```
import pandas as pd

from extractors.direct_downloads import extract_with_request, extract_with_id_crawler
from models.registry import Registry
from utils.helpers import date_to_iso, hexadecimal_to_decimal


class CZE(Registry):
    def __init__(self, **kwargs):
        """
         Child container for extracting, transforming, and loading registry data for the Czech Republic

        :param kwargs: see models.registry for details
        """

        [illegible]

        # assign constant values for final registry
        [illegible]

        # assign registry input data source and filename
        [illegible]

        # assign registry input data extraction method
        [illegible]
        # assign registry input data reading method
        [illegible]
        # assign registry data wrangling method
        [illegible]

        # extract, transform, and load registry data
        [illegible]

    def extract_registry(self):
        """
        Extracts registry input data from remote source

        :return: effect - creates [DATA_DIR|TEST_DIR]/[REGISTRY_INPUT_DIR]/[self.country_code]/[self.registry_filename] file
        """

        [illegible]
            # build test set if indicated
            [illegible]

        # extract abstracted registry input data
        [illegible]

        # extract detailed registry input data
        [illegible]

        # read registry input data
        [illegible]

    def read_registry(self, registry=None):
        """
        Reads registry input data

        :param (list) registry: set of key-value pairs representing extracted registry input data
        :return: effect - modifies self.registry
        """

        [illegible]
            # handle when registry input data has been extracted but not in current session
            [illegible]
        [illegible]

        # assign registry input data

    def wrangle_registry(self):
        """
        Transforms registry data

        :return: effect - modifies self.wrangled_records
        """

        # loop through raw registrations
        [illegible]
            # initialize storage for wrangled representation of registration in focus
            [illegible]

            [illegible]
                # update list of transformed values to populate final registry
                [illegible]
            [illegible]
                # update list of transformed values to record registrant and accommodate multiple registrants
                [illegible]

    def __add_registrant_copies(self, row, record):
        """
        Adds copies of a wrangled registration with updated registrant information to reflect all registrants
        :param (namedtuple) row: raw observation of a registration to extract registrants from
        :param (dict) record: wrangled representation of 'row' argument to duplicate
        :return: effect - modifies self.wrangled_records
        """

        # assign all owners for registration
        [illegible]
        # assign all operators for registration
        [illegible]

        # loop through registered owners
        [illegible]
            # assign registered owner in focus
            [illegible]
            # avoid modifying original wrangled representation of registration
            [illegible]

            # update registrant-related values
            [illegible]

            [illegible]
                # handle when registered owner in focus is also a registered operator
                [illegible]
            [illegible]

            # update list of transformed values to populate final registry
            [illegible]

        # loop through registered operators
        [illegible]
            # avoid modifying original wrangled representation of registration
            [illegible]

            # update registrant-related values
            [illegible]

            # update list of transformed values to populate final registry
            [illegible]

    @staticmethod
    def __get_is_active(date):
        """
        Determines whether a registration is currently active

        :param (str) date: registration expiry date
        :return: 'True' if registration is active
        """

        [illegible]
            [illegible]
        [illegible]

    @staticmethod
    def __get_name(registrant):
        """
        Formats registrant name by appending their legal ID to their display name if available

        :param (dict) registrant: registrant information data corresponding to a particular registration
        :return: formatted copy of 'registrant' argument with concatenated display and legal ID names
        """

        [illegible]

        [illegible]

        [illegible]
```