         android:duration="500" />
5 </set>
```

### slide\_out\_left.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <set xmlns:android="http://schemas.android.com/apk/res/android">
3     <translate android:fromXDelta="0" android:toXDelta="-100%"
4         android:duration="500" />
5 </set>
```



## slide\_out\_right.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <set xmlns:android="http://schemas.android.com/apk/res/android">
3     <translate android:fromXDelta="0" android:toXDelta="100%"
4         android:duration="500"/>
5 </set>
```

## drawable

### empty\_tall\_divider.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/res/android">
3     <size
4         android:height="10dp"
5         android:width="0dp"/>
6 </shape>
```

### ic\_favorite.xml

```
1 <vector xmlns:android="http://schemas.android.com/apk/res/android"
2     android:width="24dp"
3     android:height="24dp"
4     android:viewportWidth="24.0"
5     android:viewportHeight="24.0">
6     <path
7         android:fillColor="#FF000000"
8         android:pathData="M12,21.35l-1.45,-1.32C5.4,15.36 2,12.28
9         2,8.5 2,5.42 4.42,3 7.5,3c1.74,0 3.41,0.81 4.5,2.09C13.09,3.81
10        14.76,3 16.5,3 19.58,3 22,5.42 22,8.5c0,3.78 -3.4,6.86 -8.55,11.54
11        L12,21.35z"/>
12 </vector>
```

### ic\_favorite\_border.xml

```
1 <vector xmlns:android="http://schemas.android.com/apk/res/android"
2     android:width="24dp"
3     android:height="24dp"
4     android:viewportWidth="24.0"
5     android:viewportHeight="24.0">
6     <path
```

```

7         android:fillColor="#FF000000"
8         android:pathData="M16.5,3c-1.74,0 -3.41,0.81 -4.5,2.09C10
    .91,3.81 9.24,3 7.5,3 4.42,3 2,5.42 2,8.5c0,3.78 3.4,6.86
    8.55,11.54L12,21.35l1.45,-1.32C18.6,15.36 22,12.28 22,8.5 22,5.42
    19.58,3 16.5,3zM12.1,18.55l-0.1,0.1 -0.1,-0.1C7.14,14.24 4,11.39
    4,8.5 4,6.5 5.5,5 7.5,5c1.54,0 3.04,0.99 3.57,2.36h1.87C13.46,5.99
    14.96,5 16.5,5c2,0 3.5,1.5 3.5,3.5 0,2.89 -3.14,5.74 -7.9,10.05z"
    />
9 </vector>

```

### ic\_menu\_send\_white.xml

```

1 <vector xmlns:android="http://schemas.android.com/apk/res/android"
2     android:width="24dp"
3     android:height="24dp"
4     android:viewportHeight="24.0"
5     android:viewportWidth="24.0">
6     <path
7         android:fillColor="#FFFFFFF"
8         android:pathData="M2.01,21L23,12 2.01,3 2,10l15,2 -15,2z" />
9 </vector>

```

### ic\_menu\_share.xml

```

1 <vector xmlns:android="http://schemas.android.com/apk/res/android"
2     android:width="24dp"
3     android:height="24dp"
4     android:viewportHeight="24.0"
5     android:viewportWidth="24.0">
6     <path
7         android:fillColor="#FF000000"
8         android:pathData="M18,16.08c-0.76,0 -1.44,0.3 -1.96,0.77L8
    .91,12.7c0.05,-0.23 0.09,-0.46 0.09,-0.7s-0.04,-0.47 -0.09,-0.7l7
    .05,-4.11c0.54,0.5 1.25,0.81 2.04,0.81 1.66,0 3,-1.34 3,-3s
    -1.34,-3 -3,-3 -3,1.34 -3,3c0,0.24 0.04,0.47 0.09,0.7L8.04,9.81C7
    .5,9.31 6.79,9 6,9c-1.66,0 -3,1.34 -3,3s1.34,3 3,3c0.79,0
    1.5,-0.31 2.04,-0.81l7.12,4.16c-0.05,0.21 -0.08,0.43 -0.08,0.65
    0,1.61 1.31,2.92 2.92,2.92 1.61,0 2.92,-1.31 2.92,-2.92s
    -1.31,-2.92 -2.92,-2.92z" />
9 </vector>

