```
from datetime import datetime
import os
import pandas as pd

from extractors.direct_downloads import extract_with_request
from models.registry import Registry
from utils.helpers import clean_str, binary_to_decimal, date_to_iso, build_address


class CAN(Registry):
    def __init__(self, **kwargs):
        """
        Child container for extracting, transforming, and loading registry data for Canada

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

        [illegible]

        # assign constant values for final registry
        [illegible]
        [illegible]
        [illegible]
        [illegible]

        # assign registry input data source and filename
        [illegible]
        [illegible]

        # initialize storage for registrant reference data
        [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
        """

        # extract registry input data
        [illegible]
                              [illegible]

    def read_registry(self):
        """
        Reads registry input data

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

        [illegible]

        # read data layout and definitions reference
        [illegible]
            [illegible]

        # read registry input data
        [illegible]
        [illegible]
        # prep registry input data
        [illegible]

        # assign registry input data
        [illegible]

        # read registrant reference data
        [illegible]
        [illegible]
                                    [illegible]
        # prep registrant reference data
        [illegible]

        # assign registrant reference data
        [illegible]

    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]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
                [illegible]
            [illegible]

            # subset corresponding registrant references for 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, owners, record):
        """
        Adds copies of a wrangled registration with updated registrant information to reflect all registrants

        :param (pandas.DataFrame) owners: registrant reference data corresponding to a particular registration
        :param (dict) record: wrangled representation of a registration indicated by 'owners' argument to duplicate
        :return: effect - modifies self.wrangled_records
        """

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

            # update registrant-related values
            [illegible]
            [illegible]
            [illegible]
            [illegible]

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

    @staticmethod
    def __format_registry(registry, layout_refs):
        """
        Cleans registry input data and adds column names parsed from data layout and definitions reference

        :param (pandas.DataFrame) registry: registry input data to clean and add column names to
        :param (list) layout_refs: contents of data layout and definitions reference to extract column names from
        :return: cleaned copy of 'registry' argument with column names
        """

        # assign indices for column name extraction
        [illegible]
        [illegible]

        # extract and assign appropriate column names from data layout and definitions reference
        [illegible]
        [illegible]

        # remove invalid rows
        [illegible]
        [illegible]
        [illegible]

        [illegible]

    @staticmethod
    def __format_registrant_refs(registrant_refs, registry, layout_refs):
        """
        Adds column names parsed from data layout and definitions reference to registrant reference input data

        :param (pandas.DataFrame) registrant_refs: aircraft registration registrant input data to add column names to
        :param (list) layout_refs: contents of data layout and definitions reference to extract column names from
        :return: copy of 'registrant_refs' argument with column names
        """

        # assign indices for column name extraction
        [illegible]
        [illegible]

        # extract and assign appropriate column names from data layout and definitions reference
        [illegible]
        [illegible]

        [illegible]

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

        :param (str) status: registration status to check
        :return: 'True' if 'status' argument represents an active registration
        """

        [illegible]

        [illegible]
            [illegible]
        [illegible]
            [illegible]
        [illegible]
            [illegible]

    @staticmethod
    def __get_year(date):
        """
        Extracts the year from a date

        :param (str) date: registration date to get year from
        :return: year value from 'date' argument
        """

        [illegible]

        [illegible]
            [illegible]

        [illegible]
        [illegible]

        [illegible]

    @staticmethod
    def __get_is_company(category):
        """
        Determines whether a registrant represents a company based on data documentation

        :param (str) category: category label representing type of registrant
        :return: 'True' if 'category' argument represents a company
        """

        [illegible]

        [illegible]
            [illegible]
        [illegible]
            [illegible]
        [illegible]
            [illegible]
        [illegible]
            # handle when category has value "N/F"
            [illegible]

    @staticmethod
    def __get_address(owner):
        """
        Builds human-readable address in common format

        :param (namedtuple) owner: raw observation of a registrant corresponding to a particular registration
        :return: cleaned and formatted address of registrant
        """

        # assign country-specific common address format
        [illegible]
        [illegible]
            [illegible]
            [illegible]
            [illegible]
            [illegible]
            [illegible]
            [illegible]
        [illegible]

        [illegible]

        [illegible]
```