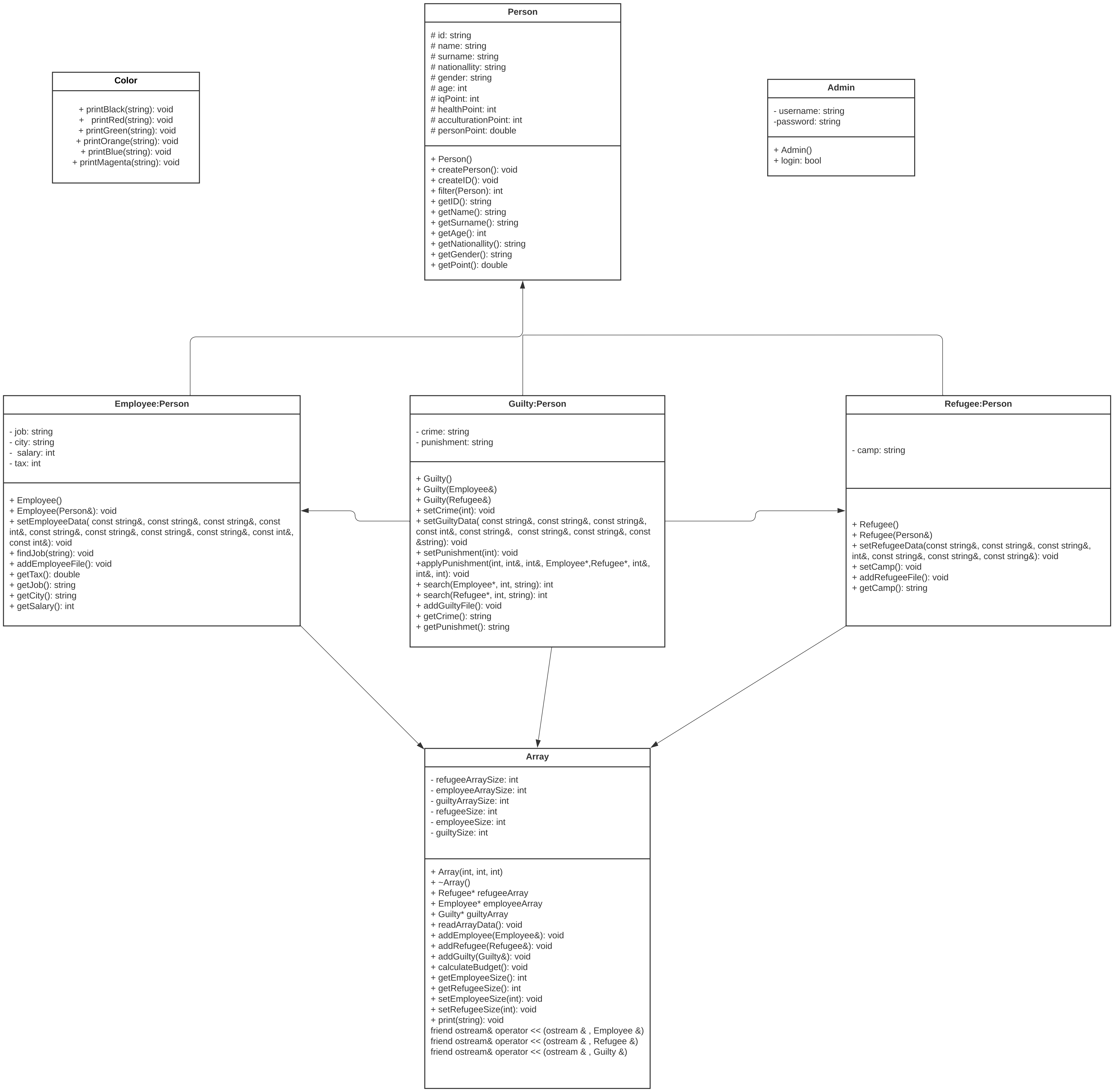
UML class Migrant System



Person

# id: string
# name: string
# surname: string
# nationallity: string
# gender: string
# age: int
# iqPoint: int
# healthPoint: int
# acculturationPoint: int
# personPoint: double

+ Person()
+ createPerson(): void
+ createID(): void
+ filter(Person): int
+ getID(): string
+ getName(): string
+ getSurname(): string
+ getAge(): int
+ getNationallity(): string
+ getGender(): string
+ getPoint(): double

Color

+ printBlack(string): void
+ printRed(string): void
+ printGreen(string): void
+ printOrange(string): void
+ printBlue(string): void
+ printMagenta(string): void

Admin

- username: string
-password: string

+ Admin()
+ login: bool

Employee:Person

- job: string
- city: string
- salary: int
- tax: int

+ Employee()
+ Employee(Person&): void
+ setEmployeeData(const string&, const string&, const string&, const int&, const string&, const string&, const string&, const string&, const int&, const int&): void
+ findJob(string): void
+ addEmployeeFile(): void
+ getTax(): double
+ getJob(): string
+ getCity(): string
+ getSalary(): int

Guilty:Person

- crime: string
- punishment: string

+ Guilty()
+ Guilty(Employee&)
+ Guilty(Refugee&)
+ setCrime(int): void
+ setGuiltyData(const string&, const string&, const string&, const int&, const string&, const string&, const string&, const &string): void
+ setPunishment(int): void
+applyPunishment(int, int&, int&, Employee*,Refugee*, int&, int&, int): void
+ search(Employee*, int, string): int
+ search(Refugee*, int, string): int
+ addGuiltyFile(): void
+ getCrime(): string
+ getPunishmet(): string

Refugee:Person

- camp: string

+ Refugee()
+ Refugee(Person&)
+ setRefugeeData(const string&, const string&, const string&, int&, const string&, const string&, const string&): void
+ setCamp(): void
+ addRefugeeFile(): void
+ getCamp(): string

Array

- refugeeArraySize: int
- employeeArraySize: int
- guiltyArraySize: int
- refugeeSize: int
- employeeSize: int
- guiltySize: int

+ Array(int, int, int)
+ ~Array()
+ Refugee* refugeeArray
+ Employee* employeeArray
+ Guilty* guiltyArray
+ readArrayData(): void
+ addEmployee(Employee&): void
+ addRefugee(Refugee&): void
+ addGuilty(Guilty&): void
+ calculateBudget(): void
+ getEmployeeSize(): int
+ getRefugeeSize(): int
+ setEmployeeSize(int): void
+ setRefugeeSize(int): void
+ print(string): void
friend ostream& operator << (ostream & , Employee &)
friend ostream& operator << (ostream & , Refugee &)
friend ostream& operator << (ostream & , Guilty &)