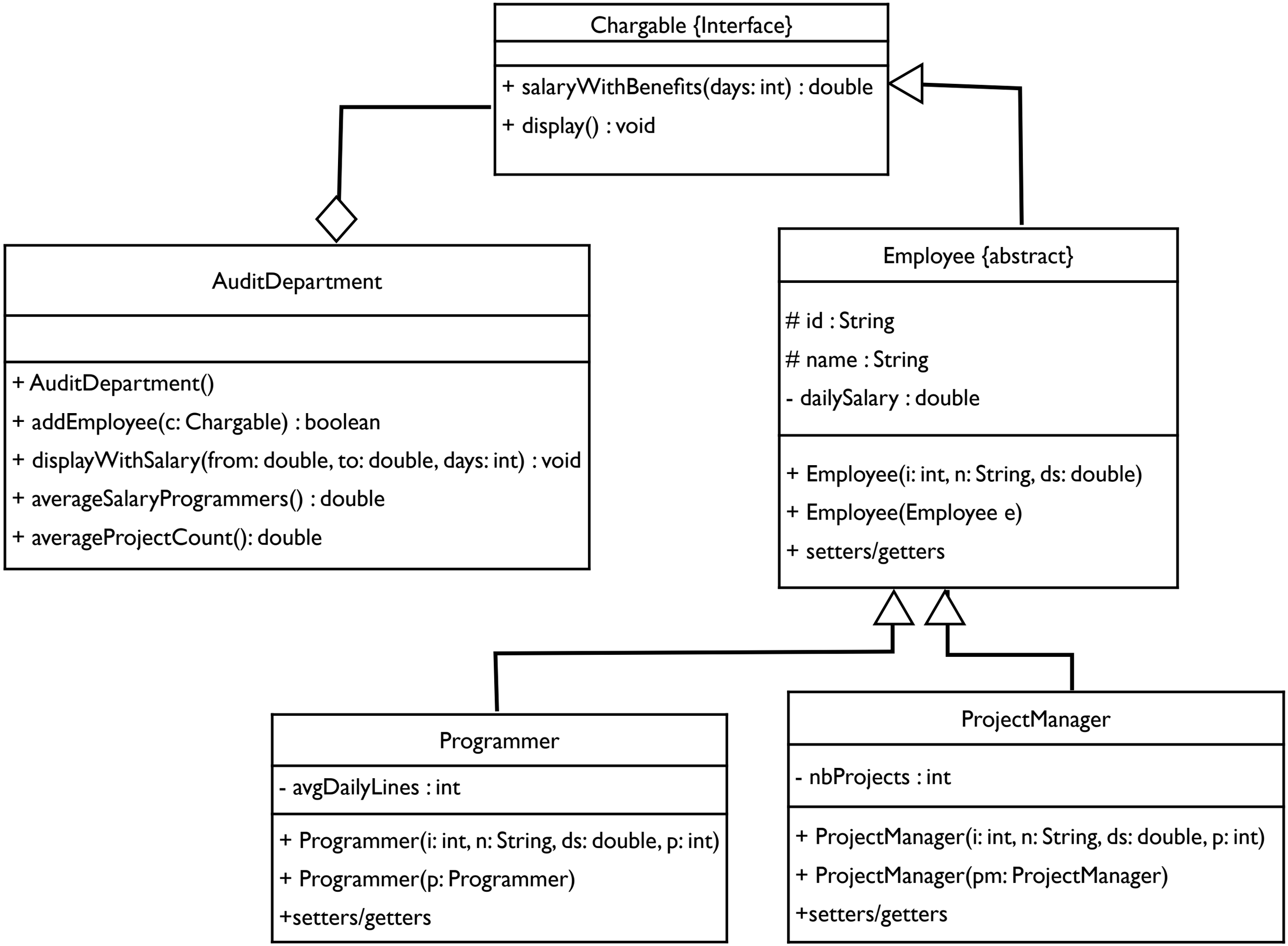
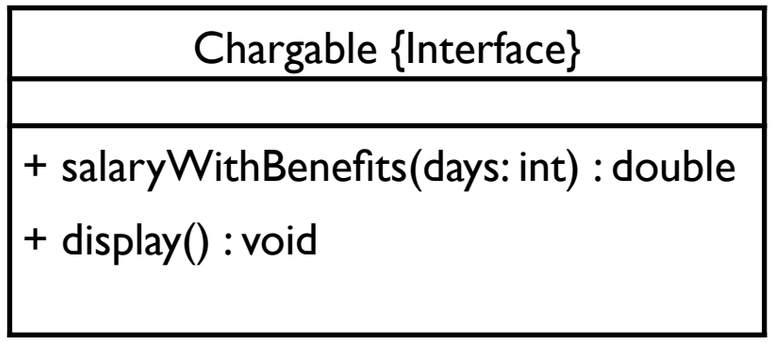
CSC 113
Tutorial 8

