1. Create an Author class and a Book class.
   Author should have: name, email, gender.
   It should have getters.
   It should have a toString method.

   Book should have: title, author(Author), price, quantity.
   It should have appropriate getters and setters.
   It should have a method: getProfit(), which calculates the profit from the book based on the price and quantity.
   It should have a toString method.

2. Create an Account class. Account should have: id, name, balance.
   It should have setters for name and balance, and getters for all fields.
   It should have a method: credit(amount), which should add amount to balance and return the updated balance.
   It should have a method: debit(amount), which should subtract the amount from the balance, if amount is less than the balance, otherwise output “Amount exceeded balance.”
   It should have a method: transferTo(anotherAccount, amount): which should subtract the amount from the account balance and add it to the given anotherAccount and return the updated balance, if amount is less than the balance, otherwise output “Amount exceeded balance.”
   It should have a static method: identifyAccounts(accountFirst, accountSecond) which gets two accounts and identifies if they are the same or not comparing all fields.
   It should have toString method.

3. Write classes: Person, Student.
   Person should have: firstName, lastName, gender, age.
   It should have appropriate getters and setters.
   It should have a method: toString().

   Student is inherited from Person. It should have program(array of { programName, grade }), year, fee.
   It should have appropriate getters and setters.
   It should have method: passExam(programName, grade). Student should contain the data about its programs and exams. passExam will update that data, so if student passed all the exams(grade is great or equal to 50), its year should be increased by one.
   It should have a toString method.

   [{programName: 'matem', grade: undefined}, { programName: 'angl', grade: undefined}])

4. Describe a model of a library. For that define classes: Library, Reader, Book.

5. To create correct hierarchies and connections, you should have a subclasses of Book such as LibraryBookBase, LibraryBook, ReaderBook.

   Book should have
   - fields
     - title - string
     - author - string
   - methods
     - getters for fields
     - toString()
     - isTheSameBook(book) - which returns true if the book title and author is the same with the current instance, false, otherwise.

   LibraryBookBase should have
   - fields
     - title - string
     - author - string
     - bookId - number
   - methods
     - getters for fields
     - toString()

   LibraryBook should have
   - fields
     - title - string
     - author - string
     - bookId - number
     - quantity - number
   - methods
     - getters for fields
     - setters for appropriate fields
     - toString()
     - increaseQuantityBy(amount - number) - increases the quantity of the book by the given amount.
     - decreaseQuantityBy(amount - number) - decrease the quantity of the book by the given amount.

   ReaderBook should have
   - fields
     - title - string
     - author - string

bookId - number
expirationDate - string
isReturned - boolean

methods
getters for fields
setters for appropriate fields
toString()

Reader should have

fields
firstName - string
lastName - string
readerId - number
books - Array of ReaderBook

methods
getters for fields
setters for appropriate fields
toString()
borrowBook(book - Book, library - Library) - requests a book from the library. If returned book is not a null and is a type of **ReaderBook**, pushes it to the books.

Library should have

fields
books - Array of LibraryBook
readers - Array of Readers

methods
getters for fields
doHaveBook(requestedBook - Book) - returns true if library has the book, false otherwise.
addBook(newBook - Book) - add new book to the library. If the book already exists, increases its quantity, otherwise adds new book of type **LibraryBook**.
addBooks(newBooks) - add new books to the library with the same logic as the addBook. Returns changed array of the books.
checkReaderId(readerId) - returns true if there exist a reader with the given id, otherwise returns false.
lendBook(book - Book, readeId) - checks whether the book exists and there is at least one at the library. Checks whether library has a reader with the given id. If the both are true, returns a book of type **ReaderBook**.

Հայերեն.

Նկարագրել գրադարանի մոդել, որպես հիմնական դասեր վերցնելով՝ Library, Reader, Book. Բոլոր հիմնական հարաբերությունները տվյալների և օբյեկտների միջև որ կունենաք, պետք է նկարագրվեն այս դասերի միջոցով: Սակայն Book դասի պարագայում նպատակահարմար է կատարել տրոհումներ և ստեղծել մի շարք ենթադասեր, որոնք ավելի կոնկռետ տվյալներ կպարունակեն ու ավելի ճիշտ կբնորոշեն գիրք օբյեկտը կախված գրադարանի(Library) կոնտեքստում ենք դիտարկում այն, թե կարդացողի(Reader):

Մոդելների նկարագրությունը(բոլոր դաշտերի կողքը նշված է տիպ, որը ինչպես կարող ա լինել պարզ՝ string, number, boolean, այնպես էլ բարդ՝ array կամ ասենք ձև որևէ դասի տիպ, օրինակ՝ Book)

Book դասը պետք է բաղկացած լինի
- դաշտեր
  - title - string
  - author - string
- մեթոդներ
  - getter-ներ բոլոր դաշտերի համար
  - toString()
  - isTheSameBook(book - Book) - որը համեմատում է արգումենտում փոխանցված գիրքը ընթացիկ(this) գրքի հետ title-ի և author-ի հիման վրա և վերադարձնում է true, եթե դրանք նույնն են, false՝ հակառակ դեպքում:

LibraryBookBase դասը պետք է բաղկացած լինի
- դաշտեր
  - title - string
  - author - string
  - bookId - number
- մեթոդներ
  - getter-ներ բոլոր դաշտերի համար
  - toString()

LibraryBook դասը պետք է բաղկացած լինի
- դաշտեր
  - title - string
  - author - string
  - bookId - number
  - quantity - number
- մեթոդներ
  - getter-ներ բոլոր դաշտերի համար

setter-ներ որոշ դաշտերի համար, որոնք փոփոխության ենթակա են
toString()
increaseQuantityBy(amount - number) - ավելացնում է գրքերի քանակը տրված չափով(amount):
decreaseQuantityBy(amount - number) - փոքրացնում է գրքերի քանակը տրված չափով(amount):

ReaderBook դասը պետք է բաղկացած լինի
դաշտեր
title - string
author - string
bookId - number
expirationDate - string (ցույց է տալիս գրքի վերադարձման ամսաթիվը)
isReturned - boolean (ցույց է տալիս արդյոք գիրքը վերադարձվել է թե ոչ)
մեթոդներ
getter-ներ բոլոր դաշտերի համար
setter-ներ որոշ դաշտերի համար, որոնք փոփոխության ենթակա են
toString()

Reader դասը պետք է բաղկացած լինի
դաշտեր
firstName - string
lastName - string
readerId - number
books - Array of ReaderBook - (բոլոր գրքերի(ReaderBook տիպի, այսինքն այդ դասի instance է) ցուցակը, որ ընթերցողը երբեք վերցրել է գրադարանից)
մեթոդներ
getter-ներ բոլոր դաշտերի համար
setter-ներ որոշ դաշտերի համար, որոնք փոփոխության ենթակա են
toString()
borrowBook(book - Book, library - Library) - Reader-ը հարցում է կատարում գրադարանից(Library) և գիրք(ReaderBook) է վերցնում: Այսինքն կոդում պետք է դիմում կատարվի գրադարանին որը կամ կվերադարձնի գիրքը որոշակի պայամններից ելնելով(տես Library դասի lendBook մեթոդը) կամ կվերադարձնի null.
Եթե գրադարանը գիրք(ReaderBook տիպի) է վերադարձրել և ոչ թե null, ապա գիրքը պիտ ավելացվի Reader-ի books զանգվածին:

Library դասը պետք է բաղկացած լինի
դաշտեր
books - Array of LibraryBook (գրադարանում գրանցված բոլոր գրքերի(LibraryBook տիպի, այսինքն այդ դասի instance է) ցուցակը)

readers - Array of Readers (գրադարանում գրանցված բոլոր ընթերցողների (Reader տիպի) ցուցակը)

մեթոդներ

getter-ներ բոլոր դաշտերի համար

doHaveBook(requestedBook - Book) - ստուգում է արդյոք գրադարանում առկա է այդ գիրքը, այսինքն առհասարակ կա նման գիրք ցուցակագրված և պահի դրությամբ գրքի քանակը զրո չի: Եթե կա, վերադարձնում է true, հակառակ դեպքում՝ false:

addBook(newBook - Book) - գարդարանի գրքերին ավելացնում է նոր գիրք: Նկատենք, որ Book տիպի գրքից պետք է ստեղծել LibraryBook տիպի օբյեկտ և նոր ավելացնել գրադարանի ցանկին: Սակայն եթե նման գիրք արդեն կա գրադարանում, ապա գրքի քանակը մեծացնում է մեկով:

addBooks(newBooks) - նոր գրքեր է ավելացնում գրքերի զանգվածին: Յուրաքանչյուր գիրքն ավելացնելու օգտվում ենք նախորդ մեթոդում նկարագրված տրամաբանությունից:

checkReaderId(readerId) - ստուգում է արդյոք գրադարանին դիմած ընթերցողը գրադարանում գրանցված է թե ոչ: Վերադարձնում է true, եթե Reader-ի readerId-ով ընթերցող կա readers ցուցակով, false հակառակ դեպքում:

lendBook(book - Book, readeId) - Ընթերցողին տալիս է հարցված գիրքը, եթե գիրքն առկա է գրադարանում և գրադարանին դիմած ընթերցողը գրանցված է այդ գրադարանում(հուշում պետք է օգտվել նախորդ մեթոդներից): Եթե նշված պայմանները տեղի ունեն, ապա այդ գիրքը վերադարձնում է, books ցուցակում քանակը մեկով կրճատելով: Նկատենք, որ գիրքը տալիս է ընթերցողին, հետևաբար Book տիպի book-ից օգտվելով իր ցուցակից գտնում է LibraryBook տիպի գիրքը, ու օգտվելով title, author ու bookId դաշտերից, ստեղծում է նոր օբյեկտ՝ ReaderBook տիպի, որտեղ isReturned-ի արժեքը ակնհայտորեն false պետք է լինի, իսկ expirationDate-ը յուրաքանչյուր գրադարան յուրովի է դնում(էս ինքներդ որոշեք ինչ արժեք(ներ) կուզեք տեղադրել):