```

### side\_nav\_bar.xml

```

1 <shape xmlns:android="http://schemas.android.com/apk/res/android"
2     android:shape="rectangle">
3     <gradient
4         android:angle="135"
5         android:centerColor="#4CAF50"
6         android:endColor="#2E7D32"
7         android:startColor="#81C784"
8         android:type="linear" />
9 </shape>

```

## layout

### activity\_browser.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.design.widget.CoordinatorLayout xmlns:android="http
3     ://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:fitsSystemWindows="true"
9     tools:context="m2b.magic2brain.com.BrowserActivity">
10
11     <android.support.design.widget.AppBarLayout
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:theme="@style/AppTheme.AppBarOverlay">
15
16         <android.support.v7.widget.Toolbar
17             android:id="@+id/toolbar"
18             android:layout_width="match_parent"
19             android:layout_height="?attr/actionBarSize"
20             android:background="?attr/colorPrimary"
21             app:popupTheme="@style/AppTheme.PopupOverlay" />
22
23     </android.support.design.widget.AppBarLayout>
24     <include layout="@layout/browser_content" />
25 </android.support.design.widget.CoordinatorLayout>

```

### activity\_card\_browser.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.design.widget.CoordinatorLayout xmlns:android="http
  ://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:fitsSystemWindows="true"
8   tools:context="m2b.magic2brain.com.CardBrowserActivity">
9
10  <android.support.design.widget.AppBarLayout
11    android:id="@+id/app_bar"
12    android:layout_width="match_parent"
13    android:layout_height="@dimen/app_bar_height"
14    android:fitsSystemWindows="true"
15    android:theme="@style/AppTheme.AppBarOverlay">
16
17    <android.support.design.widget.CollapsingToolbarLayout
18      android:layout_width="match_parent"
19      android:layout_height="match_parent"
20      android:fitsSystemWindows="true"
21      app:contentScrim="@color/colorPrimary"
22      app:layout_scrollFlags="scroll|exitUntilCollapsed">
23
24      <android.support.v7.widget.Toolbar
25        android:id="@+id/toolbar"
26        android:layout_width="match_parent"
27        android:layout_height="?attr/actionBarSize"
28        app:layout_collapseMode="pin"
29        app:popupTheme="@style/AppTheme.PopupOverlay" />
30
31      </android.support.design.widget.CollapsingToolbarLayout>
32    </android.support.design.widget.AppBarLayout>
33
34    <include layout="@layout/cardbrowser_content" />
35
36    <android.support.design.widget.FloatingActionButton
37      android:id="@+id/fab"
38      android:layout_width="wrap_content"
39      android:layout_height="wrap_content"
40      android:layout_margin="@dimen/fab_margin"

```

```

41     app:layout_anchor="@id/app_bar"
42     app:layout_anchorGravity="bottom|end"
43     app:srcCompat="@drawable/ic_favorite_border"
44     app:backgroundTint="@android:color/holo_orange_light" />
45
46 </android.support.design.widget.CoordinatorLayout>

```

### activity\_card\_image\_display.xml

```

1 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
2     xmlns:app="http://schemas.android.com/apk/res-auto"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:background="#ffffff"
7     tools:context="m2b.magic2brain.com.CardImageDisplayActivity">
8
9     <!-- The primary full-screen view. This can be replaced with
10         whatever view
11         is needed to present your content, e.g. VideoView,
12         SurfaceView,
13         TextureView, etc. -->
14     <ImageView
15         android:layout_width="match_parent"
16         android:layout_height="match_parent"
17         app:srcCompat="@drawable/brain_icon"
18         android:id="@+id/cida_imgview" />
19
20     <TextView
21         android:id="@+id/fullscreen_content"
22         android:layout_width="match_parent"
23         android:layout_height="match_parent"
24         android:gravity="center"
25         android:keepScreenOn="true"
26         android:textColor="#33b5e5"
27         android:textSize="50sp"
28         android:textStyle="bold" />
29
30     <!-- This FrameLayout insets its children based on system windows
31         using

```

```

29         android:fitsSystemWindows. —>
30     <FrameLayout
31         android:layout_width="match_parent"
32         android:layout_height="match_parent"
33         android:fitsSystemWindows="true">
34
35         <LinearLayout
36             android:id="@+id/fullscreen_content_controls"
37             style="?metaButtonBarStyle"
38             android:layout_width="match_parent"
39             android:layout_height="wrap_content"
40             android:layout_gravity="bottom|center_horizontal"
41             android:background="@color/black_overlay"
42             android:orientation="horizontal"
43             tools:ignore="UselessParent">
44
45             </LinearLayout>
46         </FrameLayout>
47 </FrameLayout>