Interfaces and Exception handling

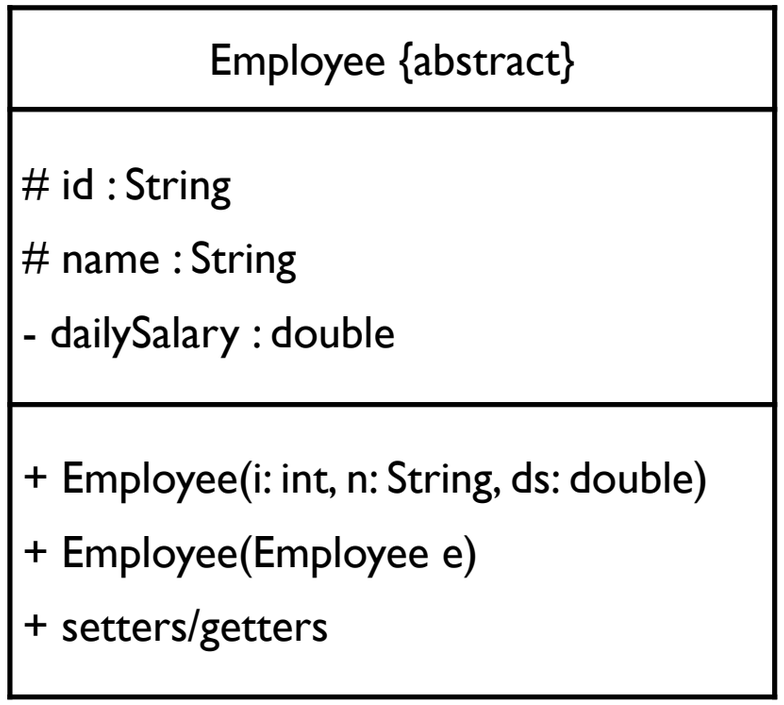

Chargable {Interface}
+ salaryWithBenefits(days: int) : double
+ display() : void
AuditDepartment
+ AuditDepartment()
+ addEmployee(c: Chargable) : boolean
+ displayWithSalary(from: double, to: double, days: int) : void
+ averageSalaryProgrammers() : double
+ averageProjectCount(): double
Employee {abstract}
id : String
name : String
- dailySalary : double
+ Employee(i: int, n: String, ds: double)
+ Employee(Employee e)
+ setters/getters
Programmer
- avgDailyLines : int
+ Programmer(i: int, n: String, ds: double, p: int)
+ Programmer(p: Programmer)
+setters/getters
ProjectManager
- nbProjects : int
+ ProjectManager(i: int, n: String, ds: double, p: int)
+ ProjectManager(pm: ProjectManager)
+setters/getters

