FixedPenalty

- fee : double

+ FixedPenalty(_date : string, _time : string, _code : int, _reg : string)
+ amount() : double
+ get_fee() : double
+ set_fee(_fee : double) : void

ParkingFine

offence_code : int
offence_date : string
offence_time : string
registration_number : string

+ ParkingFine(_date : string, _time : string, _code : int, _reg : string)
+ *amount() : double*

issues

1

ParkingAttendant

badge_number : int
fines_issued : double

+ ParkingAttendant(_badge_number : int)
+ issue(p : ParkingFine &) : void
+ *salary() : double*

ExcessCharge

- minutes : int

+ ExcessCharge(_date : string, _time : string, _code : int, _reg : string, _minutes : int)
+ amount() : double

JuniorParkingAttendant

+ JuniorParkingAttendant(_badge_number : int)
+ salary() : double

SeniorParkingAttendant

+ SeniorParkingAttendant(_badge_number : int)
+ salary() : double

Diagram: class diagram Page 1

```cpp
#include <iostream>
#include <string>

using namespace std;

class ParkingFine {
protected:
  string offence_date, offence_time;
  int offence_code;
  string registration_number;
public:
  ParkingFine(string _date, string _time, int _code, string _reg) {
    offence_date = _date;
    offence_time = _time;
    offence_code = _code;
    registration_number = _reg;
  }
  virtual double amount() = 0;
};

class FixedPenalty : public ParkingFine {
private:
  static double fee;
public:
  FixedPenalty(string _date, string _time, int _code, string _reg) : ParkingFine
(_date, _time, _code, _reg) { }
  double amount() {
    return fee;
  }
  static void set_fee(double _fee) { fee = _fee; }
  static double get_fee() { return fee; }
};

class ExcessCharge : public ParkingFine {
private:
  int minutes;
public:
  ExcessCharge(string _date, string _time, int _code, string _reg, int _minutes)
 : ParkingFine(_date, _time, _code, _reg) {
    minutes = _minutes;
  }
  double amount() {
    double result = 20 + minutes * 5;
    return (result < FixedPenalty::get_fee()) ? result : FixedPenalty::get_fee()
;
  }
};

class ParkingAttendant {
protected:
  int badge_number;
  double fines_issued;
public:
  virtual void issue(ParkingFine &p) {
    fines_issued += p.amount();
  }
  virtual double salary() = 0;
  ParkingAttendant(int _badge_number) {
    badge_number = _badge_number;
    fines_issued = 0;
  }
};
```

```cpp
class SeniorParkingAttendant : public ParkingAttendant {
public:
  SeniorParkingAttendant(int _badge_number) : ParkingAttendant(_badge_number
}
  double salary() {
    return 1200.0 + fines_issued*0.1;
  }
};

class JuniorParkingAttendant : public ParkingAttendant {
public:
  JuniorParkingAttendant(int _badge_number) : ParkingAttendant(_badge_number
}
  double salary() {
    return 900.0 + fines_issued*0.05;
  }
};

double FixedPenalty::fee = 100;

int main() {
  SeniorParkingAttendant charon(666);
  FixedPenalty fpn("16/02/2006", "09h50", 26, "LO52RNA");
  charon.issue(fpn);

  FixedPenalty::set_fee(120);

  JuniorParkingAttendant nelson(111);
  ExcessCharge ec("17/02/2006", "10h15", 14, "B14NCA", 15);
  nelson.issue(ec);

  cout << "Nelson's salary is " << nelson.salary() << endl;

  return 0;
}
```