Zadatak 1.

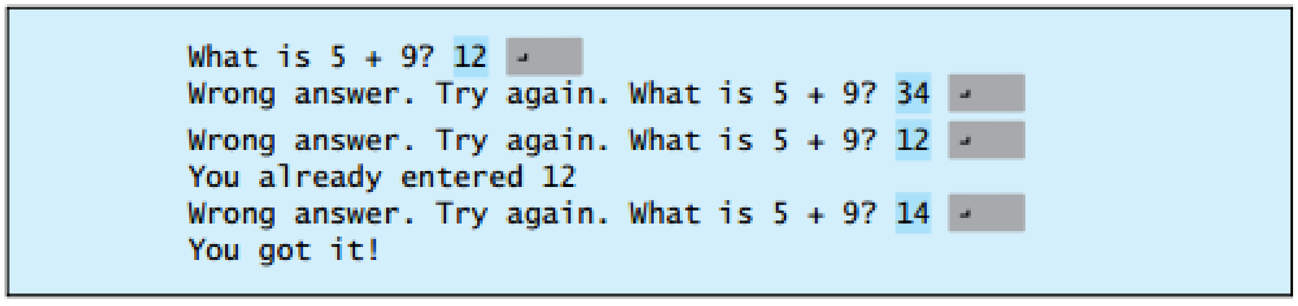
(*Sum* `ArrayList`) Write the following method that returns the sum of all numbers in an `ArrayList`:

```
public static double sum(ArrayList<Double> list)
```

Write a test program that prompts the user to enter 5 numbers, stores them in an array list, and displays their sum.

Zadatak 2.

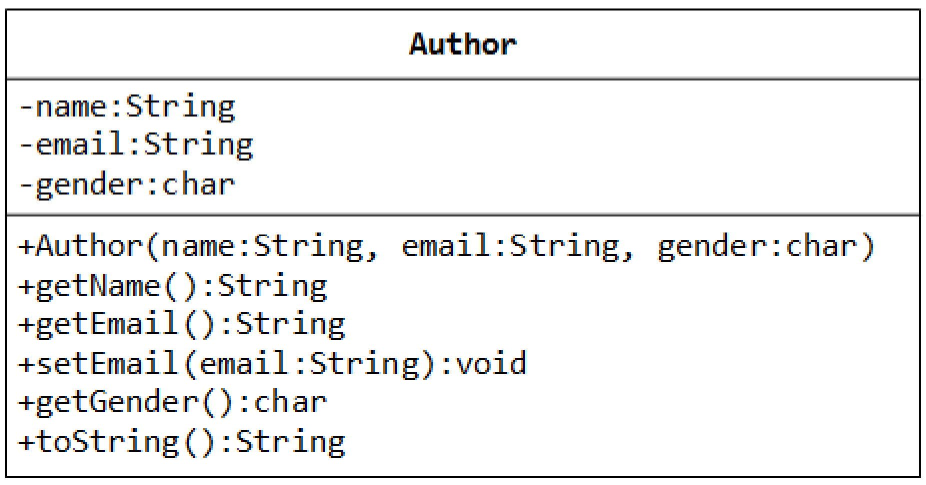
Napisati program koji pita korisnika koliki je zbir neka dva nasumicna broja, korisnik unosi svoj rezultat. Ako je unio pogresan rezultat, program mu kaze da pokusa ponovo sve dok ne unese tacan rezultat. Program takodje treba da upozori korisnika ako je neki rezultat vec unio. (*hint: koristiti ArrayList za smjestanje unesenih odgovora)*