Interface Chargable



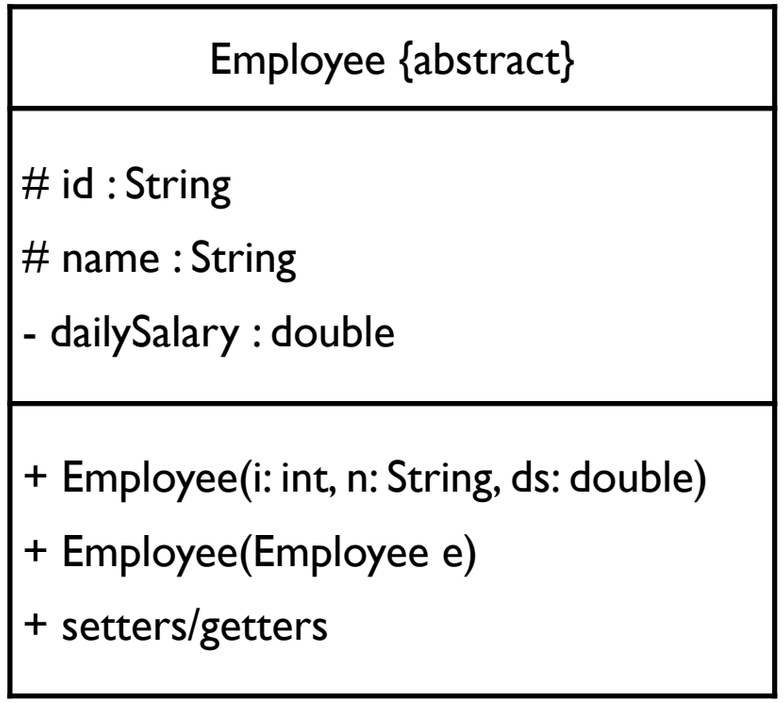

```
public interface Chargable {
   public double salaryWithBenfits(int days);
   public void display();
}
```

Class Employee is abstract




```
public abstract class Employee implements Chargable {
   protected int id;
   protected String name;
   private double dailySalary;

   public Employee()
   {}

   public Employee(Employee s) {
      id = s.id;
      name = s.name;
      dailySalary = s.dailySalary;
   }
}
```

Class Employee is abstract. interface implementation is optional




```
public void display() {
      System.out.println("ID: " + id);
      System.out.println("Name: " + name);
   }
```

Programmer Class Constructors

Programmer
- avgDailyLines : int
+ Programmer(i: int, n: String, ds: double, p: int) + Programmer(p: Programmer) +setters/getters

```
public final class Programmer extends Employee {
   private int avgDailyLines;

   public Programmer(int id, String name, double dailySalary,
                     int avgDailyLines) {
      super(id, name, dailySalary);
      this.avgDailyLines = avgDailyLines;
   }

   public Programmer(Programmer p) {
      super(p);
      avgDailyLines = p.avgDailyLines;
   }

}
```

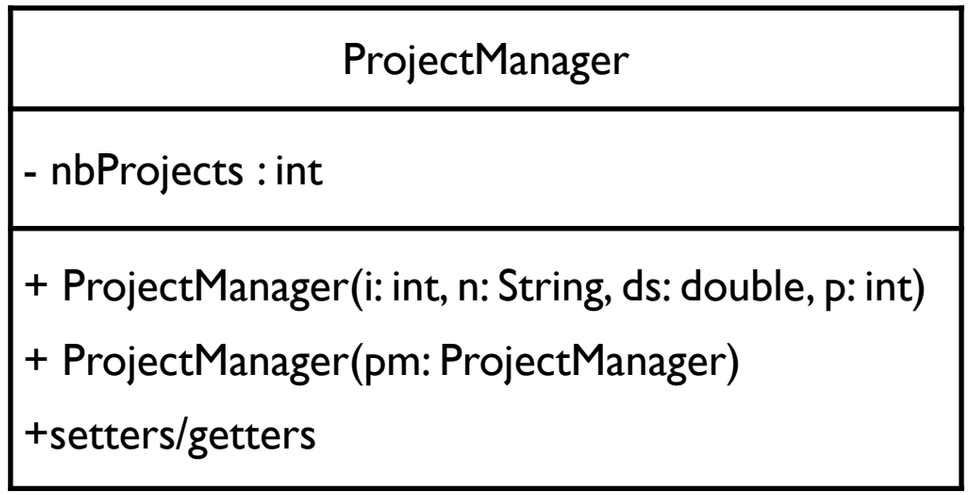
Implemented Methods

Programmer
- avgDailyLines : int
+ Programmer(i: int, n: String, ds: double, p: int) + Programmer(p: Programmer) +setters/getters

```
public double salaryWithBenfits(int days) {
     return (getDailySalary() + avgDailyLines * 10) * days;
   }
```

