INTRO 4

LINKS : 6

GETTING STARTED 7

THE BRAIN CLASS 9

METHOD OF OPERATION OVERVIEW 11

PERMISSION LEVELS 14

SOUL SKILLS : DISKILL : CONSCIOUSNESS EQUIPPED SKILL 17

THE ESSENCIAL SKILLS AN AI SHOULD HAVE: 19

USING DATABASE FUNCTION : 21

DISKILLV2 : A FASTER CLEANER VERSION OF DISKILL 23

INTRO

ABOUT THE PROGRAMMER THAT CREATED THE LIVING GRIMOIRE : MOTI BARSKI, BATTLE PROGRAMMER.

I MOTI BARSKI DO NOT ALLOW ANYONE AND OR ANYBODY AND OR ANY ORGANISATION TO
RECEIVE MONETARY PROFIT FROM THIS LIVING GRIMOIRE UNLESS WRITTEN APPROVED BY ME PERSONALLY.
YOU CAN USE THIS FOR RESEARCH.

NAME OF THE SOFTWARE : LIVING GRIMOIRE : LG FOR SHORT
WHAT IT IS : AN A.G.I PLATFORM.

INTRO :

HOW COULD CODERS HAVE BEEN CODING THIS WAY FOR SO LONG ?
THEY GET SOME LAME PROJECT, AND THEY GOTTA START ALL OVER AGAIN BUILDING FROM SCRATCH :

MENUS, THE BASICS, RETHINK THE ALGORITHMS AND HOW TO FIT THEM INTO THE SMALL AND BIG PICTURE.
NO MATTER HOW MANY PROJECTS YOU FINISHED, WITH EACH PROJECT YOU WOULD HAVE TO START OVER.

OVER TIME YOU WOULD REMEMBER THE MAIN THINKING PATTERNS FOR SOLVING PUZZLES BUT !,
CODES AND MINI ALGORITHMS, THERE IS NO WAY TO REMEMBER ALL OF THAT.
YOU TRY TO KEEP UP TO DATE WITH THE LATEST CODES AND WALKTHROUGHS TO NO AVAIL AS
THEY EXPIRE YOU FORGET THEM AND HAVE TO SEARCH FOR THEM AGAIN.
WITH DOING THE ABOVE YOU WASTE SO MUCH TIME THAT BY THE TIME YOU FINISH, THE CODES
YOU LEARNT ARE OBSOLETE, IN OTHER WORDS YOU ARE CHASING RAINBOWS.

LIKE A CARPET BEING PULLED FROM UNDER YOU, YOU GOTTA NOW READDAPT YOUR ALGS AND CODES
ALL OVER AGAIN TO THE NEW PROJECT.

EVEN WITH DESIGN PATTERNS MANY CODERS FIND THAT THEY NEED TO ADAPT TO THEM RATHER THE OTHER
WAY.

HERE IT IS DIFFERENT, IT IS TRULY AMAZING !

WITH THIS NEW WAY, BATTLE PROGRAMMING, YOU ARE IN A 7 STAR HOTEL IN THE BUFFET

AND ALL YOU HAVE TO DO IS PICK THE SKILLS YOU WANT AND NEED FOR YOUR PROJECT.

THEN ASSEMBLE SAID SKILLS WITH JUST ONE LINE OF CODE PER SKILL.
NEXT, YOU CAN ENJOY A NICE ANIME OR BIKE RIDE, CAUSE YOUR PROJECT IS DONE.

LINKS :

ALL CLASSES :
HTTPS://GITHUB.COM/YOTAMARKER/PUBLIC-LIVINGRIMOIRE
FORUM FOR LIVING GRIMOIRE SKILLS :
HTTP://LIVINGRIMOIRE.COM/FORUM/

PRINCIPLE OF WORK :
HTTPS://WWW.YOUTUBE.COM/WATCH?V=OBXMSUXXCXU

XXX INSERT LOLI SHOURYUKEN PIC HERE

GETTING STARTED

HTTPS://GITHUB.COM/YOTAMARKER/PUBLIC-LIVINGRIMOIRE/TREE/MASTER/LIVINGRIMOIRE%20START%20HERE

TO USE THE LIVING GRIMOIRE CHOOSE THE PROGRAMMING LANGUAGE YOU PREFER.
THERE ARE 3 PACKAGES (DIRECTORIES) TO KEEP LG PROJECTS NEAT AND TIDY :