```

### activity\_deck\_display.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3     xmlns:tools="http://schemas.android.com/tools"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="fill_parent"
6     android:layout_height="fill_parent"
7     tools:context="m2b.magic2brain.com.DeckDisplayActivity">
8
9     <ListView
10         android:layout_width="match_parent"
11         android:layout_height="match_parent"
12         android:id="@+id/deckdisplay" />
13
14     <android.support.design.widget.FloatingActionButton
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:clickable="true"
18         app:fabSize="normal"

```

```

19         app:srcCompat="@drawable/ic_menu_send_white"
20         android:id="@+id/fab_setlearn"
21         android:layout_marginEnd="24dp"
22         android:layout_marginBottom="23dp"
23         android:layout_alignParentBottom="true"
24         android:layout_alignParentEnd="true"
25         app:layout_anchor="@+id/deckdisplay"
26         app:layout_anchorGravity="top|left"
27         android:layout_gravity="bottom|right" />
28 </RelativeLayout>

```

### activity\_favorites.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context="m2b.magic2brain.com.MainActivity">
8
9     <ListView
10         android:layout_width="match_parent"
11         android:layout_height="match_parent"
12         android:id="@+id/favList"
13         android:layout_alignParentTop="true"
14         android:layout_alignParentStart="true" />
15
16     <android.support.design.widget.FloatingActionButton
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content"
19         android:clickable="true"
20         app:fabSize="normal"
21         app:srcCompat="@drawable/ic_menu_send_white"
22         android:id="@+id/fab_addlearn"
23         android:layout_marginEnd="24dp"
24         android:layout_marginBottom="23dp"
25         android:layout_alignParentBottom="true"
26         android:layout_alignParentEnd="true" />
27 </RelativeLayout>

```

### activity\_last\_seen.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="match_parent"
6   tools:context="m2b.magic2brain.com.MainActivity">
7
8   <ListView
9       android:layout_width="match_parent"
10      android:layout_height="match_parent"
11      android:layout_alignParentTop="true"
12      android:layout_alignParentStart="true"
13      android:id="@+id/mListView" />
14 </RelativeLayout>
```

### activity\_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.v4.widget.DrawerLayout xmlns:android="http://schemas
  .android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:id="@+id/drawer_layout"
6   android:layout_width="match_parent"
7   android:layout_height="match_parent"
8   android:fitsSystemWindows="true"
9   tools:openDrawer="start">
10
11   <include
12       layout="@layout/main_bar"
13       android:layout_width="match_parent"
14       android:layout_height="match_parent" />
15
16   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res
17   /android"
18       xmlns:app="http://schemas.android.com/apk/res-auto"
19       xmlns:tools="http://schemas.android.com/tools"
20       android:id="@+id/main_absolute"
21       android:layout_width="match_parent"
```



```

21         android:layout_height="match_parent"
22         tools:context="m2b.magic2brain.com.MainActivity2">
23     </RelativeLayout>
24
25     <android.support.design.widget.NavigationView
26         android:id="@+id/nav_view"
27         android:layout_width="wrap_content"
28         android:layout_height="match_parent"
29         android:layout_gravity="start"
30         android:fitsSystemWindows="true"
31         app:headerLayout="@layout/main_nav_header"
32         app:menu="@menu/activity_main2_drawer" />
33 </android.support.v4.widget.DrawerLayout>

```

### activity\_query.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:id="@+id/query_absolute"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context="m2b.magic2brain.com.MainActivity">
8
9     <android.support.v7.widget.Toolbar
10         android:layout_width="match_parent"
11         android:layout_height="wrap_content"
12         android:background="?attr/colorPrimary"
13         android:theme="?attr/actionBarTheme"
14         android:minHeight="?attr/actionBarSize"
15         android:layout_alignParentTop="true"
16         android:layout_alignParentStart="true"
17         android:id="@+id/toolbar_query" />
18 </RelativeLayout>

```

### activity\_search.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3     xmlns:tools="http://schemas.android.com/tools"

