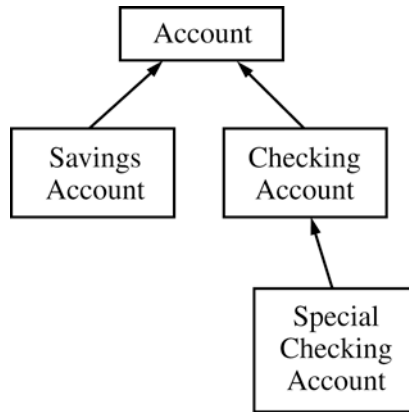


Note: The following question is somewhat longer than what may appear on the AP Computer Science Exams. In particular, a question of this type appearing on the AP Computer Science A Exam might be limited to two parts.

- *3. Consider the problem of modeling bank accounts. A diagram of the class hierarchy used to represent bank accounts is shown below.



The abstract class `Account` models a bank account with the following data and operations.

Data

- the identity number for the account (The identity number is never changed once the account has been constructed.)
- the balance in the account (The balance can change as a result of some operations.)

Operations

- create an account with a given identity number and initial balance
- return the identity number
- return the current balance
- deposit some positive amount into the account, increasing the balance
- decrease the balance by a specified positive amount; if the amount is greater than the balance, throw an `IllegalArgumentException`
- return the monthly interest due

An implementation for this class is shown below.

```
public abstract class Account
{
    private int idNum;        // identity number for this account
    private double balance;  // current balance for this account

    // precondition: startBal >= 0.0
    // postcondition: An Account with identity number idNumber
    //                and current balance of startBal has
    //                been created
    // exceptions: If startBal < 0.0, an
    //              IllegalArgumentException is thrown
    public Account(int idNumber, double startBal)
    { /* code not shown */ }

    // returns the identity number for this account
    public int idNumber()
    { /* code not shown */ }

    // returns the current balance for this account
    public double currentBalance()
    { /* code not shown */ }

    // precondition: amount >= 0.0
    // postcondition: the current balance of this account has
    //                been increasedby amount;
    // exceptions: if amount < 0.0, then current balance is
    //              unchanged and an IllegalArgumentException
    //              is thrown
    public void deposit(double amount)
    { /* code not shown */ }

    // precondition: 0.0 <= amount <= balance
    // postcondition: the current balance of this account has
    //                been decreased by amount;
    // exceptions: if amount < 0.0 or if amount > balance,
    //              then current balance is unchanged and an
    //              IllegalArgumentException is thrown
    public void decreaseBalance(double amount)
    { /* code not shown */ }

    // returns the monthly interest due for this account
    public abstract double monthlyInterest();
}
```

- (a) A savings account at a bank “is-a” bank account and is modeled by the class `SavingsAccount`. A savings account has all the characteristics of a bank account. In addition, a savings account has an interest rate, and the interest due each month is calculated from that interest rate. The operations for a savings account that differ from those specified in the class `Account` are the following.
- create a new savings account with a given annual interest rate, as well as the parameters required for all accounts
 - withdraw a positive amount that does not exceed the current balance, decreasing the balance by the amount withdrawn
 - calculate the monthly interest by multiplying the current balance by the annual interest rate divided by twelve

Write the complete definition of the class `SavingsAccount`, including the implementation of methods.

- (b) A checking account at a bank “is-a” bank account and is modeled by the class `CheckingAccount`. A checking account has all the characteristics of a bank account. In addition, a checking account can have checks written on it. Each check written decreases the account by the amount of the check plus a per-check charge. The operations for a checking account that differ from those specified in the class `Account` are the following.
- create a new checking account with a given per-check charge, as well as the parameters required for all accounts
 - clear a check for a given amount by decreasing the balance by the amount of the check plus the per-check charge
 - compute and return the monthly interest

A declaration of the class `CheckingAccount` is shown below.

```
public class CheckingAccount extends Account
{
    private double checkCharge;

    public CheckingAccount(int idNumber, double startBal,
                           double chkCharge)
    {
        super(idNumber, startBal);
        checkCharge = chkCharge;
    }

    public void clearCheck(double amount)
    {
        decreaseBalance(amount + checkCharge);
    }

    public double monthlyInterest()
    { /* code not shown */ }
}
```

A special checking account “is-a” checking account and is modeled by the class `SpecialCheckingAccount`. A special checking account has all the characteristics of a checking account. In addition, a special checking account has a minimum balance and an annual interest rate. When the balance is above the minimum balance, the per-check charge is not deducted from the balance when a check is cleared. Otherwise, a check is cleared just as it is for a checking account. In addition, when the balance is above the minimum balance when interest is calculated, interest due is calculated on the current balance. Otherwise, the interest due is the same as for a checking account. The operations for a special checking account that differ from those specified in the class `CheckingAccount` are the following.

- create a new special checking account with a given minimum balance and interest rate, as well as the parameters required for a checking account
- clear a check for a given amount according to the rules above
- calculate the monthly interest by multiplying current balance by the annual interest rate divided by twelve if the current balance is above the minimum; otherwise, calculate the interest as it is done for a checking account

Write the complete definition of the class `SpecialCheckingAccount`, including the implementation of its methods.

(c) Consider the class `Bank` partially specified below.

```
public class Bank
{
    private ArrayList<Account> accounts;
        // all accounts in this bank
        // accounts has no null entries

    // postcondition: for each account in this bank, the
    //                     monthly interest due has been deposited
    //                     into that account
    public void postMonthlyInterest()
    {
        // to be implemented in this part
    }

    // There may be fields, constructors and other methods
    //     not shown
}
```

Write the Bank method `postMonthlyInterest`, which is described as follows. For each account in this bank, `postMonthlyInterest` should calculate the monthly interest and deposit that amount into the account.

In writing `postMonthlyInterest`, you may use any of the public methods of class `Account` or its subclasses. Assume these methods work as specified. Solutions that reimplement functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method `postMonthlyInterest` below.

```
// postcondition: for each account in this bank, the monthly
//                 interest due has been deposited into
//                 that account
public void postMonthlyInterest()
```