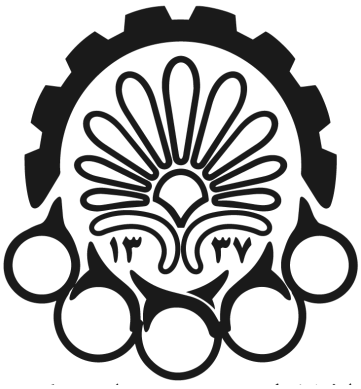

تمرین سوم درس آزمون نرم‌افزار

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)

بردیا اردکانیان

۹۸۳۱۰۷۲

استاد درس: دکتر گوهری

بهار ۱۴۰۲

Ch6.1-q3: Answer the following questions for the method search() below:

```
public static int search (List list, Object element)
// Effects: if list or element is null throw NullPointerException
//   else if element is in the list, return an index
//   of element in the list; else return -1
//   for example, search ([3,3,1], 3) = either 0 or 1
//       search ([1,7,5], 2) = -1
```

Base your answer on the following characteristic partitioning:

```
Characteristic: Location of element in list
       Block 1: element is first entry in list
       Block 2: element is last entry in list
       Block 3: element is in some position other than first or last
```

(a) "Location of element in list" fails the disjointness property. Give an example that illustrates this.

اگر $[1, 2, 3, 2, 1, 4]$ آرایه ورودی تابع باشد و به جست‌وجوی مقدار ۱ باشیم. در این حالت، عنصر ۱ هم اولین ورودی است و هم در موقعیتی غیر از اولین یا آخرین. بنابراین، به هر دو بلوک ۱ و بلوک ۳ تعلق دارد و مشخصه disjointness رعایت نمی‌شود.

(b) "Location of element in list" fails the completeness property. Give an example that illustrates this.

اگر آرایه ورودی خالی باشد و در حال جست‌وجو مقدار ۱ باشیم چون null نیست خطایی گزارش نمی‌شود ولی در این حالت، مکانی برای عنصر در لیست وجود ندارد، زیرا لیست خالی است، اما مشخصات روش فقط مواردی را که عنصر در لیست وجود دارد، در نظر می گیرد. بنابراین، مشخصه completeness رعایت نمی‌شود.

(c) Supply one or more new partitions that capture the intent of "Location of element in list" but do not suffer from completeness or disjointness problems.

یکی از راه‌های بهبود مشخصه‌ها و رسیدگی به مشکلات، تقسیم‌بندی بر اساس وجود یا عدم وجود عنصر جستجو است. نتیجه این کار دو پارتیشن زیر است:

Partition 1: The list contains the search element.

- Block 1: The search element is the first entry in the list.
- Block 2: The search element is the last entry in the list.
- Block 3: The search element is in some position other than first or last.

Partition 2: The list does not contain the search element.

در این حالت مشکلات حالت فوق پیش نمی‌آید چون اگر در لیست نباشد از تقسیم‌بندی اول تمامی بلاک‌ها False می‌شوند و تقسیم‌بندی دوم True می‌شود.

Ch6.1-q4: Derive input space partitioning test inputs for the GenericStack class with the following method signatures:

- public GenericStack ();
- public void push (Object X);
- public Object pop ();
- public boolean isEmpty ();

Assume the usual semantics for the **GenericStack**. Try to keep your partitioning simple and choose a small number of partitions and blocks.

(a) List all of the input variables, including the state variables.

برای کلاس GenericStack با متد داده شده، متغیرهای ورودی عبارتند از:

Variables

- X: the object to be pushed onto the stack.

State variables:

- stackArray: an array that holds the elements of the stack.
- topOfStack: an integer that represents the index of the top element in the stackArray.

توجه داشته باشید که متغیرهای حالت مستقیماً به عنوان متغیرهای ورودی متدها قابل دسترسی نیستند. آنها به طور ضمنی توسط روش‌هایی که اجرا می‌شوند تغییر می‌کنند.

