CS 202 – Assignment #3 **Welcome to WE CARE Hospital** **Fall 2022**

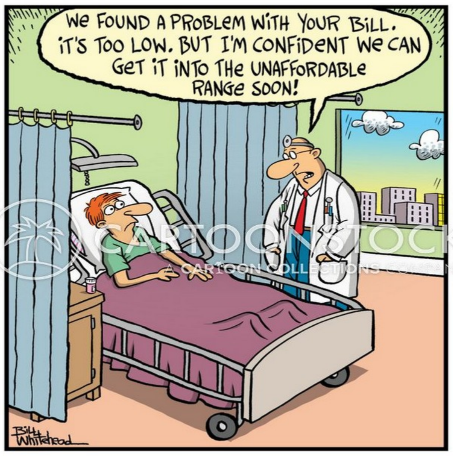
Purpose: Learn class inheritance, multi-file class implementation, and make utility.
Points: 100

Assignment:

Design and implement five C++ classes to provide a person-type, patient-type, doctor-type, date-type, and bill-type tracking functions. Each of the classes is described below.

Image Source: https://www.cartoonstock.com/cartoon?searchID=CS100844

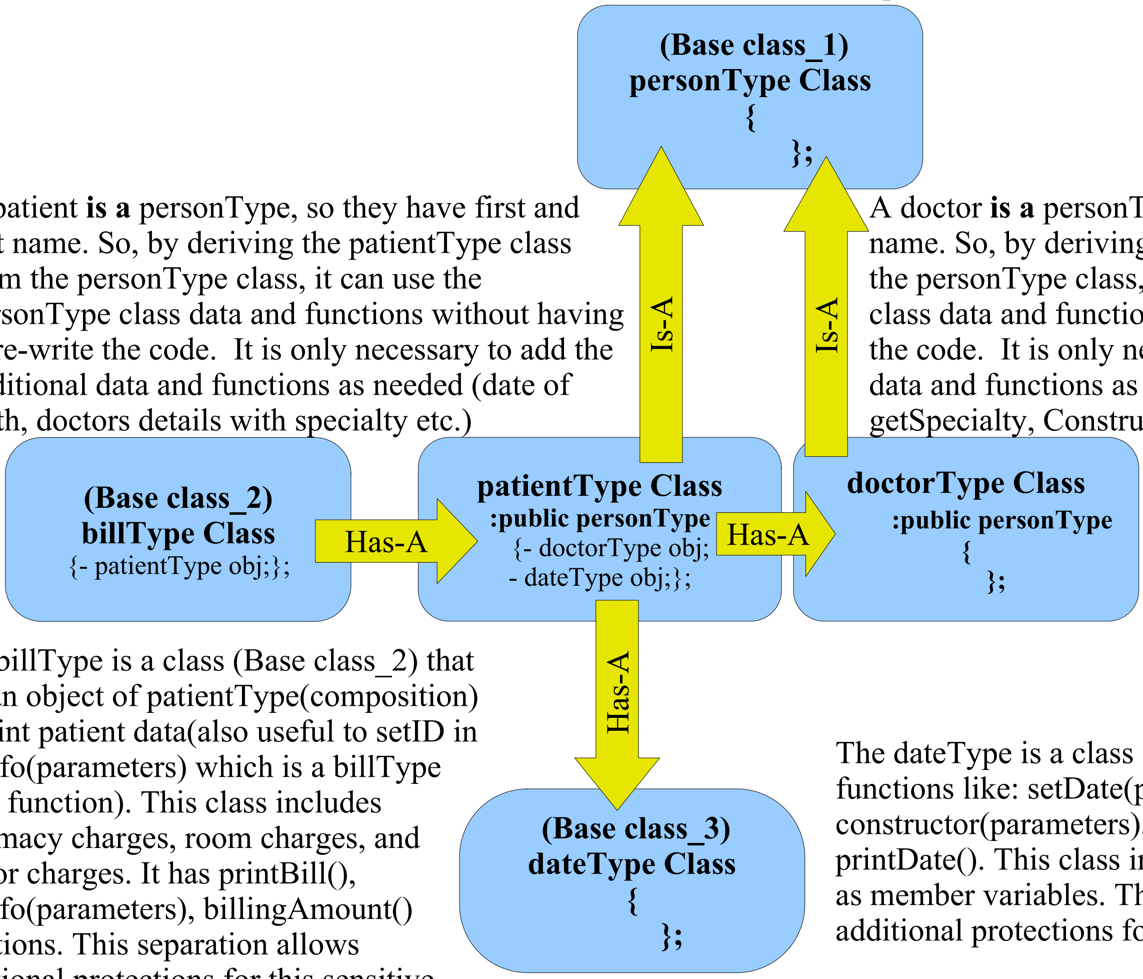
Video Link for Assignment_3: https://www.youtube.com/watch?v=P9uajz7frzc


