```python
#!/usr/bin/python3.4
# -*-coding:Utf-8 -*

from lib.inject_arguments import inject_arguments
from lib.XMLRepr import XMLRepr

from src.EvolutiveGenerator.GeneticElement import GeneticElement
from src.entities.GameEventData import GameEventData
from src.entities.ActionEventData import ActionEventData


class Neuron(GeneticElement, XMLRepr):
    """A link between an game event and an action event """

    @inject_arguments
    def __init__(self, game_event_data, action_event_data):
        """Init the neuron

        Expects:
            game_event_data to be a GameEventData or a tuple (event_name, coor)
            action_event_data to be a ActionEventData or a tuple (action_class, duration)
        """

        if type(game_event_data) is tuple:
            self.game_event_data = GameEventData(*game_event_data)
        if type(action_event_data) is tuple:
            self.action_event_data = ActionEventData(*action_event_data)


    def event_dispatcher():
        doc = "The event_dispatcher property. "

        def fget(self):
            return self._event_dispatcher

        def fset(self, event_dispatcher):
            """Register the neuron to the event_dispatcher """
            if hasattr(self, 'listener_id'):
                del self.event_dispatcher

            self._event_dispatcher = event_dispatcher
            self.listener_id = self._event_dispatcher.listen(self.game_event_data.event_name, self.onEvent)

        def fdel(self):
            """Detach the listener """
            if hasattr(self, 'listener_id'):
                self._event_dispatcher.detach(self.listener_id)
                del self.listener_id

            del self._event_dispatcher
        return locals()
    event_dispatcher = property(**event_dispatcher())

    def __del__(self):
        if hasattr(self, 'event_dispatcher'):
            del self.event_dispatcher


    def onEvent(self, event):
        if self.game_event_data.checkCoor(event):
            self.event_dispatcher.dispatch('action', self.action_event_data.buildAction(event))


    def reprJSON(self):
        return {
            'game_event_data': self.game_event_data,
            'action_event_data': self.action_event_data
        }

    def __repr__(self):
        return super().__repr__(['game_event_data', 'action_event_data'])
```