(b) Define characteristics of the input variables. Make sure you cover all input variables.

X:

- Domain: Any Object inheriting Object class
- Valid Values: Objects
- Invalid Values: Null
- Characteristics: Can be any object of any class, including null

stackArray:

- Domain: ArrayList of Objects, List of Objects
- Valid Values: Objects
- Invalid Values: Null
- Characteristics: Initially empty, can hold any number of elements of any class, including null.

topOfStack:

- Domain: Positive Integers and -1
- Valid Values: {-1, 0, 1, 2, 3, …}
- Invalid Values: (-inf, -2]
- Characteristics: initially -1 (when the stack is empty), can hold any non-negative integer value less than the size of stackArray (when the stack is not empty).

(c) Partition the characteristics into blocks.

X:

- Block 1: X is null.
- Block 2: X is not null.

stackArray:

- Block 1: stackArray is empty.
- Block 2: stackArray is not empty.

topOfStack:

- Block 1: topOfStack is -1 (empty stack).
- Block 2: topOfStack is 0 (one element in stack).
- Block 3: topOfStack is greater than 0 (multiple elements in stack).

(d) Define values for each block.

X:

- Block 1: null
- Block 2: new Object();

stackArray:

- Block 1: []
- Block 2: [1, 2, 4, 5]

topOfStack:

- Block 1: -1
- Block 2: 0
- Block 3: 2

Ch6.2-q4: Answer the following questions for the method intersection() below:

```
public Set intersection (Set s1, Set s2)
   // Effects:   If s1 or s2 is null throw NullPointerException
   //    else return a (non null) Set equal to the intersection
   //    of Sets s1 and s2

   Characteristic:  Validity of s1
      - s1 = null
      - s1 = {}
      - s1 has at least one element

   Characteristic:  Relation between s1 and s2
      - s1 and s2 represent the same set
      - s1 is a subset of s2
      - s2 is a subset of s1
      - s1 and s2 do not have any elements in common
```

(a) Does the partition “Validity of s1” satisfy the completeness property? If not, give a value for s1 that does not fit in any block.

بله، پارتیشن "Validity of s1" ویژگی completeness را برآورده می‌کند زیرا تمام مقادیر ممکن s1 را که متد می‌تواند دریافت کند را پوشش می‌دهد. این مقادیر عبارتند از ست با بیش از صفر عضو، ست خالی و ست null.

(b) Does the partition “Validity of s1” satisfy the disjointness property? If not, give a value for s1 that fits in more than one block.

با در نظر گرفتن تعریف مجموعه‌ها هر مجموعه عضو خودش هم می‌باشد پس اگر یک ست خالی داشته باشیم در اصل ست خالی که خودش می‌باشد عضو مجموعه می‌باشد در نتیجه تعداد اعضای مجموعه همواره از صفر بیشتر است مگر اینکه null باشد. می‌توان نتیجه گرفت که s1={} و s1 has at least one element با یکدیگر overlap دارند.

البته این یک دعوای خیلی قدیمی بین ریاضیدان‌ها می‌باشد که اولین بار توسط Georg Cantor مورد توجه قرار گرفت و جامعه ریاضیدان‌ها را به دو گروه intuitionists و formalists تقسیم کرد. هرچند این مشکل با اضافه کردن یک قانون به مجموعه‌ها حل شد ولی گمان کردم که سوال نیز به همین مورد اشاره می‌کند. وقت کردید به این صفحه یک نگاهی بندازید.

(c) Does the partition “Relation between s1 and s2” satisfy the completeness property? If not, give a pair of values for s1 and s2 that does not fit in any block.

