# مدیریت و مرتب‌سازی دانشجویان

در این تمرین، می‌خواهیم با مفاهیم پایه‌ای **کلاس‌ها (Classes)**، **سازنده‌ها (Constructors)** و **مرتب‌سازی داده‌ها (Sorting)** آشنا شویم.

کلاسی به نام **Student** طراحی کنید که ویژگی‌های زیر را داشته باشد:

- `name` (نام دانشجو)
- `id` (شماره دانشجویی)
- `average` (معدل)

سپس مراحل زیر را انجام دهید:

۱. یک **سازنده (constructor)** برای مقداردهی اولیهٔ ویژگی‌ها بنویسید.
۲. متدی به نام **showInfo()** بنویسید که مشخصات دانشجو را چاپ کند.
۳. در تابع `main()` ، برنامه به کاربر اجازه دهد تا اطلاعات **حداقل ۵ دانشجو** را از طریق **ترمینال** وارد کند.
۴. لیست دانشجویان را بر اساس **معدل (از بیشترین به کمترین)** مرتب کرده و چاپ کند.
۵. سپس همان لیست را بر اساس **نام (حروف الفبا)** مرتب کرده و دوباره چاپ کند.

> **نکته:** از تابع آماده‌ی `sort()` استفاده نکنید. خودتان الگوریتم مرتب‌سازی (مثل Bubble Sort یا Selection Sort) را پیاده‌سازی کنید.

در جدول زیر نمونه‌ای از ورودی و خروجی‌های برنامه آورده شده است:

**ورودی:**

```
Enter number of students: 5
Student 1:
Name: Sara
ID: 401
Average: 18.2
Student 2:
Name: Amir
ID: 402
```

```
Average: 19.4
Student 3:
Name: Reza
ID: 403
Average: 17.0
Student 4:
Name: Niloofar
ID: 404
Average: 18.9
Student 5:
Name: Babak
ID: 405
Average: 15.7
```

**خروجی:**

```
Sorted by Average (High to Low):
Name: Amir, ID: 402, Average: 19.4
Name: Niloofar, ID: 404, Average: 18.9
Name: Sara, ID: 401, Average: 18.2
Name: Reza, ID: 403, Average: 17.0
Name: Babak, ID: 405, Average: 15.7
Sorted by Name (Alphabetical):
Name: Amir, ID: 402, Average: 19.4
Name: Babak, ID: 405, Average: 15.7
Name: Niloofar, ID: 404, Average: 18.9
Name: Reza, ID: 403, Average: 17.0
Name: Sara, ID: 401, Average: 18.2
```

# اپ بانکی

## صورت سؤال:

یک برنامه طراحی کنید که ساختار ساده‌ای از یک اپ بانکی را پیاده‌سازی کند.

## توضیحات بیشتر:

کلاس پایه به نام Account ایجاد کنید که ویژگی‌ها و متدهای عمومی که برای انواع حساب‌ها نیاز است را شامل می‌شود:

ویژگی‌های accountNumber (شماره حساب) و balance (موجودی حساب) را داشته باشد.

متدهای deposit(double amount) (برای واریز پول) و withdraw(double amount) (برای برداشت پول) را داشته باشد.

در این کلاس، تابع withdraw باید چک کند که موجودی کافی برای برداشت وجود دارد یا خیر.

سپس، از کلاس Account دو کلاس مشتق‌شده بسازید:

کلاس SavingAccount (حساب پس‌انداز):

ویژگی اضافهٔ interestRate (نرخ بهره).

متدی به نام addInterest() که به موجودی حساب بهره را اضافه کند.

کلاس CheckingAccount (حساب جاری):

ویژگی اضافهٔ transactionFee (کارمزد تراکنش).

در متد withdraw، به ازای هر تراکنش برداشت، مقدار transactionFee از موجودی کسر شود.

سپس در تابع main():

یک حساب پس‌انداز (SavingAccount) بسازید و عملیات واریز و برداشت را انجام دهید.

یک حساب جاری (CheckingAccount) بسازید و عملیات واریز، برداشت و کسر کارمزد را انجام دهید.

نتیجهٔ هر عملیات (موجودی حساب) را چاپ کنید.

## ورودی‌ها و خروجی‌ها:

### ورودی‌ها:

برای هر حساب، ابتدا accountNumber و balance باید به‌طور دستی تعیین شوند.

برای حساب پس‌انداز، نرخ بهره (interest rate) و برای حساب جاری، کارمزد تراکنش (transaction fee) نیز باید داده شوند.

عملیات‌های واریز (deposit) و برداشت (withdraw) را برای هر حساب انجام دهید.

### خروجی‌ها:

پس از انجام هر عملیات، موجودی حساب باید چاپ شود.

همچنین برای حساب پس‌انداز، پس از اضافه‌کردن بهره، موجودی جدید چاپ شود.

## برای فهم بهتر ورودی و خروجی‌ها تست‌کیس‌ها را مطالعه کنید:

### ورودی ۱:

```
Account Type: SavingAccount
Deposit: 1000
Withdraw: 200
Interest Rate: 5%
```

### خروجی ۱:

```
After deposit: 1000.0
After withdrawal: 800.0
After interest: 840.0
```

توضیح: هزار تا واریزی داشتیم و دویست تا برداشت پس الان موجودی هشتصد هست.در نهایت پنج درصد موجودی حال حاضر که هشتصد است سود دریافت میکنیم که میشه چهل تا و با موجودی حال جمع میشه

## ورودی ۲:

```
Account Type: CheckingAccount
Deposit: 500
Withdraw: 100
Transaction Fee: 2
```

## خروجی ۲:

```
After deposit: 500.0
After withdrawal with fee: 398.0
```

توضیح: پونصد تا واریزی داشتیم و صد تا برداشت و به ازای برداشت در این نوع حساب دو تا باید کارمزد پرداخت کنیم که کم میشه از موجودی

# Regex Challenge

برنامه‌ای بنویسید که ورودی شامل چند رشته (نام کاربری یا جمله) از کاربر دریافت کند
و بررسی کند کدام‌یک با الگوی مشخص مطابقت دارد.

کاربر ابتدا عدد `n` را وارد می‌کند (تعداد رشته‌ها).
سپس `n` رشته وارد می‌کند.

رشته‌هایی که شامل **آدرس ایمیل معتبر** هستند (مثلاً `user@example.com` ) باید شناسایی و چاپ شوند.

از **عبارات باقاعده (Regular Expressions / RegExp)** برای تشخیص الگو استفاده کنید.

## ورودی

ورودی شامل چند خط است:

- در خط اول، عددی طبیعی `n` (تعداد رشته‌ها) وارد می‌شود.
- در `n` خط بعدی، هر خط شامل یک رشته (نام کاربری یا جمله) است.

## خروجی

برنامه باید تمام رشته‌هایی را چاپ کند که شامل آدرس ایمیل معتبر هستند.
در صورت یافتن چند ایمیل، هرکدام باید در یک خط جداگانه چاپ شود.

## مثال

### ورودی نمونه

```
5
ali@gmail.com
test123
```

```
hello@world.net
dart@flutter
parisa@yahoo.com
```

## خروجی نمونه

```
Valid emails found:
ali@gmail.com
hello@world.net
parisa@yahoo.com
```