```
import sys
from threading import Thread, Event
from time import sleep, time, strftime
from datetime import datetime
from pygame import mixer
from Display import refresh, app, AnzeigenUhr, AnzeigenInfo, SchliesseFenster

if __name__ == '__main__':
    global stopwatch_finished, timer_finished, alarm_type, stopwatch_running, timer_running, 
    twelve, twenty_four, alarm_time, reset_enabled, sound_enabled
    stopwatch_finished, stopwatch_running, timer_running, twelve, twenty_four, reset_enabled,
     sound_enabled = False, False, False, True, False, False, True
    alarm_type = "alarm"
    language = 0
    change_time = Event()
    languages = ["ENGLISH","GERMAN","FRENCH"]
    translations = [["Enter a command: ","no alarm","Please use a valid time! HH:MM:SS",
    "Languages: English, German and French.",
                     "Unknown command!","This language is not available!","Language changed."
                     ,"English is already in usage.","The Stopwatch is already running!",
                     "The Timer is already running!","Please use a valid time! HH:MM",
                     "Alarm: ","Please use a valid format! 12/24","This format does not 
                     exist! Use 12/24.",
                     "You are already using the 12-hour format.","You are already using the 
                     24-hour format.","Nothing to reset.","You are already using this type 
                     of alarm.",
                     "This type of alarm does not exist!","The sound is already on!", "The 
                     sound is already off!", "Unknown command! Try: sound on/off."],

                    ["Geben Sie einen Befehl ein: ","kein Alarm","Bitte verwenden Sie eine 
                    gültige Zeit! HH:MM:SS",
                     "Sprachen: Englisch, Deutsch und Französisch.","Unbekannter Befehl!",
                     "Diese Sprache ist nicht verfügbar!","Sprache geändert.",
                     "Deutsch wird bereits verwendet.","Die Stoppuhr läuft bereits!","Der 
                     Timer läuft bereits!","Bitte verwenden Sie eine gültige Zeit! HH:MM",
                     "Alarm: ",
                     "Bitte verwenden Sie ein gültiges Format! 12/24","Dieses Format 
                     existiert nicht! Nutzen Sie 12/24.","Sie nutzen das 12-Stunden Format 
                     bereits.",
                     "Sie nutzen das 24-Stunden Format bereits.","Nichts zum Zurücksetzen.",
                     "Du benutzt diesen Alarmtyp schon.","Dieser Alarmtyp existiert nicht!",
                     "Der Ton ist bereits an!","Der Ton ist bereits aus!","Unbekannter 
                     Befehl! Versuche: sound on/off."],

                    ["Entrez une commande: ","pas d'alarme","Veuillez utiliser une heure 
                    valide! HH:MM:SS","Langues: anglais, allemand et français.","Commande 
                    inconnue!",
                     "Cette langue n'est pas disponible!","La langue a changé.","Le français 
                     est déjà utilisé.","Le chronomètre fonctionne déjà!",
                     "Le minuteur est déjà en cours d'exécution!","Veuillez utiliser une 
                     heure valide! HH:MM","alarme: ","Veuillez utiliser un format valide! 
                     12/24",
                     "Ce format n'existe pas! Utilisez 12/24.","Vous utilisez déjà le format 
                     12 heures.","Vous utilisez déjà le format 24 heures.","Rien à 
                     réinitialiser.",
                     "Vous utilisez déjà ce type d'alarme.","Ce type d'alarme n'existe pas!",
                     "Le son est déjà activé!", "Le son est déjà désactivé!",
                     "Commande inconnue! Essayez: sound on/off."]]
```

Super Idee mit den Sprachen!

```
    def format_12():
        global twelve
        while twelve:
            sleep(1)
            format12 = strftime("%I:%M:%S %p")
            change_time.wait()
            AnzeigenUhr(format12)

    def format_24():
        global twenty_four
```

Clever!

