طراحی پشته

مشاهده‌ی اسلایدهای شش و هفت و هشت برای این تمرین الزامی است.

درمورد ساختمان داده پشته (stack) تحقیق کنید و کلاس پشته را با توجه به توابع زیر پیاده سازی کنید.

تضمین می‌شود که بیش‌تر از **صد عضو** به پشته اضافه نشود. **استفاده از کلاس Stack جاوا مجاز نیست.**

```
1 | public boolean empty()
```

اگر پشته خالی باشد، مقدار true بر می‌گرداند.

```
1 | public int size()
```

این متد اندازه‌ی پشته را بر می‌گرداند.

```
1 | public Object push(Object object)
```

این متد عنصر object را به پشته اضافه می‌کند و همان را برمی‌گرداند.

```
1 | public Object pop()
```

این متد عنصر سر پشته را برمی‌گرداند و آن را حذف می‌کند. اگر پشته خالی بود، رشته "error" را برمی‌گرداند.

```
1 | public Object peek()
```

این متد عنصر سر پشته را بدون حذف‌کردن برمی‌گرداند. اگر پشته خالی بود، رشته "error" را برمی‌گرداند.

```
1 | public int search(Object object)
```

به دنبال عنصر object می‌گردد. اگر آن‌را یافت اندیسش را برمی‌گرداند و در غیر این صورت ۱- را برمی‌گرداند. اندیس را برای اولین عضو اضافه شده صفر، برای دومین عضو اضافه شده یک و به همین ترتیب برای آخرین عضو اضافه شده به اندازه یکی کمتر از اندازه پشته در نظر بگیرید.

```
1 | public void clear()
```

پشته را به صورت کامل خالی می‌کند.

```
1 | public String toString()
```

عناصر پشته را به شکل رشته و به فرمی که در مثال‌ها مشاهده خواهید کرد، برمی‌گرداند.

```
public void pushAll(Object... objects)
```

به تعداد دلخواه کاربر عنصر می‌گیرد و آنها را به **ترتیب** به انتهای لیست اضافه می‌کند.

موضوع varargs را در اسلاید هشت بخوانید.

مثال

```
SimpleStack stack = new SimpleStack();
System.out.println(stack.empty()); // true
stack.push(1);
System.out.println(stack.toString()); // {1}
stack.pushAll(2, 3, 4, 5);
System.out.println(stack.toString()); // {1, 2, 3, 4, 5}
System.out.println(stack.peek()); //5
System.out.println(stack.pop()); //5
System.out.println(stack.pop()); //4
System.out.println(stack.toString()); // {1, 2, 3}
System.out.println(stack.empty()); // false
System.out.println(stack.size()); // 3
System.out.println(stack.search(3)); // 2
System.out.println(stack.search(5)); // -1

```

```
stack.clear();
System.out.println(stack.toString()); // {}
System.out.println(stack.size()); // 0
System.out.println(stack.empty()); // true
System.out.println(stack.pop()); // error
System.out.println(stack.peek()); // error
```

آن چه که باید آپلود کنید

آن چه که باید آپلود کنید، یک فایل zip است که وقتی آن را باز می‌کنیم، در آن **فقط** فایل SimpleStack.java را می‌بینیم.

بیگ‌نام

مشاهده‌ی اسلایدهای شش و هفت و هشت برای این تمرین الزامی است.

در این سوال باید کلاسی برای کار با اعداد خیلی بزرگ (بیش از هزار رقم!!!) طراحی کنید.

برای طراحی کلاس، به نکات زیر توجه نمایید:

- تعریف توابع در مکان لازم، نام‌گذاری صحیح، کامنت‌گذاری در مکان لازم، استفاده درست از کانستراکتورها و تمامی اصول طراحی درست از معیارهای نمره‌دهی این سوال هستند. پس لطفا به طراحی خود دقت کنید.
- گذاشتن `javadoc` برای متدها و کلاس‌ها نمره‌ی امتیازی دارد. دقت کنید که کد شما باید **فقط و فقط در جاهایی که لازم است** کامنت داشته باشد و زیادی کامنت در کد شما لزوما به معنی بهتر بودن آن نیست.
- طراحی کلاس شما باید به صورت **immutable** باشد. یعنی شی دریافت شده هیچ تغیری نمی‌کند و درصورت انجام عملیات یک شی جدید ساخته و برگردانده می‌شود.
- استفاده از کلاس‌های BigInteger و BigDecimal و موارد مشابه ممنوع است.
- پیاده‌سازی قسمت **reminder** امتیازی می‌باشد.

توجه کنید که یک طراحی ثابت وجود ندارد و ممکن است کدهای متفاوت طراحی‌های درستی داشته باشند.