WE FOUND A PROBLEM WITH YOUR BILL. IT'S TOO LOW. BUT I'M CONFIDENT WE CAN GET IT INTO THE UNAFFORDABLE RANGE SOON!
CARTOONSTOCK

Class Hierarchy

The following diagram should help understand the class hierarchy for this project.

The personType is a base class(Base class_1) encompasses basic person information like first name and last name; also functions to set and a constructor with parameters.


(Base class_1) personType Class { };
Is-A
Is-A
(Base class_2) billType Class {- patientType obj;};
Has-A
patientType Class :public personType {- doctorType obj; - dateType obj;};
Has-A
doctorType Class :public personType { };
Has-A
(Base class_3) dateType Class { };

A patient **is a** personType, so they have first and last name. So, by deriving the patientType class from the personType class, it can use the personType class data and functions without having to re-write the code. It is only necessary to add the additional data and functions as needed (date of birth, doctors details with specialty etc.)

A doctor **is a** personType, so they have first and last name. So, by deriving the patientType class from the personType class, it can use the personType class data and functions without having to re-write the code. It is only necessary to add the additional data and functions as needed (setSpecialty, getSpecialty, Constructor.)

The billType is a class (Base class_2) that has an object of patientType(composition) to print patient data(also useful to setID in setInfo(parameters) which is a billType class function). This class includes pharmacy charges, room charges, and doctor charges. It has printBill(), setInfo(parameters), billingAmount() functions. This separation allows additional protections for this sensitive data.

The dateType is a class (Base class_3) that has functions like: setDate(parameters), constructor(parameters), getter functions(), and printDate(). This class includes day, month, and year as member variables. This separation allows additional protections for this sensitive data.

The classes we will implement are as follows:

- **personType Class**
    - Implement a basic ***personType*** class(Base class_1), ***person***, will track person information and provide standard support functions for all persons. This will include first name, last name, default constructor. This will be the *base class* as this will apply to any person (patient or doctor). The personType class will inherit this functionality, so this class must be fully working before working on the patientType and doctorType classes.

- **dateType Class**
    - Implement a small class, ***dateType*** class(Base class_2), that contains dMonth, dDay, dYear as member variables. This class can set month, day, and year by using setter functions or parameterized constructor. This class isolates the sensitive birthday information of patient. This can be tested independently of other class.

- **doctorType Class**
    - This ***doctorType*** class is derived(public inheritance) from personType and sets the first name and last name of Doctor. This class contains specialty as a member variable, which is useful to set doctor's specialty. This class can set specialty using setter functions or a parameterized constructor to set first name, last name, and specialty.

- **patientType Class**
    - This ***patientType*** class is derived(public inheritance) from personType. This class sets the first name, last name of patient from personType, birth date (mm, dd, year) of patient using dateType dateOfBirth(composition), doctor's first name, last name, and specialty using doctorType attendingPhysician(composition). To set above parameters, this class can use setter functions or parameterized constructors as listed in header file. This class contains ID as a member variable, which is useful to set patient ID. The patient is 6 character long and the character should be always an upper case alphabet and the following five characters can be digits from 0 to 9. `bool checkPatientID(string patientIDTmp) const;` is used to check whether the ID is valid or invalid. The patientType parameterized constructor is the major function in this class, where this constructor is used to set the parameters accordingly and also the logic to print the following error messages like – Invalid ID, specialty missing for patient ***Williams,*** specialty missing for ***Dr. David,*** Invalid month entry for ***Thomas,*** Invalid year entry for ***Thomas.*** (Note: When you declare an object of objectType that invokes the parameterized constructor will print Error messages as above. However, if patientType obj is declared with no parameters then it prints Error messages like- Specialty is missing for Dr. ,invalid patientID for , Doctor Specialty is missing for patient. If you observe in the latter case it doesn't print the details with names as they are not set during patientType object declaration.)