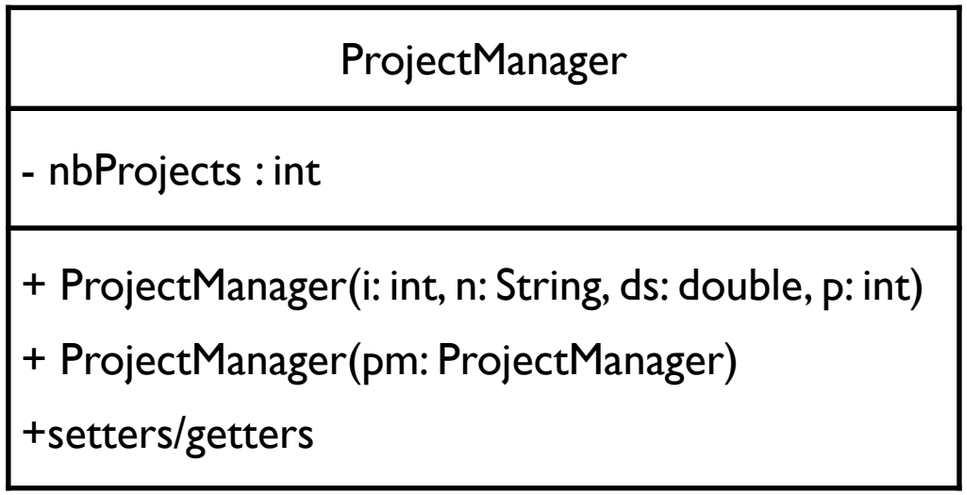
```
public void display() {
   super.display();
   System.out.println("Monthly Salary (with benfits): "
                          + salaryWithBenfits(30));
}
```

Project Manager Class Constructors



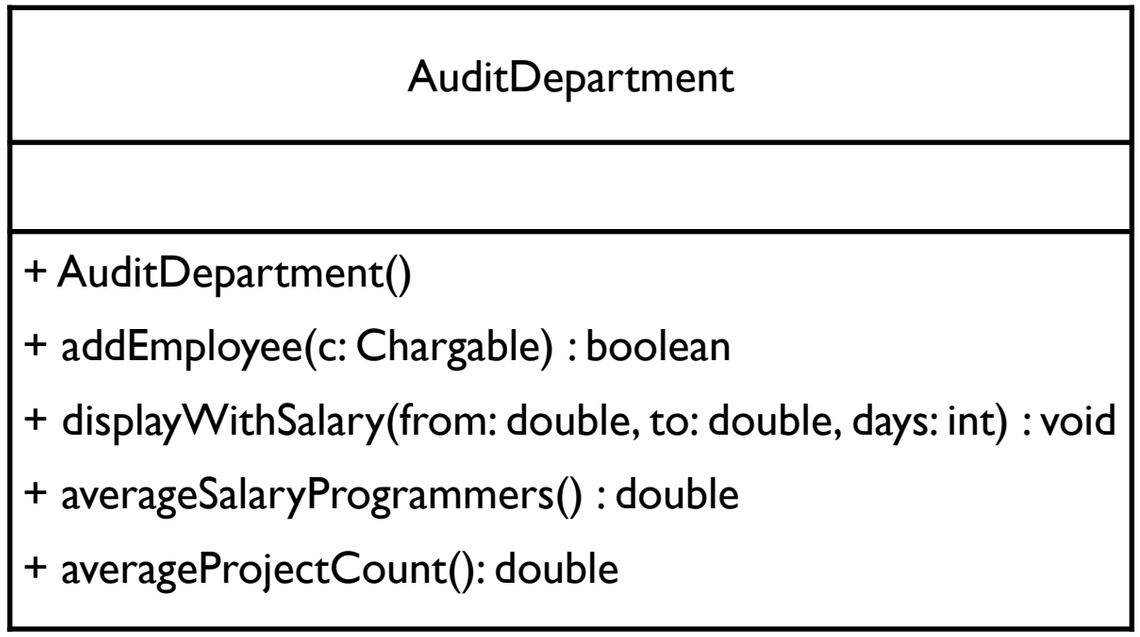

```
public final class ProjectManager extends Employee {
   int noProjects;

   public ProjectManager(int id, String name, double dailySalary,
                         int noProjects) {
      super(id, name, dailySalary);
      this.noProjects = noProjects;
   }

   public ProjectManager(ProjectManager pm) {
      super(pm);
      noProjects = pm.noProjects;
   }
}
```