کلاس BigNum شما باید حداقل توابع زیر را پیاده‌سازی کرده باشد و استفاده از توابع کمکی دیگر در کلاس، مشکلی ندارد.

```
public BigNum(String value) // Constructor (create BigNum form String)
public BigNum add(BigNum value) // return this + value
public BigNum subtract(BigNum value) // return this - value
public BigNum multiply(BigNum value) // return this * value
public BigNum remainder(BigNum value) // return this % value (value & this > 0)
```

مثال

به ازای اجرای main زیر باید خروجی موردانتظار تولید شود.

```
public static void main(String[] args) {
    BigNum bigNum1 = new BigNum("123");
    BigNum bigNum2 = new BigNum("456");
    System.out.println(bigNum1.add(bigNum2));
    System.out.println(bigNum1.multiply(bigNum2));
    System.out.println(bigNum1.subtract(bigNum2));
    System.out.println(bigNum2.remainder(bigNum1));
    System.out.println(bigNum1);
    System.out.println(bigNum2);
    System.out.println();

    bigNum1 = new BigNum("+1234567");
    bigNum2 = new BigNum("-987654");
    System.out.println(bigNum1.add(bigNum2));
    System.out.println(bigNum1.multiply(bigNum2));

```

```
        System.out.println(bigNum1.subtract(bigNum2));
        System.out.println(bigNum1);
        System.out.println(bigNum2);
        System.out.println();

        bigNum1 = new BigNum("-123");
        bigNum2 = new BigNum("123");
        System.out.println(bigNum1.add(bigNum2));
    }
```

خروجی

```
579
56088
-333
87
123
456

246913
-1219325035818
2222221
1234567
-987654

0
```

آن چه که باید آپلود کنید

تنها فایلی که باید آپلود کنید یک فایل zip است که وقتی آن را باز می‌کنیم، در آن تنها یک فایل BigNum.java باشد.

مدیریت کارخانه

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

مشاهده‌ی اسلاید شش برای این تمرین الزامی است.

مدیر یک کارخانه به شیوه‌ای که در ادامه توضیح داده می‌شود، کارخانه خود را اداره می‌کند. برنامه‌ای طراحی کنید که به او اجازه دهد صرفا با نوشتن دستوراتش، این کار را انجام دهد.

کارمندان در کارخانه‌ی معرفی‌شده، به چهار درجه با حقوق ماهانه‌ی زیر ($salary$) تقسیم می‌شوند:

1.درجه‌ی *Worker*: حقوق ۱۰۰ تومان

2.درجه‌ی *Foreman*: حقوق ۳۰۰ تومان

3.درجه‌ی *Supervisor*: حقوق ۷۰۰ تومان

4.درجه‌ی *Leader*: حقوق ۹۰۰ تومان

```
hire [name] [degree]
```

مدیر با این دستور کارمندی را با نام name و درجه‌ی degree به استخدام شرکت در می‌آورد. با استخدام هر کارمند، برای او حسابی ($credit$) باز می‌شود که حقوق‌های

دریافتی او در آن قرار می‌گیرد.

به نکات زیر توجه نمایید:

- حساب کارمندان در ابتدا خالی بوده و هیچ پولی در آن قرار ندارد.
- نام کارمندان **یکتا و با حروف کوچک** است.
- تضمین می‌شود که کارمندی با نام تکراری استخدام نمی‌شود.
- مدیر در مدت ریاست خود، حداکثر **سیصد** نفر را استخدام می‌کند.

```
pay [name]
```

زمان پرداخت حقوق هر یک از کارمندان بستگی به تصمیم مدیر کارخانه دارد. یعنی در یک زمان مشخص ممکن است یک کارمند سه بار حقوق دریافت کند درحالی که کارمند دیگری هیچ حقوقی دریافت نکرده باشد.

مدیر باید امکان پرداخت حقوق به یک کارمند را با دستور بالا داشته باشد. در نتیجه‌ی این دستور، موجودی حساب فردی به نام name، با حقوق ماهانه‌ی او جمع می‌شود.

```
get [name] [quantity]
```

افراد کارخانه می‌توانند از پول حساب خود استفاده کنند و این کار را با دستور بالا انجام می‌شود. در این صورت به میزان quantity از موجودی او کم می‌شود. اگر موجودی او کافی نباشد، پیام NotEnoughMoney چاپ می‌شود.

`special [name]`

یک کارمند در صورتی که اقدام شایسته‌ای انجام دهد به کارمند **ویژه** تبدیل می‌شود. این مزیت براساس تصمیم مدیر است و با دستور بالا انجام می‌شود.

`loan [name]`

کارمندان می‌توانند درخواست وام دهند. درخواست درصورتی پذیرفته می‌شود که تا به حال وام نگرفته باشند و جزو کارمندان ویژه باشند. اگر درخواست پذیرفته شود، به اندازه‌ی **سه** برابر حقوق ماهانه‌اش به حساب او اضافه می‌شود و پیام accepted چاپ می‌شود، در غیراین صورت پیام rejected چاپ می‌شود.