```

```

4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:paddingBottom="@dimen/activity_vertical_margin"
7      android:paddingLeft="@dimen/activity_horizontal_margin"
8      android:paddingRight="@dimen/activity_horizontal_margin"
9      android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="m2b.magic2brain.com.MainActivity">
11
12     <Button
13         android:layout_width="wrap_content"
14         android:layout_height="match_parent"
15         android:id="@+id/c_search_decks"
16         android:layout_alignParentTop="true"
17         android:layout_alignParentBottom="true"
18         android:layout_alignParentEnd="true"
19         android:layout_weight="0.51"
20         android:contentDescription="Button"
21         android:text="Search Sets"
22         android:drawableLeft="@drawable/sheets_filled_50"
23         style="?android:attr/borderlessButtonStyle" />
24
25     <Button
26         android:text="Search Cards"
27         android:layout_width="wrap_content"
28         android:layout_height="match_parent"
29         android:layout_alignParentStart="true"
30         android:id="@+id/c_search_cards"
31         android:layout_alignParentTop="true"
32         android:layout_alignParentBottom="true"
33         android:layout_toStartOf="@+id/c_search_decks"
34         android:layout_weight="0.54"
35         android:drawableLeft="@drawable/paper_filled_50"
36         style="?android:attr/borderlessButtonStyle"
37         android:elevation="0dp" />
38 </LinearLayout>

```

### activity\_search\_handler.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"

```

```

3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="match_parent"
6   android:paddingBottom="@dimen/activity_vertical_margin"
7   android:paddingLeft="@dimen/activity_horizontal_margin"
8   android:paddingRight="@dimen/activity_horizontal_margin"
9   android:paddingTop="@dimen/activity_vertical_margin"
10  tools:context="m2b.magic2brain.com.SearchHandlerActivity">
11
12  <ListView
13      android:layout_width="fill_parent"
14      android:layout_height="match_parent"
15      android:id="@+id/search_lv"
16      android:layout_alignParentStart="true"
17      android:layout_below="@+id/search_text" />
18
19  <EditText
20      android:layout_width="wrap_content"
21      android:layout_height="wrap_content"
22      android:inputType="textPersonName"
23      android:ems="10"
24      android:id="@+id/search_text"
25      android:layout_alignParentTop="true"
26      android:layout_alignParentStart="true"
27      android:layout_alignParentEnd="true"
28      android:contentDescription="Name" />
29 </RelativeLayout>

```

### browser\_content.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
   android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="fill_parent"
6   android:layout_height="fill_parent"
7   app:layout_behavior="@string/appbar_scrolling_view_behavior"
8   tools:context="m2b.magic2brain.com.BrowserActivity"
9   tools:showIn="@layout/activity_browser">
10

```

```

11 <ListView
12     android:layout_width="fill_parent"
13     android:layout_height="fill_parent"
14     android:layout_alignParentTop="true"
15     android:layout_alignParentStart="true"
16     android:id="@+id/deckbrowser_list"
17     android:headerDividersEnabled="false"
18     android:footerDividersEnabled="false" />
19 </RelativeLayout>

```

### cardbrowser\_content.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.v4.widget.NestedScrollView xmlns:android="http://
  schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     app:layout_behavior="@string/appbar_scrolling_view_behavior"
8     tools:context="m2b.magic2brain.com.CardBrowserActivity"
9     tools:showIn="@layout/activity_card_browser">
10
11     <LinearLayout
12         android:orientation="vertical"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:showDividers="middle"
16         android:divider="@drawable/empty_tall_divider">
17
18         <android.support.v7.widget.CardView
19             android:layout_width="match_parent"
20             android:layout_height="match_parent">
21
22             <ImageView
23                 android:layout_width="match_parent"
24                 android:layout_height="312dp"
25                 app:srcCompat="@drawable/
common_google_signin_btn_icon_light_normal"
26                 android:id="@+id/cbalmage" />
27

```