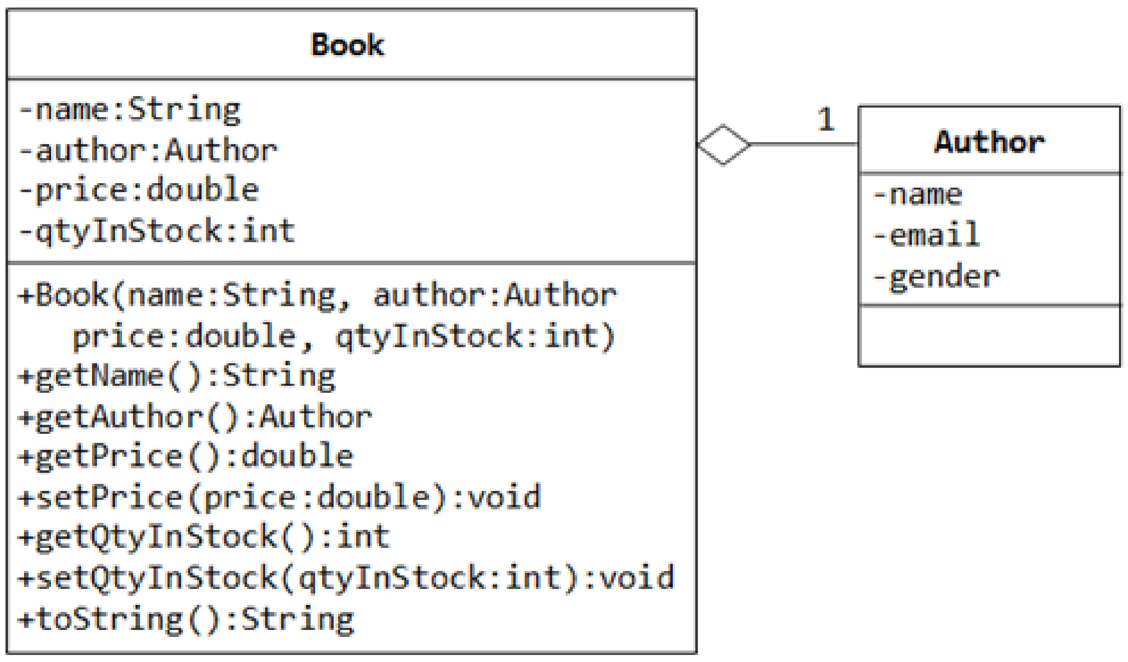
```
What is 5 + 9? 12
Wrong answer. Try again. What is 5 + 9? 34
Wrong answer. Try again. What is 5 + 9? 12
You already entered 12
Wrong answer. Try again. What is 5 + 9? 14
You got it!
```

Zadatak 3.

Napisati klase Author i Book, a zatim napisati testni program koji pravi instancu klase Author i instancu klase Book u kojoj ce data field author biti prethodno kreirana instanca Author klase, a zatim isprintati sve data field-e Book instance pozivanjem toString() metode.



```
Author
-name:String
-email:String
-gender:char
+Author(name:String, email:String, gender:char)
+getName():String
+getEmail():String
+setEmail(email:String):void
+getGender():char
+toString():String
```



```
Book
-name:String
-author:Author
-price:double
-qtyInStock:int
+Book(name:String, author:Author
   price:double, qtyInStock:int)
+getName():String
+getAuthor():Author
+getPrice():double
+setPrice(price:double):void
+getQtyInStock():int
+setQtyInStock(qtyInStock:int):void
+toString():String
```

Zadatak 4.

Napisati klasu Računar sa osobinama kolicinaRAMa, kapacitetHD, brzinaProcesora, dijagonalaMonitora, nabavnaCijena. Napisati metodu double izracunajCijenu() koja računa cijenu računara. Cijena računara je za 10% veća od nabavne cijene.

Napisati klasu Laptop koja nasleđuje klasu Računar. Ova klasa ima dodatnu osobinu trajanjeBaterije i cijena laptopa je za 15% veća od nabavne tako da se mora uraditi override metode izracunajCijenu().

RAM izraziti u gigabajtima, kapacitet hard diska takođe u gigabajtima, brzinu procesora u gigahercima, dijagonalu monitora u inčima, a trajanje baterije u satima.

Napisati testni program koji pravi instance Racunar i Laptop klase i ispisuje njihove cijene i konfiguraciju.

Zadatak 5.

(*New* `Account` *class*) An `Account` class was specified in Programming Exercise 9.7. Design a new `Account` class as follows:

- Add a new data field `name` of the `String` type to store the name of the customer.
- Add a new constructor that constructs an account with the specified name, id, and balance.
- Add a new data field named `transactions` whose type is `ArrayList` that stores the transaction for the accounts. Each transaction is an instance of the `Transaction` class. The `Transaction` class is defined as shown in Figure 11.6.

