```python
#!/usr/bin/python3.5
# -*-coding:Utf-8 -*

from json import dumps

from .JSONEncoder import JSONEncoder
from src.Writer.PathManager import PathManager


class Writer:
    """Write IA in files"""


    def onAll(self, event):
        if event.event_name != 'processus.start':
            try:
                self.writeJSON({'event_name': event.event_name}, self.getPath(event.generation_id))
            except (SystemExit, KeyboardInterrupt):
                self.writeJSON({'event_name': event.event_name}, self.getPath(event.generation_id))
                raise
    onAll.priority = 1

    def onProcessusResume(self, event):
        self.__dict__.update(event.__dict__)

    def onProcessusStart(self, event):
        self.onProcessusResume(event)

        self.writeJSON(
            {
                'generations': event.generations,
                'pop_length': event.pop_length,
                'proportion': event.proportion,
                'chance': event.chance
            },
            self.getPath()
        )

    def onCreationDone(self, event):
        for ia in event.population:
            self.writeJSON(ia, self.getPath(event.generation_id, ia.id))

    def onGradingProgress(self, event):
        with self.getPath(event.generation_id, 'grading').open('a') as grading_file:
            grading_file.write(
                dumps([event.individual.id, event.graduation], cls=JSONEncoder, sort_keys=True, indent=4)
            )

    def onSelectionDone(self, event):
        self.writeJSON(
            [(score, ia.id) for (score, ia) in event.grading],
            self.getPath(event.generation_id, 'final_grading')
        )
        self.writeJSON(
            [ia.id for ia in event.selection],
            self.getPath(event.generation_id, 'selection')
        )

    def onBreedingProgress(self, event):
        self.writeJSON(event.offspring, self.getPath(event.generation_id, event.offspring.id))
        with self.getPath(event.generation_id, 'breeding').open('a') as breeding_file:
            breeding_file.write(
                '{} + {} -> {}\n'.format(event.parents[0].id, event.parents[1].id, event.offspring.id)
            )


    def getPath(self, *args, **kwargs):
        return PathManager.getPath(self.processus_id, self.generations, *args, **kwargs)

    def write(self, text, path):
        path.write_text(text)

    def writeJSON(self, data, path):
        self.write(dumps(data, cls=JSONEncoder, sort_keys=True, indent=4), path)
```