COMP 3140 – TEAM PROJECT

Stage 2 – Class Descriptions

Prepared For:

Dr Andrew Park
TRU Computing Science
November 23, 2020

Cody Painter T00546693
Kiel Riley T00538035
Malcolm Todd T00232792

Contents

Person Abstract Class

Class Description

The Person abstract class contains those data items that are common to all people featured within the Rental Property Management Application. As an abstract class, this class is not intended to be instantiated, but rather to define the required data members to be inherited by the Tenant and Manager derived classes.

A Person consists of the name, gender, and age data members. The class provides protected visibility to the data members and the class' constructor to ensure that it can only be instantiated or accessed within any classes that are derived from it.

Class Data Members

The visibility of all data members within the Person class have been set to protected to allow for easy access from any derived classes.

name

A String type data member representing the person's full name. For example: "Damian Crase".

gender

A String type data member for the Textual representation of a person's gender. For example: "Female"

age

An int type data member representing a person's age in years. The age data member is constrained to allow for only positive values when creating a Person object. For example: 88

Class Member Functions

The Person class includes basic member functions for construction, destruction, and accessing data members. The class also includes a virtual function for getProfile() to enforce the requirement within any derived classes.

Default Constructor: *Person() = default;*

A default system generated constructor for the person class. This constructor has protected visibility and can only be called within a derived class.

Parameter Constructor: *Person(string name, int age, string gender);*

Parameter constructor with protected visibility to be used by derived class constructors. Initializes a Person object with the provided values.

Example Usage: Person("John Smith", 32, "Male");

Class Destructor: *virtual ~Person() = default;*

A default destructor to be called by the system when Person objects go out of scope. Declared as virtual to ensure proper destruction occurs when any derived class destructors are called.

Name Accessors: *string getName() const;*
void setName(string newName);

Member functions to retrieve and manipulate the name data member within the Person class.

Getter Example: Person1.getName();
Setter Example: Person1.setName("John Smith");

Age Accessors: *int getAge() const;*
void setAge(int newAge);

Member functions to retrieve and manipulate the age data member within the Person class.

Getter Example: Person1.getAge();
Setter Example: Person1.setAge(88);

Gender Accessors: *string getGender() const;*
void setGender(string newGender);

Member functions to retrieve and manipulate the gender data member within the Person class.

Getter Example: Person1.getGender();
Setter Example: Person1.setGender("Male");

Get Profile Virtual Function: *virtual string getProfile() = 0;*
A pure virtual member function to be defined in the Tenant or Manager derived classes that should return a string representation as a profile for the object to which it relates.

Example: Manager1.getProfile(); OR Tenant1.getProfile();

Tenant Class

Class Description

The Tenant class is a concrete class derived from the Person class. The Tenant adds additional data members and member functions which relate to tracking specific information relating to a tenant's employment, address, move in date, and payment history. The class provides class constructors, destructors, and accessors. Additionally, the class provides functionalities for retrieving payment status, calculating fee information, reporting unpaid fees, and printing a textual representation of a Tenant's profile.

Class Data Members

A Tenant consists of the job, unitNo, moveInDate, rentalFee, and paymentStatus data members which are created with private visibility to enforce encapsulation.

job
A String data member representing a Tenant's employment title. For example: "Dentist".
unitNo
An integer data member representing the address number within the mobile park. For example: 11

moveInDate
A String data member which stores a textual representation of the date on which a Tenant moved into the mobile park. For example: "7/25/2001"

rentalFee
A double type data member which stores a floating-point value representing the monthly rental amount in dollars charged to the applicable Tenant to maintain their occupancy. For example: 703.45

paymentsStatus
A vector of six Boolean values representing a list of statuses for whether the applicable Tenant was in good standing for each month of the current reporting period.

For example: {True, False, True, True, True, False}

Class Member Functions

The Tenant class includes basic member functions for construction, destruction, and accessing data members. The class also includes functionalities for calculating fees information in specific months or for the entire reporting period. Additionally, the class provides the getProfile() method to generate a string which summarizes the Tenant object's data members.

Default Constructor: *Tenant() = default;*
Default constructor generated by the system.

Class Destructor: *~Tenant() = default;*
Default class destructor generated by the system.

Parameter Constructor: *Tenant(string name, int age, string gender, int unitNo,*
string moveInDate, double monthlyRentalFee);

Class constructor which initializes a Tenant object using the provided information from the user.

Example Usage: Tenant("John Smith", 32, "Male", 17, "01/01/2001", 900.00);

Job Accessors: *string getJob() const;*
void setJob(string job);

Member functions to retrieve and manipulate the job string data member within the Tenant class.

Getter Example: Tenant1.getJob();
Setter Example: Tenant1.setJob(“Janitor”);

UnitNo Accessors: *int getUnitNo() const;*
void setUnitNo(int unitNo);

Member functions to retrieve and manipulate the unitNo integer data member within the Tenant class.

Getter Example: Tenant1.getUnitNo();
Setter Example: Tenant1.setUnitNo(17);

RentalFee Accessors: *double getRentalFee() const;*
void setRentalFee(double rentalFee);

Member functions to retrieve and manipulate the rentalFee double data member within the Tenant class.

Getter Example: Tenant1.getRentalFee();
Setter Example: Tenant1.setRentalFee(900.00);

MoveInDate Accessors: *string getMoveInDate() const;*
void setMoveInDate(string moveInDate);

Member functions to retrieve and manipulate the moveInDate string data member within the Tenant class.

Getter Example: Tenant1.getMoveInDate();
Setter Example: Tenant1.setMoveInDate(“01/01/2010”);

PaymentsStatus Accessors: *vector<bool> getPaymentsStatus() const;*
void setPaymentsStatus(vector<bool> paymentsStatus) const;

Member functions to retrieve and manipulate the paymentsStatus vector data member of boolean values within the Tenant class.

Getter Example: Tenant1.getPaymentsStatus();
Setter Example: Tenant1.setPaymentsStatus({True, False, True, True, True, False});

Accessors for PaymentStatus in a specific month:

bool getPaymentStatusForMonth(int month) const;
void setPaymentStatusForMonth(int month, bool status);

Provides functionality to get or set the payment status for a given month from within the Payment Status vector type data member.

Get Example: Tenant1.getPaymentStatusForMonth(1);
Set Example: Tenant1.setPaymentStatusForMonth(1, False);

Calculate Unpaid fees functionality member functions:

double calcUnpaidFeesForMonth(int month) const;
double calcUnpaidFeesTotal() const;

Provides the functionality to calculate the amounts of any unpaid fees accrued by the Tenant for a specific month, or over the entire reporting period of six months.

For Month Example: Tenant1.calcUnpaidFeesForMonth(1);
Total Example: Tenant1.calcUnpaidFeesTotal();

Calculate paid fees functionality member functions:

double calcPaidFeesforMonth(int month) const;
double calcPaidFeesTotal() const;

Provides the functionality to calculate the amounts of any paid fees provided by the Tenant for a specific month or over the entire reporting period of six months.

For Month Example: Tenant1.calcPaidFeesForMonth(3);
Total Example: Tenant1.calcPaidFeesTotal();

Test for unpaid fees member function: *bool hasUnpaidFees() const;*
Scans the Tenant’s payment history for the reporting period to check if there are any unpaid rental fees reported during the reporting period of six months.

Example Usage: Tenant1.hasUnpaidFees();

Retrieve Tenant’s profile member function: *std::string getProfile() const;*
Returns a string containing a textual representation of all data members for the applicable Tenant object as a summary of the individual’s personal information.

Example Usage: Tenant1.getProfile();

Manager Class

Class Description

The Manager class is a concrete class derived from the Person class. The Manager adds additional data members and member functions which relate to tracking specific information relating to the manager's date of hire, remuneration and expense history during the reporting the period. The class provides constructors, destructors, and accessors. Additionally, the class provides functionalities for calculating expense amounts, salary and bonus information, and printing a textual representation of a manager's profile.

Class Data Members

A Manage consists of the hireDate, salary, bonus, and monthlyExpenses data members which are created with private visibility to enforce encapsulation.

hireDate

A string type data member which stores a representation of the date upon which the applicable manager was hired. For example: "06/01/2013"

salary

A double type data member which stores a floating-point number representing the amount in dollars to which the manager in entitled to monthly for services rendered. For example: 3500.00

bonus

A double type data member which stores a floating-point number representing the amount of additional dollars to which the manager in entitled monthly as a bonus in the reporting period. For example: 500.00

monthlyExpenses

A vector of double data values representing a list of total expenses accrued by the manager from their employment duties during each month of the reporting period.

For example: {941.07, 1067.33, 1581.68, 1221.22, 1404.23, 896.63}

Class Member Functions

The Manager class includes basic member functions for construction, destruction, and accessing data members. The class also includes functionalities for calculating expense information or manager remuneration amounts in specific months or for the entire reporting period. Additionally, the class provides the getProfile() method to generate a string which summarizes the Manager object's data members.

Default Constructor: *Manager() = default;*

Default constructor generated by the system.

Default Destructor: *~Manager() = default;*

Default destructor generated by the system.

Parameter Constructor: *Manager(string name, int age, string gender, string hireDate, double salary, double bonus);*

Class constructor which initializes a Manager object using the provided information from the user.

Example Usage: Manager("John Smith", 32, "Male", "06/01/2013", 3500.00, 500.00);

HireDate Accessors: *string getHireDate() const;*
void setHireDate(string newHireDate);

Member functions to retrieve and manipulate the hireDate string data member within the Manager class.

Getter Example: Manager1.getHireDate();
Setter Example: Manager1.setHireDate("06/01/2013");

Salary Accessors: *double getSalary() const;*
void setSalary(double newSalary);

Member functions to retrieve and manipulate the salary double data member within the Manager class.

Getter Example: Manager1.getSalary();
Setter Example: Manager1.setSalary(3500.00);

Bonus Accessors: *double getBonus() const;*
void setBonus(double newBonus);

Member functions to retrieve and manipulate the bonus double data member within the Manager class.

Getter Example: Manager1.getBonus();
Setter Example: Manager1.setBonus(500.00);

MonthlyExpenses Accessors: *vector<double> getMonthlyExpenses() const;*
void setMonthlyExpenses(vector<double> monthlyExpenses);

Member functions to retrieve and manipulate the monthlyExpenses vector type data member containing a list of double values within the Manager class.

Getter Example: Manager1.getMonthlyExpenses();
Setter Example: Manager1.setMonthlyExpenses({941.07, 967.33, 1581.68, 1221.22, 1404.23, 896.63});

Accessors for Expense in a specific month:

double getExpenseForMonth(int month) const;
void setExpenseForMonth(int month, double expense);

Provides functionality to get or set the payment status for a given month from within the Payment Status vector type data member.

Get Example: Tenant1.getExpenseForMonth(1);
Set Example: Tenant1.setExpenseForMonth(1, 917.37);

Calculate paid fees functionality member function:

double calcExpenseTotal() const;

Provides functionality to calculate the total amount of any expenses accrued by the Manager during the reporting period of six months.

Example Usage: Manager1.calcExpenseTotal();

Calculate salary functionality member functions:

double calcSalaryForMonth(int month) const;
double calcSalaryTotal() const;

Provides the functionality to calculate the amounts of any salary earned by the Manager for a specific month, or over the entire reporting period of six months.

For Month Example: Manager1.calcSalaryForMonth(1);
Total Example: Manager1.calcSalaryTotal();

Calculate bonus functionality member functions:

double calcBonusForMonth(int month) const;
double calcBonusTotal() const;

Provides the functionality to calculate the amounts of any bonus earned by the Manager for a specific month, or over the entire reporting period of six months.

For Month Example: Manager1.calcBonusForMonth(1);
Total Example: Manager1.calcBonusTotal();

Calculate total remuneration (pay) functionality member functions:

double calcRemunerationForMonth(int month) const;
double calcRemunerationTotal() const;

Provides the functionality to calculate the amounts of any total pay (salary and bonus) earned by the Manager for a specific month, or over the entire reporting period of six months.

For Month Example:	Manager1.calcRemunerationForMonth(1);
Total Example:	Manager1.calcRemunerationTotal();

Retrieve Manager's profile member function:	*string getProfile() const;*
Returns a string containing a textual representation of all data members for the applicable Manager object as a summary of the individual's personal information.

Example Usage:	Manager1.getProfile();

RentalPropertyManager Class

Class Description

The RentalPropertyManager class is a concrete class. The class is intended to encapsulate an applicable six-month summary period that the managers are required provide to the owner. Instance of this class are to be instantiated within the application file and will provide the Rental Management System with the key functionalities to meet the specified reporting requirements. The class provides constructors, a destructor, member functions to allow data to be imported from CSV files, and member functions to provide useful reporting figures from the imported data.

Class Data Members

A RentalPropertyManager consists of a tenantList data member using a vector of Tenant objects, and a managerList data member using a vector of Manager objects. These vectors are set to private visibility to enforce encapsulation.

tenantList
A vector type data member containing a list of Tenant objects representing the habitants of the mobile park being managed. For Example {Tenant1, Tenant2, Tenant3, … , TenantN}

managerList
A vector type data member containing a list of Manager objects representing those managers currently employed to manage the applicable mobile park. For example: {Manager1, Manager2}

Class Member Functions

The RentalPropertyManager class includes basic member functions for construction, destruction, and accessing data members. The class also includes functionalities for reading tenant and manager information from CSV files to populate the class' vectors, and reporting functionalities relating to calculating revenue, expense, delinquent account details, and printing profile information.

Default Constructor: RentalPropertyManager() = default;
Default constructor generated by the system.

Default Constructor: ~RentalPropertyManager() = default;
Default destructor generated by the system is sufficient as there is no manual memory management done within the class.

Member functions to provide file input functionality:

void loadManagers(string managerDataFile);
void loadTenants(string tenantDataFile);

Member functions to facilitate the import of tenant and manager data from CSV files

Manager Useage Example: PropertyManager1.loadManagers("Manager.csv");
Tenant Usage Example: PropertyManager1.loadTenants("Tenants.csv");

Member functions to provide revenue calculations (from rent):

double calcCollectedRentTotal() const;
double calcCollectedRentForMonth(int month) const;
string generateCollectedRentMonthlySummary() const;

Member functions that calculate the rent for all tenants for all six months, a specific month, or that can generate a month-by-month summary of collected tenant rents.

Total Example:	PropertyManager1.calcCollectedRentTotal();
For Month Example:	PropertyManager1.calcCollectedRentForMonth(1);
Summary Example:	PropertyManager1.generateCollectedRentMonthlySummary();

Member functions to provide lost revenue calculations (from unpaid rent):

double calcUnpaidRentTotal() const;
double calcUnpaidRentForMonth(int month) const;
string generateUnpaidRentMonthlySummary() const;

Member functions that calculate the unpaid rent amounts for all tenants for all six months, a specific month, or that can generate a month-by-month summary of uncollected tenant rent amounts.

Total Example:	PropertyManager1.calcUnpaidRentTotal();
For Month Example:	PropertyManager1.calcUnpaidRentForMonth(1);
Summary Example:	PropertyManager1.generateUnpaidRentMonthlySummary();

Member functions to provide management expense calculations (from manager expenses):

double calcManagerExpenseTotal() const;
double calcManagerExpenseForMonth(int month) const;
string generateManagerExpenseMonthlySummary() const;

Member functions that calculate the accrued expenses from managers for all six months, a specific month, or that can generate a month-by-month summary of manager expense amounts.

Total Example:	PropertyManager1.calcManagerExpenseTotal();
For Month Example:	PropertyManager1.calcManagerExpenseForMonth(1);
Summary Example:	PropertyManager1.generateManagerExpenseMonthlySummary();

Member functions to provide management remuneration expense calculations (from salary & bonus):

double calcManagerRemunerationTotal() const;
double calcManagerRemunerationForMonth(int month) const;
string generateManagerRemunerationMonthlySummary() const;

Member functions that calculate the remuneration (salary and bonus) amounts as an expense to employ the managers for all six months, a specific month, or that can generate a month-by-month summary of manager total remuneration cost amounts.

Total Example:	PropertyManager1.calcManagerRemunerationTotal();
For Month Example:	PropertyManager1.calcManagerRemunerationForMonth(1);
Summary Example:	PropertyManager1.generateManagerRemunerationMonthlySummary();

Member functions to provide net income calculations (revenues less expenses):

double calcNetIncomeTotal() const;
double calcNetIncomeForMonth(int month) const;
string generateNetIncomeMonthlySummary() const;

Member functions that calculate the net income amounts experienced by the mobile park being managed for all six months, a specific month, or that can generate a month-by-month summary of total net income amounts for the owner.

Total Example:	PropertyManager1.calcNetIncomeTotal();
For Month Example:	PropertyManager1.calcNetIncomeForMonth(1);
Summary Example:	PropertyManager1.generateNetIncomeMonthlySummary();

Member function to generate a list of all profiles:vector<   string> getAllProfiles();
Member function which returns a vector of String objects containing all profile details from managers and tenants that are currently in the system.

Example Usage: PropertyManager1.getAllProfiles();

Member function to list tenants with amounts outstanding (unpaid rents):

vector<Tenant> getTenantsWithUnpaidAmounts() const;

Member function which returns a vector of Tenant objects for any tenant's with amount owing in the six-month report period.

Example Usage: PropertyManager1.getTenantsWithUnpaidAmounts();