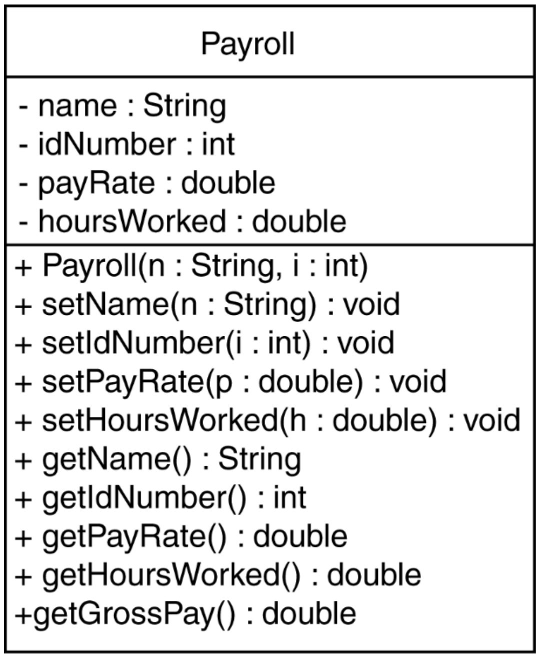
Department of Computer Science
420-101-VA Programming 1
Teacher: Nagat Drawel

Week 10 part 2: User-defined Classes

1. Design a Payroll class that has fields for an employee’s name, ID number, hourly pay rate, and number of hours worked. Write the appropriate accessor and mutator methods and a constructor that accepts the employee’s name and ID number as arguments. The class should also have a method that returns the employee’s gross pay, which is calculated as the number of hours worked multiplied by the hourly pay rate. Write a program that demonstrates the class by creating a payroll object, then asking the user to enter the data for an employee. The program should display the amount of gross pay earned.


Payroll
- name : String
- idNumber : int
- payRate : double
- hoursWorked : double
+ Payroll(n : String, i : int)
+ setName(n : String) : void
+ setIdNumber(i : int) : void
+ setPayRate(p : double) : void
+ setHoursWorked(h : double) : void
+ getName() : String
+ getIdNumber() : int
+ getPayRate() : double
+ getHoursWorked() : double
+getGrossPay() : double

2. Write a class named Employee that has the following fields:

   name. The name field references a String object that holds the employee’s name.

   idNumber. The idNumber is an int variable that holds the employee’s ID number.

   department. The department field references a String object that holds the name of the department where the employee works.

position. The position field references a String object that holds the employee's job title. The class should have the following constructors:

A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name, employee's ID number, department, and position.

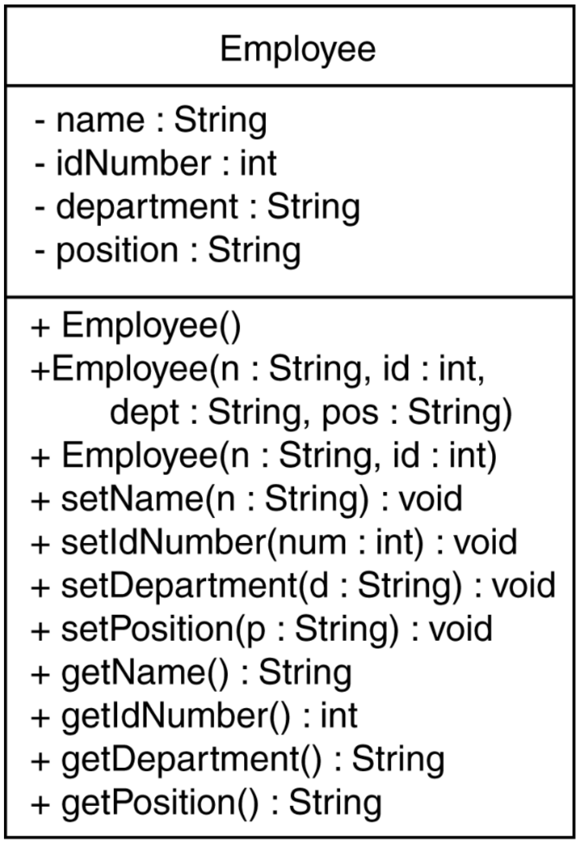
A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name and ID number. The department and position fields should be assigned an empty string ("").

A no-arg constructor that assigns empty strings ("") to the name, department, and position fields, and 0 to the idNumber field.

Write appropriate mutator methods that store values in these fields and accessor methods that return the values in these fields. Once you have written the class, write a separate program that creates three Employee objects to hold the following data:

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects and then display the data for each employee on the screen.



```
Employee
- name : String
- idNumber : int
- department : String
- position : String
+ Employee()
+Employee(n : String, id : int,
      dept : String, pos : String)
+ Employee(n : String, id : int)
+ setName(n : String) : void
+ setIdNumber(num : int) : void
+ setDepartment(d : String) : void
+ setPosition(p : String) : void
+ getName() : String
+ getIdNumber() : int
+ getDepartment() : String
+ getPosition() : String
```

3. Write a class named Car that has the following fields:

   yearModel. The yearModel field is an int that holds the car’s year model.

   make. The make field references a String object that holds the make of the car.

   speed. The speed field is an int that holds the car’s current speed.

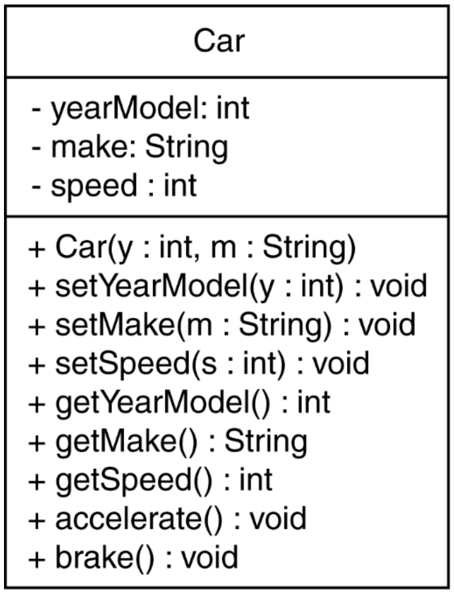
   In addition, the class should have the following constructor and other methods.

   Constructor. The constructor should accept the car’s year model and make as arguments. These values should be assigned to the object’s yearModel and make fields. The constructor should also assign 0 to the speed field.

   Accessors. Appropriate accessor methods should get the values stored in an object’s yearModel, make, and speed fields.

   accelerate. The accelerate () method should add 5 to the speed field each time it is called.   brake. The brake method should subtract 5 from the speed field each time it is called.

   Demonstrate the class in a program that creates a Car object, and then calls the accelerate method five times. After each call to the accelerate method, get the current speed of the car and display it. Then call the brake method five times. After each call to the brake method, get the current speed of the car and display it.


Car
- yearModel: int
- make: String
- speed : int
+ Car(y : int, m : String)
+ setYearModel(y : int) : void
+ setMake(m : String) : void
+ setSpeed(s : int) : void
+ getYearModel() : int
+ getMake() : String
+ getSpeed() : int
+ accelerate() : void
+ brake() : void