- **billType Class**
    - Implement a small class, ***billType*** class(Base class_3), that contains ID, pharmacyCharges, roomRent, doctorFee, patientType pType(This object is useful to access print function of patientType) as member variables. This class sets the above parameters using setter functions or parameterized constructor. This class has `double billingAmount();` This function returns pharmacyCharges + roomRent + docFee. This class has `void printBill();` function to print the patients data like first name, last name, doctor details like first name, last name, and specialty. This `printBill()` function also includes to print pharmacyCharges, roomRent, doctorFee and total charges.

Inheritance, composition and fundamental concepts are important in object oriented programming and it is essential to fully understand. Refer to the text and class lectures for additional information regarding inheritance and composition.

Development and Testing

In order to simplify development and testing, the project is split into five parts: personType, patientType, doctorType, billType and dateType classes.

- The dateType can be developed first and this class can be tested independently of the other class.
- The personType must be developed next (before the derived classes patientType and doctorType). This class can be tested independently of the other class.
- The doctorType must be developed next and this class sets first name and last name using personType; it sets doctor's specialty through member variable(specialty). doctorType parameterized constructor is the major function in this class.
- The patientType must be developed next and this class sets first name and last name using personType; sets ID using member variable; birth date(dd, mm, year) using dateType dateOfBirth; doctors first name, last name, and speciality using doctorType attendingPhysician. patientType parameterized constructor is the major function in this class, where this constructor is used to set the parameters accordingly and also the logic to print the error messages. For more details or explanation refer to page 2.
- The billType must be developed last as this has the patientType pType(composition). The billType parameterized constructor is the major function in this class, where this constructor is used to set all parameters accordingly.

  Note: An object(patientType, doctorType, or billType) with no parameters prints error messages like doctor specialty missing for patient, specialty missing for Dr. , invalid patient ID by default. For such objects, you will set all the parameters using setter functions only. While using setters, if it has any invalid data like ID, birth date details, specialty missing, then those setter functions should print error messages accordingly. So the setter functions should have error checking logic. Check the sample output for reference and match your output accordingly (Formatting can be tricky if you are unaware of program flow; you should be clear how the print() functions are invoked).

Submission:

- All files must compile and execute on Ubuntu and compile with C++11.

- Submit source files
  - Submit a copy of the **source** files via the on-line submission.
  - *Note*, do ***not*** submit the provided mains (we have them).

- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.

- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All program and header files must include your name, section number, assignment, NSHE number, input and output summary. The required format is as follows:

```
/*
```

```
    Name: MY_NAME, NSHE, CLASS-SECTION, ASSIGNMENT
    Description: <per assignment>
    Input: <per assignment>
    Output: <per assignment>
    */
```

Failure to include your name in this format will result in a loss of up to 5%.

Code Quality Checks

A C++ linter[1] is used to perform some basic checks on code quality. These checks include, but are not limited to, the following:

- Unnecessary 'else' statements should not be used.
- Identifier naming style should be either camelCase or snake_case (consistently chosen).
- Named constants should be used in place of literals.
- Correct indentation should be used.
- Redundant return/continue statements should not be used.
- Selection conditions should be in the simplest form possible.
- Function prototypes should be free of top level *const*.

Not all of these items will apply to every program. Failure to to address these guidelines will result in a loss of up to 5%.

personType Class

Implement a basic *personType* class to provide personType functions. The *personType* UML class diagram is as follows:

personType
-lastName, firstName: string
+personType(string first="", string last="")
+setName(string first, string last): void
+getFirstName() const: string
+getLastName() const: string
+print() const: void

Function Descriptions:

The following are more detailed descriptions of the required functions.

