به نام خدا

|  |  |  |  |
| --- | --- | --- | --- |
| مدرس: مظفر بگ محمدي | دانشگاه ايلام | ترم دوم سال تحصيلي 02-01 | سری هفتم تمرینات برنامه‌نویسی جاوا |

1. **Generic و استثناء:** پادشاه گوگولستان دختر زیبایی به نام فرنوش داشت که همه‌ی جوانان شهر آرزوی ازدواج با او را در سر می‌پروراندند. روزها یکی پس از دیگری سپری می‌شد و عده‌ی هواخواهان فرنوش هر روز بیش‏تر و بیشتر می‌شد و تصمیم‌گیری برای او سخت‏تر و سخت‌تر. تا این‌که ماجرایی رخ داد که سبب آشنایی فرنوش را با عده‏ای فراهم کرد که می‌توانستند مشکل او را حل‌وفصل نمایند. خوشبختانه او توانسته بود شما را که همواره مشتاق همیاری و کمک در انجام امور خیر هستید، پیدا کند! پس بیایید برای خوشبخت شدن هرچه بیشتر فرنوش بکوشید و دستان پرتوان خود را به‌کارگیرید تا شاید روزی کسی این کار نیک را در حق شما جبران نماید چنان‌که گفته‌اند: "تو نیکی کن و در دجله انداز که ایزد در بیابانت دهد باز". برای آن‌که در این امر ما نیز سهمی داشته باشید و از برکات این امر خیر ذره‌ای چشیده باشیم چگونگی این کمک را تا حدی شرح می‌دهیم، "باشد که مقبول افتد ": در گام اول اطلاعات خواستگاران فرنوش را در یک داده ساختار ذخیره کنید. مطابق علایق شخصی فرنوش، این داده ساختار یک "درخت جست‌وجوی دودویی" است. از آنجا که معیارهای فرنوش در انتخاب همسرش برای ما به‌طور دقیق مشخص نیست این داده ساختار باید به‌صورت Generic تعریف شود. برای مثال اگر قلی خواستگار فرنوش باشد می‌تواند ویژگی‌هایی همچون قد، وزن، رنگ چشم و ... داشته باشد و برنامه شما باید آن‌ها را تا حد خوبی پشتیبانی نماید. نام داده ساختار و تعریف آن این‌گونه است:
2. Public class BST < T extends Comparable > {}

عملیاتی که این کلاس باید بتواند انجام دهد به‌صورت توابعی به همراه تعریف هریک، در زیر آورده شده است:

* این متد تعداد عناصر موجود در داده ساختار را نمایش می‌دهد:

1. public int size();

* باید بتوان عنصری جدید را به این داده ساختار افزود:

1. Public void add(T element);

* این متد عنصری از جنس T گرفته و اگر آن عنصر داخل داده ساختار موجود بود true و در غیر این صورت false برمی‌گرداند:

1. Public T containts(T element);

* کوچک‌ترین عنصر موجود در داده ساختار را برمی‌گرداند (در درخت جستجوی دودویی این عنصر همان چپ‌ترین برگ است):

1. Public T getMinElement() throws IllegalStateException;

* بزرگ‌ترین عنصر موجود در درخت که همان راست‌ترین برگ را برمی‌گرداند:

1. Public T getMaxElement() throws IllegalStateException;

در پیاده‌سازی این داده ساختار به نکات زیر توجه کنید:

* در توابع getMaxElement و getMinElement اگر هیچ عنصری وجود نداشته باشد باید یک استثنا از نوع IllegalStateException پرتاب شود.
* پیاده‌سازی بدنه‌ی متدها اختیاری بوده و بر عهده‌ی خودتان است.
* برای آن‌که از درستی تست‌های نوشته‌شده مطمئن شوید باید تعدادی تست در قالب Unit Test (حداقل 2 تست برای هر متد) طراحی کنید. سعی کنید تست‌های جامع و کاملی طراحی کنید چراکه در نمره دهی شما مؤثر است.

1. **واسط و استثنا**: در این تمرین شما باید یک فروشگاه مجازی کتاب درست کنید. فروشگاه شما دارای لیستی از مشتریان و لیستی از کتابهای موجود است. هر مشتری دارای یک id یکتا، کارت اعتباری، اسم، آدرس و کلمه ی عبور است. آدرس شامل اسم خیابان ، شهر، استان و کد پستی است. کارت اعتباری دارای سقف اعتبار برابر 500000 تومان می باشد. با توجه به واسطهای زیر کلاسهای فروشگاه، مشتری، آدرس و کارت اعتباری خرید را پیاده کنید. از واسطها و کلاسهای زیر استفاده کنید.

دقت کنید که در هنگام فراخوانی متد buy اگر موجودی کتاب کافی نباشد، باید استثناء NotEnoughBook تولید شود. اگر موجودی کارت اعتباری مشتری کافی نباشد، باید استثناء NotEnoughMoney تولید شود.

interface Store {

public void register(Customer customer);

public void remove(Customer customer);

public Customer getCustomer(int id);

}

//----

interface BookStore extends Store {

// invoke this method when a customer tries to buy books.

public boolean buy(Customer customer, Book book, int amount);

// The BookStore tries to get more books by this methods.

public void importBooks(Book book, int amount);

// get all the books with the specified title from the book list

// we assume no repeated book title in the book list.

public Book getBooks(String title);

// get the amount with the specified book title.

public int getBookAmount(String title);

// get all the books in the book list.

public Book[] getAllBooks();

}

//-------

class MyStore implements BookStore {

private MyCustomer [] customers;

private MyBook [] books;

//how many books are available for each title

private int [] numberOfBook;

}

//-----

interface Book {

// get the title of this book.

public String getTitle();

// get the date of publishing this book.

public java.util.Date getDate();

// get the edition of this book.

public int getEdition();

// get the brief description of this book.

public String getDescription();

// get the price of this book.

public float getPrice();

}

//----

public class MyBook implements Book{

private String title;

private java.util.Date date;

private int edition;

private String description;

private float price;

public MyBook (String title, java.util.Date date, int edition, String desc, float price ) {

// a constructor to create a new MyBook.

}

}