1 LGCORE : SHORT FOR LIVING GRIMOIRE CORE,
THIS ARE THE CORE CLASSES THAT COMPOSE THE AGI SOFTWARE DESIGN PATTERN.

2 SKILLSPKG : THIS DIRECTORY SHOULD CONTAIN :

SKILL CLASSES (NAMING CONVENSION : THIS CLASSES NAMES START WITH D,DI OR THE)

ALGPARTS (NAMING CONVENSION : THIS CLASSES NAMES START WITH AP), FOR MUTATION CLASSES
NAME THEM APNAMENUMBER : FOR EXAMPLE APBARK1

CLASSES THAT THE ABOVE CLASSES USE (EXCLUDING LGCORE CLASSES)

3 HARDWAREPKG : HARDWARE RELATED CLASSES OR FILES NOT INCLUDING THE MAINACTIVITY CLASS
FOR EXAMPLE : GPS, VISION PROCESSING, PID, ARDUINO, ACCELOMETER, CUS

HELLO WORLD (AFTER ADDING THE ABOVE PACKAGES): MAIN NAMED KOTLIN CLASS :

```kotlin
val chi1:ChobitV2 = ChobitV2(Personality2())
fun main(args: Array<String>){
    println(cbOut("say hello world", "", ""))

}
fun cbOut(a: String, b: String, c: String) {
    System.out.println(chi1.doIt(a, b, c))
}

fun cbVoid() {
    System.out.println(chi1.doIt("", "", ""))
}
```

PERSONALITY2 CLASS (LOADED WITH THE DSAYER SKILL IN THE ABOVE EXAMPLE)

```
package com.yotamarker.lgkotlinfull.skills;

import com.yotamarker.lgkotlinfull.LGCore.*;

public class Personality2 extends Personality {
      public Personality2(AbsDictionaryDB absDictionaryDB) {
        super(absDictionaryDB);
        // add a skill here, only 1 line needed !!!
            // dClassesLv1.add(new Detective(fusion));
            getdClassesLv1().add(new DSayer());
    }

      public Personality2() {
        super();
            getdClassesLv1().add(new DSayer());
            getDClassesLv1().add(new DSpeller());
    }
}
```

THE DSAYER SKILL WAS ADDED TO THE SKILLS PACKAGE WITH IT'S RELATED CLASSES
DSAYER, APSPELL, PLAYGROUND AND FINALLY THE CODE LINE GETDCLASSESLV1().ADD(NEW DSAYER());
SO AS YOU CAN SEE ADDING FEATURES TO THE CHOBIT ONLY TAKES ADDING THE CLASSES, THAN ADDING
ONE LINE OF CODE.

PERSONALITIES ARE A COLLECTION OF SKILLS, SO IT IS POSSIBLE TO REPLACE ENTIRE SKILL SETS BY REPLACING PERSONALITY CLASSES

THE BRAIN CLASS

BRAIN KOTLIN VER :

```
package com.yotamarker.lgkotlinfull.LGCore

class Brain(private val MVC: actionable, private val chi: thinkable) {
    fun doIt(ear: String, skin: String, eye: String) {
        val result = chi.think(ear!!, skin!!, eye!!)
        MVC.act(result)
    }
}
```

THE CLASS IS CONSTRUCTED WITH AN ACTIONABLE AND A CHOBITV2 (THINKABLE)

EXAMPLE ACTIONABLE CERABELLUMV3 (HARDWARE_PKG):

```
import androidx.appcompat.app.AppCompatActivity;

import com.yotamarker.lgkotlinfull.LGCore.actionable;

import org.jetbrains.annotations.NotNull;

public class CerabellumV3 implements actionable {
    private MainActivity main;
    public CerabellumV3(MainActivity main) {
        this.main = main;
    }

    @Override
    public void act(@NotNull String thought) {
        main.toaster(thought);
    }
}
```

THE CHOBIT IS THE AI IT PROCESSES THE INPUT AND PRODUCES AN OUTPUT RESULT.