`promote`

هر زمانی که مدیر تصمیم بگیرد، تمام کسانی که شرط $credit > 2 \times salary$ را داشته باشند، ترفیع می‌گیرند. مثلا Workerها تبدیل به Foreman می‌شود و Leaderها بازنشسته (Retired) می‌شوند. به این عملیات promote گفته می‌شود و با دستور بالا قابل انجام است.
حساب فرد بازنشسته محفوظ مانده و می‌تواند از آن برداشت نماید. همچنین اگر حداقل یک **Leader ویژه** در عملیات promote بازنشسته شود، در کارخانه جشنی برگذار می‌شود و همه کارمندان **ویژه** که قبلا وام گرفته‌اند، بعد از آن می‌توانند دوباره وام بگیرند.

`regress`

هر زمانی که مدیر تصمیم بگیرد، تمام کسانی که شرط $credit < \lfloor \frac{salary}{2} \rfloor$ را داشته باشند، تنزل مقام می‌گیرند. مثلا Foremanها تبدیل به Worker می‌شود و Workerها اخراج (Fired) می‌شوند. به این عملیات regress گفته می‌شود و با دستور بالا قابل انجام است.

حساب فرد اخراج شده محفوظ مانده و می‌تواند از آن برداشت نماید.

```
report [degree]
```

هر زمان که مدیر گزارشی از کارمندانی با یک درجه‌ی خاص بخواهد، دستور report را وارد می‌کند. گزارش شامل افراد اخراج‌شده (Fired) و بازنشسته (Retired) هم می‌شود. کارمندان ویژه (حتی کارمندان سابق) با لفظ special که قبل از نامشان می‌آید شناخته می‌شوند.
خروجی دستور مطابق زیر است:

```
[name]([degree]) [credit]
```

ورودی

ورودی شامل دستورات دلخواه مدیر است. تا وقتی که دستوری وارد شود، برنامه ادامه دارد.

خروجی

خروجی شامل مواردی است که در هر دستور توضیح داده شد.

مثال

ورودی نمونه ۱

```
hire ali Worker
hire sara Foreman
hire bahar Supervisor
hire taghi Leader
pay ali
pay taghi
pay ali
pay taghi
pay ali
pay taghi
pay ali
pay taghi
promote
report Worker
report Foreman
report Supervisor
report Leader
loan sara
special sara
report Worker
report Foreman
report Supervisor
report Leader
loan sara
loan sara
pay bahar
pay bahar
```

```
pay bahar
promote
special bahar
report Worker
report Foreman
report Supervisor
report Leader
pay bahar
pay bahar
pay bahar
promote
report Worker
report Foreman
report Supervisor
report Leader
loan sara
hire mina Worker
report Worker
report Foreman
report Supervisor
report Leader
get mina 200
get ali 200
report Worker
report Foreman
report Supervisor
report Leader
```

```
regress
report Worker
report Foreman
report Supervisor
report Leader
```

خروجی نمونه ۱

```
ali(Foreman) 400
sara(Foreman) 0
bahar(Supervisor) 0
rejected
ali(Foreman) 400
special sara(Foreman) 0
bahar(Supervisor) 0
accepted
rejected
ali(Foreman) 400
special sara(Supervisor) 900
special bahar(Leader) 2100
ali(Foreman) 400
special sara(Supervisor) 900
accepted
mina(Worker) 0
ali(Foreman) 400
special sara(Supervisor) 3000
```

```
NotEnoughMoney
mina(Worker) 0
ali(Foreman) 200
special sara(Supervisor) 3000
ali(Foreman) 200
special sara(Supervisor) 3000
```

دنباله جادویی (امتیازی)(C++)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

فرض کنید دنباله‌ی $a_1a_2a_3\dots a_n$ را داشته باشیم.

زیردنباله $[i,j]$ $(1 \le i \le j \le n)$ به صورت $a_ia_{i+1}a_{i+2}\dots a_j$ تعریف می‌شود. برای یک زیردنباله، جمع آن، جمع تمام ارقام آن دنباله تعریف می‌شود.

دنباله‌ی $a_1a_2a_3\dots a_n$ را دنباله‌ی جادویی گویند اگر بتوان آن را به **دو** زیردنباله یا بیش‌تر با جمع مساوی تقسیم نمود. برای مثال دنباله‌ی 350178 یک دنباله‌ی جادویی است زیرا می‌تواند به سه زیردنباله‌ی 350 و 17 و 8 تقسیم شود که:

$$350 : 3+5+0=8$$

$$17 : 1+7=8$$

$$8 : 8=8$$

دقت کنید که زیر دنباله‌ها دارای اشتراک نیستند.

ورودی

ابتدا عدد n که تعداد رقم‌های دنباله است، داده می‌شود. در خط بعدی یک دنباله‌ی n تایی از ارقام 0 تا 9 داده می‌شود.

$$2 \leq n \leq 100$$

$$0 \leq a_i \leq 9$$

خروجی

اگر دنباله، یک دنباله‌ی جادویی است Yes وگرنه No چاپ کنید.

مثال

ورودی نمونه ۱

```
5
73452
```

خروجی نمونه ۱

```
Yes
```

ورودی نمونه ۲

```
4
```

1248

خروجی نمونه ۲

No