```

28     </android.support.v7.widget.CardView>
29
30     <android.support.v7.widget.CardView
31         android:layout_width="match_parent"
32         android:layout_height="match_parent" >
33
34         <LinearLayout
35             android:orientation="vertical"
36             android:layout_width="match_parent"
37             android:layout_height="match_parent">
38
39             <TextView
40                 android:text="Legendary"
41                 android:layout_width="match_parent"
42                 android:layout_height="wrap_content"
43                 android:id="@+id/cbaType"
44                 android:textAlignment="center"
45                 android:textStyle="normal|bold"
46                 android:textAllCaps="true" />
47             </LinearLayout>
48     </android.support.v7.widget.CardView>
49
50     <android.support.v7.widget.CardView
51         android:layout_width="match_parent"
52         android:layout_height="match_parent">
53
54         <LinearLayout
55             android:orientation="vertical"
56             android:layout_width="match_parent"
57             android:layout_height="match_parent">
58
59             <TextView
60                 android:text="Mana Cost"
61                 android:layout_width="match_parent"
62                 android:layout_height="wrap_content"
63                 android:textStyle="normal|bold"
64                 android:textSize="18sp"
65                 android:textAlignment="center"
66                 android:textColor="@android:color/black" />
67
68         <LinearLayout

```

```

69         android:orientation="horizontal"
70         android:layout_width="match_parent"
71         android:layout_height="match_parent"
72         android:id="@+id/cba_mcost_layout">
73
74     </LinearLayout>
75 </LinearLayout>
76
77 </android.support.v7.widget.CardView>
78
79 <android.support.v7.widget.CardView
80     android:layout_width="match_parent"
81     android:layout_height="match_parent"
82     android:layout_weight="1">
83
84     <LinearLayout
85         android:orientation="vertical"
86         android:layout_width="match_parent"
87         android:layout_height="match_parent">
88
89         <TextView
90             android:text="Text"
91             android:layout_width="match_parent"
92             android:layout_height="wrap_content"
93             android:textStyle="normal|bold"
94             android:textSize="18sp"
95             android:textAlignment="center"
96             android:textColor="@android:color/black" />
97
98         <TextView
99             android:text="TextView"
100             android:layout_width="match_parent"
101             android:layout_height="wrap_content"
102             android:id="@+id/cbaText" />
103
104     </LinearLayout>
105 </android.support.v7.widget.CardView>
106
107 <android.support.v7.widget.CardView
108     android:layout_width="match_parent"
109     android:layout_height="match_parent">

```

```

110
111     <LinearLayout
112         android:orientation="vertical"
113         android:layout_width="match_parent"
114         android:layout_height="match_parent">
115
116         <TextView
117             android:text="Flavor"
118             android:layout_width="match_parent"
119             android:layout_height="wrap_content"
120             android:textStyle="normal|bold"
121             android:textAlignment="center"
122             android:textSize="18sp"
123             android:textColor="@android:color/black" />
124
125         <TextView
126             android:text="TextView"
127             android:layout_width="match_parent"
128             android:layout_height="wrap_content"
129             android:id="@+id/cbaFlavor"
130             android:textStyle="normal|italic" />
131     </LinearLayout>
132 </android.support.v7.widget.CardView>
133
134 <android.support.v7.widget.CardView
135     android:layout_width="match_parent"
136     android:layout_height="wrap_content">
137
138     <LinearLayout
139         android:orientation="vertical"
140         android:layout_width="match_parent"
141         android:layout_height="match_parent">
142
143         <TextView
144             android:text="Informations"
145             android:layout_width="match_parent"
146             android:layout_height="wrap_content"
147             android:textStyle="normal|bold"
148             android:textAlignment="center"
149             android:textSize="18sp"
150             android:textColor="@android:color/black" />

```

```

151
152         <TextView
153             android:text="TextView"
154             android:layout_width="match_parent"
155             android:layout_height="wrap_content"
156             android:id="@+id/cbaInfo"
157             android:textAlignment="center" />
158
159     </LinearLayout>
160 </android.support.v7.widget.CardView>
161 </LinearLayout>
162 </android.support.v4.widget.NestedScrollView>

```

### main\_bar.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.design.widget.CoordinatorLayout xmlns:android="http
  ://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:fitsSystemWindows="true"
8     tools:context="m2b.magic2brain.com.MainActivity">
9
10     <android.support.design.widget.AppBarLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:theme="@style/AppTheme.AppBarOverlay">
14
15         <android.support.v7.widget.Toolbar
16             android:id="@+id/toolbar"
17             android:layout_width="match_parent"
18             android:layout_height="?attr/actionBarSize"
19             android:background="?attr/colorPrimary"
20             app:popupTheme="@style/AppTheme.PopupOverlay" />
21
22     </android.support.design.widget.AppBarLayout>
23     <include layout="@layout/main_content" />
24
25     <android.support.design.widget.FloatingActionButton