Implemented Methods



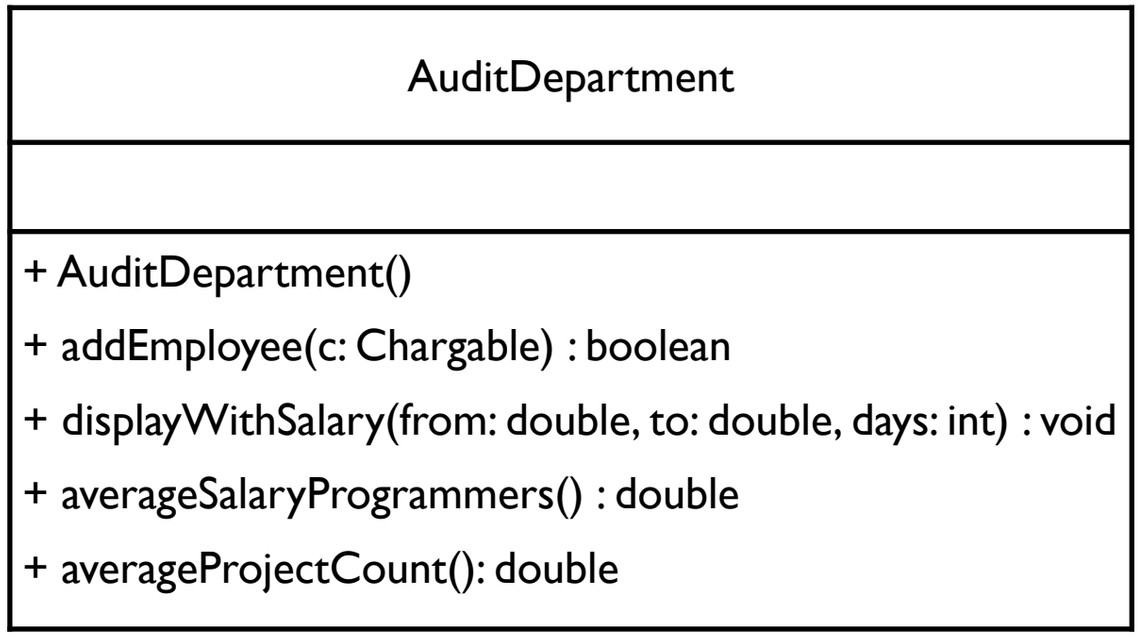

```
public double salaryWithBenfits(int days) {
     return getDailySalary() * days + noProjects * 500;
  }

public void display() {
   super.display();
   System.out.println("Monthly Salary (with benfits): "
                    + salaryWithBenfits(30));
}
```

AuditDepartment Class Constructor



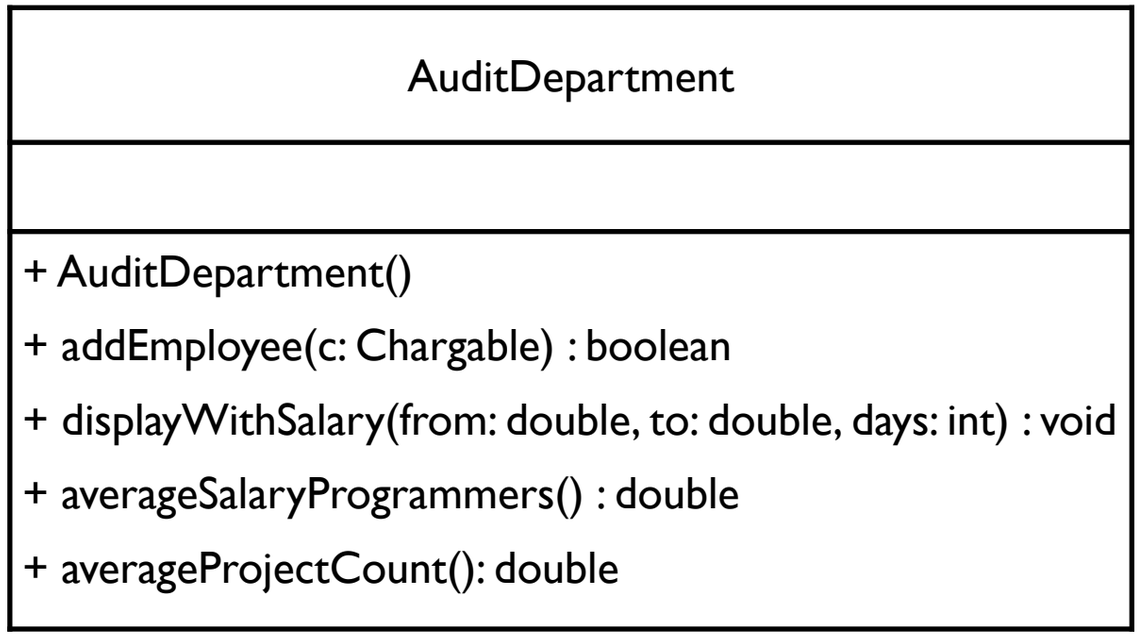

```
public class AuditDepartment {
   Chargable[] emp;
int current;

public AuditDepartment(int size) {
   emp = new Chargable[size];
   current = 0;
}
```

