

Segundo examen parcial

Computacional II

Nombre alumno:

Resuelva los siguientes ejercicios. Por favor, tenga en cuenta todos los detalles de programación orientada a objetos (POO) discutidos durante el curso.

1. Clases y Herencia.

Cree una jerarquía de herencia que podría usar un banco para representar las cuentas bancarias de los clientes. Todos los clientes en este banco pueden depositar (es decir, abonar) dinero en sus cuentas, y retirar (es decir, cargar) dinero de ellas. También existen tipos más específicos de cuentas. Por ejemplo, las cuentas de ahorro obtienen intereses sobre el dinero que contienen. Por otro lado, las cuentas de cheques cobran una cuota por transacción (es decir, abono o cargo).

Cree una jerarquía de herencia que contenga la clase base "Cuenta", junto con las clases derivadas "CuentaAhorros" y "CuentaCheques" que hereden de la clase Cuenta. La clase base Cuenta debe incluir un miembro de datos de tipo double para representar el saldo de la cuenta. La clase debe proporcionar un constructor que reciba un saldo inicial y lo utilice para inicializar el miembro de datos. El constructor debe validar el saldo inicial, para asegurar que sea mayor o igual a 0.0 . De no ser así, el saldo debe establecerse en 0.0 y el constructor debe mostrar un mensaje de error, indicando que el saldo inicial es inválido. La clase debe proporcionar tres funciones miembro. La función miembro abonar debe sumar un monto al saldo actual. La función miembro cargar debe retirar dinero de la Cuenta y asegurar que el monto a cargar no exceda el saldo de la Cuenta . Si lo hace, el saldo debe permanecer sin cambio y la función debe imprimir el mensaje "El monto a cargar excedió el saldo de la cuenta." La función miembro getSaldo debe devolver el saldo actual.

La clase derivada CuentaAhorros debe heredar la funcionalidad de una Cuenta, pero también debe incluir un miembro de datos de tipo double que indique la tasa de interés (porcentaje) asignada a la Cuenta. El constructor de CuentaAhorros debe recibir el saldo inicial, así como un valor inicial para la tasa de interés de CuentaAhorros. CuentaAhorros debe proporcionar una función miembro pública llamada calcularInteres, que devuelva un valor double que indique el monto de interés obtenido por una cuenta. La función miembro calcularInteres debe determinar este monto, multiplicando la tasa de interés por el saldo de la cuenta. (Nota: CuentaAhorros debe heredar las funciones miembro abonar y cargar como están, sin redefinirlas).

La clase derivada CuentaCheques debe heredar de la clase base Cuenta e incluir un miembro de datos adicional de tipo double, que represente la cuota que se cobra por transacción. El constructor de CuentaCheques debe recibir el saldo inicial, así como un parámetro que indique el monto de la cuota. La clase CuentaCheques debe redefinir las funciones miembro abonar y cargar de manera que resten la cuota del saldo de la cuenta, cada vez que se realice una de esas transacciones con éxito. Las versiones de CuentaCheques de estas funciones deben invocar la versión de la clase base Cuenta para realizar las actualizaciones en el saldo de una cuenta. La función cargar de CuentaCheques debe cobrar una cuota sólo si realmente se retiró dinero (es decir, que el monto a cargar no exceda el saldo de la cuenta). (Sugerencia: defina la función cargar de Cuenta de manera que devuelva un valor bool que indique si se retiró dinero. Después use el valor de retorno para determinar si se debe cobrar una cuota).

Después de definir las clases en esta jerarquía, escriba un programa para crear objetos de cada clase y evaluar sus funciones miembro (se refiere a un "main"). Agregue interés al objeto CuentaAhorros , primero invocando a su función calcularInteres y después pasando el monto de interés devuelto a la función abonar del objeto.

2. En un alejado lugar la Secretaría de Turismo cuenta con datos almacenados dentro de un archivo que registra las camas de hotel ocupadas diariamente a nivel regional. El archivo representa los datos mediante la siguiente estructura de 3 campos:

`fecha;codigohotel;numerodecama`

El valor del campo fecha corresponde a la fecha (en formato aaaammdd) en que fue generado el registro. El valor del campo codigohotel corresponde a un identificador único para cada hotel de la región y el valor del campo numerodecama es la identificación de la misma. Al respecto, una cama es única en un hotel; si el mismo número se repite en dos hoteles distintos se trata, entonces, de dos camas diferentes.

Lo que se debe hacer:

- a) `leeProcesa(.....)`: Al iniciarse la ejecución del programa, este invoca a la función `leeProcesa` que leerá el contenido del archivo y almacenará su contenido completo en la memoria (de la forma que usted considere mas conveniente).
- b) `nuevaListaSinDuplicados(.....)`: Escribir la funcionalidad `nuevaListaSinDuplicados` que a partir de la información guardada antes genera una nueva lista sin repetidos, dejando sólo una ocurrencia de cada información presente en el archivo (A partir del literal "c", esta nueva información será con la que trabajaremos).
- c) `seleccionar(.....)`: Proveer la funcionalidad `seleccionar`, que a partir de los datos guardados en el contenedor del punto "b" y un código de un hotel pasado como parámetro, creará un nueva lista únicamente con los datos de las camas del hotel referido.
- d) `estadistico(.....)`: Proveer la funcionalidad `estadistico`, que dada la información guardada en el contenedor del punto "b" permitirá mostrar en pantalla la cantidad de camas registradas por cada hotel. A modo de ejemplo:
Hotel 20 tiene en total 2 cama(s) distintas
Hotel 15 tiene en total 1 cama(s) distintas
Hotel 16 tiene en total 6 cama(s) distintas
- e) `contar(.....)`: Proveer la funcionalidad `contar`, que dada la información guardada en el contenedor del punto "b" y una fecha (dia, mes, año) retorna la cantidad de camas ocupadas existentes de esa fecha.
- f) `eliminarHotel(.....)`: Proveer la funcionalidad `eliminarHotel`, que dada la información guardada en el contenedor del punto "b" y un identificador de hotel eliminará todas las ocurrencias asociadas a dicho hotel.
- g) Salir.

Proveer de un menú de diálogo para el usuario que le permita acceder a las funciones definidas en los puntos anteriores. Nota: Al seleccionar cada opción, debe mostrar la información respectiva (de nuevo, de la forma que usted considere mas conveniente).