```



```

26         android:id="@+id/fab"
27         android:layout_width="wrap_content"
28         android:layout_height="wrap_content"
29         android:layout_gravity="bottom|end"
30         android:layout_margin="@dimen/fab_margin"
31         app:srcCompat="@drawable/ic_action_reload" />
32
33 </android.support.design.widget.CoordinatorLayout>

```

### main\_content.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:paddingBottom="@dimen/activity_vertical_margin"
8     android:paddingLeft="@dimen/activity_horizontal_margin"
9     android:paddingRight="@dimen/activity_horizontal_margin"
10    android:paddingTop="@dimen/activity_vertical_margin"
11    app:layout_behavior="@string/appbar_scrolling_view_behavior"
12    tools:context="m2b.magic2brain.com.MainActivity"
13    tools:showIn="@layout/main_bar">
14 </RelativeLayout>

```

### main\_nav\_header.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     android:layout_width="match_parent"
5     android:layout_height="@dimen/nav_header_height"
6     android:background="@drawable/side_nav_bar"
7     android:gravity="bottom"
8     android:orientation="vertical"
9     android:paddingBottom="@dimen/activity_vertical_margin"
10    android:paddingLeft="@dimen/activity_horizontal_margin"
11    android:paddingRight="@dimen/activity_horizontal_margin"
12    android:paddingTop="@dimen/activity_vertical_margin"

```

```

13     android:theme="@style/ThemeOverlay.AppCompat.Dark">
14
15     <TextView
16         android:layout_width="match_parent"
17         android:layout_height="wrap_content"
18         android:paddingTop="@dimen/nav_header_vertical_spacing"
19         android:text="Magic 2 Brain"
20         android:textAppearance="@style/TextAppearance.AppCompat.Body1"
21     "> />
22
23     <TextView
24         android:layout_width="wrap_content"
25         android:layout_height="wrap_content"
26         android:text="Your MTG learning companion" />
27 </LinearLayout>

```

## menu

### activity\_main2\_drawer.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android">
3     <group android:checkableBehavior="single">
4         <item
5             android:icon="@drawable/ic_home"
6             android:title="Home" />
7         <item
8             android:id="@+id/nav_search"
9             android:icon="@drawable/ic_search"
10            android:title="Search" />
11        <item
12            android:id="@+id/nav_browser"
13            android:icon="@drawable/ic_view_carousel"
14            android:title="Set Browser" />
15    </group>
16
17    <item android:title="Learn">
18        <menu>
19            <item
20                android:id="@+id/nav_quick_learn"

```

```

21         android:icon="@drawable/ic_timer"
22         android:title="Quick Learn" />
23     <item
24         android:id="@+id/nav_favorite"
25         android:icon="@drawable/ic_star_rate"
26         android:title="Favorites" />
27     <item
28         android:id="@+id/nav_history"
29         android:icon="@drawable/ic_history"
30         android:title="Recently Learned" />
31 </menu>
32 </item>
33
34 <item android:title="Communicate">
35     <menu>
36         <item
37             android:id="@+id/nav_share"
38             android:icon="@drawable/ic_menu_share"
39             android:title="Share" />
40     </menu>
41 </item>
42 </menu>

```

### query.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto">
4     <item
5         android:id="@+id/restart_all"
6         android:orderInCategory="100"
7         android:title="Restart"
8         app:showAsAction="never" />
9
10    <item android:id="@+id/query_lands"
11        android:orderInCategory="98"
12        android:title="Query Lands"
13        android:checkable="true"
14        android:checked="true" />
15
16    <item android:id="@+id/query_revers"

```

```

17     android:orderInCategory="99"
18     android:title="Reverse query"
19     app:showAsAction="never"/>
20 </menu>

```

## values

### attr.xml

```

1 <resources>
2     <!-- Declare custom theme attributes that allow changing which
3          styles are
4          used for button bars depending on the API level.
5          ?android:attr/buttonBarStyle is new as of API 11 so this is
6          necessary to support previous API levels. -->
7     <declare-styleable name="ButtonBarContainerTheme">
8         <attr name="metaButtonBarStyle" format="reference" />
9         <attr name="metaButtonBarButtonStyle" format="reference" />
10    </declare-styleable>
11 </resources>