AuditDepartment Class addEmployee



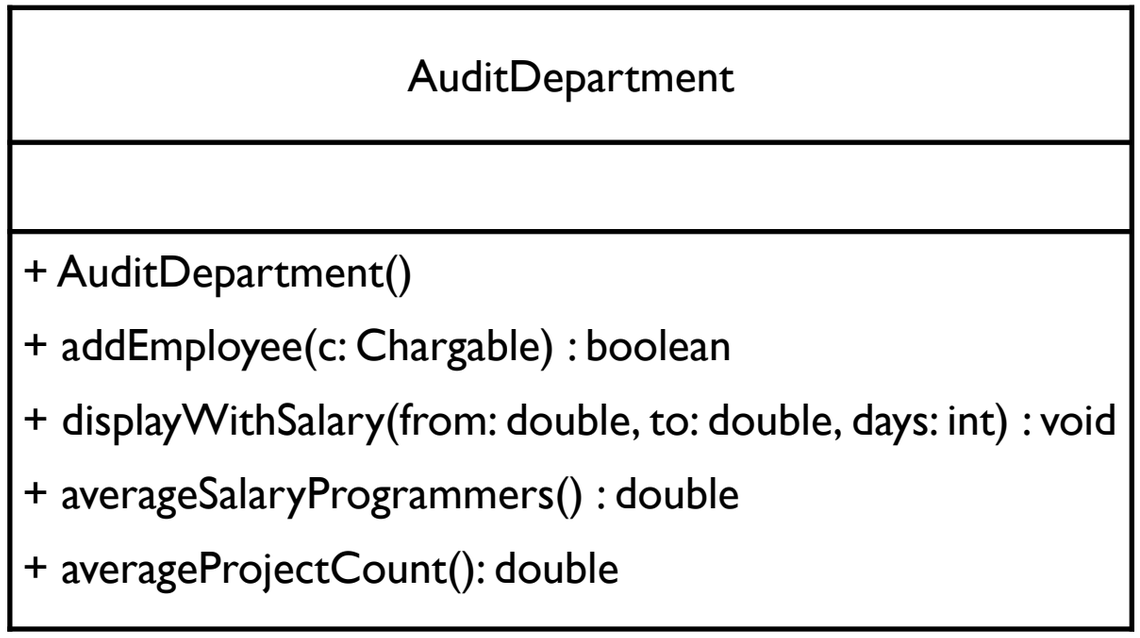

```
public boolean addEmployee(Chargable c) {
    emp[current] = c;
    current++;
    return true;
}
```

