

+setDependents(in dep: string): void

, in ben : Benefit)

**Pseudocode**

//Program Header

//Program Name: Class Development

//Programmer: Your Name

//CIS247C, Week 4 Lab

//Program Description: PROVIDE A DESCRIPTITON OF THE PROGRAM

**Define class – Benefit**

private access specifier

// declare data members

healthInsurance as a string

lifeInsurance as a double

vacation as an integer

public access specifier

// declare methods

default constructor Benefit()

constructor with parameters Benefit(string, double, int)

declaration displayBenefits()

declaration getHealthInsurance() returns a string

declaration setHealthInsurance(string)

declaration getLifeInsurance() returns a double

declaration setLifeInsurance(double)

declaration getVacation() returns an integer

declaration setVacation(int)

**end define class**

**start define constructors/methods**

default constructor Benefit()

set the attributes as follows: healthInsurance = "not provided", lifeInsurance = 0.0, vacation = 14.

end constructor

constructor with parameters Benefit(string health, double life, int vacation)

should initialize all of the attributes using values passed in using its parameter list.

end constructor

procedure displayBenefits()

should display all the attributes of the Benefit object in a well-formatted string with logical labels applied to each attribute.

end procedure

procedure getHeathInsurance() returns a string

return value of healthInsurance data member.

end procedure

procedure setHealthInsurance(string hins)

assign value to healthInsurance data member using value passed in parameter list.

end procedure

procedure getLifeInsurance() returns a double

return value of lifeInsurance data member.

end procedure

procedure setLifeInsurance(double lifeIns)

assign value to lifeInsurance data member using value passed in parameter list.

end procedure

procedure getVacation() returns an integer

return value of vacation data member.

end procedure

procedure setVacation(int vaca)

assign value to vacation data member using value passed in parameter list.

end procedure

**end define constructors/methods**

**Define class – iEmployee**

public access specifier

declaration pure virtual calculatePay() returns a double

**end define class**

**Define class – Employee and inherit iEmployee as public**

private access specifier

// declare data members

firstName as a string

lastName as a string

gender as a character

dependents as an integer

annualSalary as a double

numEmployees as a static integer value of 0

public access specifier

default constructor Employee()

constructor with parameters Employee(string, string, char, int, double, Benefit)

declaration calculatePay()

declaration displayEmployee()

declaration getFirstName()

declaration setFirstName(string)

declaration getLastName()

declaration setLastName(string)

declaration getGender()

declaration setGender(char)

declaration getDependents()

declaration setDependents(int)

declaration getAnnualSalary()

declaration setAnnualSalary(double)

declaration static getNumEmployees()

declaration setDependents(string)

declaration setAnnualSalary(string)

benefit as a Benefit class composite object

**end define class**

**start define constructors/methods**

default constructor Employee() : benefits()

set the attributes as follows: firstName = "not given", lastName = "not given", gender = "U" (for unknown), dependents = 0, and annualSalary = 20,000.

Increment numEmployees by 1 in each of the constructors.

end constructor

constructor with parameters Employee(string first, string last, char gen, int dep,

double salary, **Benefit ben**)

should initialize all of the attributes using values passed in using its parameter list and pass on to the benefits object its attribute values.

Increment numEmployees by 1 in each of the constructors.

end constructor

procedure calculatePay()

should return the value of annual salary divided by 52 (return annualSalary / 52;).

end procedure

procedure displayEmployee()

should display all the attributes of the Employee object in a well-formatted string with logical labels applied to each attribute. Don't forget to call calculatePay from within the displayEmployee method in order to display the Employee's weekly pay as well!

should call the benefit object’s displayBenefit method.

end procedure

procedure getFirstName()

return value of firstName data member.

end procedure

procedure setFirstName(string first)

assign value to firstName data member using value passed in parameter list.

end procedure

procedure getLastName()

return value of lastName data member.

end procedure

procedure setLastName(string last)

assign value to lastName data member using value passed in parameter list.

end procedure

procedure getGender()

return value of gender data member.

end procedure

procedure setGender(char gen)

assign value to gender data member using value passed in parameter list.

end procedure

procedure getDependents()

return value of dependents data member.

end procedure

procedure setDependents(int dep)

assign value to dependents data member using value passed in parameter list.

end procedure

procedure setDependents(string dep)

Create an overloaded setDependents method and this time make the parameter a string.

end procedure

procedure getAnnualSalary()

return value of annualSalary data member.

end procedure

procedure setAnnualSalary(double salary)

assign value to annualSalary data member using value passed in parameter list.

end procedure

procedure setAnnualSalary(string salary)

Create an overloaded setAnnualSalary method and this time make the parameter a string.

end procedure

procedure getNumEmployees()

Make the getNumEmployees a static method. (This way, you can call it with the class name instead of an object name.)

end procedure

**end define constructors/methods**

**Start main**

1. Create an Employee object using the default constructor.
2. Prompt for and then set the first name, last name, and gender. Consider using your getInput method from Week 1 to obtain data from the user for this step as well as Step 3.
3. Prompt for and then set the dependents and annual salary using the overloaded setters that accept Strings.
4. **Prompt for and set healthInsurance, lifeInsurance, and vacation.**
5. Using your code from Week 1, display a divider that contains the string "Employee Information".
6. Display the Employee Information.
7. Display the number of employees created using getNumEmployees. Remember to access getNumEmployees using the class name, not the Employee object.
8. **Create a Benefit object called benefit1 using the multi-arg construction. Use any information you want for health insurance, life insurance, and vacation.**
9. Create a second Employee object using the multi-arg constructor, setting each of the attributes with the following values: **"Mary", "Noia", 'F', 5, 24000.0, benefit1**
10. Using your code from Week 1, display a divider that contains the string "Employee Information".
11. Display the employee information for the second Employee object.
12. Display the number of employees created using getNumEmployees. Remember to access getNumEmployees using the class name, not the Employee object.

**end main program**

**procedure DisplayApplicationInformation**

display “Welcome the Basic User Interface Program”

display “CIS247, Week 1 Lab”

display “Name: YOUR NAME”

display “This program accepts user input as a string, then makes the appropriate data conversion”

**end procedure**

**procedure DisplayDivider(string outputTitle)**

display “\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* “ + outputTitle + “\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”

**end procedure**

**function GetInput(string inputType) as string**

strInput as string

display “Enter the “ + inputType

get strInput

return strInput

**end function**

**procedure TerminateApplication**

display “Thank you for using the Basic User Interface program”

exit application

**end procedure**

Example output:

