Patient Charges

Stephen Gerkin
September 2, 2019
CIST 2362 – C++ II
CRN 22282
Programming Lab 03

Program Description

Patient charges demonstrates the **Patient** class and **Procedure** class by creating instantiating a **Patient** object and three **Procedure** objects stored in an array from the sample data provided. The patient information is then printed to the console to demonstrate the data stored by the **Patient** object.

Each procedure is then printed to the console using a *foreach* loop to iterate over the array of **Procedure** objects. While iterating, the total charges stored by the procedures are summed and stored in a variable to be displayed at the conclusion of the program.

Class Documentation

Patient Class Reference

Patient class holds information about a patient.

Public Member Functions

- **Patient** (const std::string &firstName, const std::string &middleName, const std::string &lastName, const std::string &address, const std::string &city, const std::string &state, const std::string &zipCode, const std::string &phoneNumber, const std::string &emergencyContactName, const std::string &emergencyContactPhone)
  *Default **Patient** constructor.*

- **Patient** (const **Patient** &)
  Copy constructor.

- std::string **getFirstName** ()
  *Getter for **Patient** first name.*

- void **setFirstName** (const std::string &firstName)
  *Setter for **Patient** first name.*

- std::string **getMiddleName** ()
  *Getter for **Patient** middle name.*

- void **setMiddleName** (const std::string &middleName)
  *Setter for **Patient** middle name.*

- std::string **getLastName** ()
  *Getter for **Patient** last name.*

- void **setLastName** (const std::string &lastName)
  *Setter for **Patient** last name.*

- std::string **getAddress** ()
  *Getter for **Patient** address.*

- void **setAddress** (const std::string &address)
  *Setter for **Patient** address.*

- std::string **getCity** ()
  *Getter for **Patient** city.*

- void **setCity** (const std::string &city)
  *Setter for **Patient** city.*

- std::string **getState** ()
  *Getter for **Patient** state.*

- void **setState** (const std::string &state)
  *Setter for **Patient** state.*

- std::string **getZipCode** ()
  *Getter for **Patient** zip code.*

- void **setZipCode** (const std::string &zipCode)
  *Setter for **Patient** zip code.*

- std::string **getPhoneNumber** ()
  *Getter for **Patient** phone number.*

- void **setPhoneNumber** (const std::string &phoneNumber)
  *Setter for **Patient** phone number.*

- std::string **getEmergencyContactName** ()
  *Getter for **Patient** emergency contact name.*

- void **setEmergencyContactName** (const std::string &emergencyContactName)
  *Setter for **Patient** emergency contact name.*

- std::string **getEmergencyContactPhone** ()
  *Getter for **Patient** emergency contact phone number.*

- void **setEmergencyContactPhone** (const std::string &emergencyContactPhone)
  *Setter for **Patient** emergency contact phone number.*

- void **print** ()
  *Prints *this to the console.*

- **~Patient** ()
  Destructor.

Detailed Description

Patient class holds information about a patient.

A class named **Patient** that has member variables for the following data:

- First name, middle name, last name
- Address, city, state, and ZIP code
- Phone number

Patient class has a constructor that accepts an argument for each member variable. The **Patient** class also has getter and setter functions for each member variable.

Constructor & Destructor Documentation

Patient::Patient (const std::string &*firstName*, const std::string &*middleName*, const std::string &*lastName*, const std::string &*address*, const std::string &*city*, const std::string &*state*, const std::string &*zipCode*, const std::string &*phoneNumber*, const std::string &*emergencyContactName*, const std::string &*emergencyContactPhone*)

Default **Patient** constructor.

The default constructor (and only constructor besides a deep-copy) for the **Patient** class. Requires all member variables to be initialized at instantiation. No implementation other than initializing the member variables.

IMPORTANT: Input is not validated for null values or empty strings.

Parameters

firstName	- The patient's first name
middleName	- The patient's middle name
lastName	- The patient's last name
address	- The patient's street address
city	- The patient's city of residence
state	- The patient's state of residence
zipCode	- the patient's zip code
phoneNumber	- The patient's phone number
emergencyContactName	- The name of the emergency contact for patient

emergencyContactPhone	- The phone number for patient emergency contact

Patient::Patient (const Patient &*obj*)

Copy constructor.