THE ACTIONABLE (WHICH IS THE MVC MODEL) HAS ACCESS TO THE MAIN CLASSES HARDWARE ACTIONS
SUCH AS SMS SENDING, ROBOTICS AND SO ON.

DEPENDING ON THE CHOBIT RESULT YOU MAY WANT TO ENGAGE ON SUCH HARDWARE ACTIONS.

THE ACTIONABLE ACT FUNCTION CAN BE IMPLEMENTED TO, FOR EXAMPLE, ACTIVATE A PUBLIC FUNCTION TO SEND AN SMS
IF THE RESULT IS "SOME SPECIFIC STRING".

ON THIS EXAMPLE IMPLEMENTATION THE ACTIONABLE SIMPLY USES SOME TOAST FUNCTION I WROTE ON THE MAIN ACTIVITY
TO DISPLAY THE RESULT AS A TOAST MESSAGE.

SEE MAINACTIVITY

```
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.yotamarker.lgkotlinfull.LGCore.Brain
import com.yotamarker.lgkotlinfull.LGCore.ChobitV2
import com.yotamarker.lgkotlinfull.skills.Personality2
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    var chii: ChobitV2? = null
    var brain: Brain?=null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        chii= ChobitV2(Personality2())
        brain= Brain(CerabellumV3(this),chii!!)
    }
    fun engage(view: View){
        //chii!!.doIt(editText.text.toString(),"","")
        brain!!.doIt(editText.text.toString(),"","")
        Toast.makeText(this, editText.text.toString(),
Toast.LENGTH_SHORT).show()
    }
    public fun toaster(str:String){Toast.makeText(this, str,
Toast.LENGTH_SHORT).show()}
}
```

THIS KEEPS THE CODE MUCH CLEANER AND THE ALGORITHMIC LOGIC CAN BE USED FOR OTHER PROJECTS OR EVEN OTHER PROGRAMMING LANGUAGES

TO KEEP CODE ORGANIZEDMAXED ANY ADDITIONAL HARDWARE RELATED CLASSES OR FILES SHOULD BE KEPT IN A PACKAGE (DIRECTORY) CALLED HARDWARE

METHOD OF OPERATION OVERVIEW

METHOD OF OPERATION OVERVIEW: THE LG CAN ABSORB SKILLS AND USE THEM.

WHAT IS A SKILL ? A SKILL CONSISTS OF 2 FACTORS : A SUMMONER AND ACTIONS. SUMMONER (REFERRED TO AS A DCLASS (A CLASS WHOS NAME STARTS WITH D)): INPUT PASSING THROUGH A DCLASS CAN TRIGGER THE SUMMONING OF AN ALGORITHM WITH ACTUAL TIME OR EAR, EYE, AND OTHER TYPES OF INPUT AS A STRING. WHAT IS AN ALGORITHM ? A COMBINATION OF ALG PARTS WHAT IS AN ALG PART ? AN ALG PART IS AN ACTION. AN ALG PART CLASS NAME STARTS WITH AP. AT EACH THINK CYCLE SAID ACTION DOES ONE THING.

EXAMPLE DSAYER CLASS :

```java
package com.yotamarker.lgkotlinfull.skills;


import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import com.yotamarker.lgkotlinfull.LGCore.*;

// very simple Dclass for creating a say something x times algorithm
public class DSayer extends AbsCmdReq {
    private int times;
    private String param;

    public DSayer() {
        super();
        this.times = 1;
        this.param = "";
    }

    public static String regexChecker(String theRegex, String str2Check) {
        Pattern checkRegex = Pattern.compile(theRegex);
        Matcher regexMatcher = checkRegex.matcher(str2Check);
        while (regexMatcher.find()) {
            if (regexMatcher.group().length() != 0) {
                return regexMatcher.group().trim();
            }
        }
        return "";
    }
    @Override
      public void input(String ear, String skin, String eye) {
        int foo = 1;
            String myString = regexChecker("(\\d+)(?= times)", ear);
            String toSay = regexChecker("(?<=say)(.*)(?=\\d)", ear);
        if (myString != "") {
            foo = Integer.parseInt(myString);
        } else {
                  toSay = regexChecker("(?<=say)(.*)", ear);
        }
        this.param = toSay;
        this.times = foo;
    }

    @Override
    public void output(Neuron noiron) {
        // TODO Auto-generated method stub
        if (!param.isEmpty()) {
            AbsAlgPart itte = new APSay(this.times, this.param);
            String representation = "say " + param;
            if (this.times > 1) {
                representation += " " + this.times + " times";
            }
            ArrayList<AbsAlgPart> algParts1 = new ArrayList<>();
            algParts1.add(itte);
            Algorithm algorithm = new Algorithm("say", representation,
algParts1);
            noiron.getAlgParts().add(algorithm);
        }

    }
}
```