- `personType(string first = "", string last = "");` This is a parameterized constructor that sets firstName and lastName according to the parameters. The default values of the parameters are empty strings.
- `void setName(string first, string last);` This function is to set firstName and lastName according to the parameters.
- `string getFirstName() const;` This function is to return the first name.
- `string getLastName() const;` This function is to return the last name.
- `void print() const;` This function is to output the first name and last name in the form firstName lastName.

dateType Class

Implement a basic student financial class named ***dateType*** to provide student financial support functions. The *money* UML class diagram is as follows:

1 For more information, refer to: https://en.wikipedia.org/wiki/Lint_(software)

dateType
-dMonth: int
-dDay: int
-dYear: int
+dateType(int=1, int=1, int=1910)
+setDate(int, int, int): void
+getDay() const: int
+getMonth() const: int
+getYear() const: int
+printDate() const: void

Function Descriptions:
The following are more detailed descriptions of the required functions.

- `dateType(int month = 1, int day = 1, int year = 1910);` This is a parameterized constructor to set the date. The member variables dMonth, dDay, and dYear are set according to the parameters. The default values of the parameters are as given in that parameterized constructor.
- `void setDate(int month, int day, int year);` This function is to set the date. The member variables dMonth, dDay, and dYear are set according to the parameters.
- `int getDay() const;` This function to return the day.
- `int getMonth() const;` This function to return the month.
- `int getYear() const;` This function to return the year.
- `void printDate() const;` This function to output the date in the form mm-dd-yyyy.

doctorType Class

The ***doctorType*** UML class diagram is as follows:

doctorType
-specialty: string
+doctorType(string"", string="", string="")
+setSpecialty(string): void
+getSpecialty()const: string
+print()const: void

Function Descriptions:
The following are more detailed descriptions of the required functions.

- `doctorType(string first = "", string last = "", string spl = "");` This is a parameterized constructor that sets firstName, lastName and specialty according to the parameters. This function should print error message if doctor’s specialty(3rd parameter) is missing.
- `void setSpecialty(string spl);` This function is to set specialty according to the parameter.
- `string getSpecialty() const;` This function is to return the specialty.
- `void print() const;` This function is to output the doctor's first name, last name and specialty

patientType Class

The ***patientType*** UML class diagram is as follows:

patientType
-ID: string
-dateOfBirth: dateType
-attendingPhysician: doctorType
+patientType(string = " ", string = " ", string = " ", int = 1, int = 1, int = 1910, string = " ", string = " ", string = " ")
+setInfo(string = " ", string = " ", string = " ", int = 1, int = 1, int = 1910, string = " ", string = " ", string = " "): void
+setID(string = " "): void
+setBirthDate(int=1, int=1, int=1910): void
+setDoctorName(string = " ", string = " "): void
+setDoctorSpl(string = " "): void
+getID() const: string
+getBirthYear() const: int
+getDoctorFName() const: string
+print()const: void
+checkPatientID(string)const: bool

Function Descriptions:
The following are more detailed descriptions of the required functions.

- `patientType(string id = "", string fName = "", string lName = "", int bDay = 1, int bMth = 1, int bYear = 1910, string docFrName = "", string docLaName = "", string docSpl = "");` This is a Parameterized Constructor to set the patient details like ID, first name, last name, birthday(dd,mm,yr); doctor details like first name, last name, specialty are set according to the parameters passed. This constructor should have the logic to check if ID is valid, check missing specialty and print error message if it has no specialty, check the range of month and year, and print error messages if they are not in range.
  Month Range: 1 to 12 is valid
  Year Range: 1910 to 2022 is valid
- `void setInfo(string id = "", string fName = "", string lName = "", int bDay = 1, int bMth = 1, int bYear = 1910, string docFrName = "", string docLaName = "", string docSpl = "");` This is a Parameterized Constructor to set the patient details like ID, first name, last name, birthday(dd,mm,yr); doctor details like first name, last name, specialty are set according to the parameters passed. Check all possible error conditions just like above constructor cases.
- `void setID(string id);` This function is to set the ID according to the parameter. If the ID is valid set the ID else print Error message.
- `void setBirthDate(int bDay = 1, int bMth = 1, int bYear = 1910);` This function is to set the birthDate. bDay, bMth, bYear are set according to the parameters. If none given they are set to the above default values. Check the range of month and year, and print error messages if they are not in range.
  Month Range: 1 to 12 is valid