Beide Formate hätten in einen Thread gesteckt werden können anhand einer Variablen "uhrzeit_format".

```
        while twenty_four:
            sleep(1)
            change_time.wait()
            AnzeigenUhr(str(datetime.now().strftime('%H:%M:%S')))

    print("Start")
    result = 0
    wait_start_event = Event()
    thread = Thread(target=refresh, args=(wait_start_event,))
    thread.start()
    wait_start_event.wait()
    AnzeigenInfo(translations[language][1])
    standard_time = Thread(target=format_12)
    standard_time.start()
    reset = Event()
    change_time.set()
    print("")

    def sound():                                    datetime.now().hour
        global sound_enabled, alarm_type
        if sound_enabled:
            if 8 <= int(datetime.now().strftime('%H')) <= 20:
                if alarm_type == "alarm":
                    sound = "alarm_loud.mp3"
                elif alarm_type == "music":
                    sound = "music_loud.mp3"
            else:
                if alarm_type == "alarm":
                    sound = "alarm_silent.mp3"
                elif alarm_type == "music":
                    sound = "music_silent.mp3"
            mixer.init()
            mixer.music.load(sound)
            mixer.music.play(-1)
        else:
            print(translations[language][11].split(":")[0])

    def set_alarm(arg):

        global alarm_time, reset_enabled

        alarm_info = translations[language][11]+str(arg)
        AnzeigenInfo(alarm_info)
        alarm_time = ~~str~~(arg) nicht nötig. arg ist schon ein String. Idem unten.
        hours = int(arg[0:2])
        minutes = int(arg[3:5])

        total_seconds_alarm = ((hours * 60) * 60) + (minutes * 60)

        hours_now = int(datetime.now().strftime('%H'))
        minutes_now = int(datetime.now().strftime('%M'))
        seconds_now = int(datetime.now().strftime('%S'))

        total_seconds_now = ((int(hours_now) * 60) * 60) + (int(minutes_now) * 60) +
        seconds_now
                          Dieser Test ist nicht ausreichend. Der Test würde besser auf die berechneten Sekunden gemacht.
        if hours < hours_now:
            total_seconds_next_day = ((24*60)*60)-total_seconds_now
            time_to_alarm = total_seconds_next_day+total_seconds_alarm

            sleep(time_to_alarm)
        else:
            time_to_alarm = total_seconds_alarm-total_seconds_now

            sleep(time_to_alarm)

        if alarm_time == ~~str~~(arg): -> Die Kontrolle, ob der Alarm noch immer gesetzt ist, ist sehr wichtig. Besser wäre
            sound()                      aber noch, einen Mechanismus einzubauen, der es ermöglicht, den Alarm abzubrechen.
            reset_enabled = True         Wie könnte das gehen?
```

```python
        reset.wait()
        reset.clear()
        mixer.music.stop()
        AnzeigenInfo(translations[language][1])
        reset_enabled = False

    def set_timer(arg):

        global timer_running, ~~twelve~~

        change_time.clear()
        hours = int(arg[0:2])
        minutes = int(arg[3:5])
        seconds = int(arg[6:8])
```

→ Die Eingabe sollte direkt schon in Zahlen umgewandelt werden.

```python
        start = time()

        total_seconds = ((~~int(~~hours) * 60) * 60) + (~~int(~~minutes) * 60) + ~~int(~~seconds)

        while str(time() - start).split(".")[0] != ~~str~~(total_seconds):
```

↳ Wenn man nummerische Wert vergleicht, dann sollte man diese nicht als Strings vergleichen.

```python
            timer_run = time() - start
            timer_left = total_seconds - timer_run
            time_parts = str(timer_left).split(".")
```

→ Auch würde besser eine numerische Umwandlung genutzt.

```python
            seconds = int(time_parts[0])
            milliseconds = time_parts[1][0:2]

            hours = str(seconds / 3600).split(".")[0]
            seconds -= int(hours) * 3600
            minutes = str(seconds / 60).split(".")[0]
            seconds -= int(minutes) * 60

            if len(str(hours)) == 1:
                hours = "0" + str(hours)
            if len(str(minutes)) == 1:
                minutes = "0" + str(minutes)
            if len(str(seconds)) == 1:
                seconds = "0" + str(seconds)
```

→ Nutzt die String-Funktion just().

```python
            time_content = ~~str~~(hours) + ":" + ~~str~~(minutes) + ":" + ~~str~~(seconds) + ":" + ~~str~~(
            milliseconds)
            AnzeigenUhr(time_content)

        sound()

        reset.wait()
        reset.clear()
        change_time.set()
        mixer.music.stop()
        timer_running = False

    def stopwatch():
        global stopwatch_finished, stopwatch_running
        change_time.clear()
        start = time()

        while not stopwatch_finished:
            meanwhile_time = time()
            timer_run = meanwhile_time - start
            time_parts = str(timer_run).split(".")

            seconds = int(time_parts[0])

            hours = str(seconds / 3600).split(".")[0]
            seconds -= int(hours) * 3600
            minutes = str(seconds / 60).split(".")[0]
            seconds -= int(minutes) * 60
```

Siehe oben!

```
            if len(str(hours)) == 1:
                hours = "0" + str(hours)
            if len(str(minutes)) == 1:
                minutes = "0" + str(minutes)
            if len(str(seconds)) == 1:
                seconds = "0" + str(seconds)

            time_content = ~~str~~(hours) + ":" + ~~str~~(minutes) + ":" + ~~str~~(seconds) + ":" +
            ~~time_parts[1][0:2]~~
            AnzeigenUhr(time_content)

        reset.wait()
        reset.clear()
        change_time.set()

        stopwatch_finished = False
        stopwatch_running = False


    while True:
        user_input = input(translations[language][0])
        input_list = user_input.split(" ")
        if input_list[0].upper() == "END":
            SchliesseFenster()
            break
```