Provides a deep copy of a **Patient** object

Parameters

obj	- The **Patient** object to be copied.

Patient::~Patient ()

Destructor.

No implementation needed during destruction.

Procedure Class Reference

Procedure class holds information about procedures.

Public Member Functions

- **Procedure** (const std::string &name, const std::string &date, const std::string &practitionerName, const double &charge)
  *Default **Procedure** constructor.*

- **Procedure** (const **Procedure** &obj)
  Copy constructor.

- std::string **getName** ()
  *Getter for **Procedure** name.*

- void **setName** (const std::string &name)
  *Setter for **Procedure** name.*

- std::string **getDate** ()
  *Getter for **Procedure** date.*

- void **setDate** (const std::string &date)
  *Setter for **Procedure** date.*

- std::string **getPractitionerName** ()
  *Getter for **Procedure** practitioner name.*

- void **setPractitionerName** (const std::string &practitionerName)
  *Setter for **Procedure** practitioner name.*

- double **getCharge** ()
  *Getter for **Procedure** charge.*

- void **setCharge** (const double &charge)
  *Setter for **Procedure** charge.*

- void **print** ()
  *Prints *this to the console.*

- **~Procedure** ()
  Destructor.

Detailed Description

Procedure class holds information about procedures.

A class named **Procedure** that has member variables for the following data:

- Name of the procedure
- Date of the procedure
- Name of the practitioner who performed the procedure
- Charges for the procedure

Procedure class has a constructor that accepts an argument for each member variable. The **Procedure** class also has getter and setter functions for each member variable.

Constructor & Destructor Documentation

Procedure::Procedure (const std::string &*name*, const std::string &*date*, const std::string &*practitionerName*, const double &*charge*)

Default **Procedure** constructor.

The default constructor (and only constructor besides a deep-copy) for the **Procedure** class. Requires all member variables to be initialized at instantiation. No implementation other than initializing the member variables.

IMPORTANT: Input is not validated for null values or empty strings.

Parameters

name	- The procedure name
date	- The date of the procedure
practitionerName	- The name of the practitioner who performed the procedure
charge	- The charges incurred for the procedure

Procedure::Procedure (const Procedure &*obj*)

Copy constructor.

Provides a deep copy of a **Procedure** object

Parameters

obj	- The **Procedure** object to be copied.

Procedure::~Procedure ()

Destructor.

No implementation needed during destruction.

File Documentation

InputValidation.h File Reference

Functions

- bool **validYN** (const std::string &input)
  Validates Yes or No option selection.

- bool **validUnsignedInt** (const std::string &input)
  Validates Unsigned Integer numbers.

- bool **validUnsignedFloat** (const std::string &input)
  Validates Unsigned Floating-point numbers.

Function Documentation

bool validUnsignedFloat (const std::string &*input*)

Validates Unsigned Floating-point numbers.

Uses regular expression parsing to match input against valid unsigned floating-point number values.

Parameters

input	- String value to validate.

Returns

True if the input consists only of an unsigned floating-point number value.

bool validUnsignedInt (const std::string &*input*)

Validates Unsigned Integer numbers.

Uses regular expression parsing to match input against valid unsigned integer values.

Parameters

input	- String value to validate.

Returns

True if input consists only of an unsigned integer value.

bool validYN (const std::string &*input*)

Validates Yes or No option selection.

Uses regular expression parsing to match input against valid “Y” or “N” entry only. Case insensitive.

Parameters

input	– String value to validate.

Returns

True if input matches “Y” or “N”

patientCharges.cpp File Reference

Functions

- void **patientCharges** ()
  Demonstration function.

- bool **repeatProgram** ()
  Function to repeat program.

- int **main** ()
  Main entry point for program.

Function Documentation

int main ()

Main entry point for program.

void patientCharges ()

Demonstration function.

Demonstrates the **Patient** and **Procedure** classes by instantiating a **Patient** object and an array of **Procedure** objects with sample data. Prints the objects to the screen and calculates total charges for procedures then displays that.

bool repeatProgram ()

Function to repeat program.

Displays prompt to user asking if they would like to repeat the program. Waits for input consisting of "y" or "n" (case insensitive).

Returns

True if user wants to repeat program.

Sample Program Output