Year Range: 1910 to 2022 is valid
- `void setDoctorName(string fName = "", string lName = "");` This function is to set the doctor's first name and last name according to the parameters.
- `void setDoctorSpl(string spl = "");` This function is to set the doctor's specialty according to the parameter.
- `int getBirthYear() const;` This function is to return the year
- `string getDoctorFName() const;` This function is to return doctor's first name.
- `void print() const;` This function is to output the first name and last name of patient; output the doctors first name, last name, and specialty using doctorType object(attendingPhysicain).
- `bool checkPatientID(string patientIDTmp) const;` This function is to check patientID; It has totally 6 characters; First character is upper case alphabet; Characters from 2 to 6 can be digits from 0 to 9; Failing to fulfill above conditions should return false else true.

<u>billType Class</u>

The ***billType*** UML class diagram is as follows:

billType
-ID: string
-pharmacyCharges: double
-roomRent: double
-doctorFee: double
-pType: patientType
+billType()
+billType(string, double, double, double, patientType)
+setInfo(string, double, double, double, patientType): void
+setID(string): void
+billingAmount(): double
+getID(): string
+printBill()const: void

<u>Function Descriptions:</u>
The following are more detailed descriptions of the required functions.
- `billType();` Default Constructor. The member variables ID, pharmacyCharges, roomRent, doctorFee, have to be initialized; no need to initialize pType here.
- `billType(string id, double phCharges, double rRent, double docFee, patientType patientObj);` This is Parameterized Constructor. The member variables ID, pharmacyCharges, roomRent, doctorFee, are set according to the parameters. Also, set pType with patientObj.
- `void setInfo(string id, double phCharges, double rRent, double docFee);` This function is to setInfo of a bill. The member variables ID, pharmacyCharges, roomRent, doctorFee, are set according to the parameters.
- `void setID(string id);` This function is to setID according to the parameter
- `string getID() const;` This function is to return the ID.
- `double billingAmount();` This function is to return pharmacyCharges + roomRent + doctorFee

- `void printBill();` This function is to output the first name and last name using pType obj. You can call the functions of personType using pType obj. This is to output Billing details like ID, pharmacyCharges, roomRent, doctorFee and finally total charges. This function prints any patients data only if the ID of patient match with the ID of billType obj.
- If a billType obj is declared with no parameters, then it should print Error messages(please check the sample execution at the end of document)

The personType class and dateType class must be fully completed prior to doctorType, patientType, and billType classes. Refer to the example executions for formatting. Make sure your program includes the appropriate documentation.

Note: Your implementation files should be fully commented.

Files Provided:
The following files are available to download for this assignment.
- makefile.
- main.cpp.
- personType.h
- dateType.h
- doctorType.h
- patientType.h
- billType.h
- pdf of Assignment3
- Ast3_Sample_Output

Submission:
Submit the following files for this assignment.
- personTypeImp.cpp.
- dateTypeImp.cpp.
- doctorTypeImp.cpp
- patientTypeImp.cpp.
- billTypeImp.cpp

How to Compile:
The provided make file assumes the source file names are as described. If you change the names, the make file will need to be edited. To build the given classes, one main provided and it has good and bad data. To compile, simply type:
- **`make`**

Which will create the *main* for executables respectively.