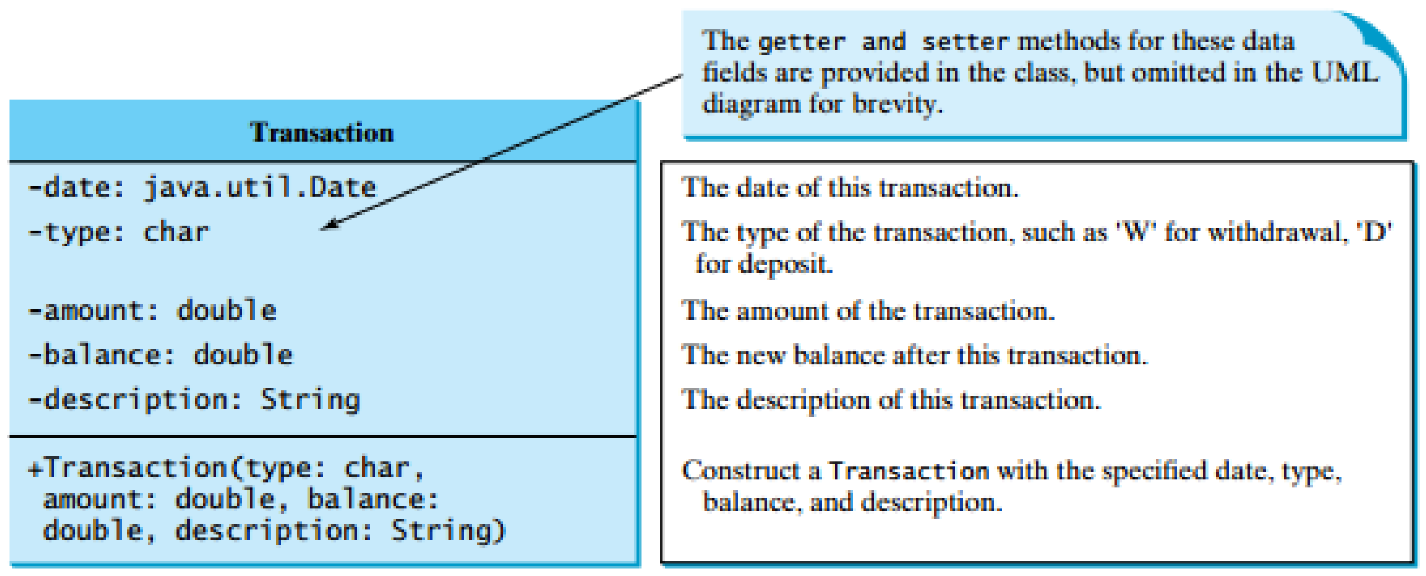

The getter and setter methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.
Transaction
-date: java.util.Date
-type: char
-amount: double
-balance: double
-description: String
+Transaction(type: char, amount: double, balance: double, description: String)
The date of this transaction.
The type of the transaction, such as 'W' for withdrawal, 'D' for deposit.
The amount of the transaction.
The new balance after this transaction.
The description of this transaction.
Construct a Transaction with the specified date, type, balance, and description.

FIGURE 11.6 The `Transaction` class describes a transaction for a bank account.

- Modify the `withdraw` and `deposit` methods to add a transaction to the `transactions` array list.
- All other properties and methods are the same as in Programming Exercise 9.7.

Write a test program that creates an `Account` with annual interest rate `1.5%`, balance `1000`, id `1122`, and name `George`. Deposit $30, $40, and $50 to the account and withdraw $5, $4, and $2 from the account. Print an account summary that shows account holder name, interest rate, balance, and all transactions.

Ako neko nije napisao staru Account klasu, evo sta treba da uradi:

(*The* `Account` *class*) Design a class named `Account` that contains:

- A private `int` data field named `id` for the account (default `0`).
- A private `double` data field named `balance` for the account (default `0`).
- A private `double` data field named `annualInterestRate` that stores the current interest rate (default `0`). Assume all accounts have the same interest rate.
- A private `Date` data field named `dateCreated` that stores the date when the account was created.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id and initial balance.
- The accessor and mutator methods for `id`, `balance`, and `annualInterestRate`.
- The accessor method for `dateCreated`.
- A method named `getMonthlyInterestRate()` that returns the monthly interest rate.
- A method named `getMonthlyInterest()` that returns the monthly interest.
- A method named `withdraw` that withdraws a specified amount from the account.
- A method named `deposit` that deposits a specified amount to the account.

Draw the UML diagram for the class and then implement the class. (*Hint*: The method `getMonthlyInterest()` is to return monthly interest, not the interest rate. Monthly interest is `balance * monthlyInterestRate`. `monthlyInterestRate` is `annualInterestRate / 12`. Note that `annualInterestRate` is a percentage, e.g., like 4.5%. You need to divide it by 100.)

Write a test program that creates an `Account` object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the `withdraw` method to withdraw $2,500, use the `deposit` method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.