خیر، پارتیشن ویژگی completeness را برآورده نمی‌کند زیرا موردی را که s2 یک proper subset از s1 است را پوشش نمی‌دهد (یعنی s2 زیر مجموعه ای از s1 است، اما s1 حداقل یک عنصر دارد که در s2 نیست). در اینجا نمونه ای از یک جفت مجموعه است که در هیچ بلوکی قرار نمی‌گیرد:

s1={1, 2, 3} & s2={1, 2}

(d) Does the partition “Relation between s1 and s2” satisfy the disjointness property? If not, give a pair of values for s1 and s2 that fits in more than one block.

خیر پارتیشن ویژگی disjointness را رعایت نمی‌کند چون اگر $s1=\{1,2\}$ و $s2=\{2,1\}$ باشد هم بلاک s1 and s2 represent the same set را در خود دارد هم s1 is a subset of s2 که این ویژگی را نقض می‌کند.

(e) If the “Base Choice” criterion were applied to the two partitions (exactly as written), how many test requirements would result?

$$Partition\ 1\ has\ 3\ blocks, Partiton\ 2\ has\ 4\ blocks \rightarrow 3 \times 4 = 12$$

Ch6.2-q6: Derive input space partitioning test inputs for the BoundedQueue class with the following signature:

- public BoundedQueue (int capacity); // The maximum number of elements
- public void enQueue (Object X);
- public Object deQueue ();
- public boolean isEmpty ();
- public boolean isFull ();

Assume the usual semantics for a queue with a fixed, maximal capacity. Try to keep your partitioning simple—choose a small number of partitions and blocks.

(a) List all of the input variables, including the state variables.

Variables:

1. capacity: an integer representing the maximum number of elements the queue can hold.
2. X: an object to be enqueued in the queue.

State Variables

1. elements: an array of objects.
2. size: an integer representing the current number of elements in the queue.

(b) Define characteristics of the input variables. Make sure you cover all input variables.

capacity:

- Domain: Positive integers.
- Valid values: Any positive integer up to the maximum value allowed by the implementation.
- Invalid values: Negative integers, zero, or values greater than the maximum allowed by the implementation.
- Characteristic: Positive integer representing maximum number of elements in Queue

X:

- Domain: Any object.
- Valid values: Any non-null object.
- Invalid values: Null.
- Characteristic: Can be any object of any class, including null

(c) Partition the characteristics into blocks. Designate one block in each partition as the “Base” block.

capacity:

- Block 1: capacity less than the maximum value allowed by the implementation. (Base block)
- Block 2: capacity equal to the maximum value allowed by the implementation.
- Block 3: capacity greater than the maximum value allowed by the implementation.

X:

- Block 1: non-null object. (Base block)
- Block 2: null object.

(d) Define values for each block.

capacity:

- Block 1: capacity = 10.
- Block 2: capacity = Maximum value allowed by the implementation, 2147483647.
- Block 3: capacity = Maximum value allowed by the implementation 2147483647 + 1.

X:

- Block 1: X = Any non-null object.
- Block 2: X = Null.

(e) Define a test set that satisfies Base Choice Coverage (BCC). Write your tests with the values from the previous step. Be sure to include the test oracles.

Capacity = [c1, c2, c3]

X = [x1, x2]

Base Choice = [c1, x1]

Test Set: BASE([c1, x1]), [c1, x2,], [c2, x1], [c3, x1]

Test BASE([c1, x1]):

- Capacity = 10
- X = new Integer(5)
- Operation = enQueue(X)
- Oracle = call deQueue() and verity it returned the same value as X.

Test [c1, x2]:

- Capacity = 10
- X = null
- Operation = enQueue(X)
- Oracle = Verify that the enQueue(X) will throw NullPointerException

Test [c2, x1]:

- Capacity = 2147483647

- X = new Integer(5)
- Operation = while (!BoundedQueue.isFull()) enQueue(X);
- Oracle = Verify that when trying to enQueue(X) one more time an error is thrown.

Test [c3, x1]:

- Capacity = 2147483648
- X = new Integer(5)
- Operation = while (!BoundedQueue.isFull()) enQueue(X);
- Oracle = Verify that when calling the construction of Queue OutOfBoundException will be thrown