```python
from hourly_employee import HourlyEmployee
from salaried_employee import SalariedEmployee

if __name__ == "__main__":

    fred = SalariedEmployee(14, 'Fred Flintstone', 454121883, 'weekly', 5200)
    barney = HourlyEmployee(23, 'Barney Rubble', 333333333, 'weekly', 1.25)
    boss = SalariedEmployee(99, 'Mr Slate', 999999999, 'monthly', 120000)

    weekly_employees = [fred, barney]
    monthly_employees = [boss]

    # Week of Jan 7th.
    barney.log_hours(8)
    barney.log_hours(7)
    barney.log_hours(10)
    barney.log_hours(8)
    barney.log_hours(8)
    for e in weekly_employees:
        e.record_pay('Jan 7, 3000 BC')

    # Week of Jan 14th.
    barney.log_hours(42)
    for e in weekly_employees:
        e.record_pay('Jan 14, 3000 BC')

    # Week of Jan 21st.
    barney.log_hours(37)
    for e in weekly_employees:
        e.record_pay('Jan 21, 3000 BC')

    # Week of Jan 28th.  End of month so pay the monthly employees too.
    barney.log_hours(40)
    for e in weekly_employees:
        e.record_pay('Jan 28, 3000 BC')
    for e in monthly_employees:
        e.record_pay('Jan 28, 3000 BC')

    # Sum up the company payroll for January.
    grand_total = 0
    for e in weekly_employees + monthly_employees:
        pay = e.total_pay()
        print("{} made ${}".format(e.name, pay))
        grand_total += pay
    print("In total, employees made ${}".format(grand_total))
```

```python
class Employee():
    """
    An employee.
        eid: int        - This employee's ID number
        name: str       - This employee's name
        SIN: int        - This employee's social insurance number
        pay_period: str - This employee's pay period, either 'weekly' or
                          'monthly'

    This is an abstract class.  Only child classes should be instantiated.
    """

    def __init__(self, eid, name, SIN, pay_period):
        """
        (Employee) -> NoneType
        """

        self.eid = eid
        self.name = name
        self.SIN = SIN
        self.pay_period = pay_period


    def amount_of_pay(self):
        """
        (Employee) -> float

        Return the amount that this Employee should be paid in the next
        pay period.
        """


    def record_pay(self, date):
        """
        (Employee, str) -> NoneType

        Record the amount this Employee should be paid for the next pay
        period in their pay history, with the associated date.
        """


    def total_pay(self):
        """
        (Employee) -> float

        Return the total amount of pay this Employee has received.
        """


if __name__ == '__main__':
    import doctest
    doctest.testmod()
```