Wenn das Fenster schließt, müssen vorher alle Threads gestoppt werden.

```
        elif input_list[0].upper() == "SET_TIME":
            time = input_list[1]
            print(time)
            AnzeigenUhr(time)
```

Diese Funktion wird wahrscheinlich nicht mehr benötigt.

```
        elif input_list[0].upper() == "SET_ALARM":
            if len(user_input) == 15:
                alarmThread = Thread(target=set_alarm, args=(input_list[1],))
                alarmThread.start()
            else:
                print(translations[language][10])
        elif input_list[0].upper() == "SET_TIMER":
            if len(user_input) == 18:
                if timer_running:
                    print(translations[language][9])
                else:
                    timerThread = Thread(target=set_timer, args=(input_list[1],))
                    timerThread.start()
                    timer_running = True
            else:
                print(translations[language][2])
        elif input_list[0].upper() == "STOPWATCH":
            if input_list[1].upper() == "START":
                if stopwatch_running:
                    print(translations[language][8])
                else:
                    stopwatchThread = Thread(target=stopwatch)
                    stopwatchThread.start()
                    stopwatch_running = True
            elif input_list[1].upper() == "STOP":
                stopwatch_finished = True
        elif input_list[0].upper() == "LANGUAGES":
            print(translations[language][3])
        elif input_list[0].upper() == "SET_LANGUAGE":
            prev_lan_number = language
            language = languages.index(input_list[1].upper())
            try:
                if prev_lan_number == language:
                    print(translations[language][7])
                else:
                    print(translations[language][6])
            except ValueError:
                print(translations[language][5])
            AnzeigenInfo(translations[language][1])
```

```python
        elif input_list[0].upper() == "SET_FORMAT":
            if len(user_input) == 13:
                if input_list[1].upper() == "12":
                    if twelve:
                        print(translations[language][14])
                    else:
                        twelve = True
                        twenty_four = False
                        format_thread12 = Thread(target=format_12)
                        format_thread12.start()
                elif input_list[1].upper() == "24":
                    if twenty_four:
                        print(translations[language][15])
                    else:
                        twelve = False
                        twenty_four = True
                        format_thread24 = Thread(target=format_24)
                        format_thread24.start()
                else:
                    print(translations[language][13])
            else:
                print(translations[language][12])
        elif input_list[0].upper() == "RESET":
            if len(user_input) == 5:
                if not change_time.is_set() or reset_enabled:
                    reset.set()
                else:
                    print(translations[language][16])
            else:
                print(translations[language][4])
        elif input_list[0].upper() == "SET_ALARM_TYPE":
            if input_list[1].upper() == "ALARM":
                if alarm_type == "alarm":
                    print(translations[language][17])
                else:
                    alarm_type = "alarm"
            elif input_list[1].upper() == "MUSIC":
                if alarm_type == "music":
                    print(translations[language][17])
                else:
                    alarm_type = "music"
            else:
                print(translations[language][18])
        elif input_list[0].upper() == "SOUND":
            if input_list[1].upper() == "ON":
                if sound_enabled:
                    print(translations[language][19])
                else:
                    sound_enabled = True
            elif input_list[1].upper() == "OFF":
                if not sound_enabled:
                    print(translations[language][20])
                else:
                    sound_enabled = False
            else:
                print(translations[language][21])
        else:
            print(translations[language][4])

    sys.exit(result)
```

```python
import sys
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *

class Display(QWidget):
    def __init__(self, parent=None):
        super(Display, self).__init__(parent)
 
        # Textfeld
        self.uhrzeitLabel = QLabel()
        self.uhrzeitLabel.setStyleSheet("font-size:60pt;")
        self.uhrzeitLabel.setAlignment(Qt.AlignCenter)

        # Textfeld
        self.infoLabel = QLabel()
        self.infoLabel.setStyleSheet("font-size:20pt;")
        self.infoLabel.setAlignment(Qt.AlignCenter)

        # Fensterlayout
        mainLayout = QVBoxLayout()
        mainLayout.addWidget(self.uhrzeitLabel)
        mainLayout.addWidget(self.infoLabel)
        self.setLayout(mainLayout)
 
        # Fensterlayout
        self.setWindowTitle("Uhr")
        self.resize(800,400)
        self.setWindowIcon(QIcon('uhr.png'))
        
    def AnzeigenUhr(self, text):
        self.uhrzeitLabel.setText(text)
        
    def AnzeigenInfo(self, text):
        self.infoLabel.setText(text)
 
def refresh(event):
    global result,app,screen
    print("Starte Thread",app,screen)
    app = QApplication(sys.argv)
    screen = Display()
    screen.show()
    event.set()
    print("OK")
    result = app.exec_()
    print("Ende")
    
def AnzeigenUhr( text):
    global screen
    screen.AnzeigenUhr(text)
    
def AnzeigenInfo(text):
    global screen
    screen.AnzeigenInfo(text)
    
def SchliesseFenster():
    global app
    app.quit()
    app = None
    
app,screen = None,None
```