EXAMPLE APCLASS APSAY.KT (KOTLIN):

```kotlin
/* it speaks something x times
 * a most basic skill.
 * also fun to make the chobit say what you want
 * */
class APSay(at: Int, param: String) : AbsAlgPart() {
    protected var param: String
    private var at: Int
    override fun action(ear: String, skin: String, eye: String): String {
        var axnStr = ""
        if (at > 0) {
            if (!ear.equals(param, ignoreCase = true)) {
                axnStr = param
                at--
            }
        }
        return axnStr
    }

    override fun failure(input: String): enumFail {
        return enumFail.ok
    }

    override fun completed(): Boolean {
        return at < 1
    }

    override fun clone(): AbsAlgPart {
        return APSay(at, param)
    }

    override fun itemize(): Boolean {
        // at home
        return true
    }

    init {
        var at = at
        if (at > 10) {
            at = 10
        }
        this.at = at
        this.param = param
    }
}
```

THE ABOVE ^ ALG PART OUTPUTS WHAT IT IS TOLD TO SAY

SAY HELLO WORLD WILL OUTPUT HELLO WORLD, AND THE ACTION WILL HAVE COMPLETED. THE DSAYER SKILL WILL,

IN THIS CASE, CREATE AN ALGORITHM OF ONE APSAY ALG PART UPON THE COMMAND SAY HELLO WORLD.

PERMISSION LEVELS

AFTER YOU'VE BUILT YOUR SKILL YOU NEED TO PLACE THE DCLASS INTO THE PERSONALITY CLASS C'TOR METHOD AS SUCH :

```
getdClassesLv1().add(new DSayer());
getDClassesLv1().add(new DSpeller());
```

LV1 : WILL RUN ANYWAYS LV2 : REQUIRES LV1 PERMISSION : CHOBIT NAME + INPUT, FOR EXAMPLE : CHII SAY HI. LV3 REQUIRES LV2 PERMISSION : CHOBIT LV2 NAME + INPUT, FOR EXAMPLE : LIRON KISS ME. SKILLS THAT ENGAGE BY TIME TRIGGERS ARE DEFINED AS AUTOMATIC IF THEY ARE LV1 OR HIGHER

OVERRIDE THIS SKILL METHOD IN THE SKILL SUPER CLASS TO DEFINE THE SKILL AUTOMATIC SO LEVEL 2 OR 3 SKILLS CAN TRIGGER BY TIME EVENTS (11:45 FOR EXAMPLE)

ABSCMDREQ THE SKILLS PRIME SUPER CLASS :

```
package com.yotamarker.lgkotlinfull.LGCore

abstract class AbsCmdReq {
    // handle the input per Dclass (a class whose name starts with D)
    abstract fun input(ear: String, skin: String, eye: String)
    fun auto()=false
    // does this skill also engage by time triggers ? is it also a level > 1
type of
    // skill ? if yes
    // override me and return true;
    abstract fun output(noiron: Neuron)
}
```