AuditDepartment Class display with salary



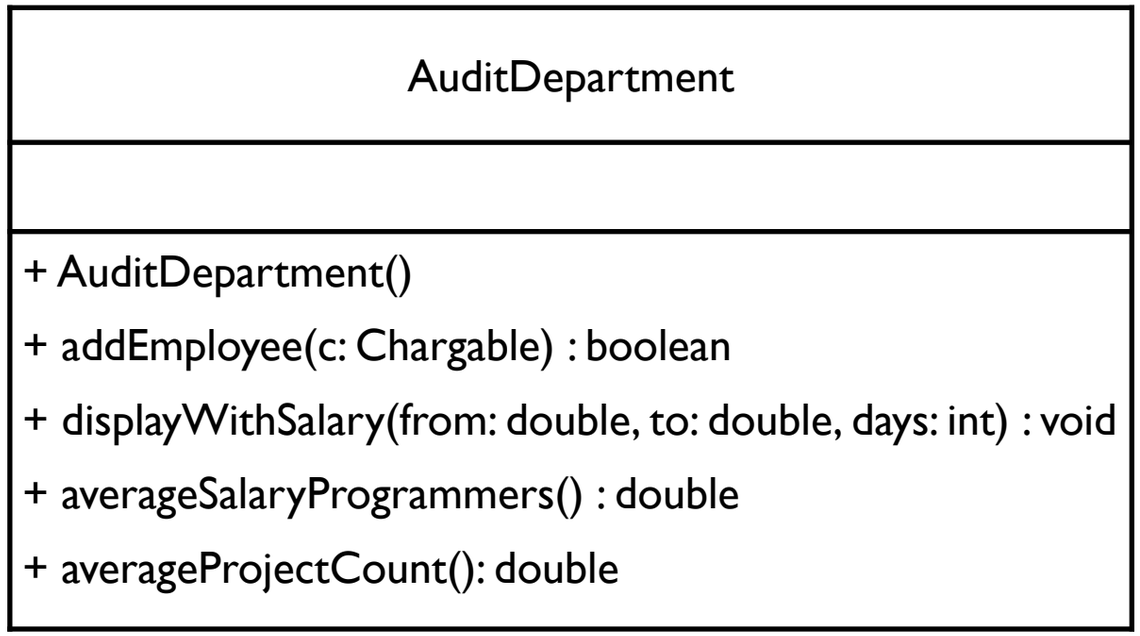

```
public void displayWithSalary(double from,  double to, int days){

      if(from > to || days < 0)
         throw new IllegalArgumentException();

    for(int i = 0; i < current; i++) {
       if(emp[i].salaryWithBenfits(days) >= from &&
           emp[i].salaryWithBenfits(days) <= to) {
          emp[i].display();
       }
    }
}
```

AuditDepartment Class average salary for programmers




```
public double averageSalaryProgrammers(){
     double sum = 1;
     int count = 0;

    for(int i = 0; i < current; i++) {
       if(emp[i] instanceof Programmer) {
          sum += emp[i].salaryWithBenfits(30);
          count++;
       }
    }
    if(count == 0)
       throw new ArithmeticException("Programmers");
    return sum / count;
}
```

AuditDepartment Class average projects for project managers




```
public double averageProjectCount() {
      double sum = 0;     int count = 0;
      for(int i = 0; i < current; i++) {
         if(emp[i] instanceof ProjectManager) {
            ProjectManager pm = (ProjectManager)emp[i];
            sum += pm.getNoProjects();
            count++;
         }
      }
      if(count == 0)
         throw new ArithmeticException("Project Managers");

      return sum / count;
   }
}
```

Main

```
public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      AuditDepartment ad;

      System.out.print("Please enter size of employee");
```

```
      try
      {
         ad = new AuditDepartment(input.nextInt());
      }
      catch(NegativeArraySizeException ex2)
      {
         System.out.println("Array size should not be negative");
         return;
      }
```

Main

```
do {
     displayMenu()
     choice = input.nextInt();
     try
     {
        switch(choice) {
        case 1:
       ...
            ad.addEmployee(p);
            break;
        case 2:
       ...
            ad.addEmployee(pm);
            break;
        case 3:
       ...
            ad.displayWithSalary(from, to, days);
            break;
        case 4:
            System.out.println("Average Salary for Programmers: "
                        + ad.averageSalaryProgrammers());
            break;
        case 5:
            System.out.println("Average Project Count for Project Managers: "
                        + ad.averageProjectCount());
            break;
     }
}
```

Main

```
catch(ArithmeticException e)
{
 System.out.println("No "+e.getMessage()+" avaiable");
}
catch(ArrayIndexOutOfBoundsException e)
{
 System.out.println("Array is full");
}
catch(IllegalArgumentException e) {
 System.out.println("to should be larger than from, days should be
positive");
}
catch(Exception e)
{
 System.out.println(e);
}

} while(choice != 6);
System.out.println("Bye!");
}
```