How to Run your code:
Use the following commands to run.
- ./main (This will print from patientType class, doctorType class, billType class and test cases from main.cpp)

Example how the Composition Works:

From patientType class
```
patientType(string id = "", string fName = "", string lName =
"",int bDay = 1, int bMth = 1, int bYear = 1910,string docFrName
= "", string docLaName = "", string docSpl = "");
```

From billType class

```
billType(string id, double phCharges, double rRent,
      double docFee, patientType patientObj);
```

billType class is having and object of patientType(composition). So this patientObj can access the functions related to patientType

From main.cpp

```
patientType paul("P33357", "Paul", "John", 8, 11, 1985,
        "Jose", "Clark", "Cardiac");

billType paulBill("P33357", 5897.65, 1720.50, 1100.90, paul);
```

If you examine the above code, patientType paul is passed to billType paulBill. (see billType(paramters) parameterized constructor for more details). Here paul.print() prints the patient details and doctor details with specialty only.

```
*************************************************************
Patient: Paul John
Patient ID: P33357
Attending Physician: Dr.Jose Clark; Specialty: Cardiac
_______________________________________________
```

paulBill.printBill() prints patient details, doctor details with specialty and also Billing details of that patient. PaulBill.printBill() will print the following only if the ID of paul matches with the ID of billType paulBill.

```
*************************************************************
Patient: Paul John
Patient ID: P33357
Attending Physician: Dr.Jose Clark; Specialty: Cardiac
_______________________________________________

Billing Charges for ID: P33357
Pharmacy Charges: $5897.65
Room Charges:     $1720.50
Doctor's Fees:    $1100.90
____________________________
Total Charges:    $8719.05

*************************************************************
```

<u>Example Execution:</u> (Note: I highlight some text in color to emphasize errors due to class objects with no parameters; Your original text color for output should be black only.)
Below is an example program execution output for the *main* program.

```
kishore@kishore-virtual-machine:/CS202_ast3_f22$ make
g++ -Wall -Wextra -pedantic -std=c++11 -g -c main.cpp
g++ -Wall -Wextra -pedantic -std=c++11 -g -c patientTypeImp.cpp
g++ -Wall -Wextra -pedantic -std=c++11 -g -c doctorTypeImp.cpp
```

```
g++ -Wall -Wextra -pedantic -std=c++11 -g -c dateTypeImp.cpp
g++ -Wall -Wextra -pedantic -std=c++11 -g -c billTypeImp.cpp
g++ -Wall -Wextra -pedantic -std=c++11 -g -o main main.o personTypeImp.o
patientTypeImp.o doctorTypeImp.o dateTypeImp.o billTypeImp.o
kishore@kishore-virtual-machine:/CS202_ast3_f22$ ./main

######################################################################
                    Welcome to WE CARE Hospital
######################################################################
Specialty is missing for Dr.David
Doctor Specialty is missing for patient William
Error: Invalid month entry for Thomas
Error: Invalid year entry for Thomas
Error: C9832 is invalid patientID for Charles
Error: d88888 is invalid patientID for Dennis
Specialty is missing for Dr.
Error:  is invalid patientID for
Doctor Specialty is missing for patient
Error: Invalid year entry for Lara
```

PatientType lara;
These errors are due to patientType lara with no parameters in main.cpp.

```
*****************************************************************
Patient: Lara King
Patient ID: L48532
Attending Physician: Dr.Jerry Nguyen; Specialty: Neurology
__________________________________________________
```

lara.print()

```
*****************************************************************
Patient: Lara King
Patient ID: L48532
Attending Physician: Dr.Jerry Nguyen; Specialty: Neurology
__________________________________________________

Billing Charges for ID: L48532
Pharmacy Charges: $1896.50
Room Charges:     $225.00
Doctor's Fees:    $345.95
______________________________
Total Charges:    $2467.45

*****************************************************************
```

laraBill.printBill()

```
Specialty is missing for Dr.
Error:  is invalid patientID for
Doctor Specialty is missing for patient
```