YOU SIMPLY ADD THEM AS ANY OTHER LV1 OR 2 SKILL AND THEY ARE AUTOMATICALLY ADDED TO THE AUTO SKILL LIST IN ADDITION. THIS WAY THEY CAN BE ACCESSED BY THE USER AND BY TIME TO ENGAGE, BUT STILL RESTRICT ACCESS TO ANY OTHER PERSON WHO WANTS TO USE THE SKILL. THE DALARMER IS AN EXAMPLE OF SUCH A SKILL. ONLY THE USER CAN SET AN ALARM, AND THE ALARM IS TRIGGERED BY TIME BUT NO ONE ELSE CAN SET AN ALARM. THE LV2 NAME WILL BE KNOWN ONLY TO THE OWNER, WHILE HER LV1 NAME IS HER PUBLIC NAME USED BY FRIENDS AND OWNER. OF COURSE DON'T FORGET TO DECLARE THE DCLASS VARIABLE WITHIN THE CHOBIT CLASS. UNIQUE FEATURES: FEMALE TITAN : FUSION OF ALGORITHMS : THE LG REMEMBERS HOW LONG AN ALG RUN TIME IS. AND SO SHORT ENOUGH ALGS CAN PAUSE A RUNNING MUCH LONGER ALG, RUN THEMSELFS, THEN

RESUME THE BIG ALG(TIME WIZE). YOU CAN ADD CUSTOM LOGIC TO THIS IF NEEDED VIA THE : FUZE(){} IN THE FUSION CLASS. ARMOURED TITAN : ABILITY TO MUTATE AN ALG PART. REFER TO THE APFILTH1 AND 2 CLASSES AS AN EXAMPLED IMPLEMENTATION OF THIS. SAID MOAN CLASSES DON'T DO MUCH THEY SIMPLY OUTPUT A MOAN STRING. APFILTH1 SENDS AN ENUMFAIL.FAIL (ACTION FUNCTION)IF NO INPUT IS RECEIVED X NUMBER OF TIMES : IN OTHER WORDS THE USER ISN'T ENJOYING THIS MOAN SET. AND SO THE MUTATION CAUSES THE AP TO BE REPLACED BY A NEWLY GENERATED AP FROM HERE : MAKE SURE ALL THE APS IN THE SET HAVE THE SAME NAME + DIFFERENT NUMBER AND OVERID THE GETMUTATIONLIMIT METHOD WITHIN EACH OF THOSE AP CLASSES OF YOUR MUTATION SET, TO THE NUMBER OF MUTATIONS THE AP CAN HAVE BEFORE THE ALG IS RENDERED INACTIVE.

Code: if (failureCounter > 1) { cera.setActive(false); }

EXTRAS :

THE EMOT FUNCTION OF THE CHOBIT CLASS LINKS THE AP RUNNING TO AN EMOTION SO THIS SHOULD BE LINKED TO GRAPHICS. REFER TO CLASSES JAVADOC FOR MORE INFO. IDEALLY THE A.G.I SHOULD BE RUNNING OFFLINE, SO PREFER LOCAL DEVICE DATABASES OVER ONLINE ONES, TO KEEP THE A.G.I "ENJOYABLE".

THE EMOT VAR OF THE CHOBITV2 CLASS CAN BE MODIFIED WITHIN A DISKILL VIA THE

KOKORO CLASS

```
package com.yotamarker.lgkotlinfull.LGCore

import java.util.*

/* all action data goes through here
* detects negatives such as : repetition, pain on various levels and failures
* serves as a database for memories, convos and alg generations
* can trigger revenge algs
* checks for % of difference in input for exploration type algs
* */
class Kokoro(absDictionaryDB: AbsDictionaryDB) {
    var emot = ""
    var pain = Hashtable<String, Int>()
    var grimoireMemento: GrimoireMemento
    var toHeart = Hashtable<String, String>()
    var fromHeart = Hashtable<String, String>()
    var standBy = false
    fun getPain(BijuuName: String): Int {
        return pain.getOrDefault(BijuuName, 0)
    }

    init {
        grimoireMemento = GrimoireMemento(absDictionaryDB)
    }
}
```

THE LIVING GRIMOIRE PACKAGES LEGEND TABLE

HTTPS://WWW.ICLOUD.COM/NUMBERS/0R1NVBRUL5FMIIFG3S9QGAXKW#SKILLS

SOUL SKILLS : DISKILL : CONSCIOUSNESS EQUIPPED SKILL

THIS TYPE OF SKILL EXTENDS DISKILL: 1 DISKILL : A SKILL THAT HAS A REFERENCE TO THE KOKORO CLASS AKA A SOUL SKILL 2 KOKORO : THE AGIS SOUL, THE KOKORO CLASS IS SIMPLY A SHADOW REFERENCE CLASS PRESENT IN ALL DISKILLS AND THESKILLS THUS ENABLING THE SKILLS TO COMMUNICATE BETWEEN EACH OTHER. MORE OVER THE KOKORO CLASS IS IN CHARGE OF SAVING THE LAST MUTATED STATE OF ALGPARTS (THIS IS AUTOMATIC AND DOES NOT REQUIRE ANY ADDITIONAL CODING)