```

### colors.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="colorPrimary">#3f51b5</color>
4     <color name="colorPrimaryDark">#303F9F</color>
5     <color name="colorAccent">#FF4081</color>
6     <color name="white">#FFFFFF</color>
7     <color name="black_overlay">#66000000</color>
8 </resources>

```

### dimens.xml

```

1 <resources>
2     <!-- Default screen margins, per the Android Design guidelines.
3          -->
4     <dimen name="activity_horizontal_margin">16dp</dimen>
5     <dimen name="activity_vertical_margin">16dp</dimen>
6     <!-- Default screen margins, per the Android Design guidelines.
7          -->
8     <dimen name="nav_header_vertical_spacing">16dp</dimen>

```

```

7     <dimen name=" nav_header_height">160dp</dimen>
8     <dimen name=" fab_margin">16dp</dimen>
9     <dimen name=" appBar_padding_top">8dp</dimen>
10    <dimen name=" app_bar_height">180dp</dimen>
11    <dimen name=" text_margin">16dp</dimen>
12 </resources>

```

## strings.xml

```

1 <resources>
2     <string name=" app_name">Magic2Brain</string>
3     <string name=" navigation_drawer_open">Open navigation drawer</
string>
4     <string name=" navigation_drawer_close">Close navigation drawer</
string>
5     <string name=" title_activity_card_browser">CardBrowserActivity</
string>
6     <string name=" title_activity_browser">BrowserActivity</string>
7     <string name=" title_activity_card_image_display">
CardImageDisplayActivity</string>
8     <string name=" send_to">Send to</string>
9     <string name=" BrowserActivity_title">Set Browser</string>
10    <string name=" remove_fav">Removed from your favorites!</string>
11    <string name=" add_fav">Added to your favorites!</string>
12    <string name=" image_link_1">http://gatherer.wizards.com/Handlers/
Image.ashx?multiverseid=</string>
13    <string name=" image_link_2"><![CDATA[&type=card]]></string>
14    <string name=" FavoritesActivity_title">Favorites</string>
15    <string name=" No_Favs_to_learn">You have no favorites to learn!</
string>
16    <string name=" LastSeenActivity_title">Recently Learned</string>
17    <string name=" SearchActivity_title">Search</string>
18    <string name=" SearchHandler_hint_1">Cardname</string>
19    <string name=" SearchHandler_hint_2">Setname</string>
20    <string name=" error">error</string>
21    <string name=" play_store_link">http://play.google.com/store/apps/
details?id=m2b.magic2brain.com</string>
22    <string name=" query_mode0_hint">What is the name of this card?</
string>
23 </resources>

```

## styles.xml

```
1 <resources>
2     <!-- Base application theme. -->
3     <style name="AppTheme" parent="Theme.AppCompat.Light.
4         DarkActionBar">
5         <!-- Customize your theme here. -->
6         <item name="colorPrimary">@color/colorPrimary</item>
7         <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
8         <item name="colorAccent">@color/colorAccent</item>
9         <item name="android:windowBackground">@color/white</item>
10    </style>
11
12    <style name="AppTheme.NoActionBar">
13        <item name="windowActionBar">false</item>
14        <item name="windowNoTitle">true</item>
15    </style>
16
17    <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.
18        AppCompat.Dark.ActionBar" />
19
20    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.
21        AppCompat.Light" />
22
23    <style name="splashScreenTheme" parent="Theme.AppCompat.Light.
24        DarkActionBar">
25        <item name="android:windowBackground">@drawable/splash</item>
26        <item name="colorPrimaryDark">@color/colorAccent</item>
27    </style>
28
29    <style name="FullscreenTheme" parent="AppTheme">
30        <item name="android:actionBarStyle">@style/
31        FullscreenActionBarStyle</item>
32        <item name="android:windowActionBarOverlay">true</item>
33        <item name="android:windowBackground">@null</item>
34        <item name="metaButtonBarStyle">?android:attr/buttonBarStyle
35    </item>
36        <item name="metaButtonBarButtonStyle">?android:attr/
37        buttonBarButtonStyle</item>
38    </style>
39
```

```
33     <style name="FullscreenActionBarStyle" parent="Widget.AppCompat.  
    ActionBar">  
34         <item name="android:background">@color/black_overlay </item>  
35     </style>  
36 </resources>
```