PatientType frank;
These errors are due to patientType frank with no parameters in main.cpp.

```
*****************************************************************
Patient: Frank Lee
Patient ID: F65780
Attending Physician: Dr.Bobby Scott; Specialty: Pediatric
__________________________________________________

Billing Charges for ID: F65780
Pharmacy Charges: $986.50
Room Charges:     $76.98
Doctor's Fees:    $124.95
______________________________
Total Charges:    $1188.43

*****************************************************************


*****************************************************************
Patient: William John
Patient ID: W12345
Attending Physician: Dr.David Jones; Specialty:
__________________________________________________

Billing Charges for ID: W12345
Pharmacy Charges: $245.50
Room Charges:     $76.98
```

```
Doctor's Fees:    $124.95
______________________________
Total Charges:    $447.43

*****************************************************************


*****************************************************************
Patient: Thomas Rodriguez
Patient ID: T12395
Attending Physician: Dr.Joe Roberts; Specialty: Ophthalmology
__________________________________________________

Billing Charges for ID: T12395
Pharmacy Charges: $546.50
Room Charges:     $120.00
Doctor's Fees:    $234.55
______________________________
Total Charges:    $901.05

*****************************************************************

ID mismatch or Invalid ID is passed; ID from BillType: C98327

ID mismatch or Invalid ID is passed; ID from BillType: d88888


*****************************************************************
Patient: Paul John
Patient ID: P33357
Attending Physician: Dr.Jose Clark; Specialty: Cardiac
__________________________________________________

Billing Charges for ID: P33357
Pharmacy Charges: $5897.65
Room Charges:     $1720.50
Doctor's Fees:    $1100.90
______________________________
Total Charges:    $8719.05

*****************************************************************


*****************************************************************
Patient: Henry Hall
Patient ID: H97651
Attending Physician: Dr.Eric Young; Specialty: Neurology
__________________________________________________

Billing Charges for ID: H97651
Pharmacy Charges: $1295.00
Room Charges:     $250.00
Doctor's Fees:    $600.90
______________________________
Total Charges:    $2145.90

*****************************************************************


*****************************************************************
Patient: Brian Jackson
Patient ID: B97893
Attending Physician: Dr.Walter Flores; Specialty: Cardiac
__________________________________________________

Billing Charges for ID: B97893
Pharmacy Charges: $3456.75
Room Charges:     $2000.00
Doctor's Fees:    $1300.00
______________________________
```

```
Total Charges:    $6756.75

*****************************************************************


*****************************************************************
Patient: Jeffrey Lewis
Patient ID: J91761
Attending Physician: Dr.Fred Baker; Specialty: Radiology
__________________________________________________

Billing Charges for ID: J91761
Pharmacy Charges: $4256.90
Room Charges:     $100.00
Doctor's Fees:    $180.00
______________________________
Total Charges:    $4536.90

*****************************************************************

Specialty is missing for Dr.
Error:  is invalid patientID for
Doctor Specialty is missing for patient
```

billType.adamBill;

```
*****************************************************************
Patient:
Patient ID: A93472
Attending Physician: Dr. ; Specialty:
__________________________________________________

Billing Charges for ID: A93472
Pharmacy Charges: $2356.75
Room Charges:     $425.75
Doctor's Fees:    $125.25
______________________________
Total Charges:    $2907.75

*****************************************************************
End of Patients Data
```

adamBill.printBill();
Patient and doctor data can not be set using billType obj and so no data.
But you can print billing details as you can set data using adamBill.setInfo(parameters)

```
######################################################################
Specialty is missing for Dr.Kevin
Specialty is missing for Dr.David
```

doctorType bad data
doctor speciality missing for kevin and david

```
######################################################################
Dr.Kevin Perez; Specialty:
Dr.David Jones; Specialty:
Dr.Joe Roberts; Specialty: Opthamology
Dr.Donald Wilson; Specialty: Pediatrics
Dr.Jose Clark; Specialty: Cardiac
Dr.Patrick Green; Specialty: Pathology
Dr.Eric Young; Specialty: Neurology
Dr.Walter Flores; Specialty: Cardiac
Dr.Fred Baker; Specialty: Radiology
Dr.Steven Taylor; Specialty: Radiology
Dr.Anthony Lee; Specialty: Anesthesiology

######################################################################
kishore@kishore-virtual-machine:/CS202_ast3_f22$
```