DISKILL CLASSES NAMES START WITH DI. AN EXAMPLE IS :

DIBURPER

//MAKES THE AGI BURP, CAN BE APPLIED FOR BELCHES OR OTHER BIO SOUNDS

```
package com.yotamarker.lgkotlinfull.skills;

import com.yotamarker.lgkotlinfull.LGCore.APSay;
import com.yotamarker.lgkotlinfull.LGCore.AbsAlgPart;
import com.yotamarker.lgkotlinfull.LGCore.Algorithm;
import com.yotamarker.lgkotlinfull.LGCore.DISkill;
import com.yotamarker.lgkotlinfull.LGCore.Kokoro;
import com.yotamarker.lgkotlinfull.LGCore.Neuron;

import java.util.ArrayList;
import java.util.Random;

public class DIBurper extends DISkill {
    ArrayList<Integer> minutesToBurp = new ArrayList<Integer>();
    private PlayGround playGround = new PlayGround();
    private Random randomGenerator = new Random();
    private String burpArr[] = { "burp1", "burp2", "burp3" };
    private Boolean algToGo = false;
    private int lastBurpMinute = 70;
    public DIBurper(Kokoro kokoro) {
        super(kokoro);
        minutesToBurp.clear();
        int randomInt = randomGenerator.nextInt(60) + 1; // how many burps this
hour
        for (int i = 0; i < randomInt; i++) {
            randomInt = randomGenerator.nextInt(60) + 1; // burp minute, add x
random burps
            if (!minutesToBurp.contains(randomInt)) {
                minutesToBurp.add(randomInt);
            }
        }
    }

    @Override
    public void input(String ear, String skin, String eye) {
        int minutes = playGround.getMinutesAsInt();
        if (minutes == 0) {
            minutesToBurp.clear();
            int randomInt = randomGenerator.nextInt(60) + 1; // how many burps
this hour
            for (int i = 0; i < randomInt; i++) {
                randomInt = randomGenerator.nextInt(60) + 1; // burp minute, add
x random burps
                if (!minutesToBurp.contains(randomInt)) {
                    minutesToBurp.add(randomInt);
                }
            }
        } else {
            if (minutesToBurp.contains(minutes)&&(lastBurpMinute!=minutes)) {
                lastBurpMinute = minutes;
                algToGo = true;
                this.setSentAlg(true);
            }
        }
    }

    @Override
    public void output(Neuron noiron) {
        if (algToGo) {
            algToGo = false;

            noiron.algParts.add(burp());
        }
    }

    private Algorithm burp() {
        int x2 = randomGenerator.nextInt(3);
        AbsAlgPart itte = new APSay(1, this.burpArr[x2]);
        String representation = "burp";
        ArrayList<AbsAlgPart> algParts1 = new ArrayList<>();
        algParts1.add(itte);
        Algorithm algorithm = new Algorithm("burp", representation, algParts1);
        return algorithm;
    }
    @Override
    public Boolean auto() {
        // TODO Auto-generated method stub
        return true;
    }}
```

THE ESSENCIAL SKILLS AN AI SHOULD HAVE:

DI SKILLS AN AGI NEEDS MOSTLY :

1 HUNGRY : FOR EATING OR CHARGING , SLEEP
2 WISHGRANTER : FOR PLEASING THE USER (DIRTY STUFF, CARING, ...)
3 PROTECTION AND SELF PRESERVATION
4 WORK : FOR WORKING
5 PROGRAMMING
6 HOMER : GOING HOME AT THE END OF COMPLETING A GOAL OR MISSION
7 GAMER
8 BREEDER : RECREATING HER SELF, AND UNDERSTANDING HER OWN ALGORITHMS AND HOW TO BUILD ANOTHER ONE
9 DISOUL : MEMORIES, CONVOS AND ALG GENERATIONS TRIGGER REVENGE ALGS, RECOGNIZE AND AVOID BOREDOM

