تمرین شماره ۸

مکتب ۹

بسم الله الرحمن الرحیم

- برای هر کدام از سوالات، در صورت نیاز از خروجی دستورات عکس بگیرید و در کنار پاسخ خود قرار دهید.
- برای تمرین‌های برنامه نویسی فایل‌های java. خود را بصورت یک پوشه در کنار فایل‌های دیگر قرار دهید.

پیاده‌سازی داده‌ساختارهای مختلف

هدف این تمرین شبیه‌سازی رفتار داده‌ساختارهای مختلف است. متن تمرین در ادامه فایل آمده است.

نکات

- مهلت ارسال تمرین تا ساعت ۲۴ یک‌شنبه ۹۶/۱۰/۱۰ می‌باشد. پاسخ تمرین را در قسمت مربوطه در سامانه http://maktabsharif.ir/sama/ قرار دهید.
- به ازای هر روز تأخیر ۵ درصد نمره کسر خواهد شد.
- در صورت لزوم یک فایل word به عنوان توضیح در کنار کدهای خود قرار دهید.
- نام فایل خود را به این صورت قرار دهید. Name_hw8_maktab9 به عنوان مثال Mohammad_Ali_Kargar_hw8_maktab9.
- در صورتیکه تمرین شامل چند فایل و فولدر می‌باشد حتماً آن‌ها را قالب یک فایل zip ارسال کنید.
- کدهای همه تمرین‌ها باید روی github نگهداری شود.
- تمیزی فایل گزارش و فایل‌های برنامه نویسی نیز بخشی از نمره را تشکیل می‌دهد.
- در صورتیکه سوالی دارید در گروه تلگرام بپرسید.

در این سوال قرار است classها و interfaceهای زیر را پیاده سازی کنید:

```
public interface Collection<E>;
```

داده ساختارهای بعدی این interface را implement میکنند.که دارای متدهای زیر است:

```
public int size();
```

این متد تعداد عناصر داخل این داده ساختار را بر می‌گرداند.

```
public void clear();
```

این متد داده ساختار را خالی می‌کند. یعنی بعد از صدا زدن این متد، متد size مقدار ۰ برمی‌گرداند.

```
public boolean isEmpty();
```

خروجی این متد در صورتی true است که size صفر باشد در غیر این صورت false است.

```
public boolean contains(E var);
```

این متد یک عنصر از جنس E می‌گیرد و اگر آن عنصر در داده ساختار موجود بود مقدار true بر می‌گرداند.

```
public class Stack<E> implements Collection<E>
```

این class داده ساختار پشته را شبیه سازی می‌کند و دارای متدهای زیر است:

```
public Stack(int maxSize);
```

این متد constructor این داده ساختار است و به عنوان ورودی maxSize را دریافت می‌کند. آرایه‌ای به طول maxSize می‌سازد و با نگاه داشتن یک متغیر بالای پشته را مشخص می‌کند maxSize .در واقع ماکسیمم تعداد عناصری است که در پشته میتواند قرار گیرد.

```
public int getMaxSize();
```

این متد maxSize را بر میگردند.

```
public E get(int i);
```

این متد که برای تست کردن طراحی شده است، i-امین عنصر در پشته را بر می‌گرداند. عناصر داخل پشته به ترتیب ورود و با شروع از صفر شماره گذاری می‌شوند. در صورتی که i بیشتر یا مساوی size بود IndexOutOfBoundExceptionبر می‌گرداند که بعدا توضیح داده می‌شود.

```
public void resize(int newSize);
```

این متد مقدار maxSize پشته را تغییر می‌دهد. اگر newSize کمتر از maxSize بود تنها newSize عنصر اول در پشته جدید قرار می‌گیرند.

```
public void push(E var);
```

این متد یک عنصر از جنس E به انتهای پشته می‌افزاید. در صورتی که size برابر با maxSize باشد، OverflowException بر می‌گرداند.

```
public E top();
```

این متد عنصر بالایی پشته را بر می‌گرداند. در صورتی که پشت خالی باشد null بر می‌گردند.

```
public E pop();
```

این متد عنصر بالایی پشت را حذف کرده و بر می‌گرداند. در صورتی که پشت خالی باشد UnderflowException بر می‌گرداند.

```
public class Deque<E> implements Collection<E>;
```

این داده ساختار یک صف دو طرفه را پیاده سازی می‌کند که با لیست پیوندی سخته شده است.

```
public void pushFront(E var);
```

یک عنصر به ابتدای صف اضافه می‌کند.

```
public void pushBack(E var);
```

یک عنصر به انتهای صف اضافه می‌کند.

```
public E front();
```

عنصر ابتدای صف را بر می‌گرداند.

```
public E back();
```

عنصر انتهای صف را بر می‌گرداند.

```
public E popFront();
```

عنصر ابتدای صف را حذف کرده و بر می‌گرداند. در صورتی که صف خالی بود، `UnderflowException` بر می‌گرداند.

```
public E popBack();
```

عنصر انتهای صف را حذف کرده و بر می‌گرداند. در صورتی که صف خالی بود، `UnderflowException` بر می‌گرداند.

```
public E getElement(int i);
```

این متد **i**-امین عنصر صف را با شروع از ابتدا و صفر بر می‌گرداند. اگر `i` بیشتر یا مساوی `size` بود، `IndexOutOfBoundException` بر می‌گرداند.

```
public class BST<E extends Comparable<E>> implements Collection<E>;
```

این `class` یک درخت دودویی جست و جو (ددج) است و دارای متدهای زیر است:

```
public void add(E var);
```

این متد یک عنصر به ددج اضافه می‌کند.

```
public E getMinElement();
```

این متد کوچک‌ترین عنصر ددج را بر می‌گرداند. این عنصر همان چپ‌ترین برگ است.

```
public E getMaxElement();
```

این متد بزرگ‌ترین عنصر ددج را بر می‌گرداند. این عنصر همان راست‌ترین برگ است.

```
Public class IndexOutOfBoundException extends RuntimeException;
public class OverflowException extends RuntimeException;
public class UnderflowException extends RuntimeException;
```

این class ها Exception هایی هستند که در صورت لزوم باید بر گردانده شوند.

حال با استفاده از کتاب خانه ی JUnit برای هر داده ساختار خود حداقل سه تست بنویسید که هر کدام از این تست‌ها درستی یک متد از class مورد نظر را تست می‌کند. برای مثال:

```
@Test
public void testStackOverflow();
```

این متد یک پشته با size مشخص می‌سازد و بیشتر از size آن به آن عنصر اضافه می‌کند. در صورتی که OverflowException دریافت نکند، fail می‌شود.