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

from abc import ABCMeta, abstractmethod


class Graduator(metaclass=ABCMeta):
    """Graduate individuals

    This is an abstract class to inherit.
    Assess individual's performances and assign them a score.
    The Graduator is to think as a bridge between the Generator and the software.
    It is designed to use the software to make evolute individuals.
    IT IS THE NATURE.
    Individuals are represented by root GeneticElement instances.
    """


    @abstractmethod
    def grade(self, individual, generation_id):
        """Assign a score to a individual

        Has to be implemented.

        Expects:
            individual to be an GeneticElement

        return int or any sortable object The score
        """

        raise NotImplementedError


    def gradeAll(self, individuals, generation_id, dispatch):
        """Assign a score to each individual

        Expects:
            individuals to be a list of GeneticElement

        Return a list of couple (score, GeneticElement)
        """

        grading = []
        for individual in individuals:
            graduation = self.grade(individual, generation_id)
            grading.append((graduation, individual))
            dispatch(individual, graduation)
        return grading
```