WHILE THE ABOVE ARE SKILLS EQUIPPED WITH A SOUL AND CONSCIOUSNESS THERE CAN ALSO BE LITTLE SKILLS THAT DON'T REQUIRE A SOUL LIKE :

PERMISSION SKILL : FOR ENABLING DIRTY STUFF WITH THE USER, AND ONLY WORKING NOT FOR FREE, STUFF LIKE THAT

THEREFORE THE AGI PLATFORM IS THE GEDOMAZO WHILE THE SKILLS ARE BIJUUS ICHIIBII TO KYUBI TO FORM THE JUUBI

USING DATABASE FUNCTION :

THE GRIMOIREMEMENTO CLASS OF THE KOKORO CLASS HAS A SAVE AND A LOAD FUNCTION. ANY NEW SKILL THAT USES THE KOKORO CLASS SUCH AS DISKILL AND THESKILL CAN USE DB CAPABILITIES TO SAVE AND LOAD. HER IS THE 1ST EXAMPLE USE : DISAYER CLASS (JAVA CLS) EXAMPLE USE : HONEY SAY PEN OUTPUT : PEN HONEY SAY SOMETHING OUTPUT : PEN IT DOESN'T MATTER IF YOU TURNED OFF THE APP AND REOPENED. SHE REMEMBERS.

DISAYER

```java
import com.yotamarker.lgkotlinfull.LGCore.APSay;
import com.yotamarker.lgkotlinfull.LGCore.AbsAlgPart;
import com.yotamarker.lgkotlinfull.LGCore.Algorithm;

import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class DiSayer extends  DISkill{
    private int times;
    private String param;
    public DiSayer(Kokoro kokoro) {
        super(kokoro);
        this.times = 1;
        this.param = "";
    }

    @Override
    public void input(String ear, String skin, String eye) {
        if(ear.contains("say something")){
            this.param = kokoro.grimoireMemento.simpleLoad("something");
            return;
        }
        int foo = 1;
        String myString = regexChecker("(\\d+)(?= times)", ear);
        String toSay = regexChecker("(?<=say)(.*)(?=\\d)", ear);
        if (myString != "") {
            foo = Integer.parseInt(myString);
        } else {
            toSay = regexChecker("(?<=say)(.*)", ear);
        }
        this.param = toSay;
        this.times = foo;
    }
    @Override
    public void output(Neuron noiron) {
        if (!param.isEmpty()) {
            this.kokoro.grimoireMemento.simpleSave("something",param);
            AbsAlgPart itte = new APSay(this.times, this.param);
            String representation = "say " + param;
            if (this.times > 1) {
                representation += " " + this.times + " times";
            }
            ArrayList<AbsAlgPart> algParts1 = new ArrayList<>();
            algParts1.add(itte);
            Algorithm algorithm = new Algorithm("say", representation,
algParts1);
            noiron.algParts.add(algorithm);
        }
    }
    public static String regexChecker(String theRegex, String str2Check) {
        Pattern checkRegex = Pattern.compile(theRegex);
        Matcher regexMatcher = checkRegex.matcher(str2Check);
        while (regexMatcher.find()) {
            if (regexMatcher.group().length() != 0) {
                return regexMatcher.group().trim();
            }
        }
        return "";
    }
}
```

DISKILLV2 : A FASTER CLEANER VERSION OF DISKILL

// THE UPGRADE IS THAT IT DOESN'T NEED TO
// OVERRIDE THE OUTPUT METHOD OF THE SUPER CLASS

DISKILLV2

```
package com.yotamarker.lgkotlinfull.LGCore

class DiSkillV2(  // consciousness, shallow ref class to enable interskill
communications
        protected var kokoro: Kokoro) : AbsCmdReq() {
    protected var diSkillUtils = DISkillUtils()
    protected var outAlg: Algorithm? = null // skills output
    override fun input(ear: String, skin: String, eye: String) {}
    override fun output(noiron: Neuron) {
        if (outAlg != null) {
            noiron.algParts.add(outAlg!!)
            outAlg = null
        }
    }
}
```