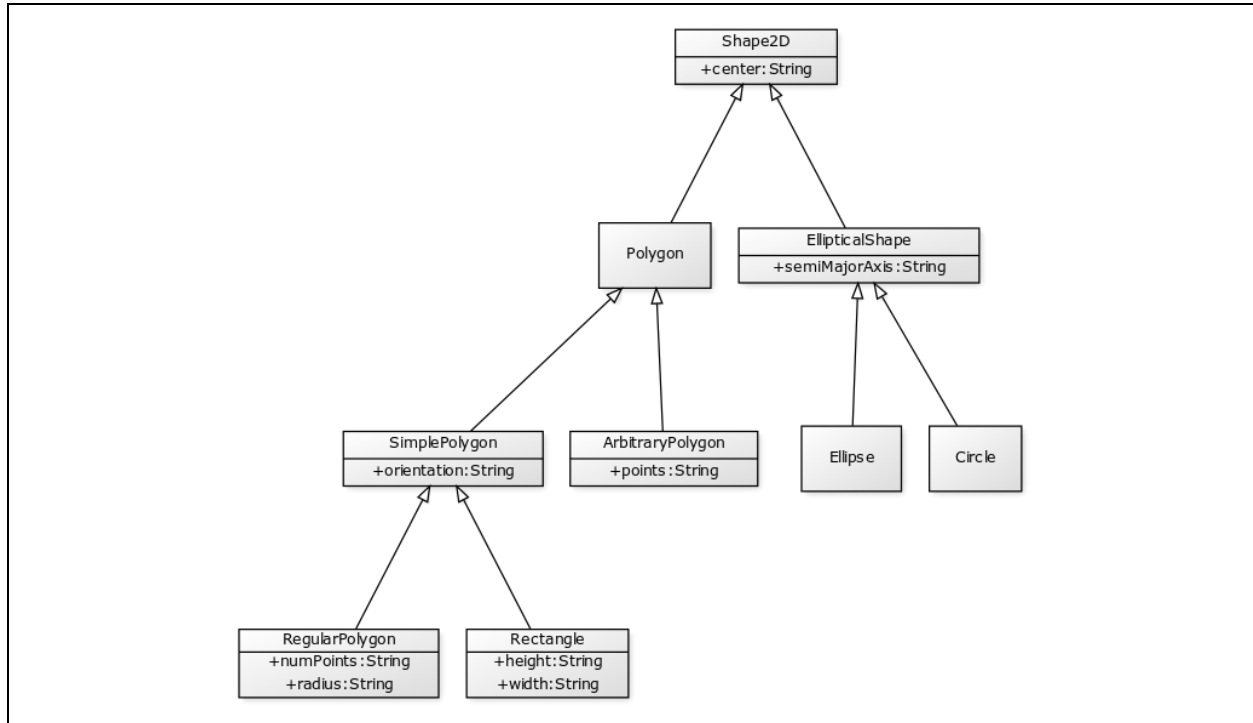


ITI 1121 Sample Quiz - Winter 2016

This is simply a practice quiz and does not represent what could or could not be on the actual final.

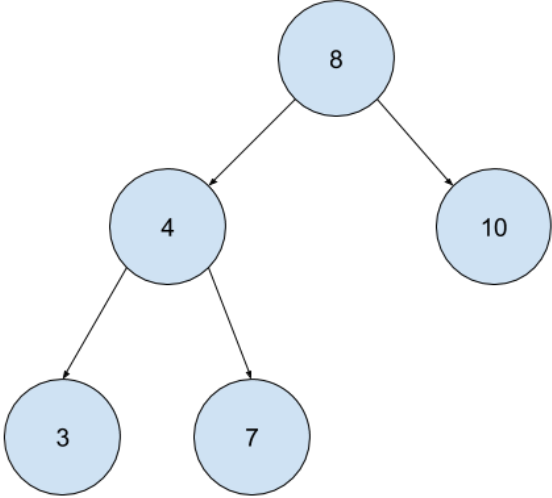
Section A: Short Answer

1. Given the following UML diagram, circle all the choices that are true.



a) A Rectangle is a superclass of Polygon	e) Ellipse is a subclass of Shape2D
b) Polygon is a superclass of SimplePolygon	f) This code is valid: <pre>EllipticalShape es = new Circle(); Ellipse e = (Ellipse) es;</pre>
c) The class RegularPolygon has an orientation field	g) This code is valid: <pre>LinkedList<Polygon> list = new LinkedList<>(); list.add(new Rectangle()); list.add(new RegularPolygon());</pre>
d) This code is valid <pre>Polygon p = new Rectangle(); System.out.println(p.getCenter());</pre>	

2. Give two (2) possible insertion order for the following binary search tree:

 <pre> graph TD 8((8)) --> 4((4)) 8 --> 10((10)) 4 --> 3((3)) 4 --> 7((7)) </pre>	<p>Possibility 1:</p> <hr/> <p>Possibility 2:</p> <hr/> <p>Extra:</p> <hr/>
--	---

3. Given the following printouts, draw a possible tree and a possible insertion order:

Preorder: 5, 3, 1, 4, 8, 6, 7

Inorder: 1, 3, 4, 5, 6, 7, 8

Postorder: 1, 4, 3, 7, 6, 8, 5

<p>Tree drawing:</p>	<p>Possibility:</p> <hr/> <p>Extra:</p> <hr/>
----------------------	---

4. Solve the following Reverse Polish Notation expressions:

a) 3 3 3 3 + x - x 3 ÷ _____

b) 7 7 x 1 + 5 ÷ _____

c) 5 1 2 + 4 x + 3 - _____

Section B - Design

1. Design a special kind of stack (**MinStack**) that can allow you to get the smallest value in the stack. Assume you have access to the data structure **SimpleLinkStack** which follows the **Stack** interface.

All methods must be $O(1)$ time! (do not search the stack for any results)

Stack interface, for both your stack and the existing stack.

```
public interface Stack<T> {  
    public T pop();  
    public T peak();  
    public void push(T elem);  
    public boolean isEmpty();  
    public int size();  
}
```

Your stack must also include the method:

```
public T min();
```

You may use the following interface on for the stack elements:

```
public interface Comparable<T> {  
    // if a < b --> a.compareTo(b) < 0  
    // if a > b --> a.compareTo(b) > 0  
    public int compareTo(T other);  
}
```

Exceptions are not needed for this example.

2. Implement the `remove()` method on a node in a binary search tree. For this implementation of a binary search tree, insertion, deletion and searching are done using recursive **BSTNode** methods. The basic implementation of a **BSTNode** is nested in the **LinkedBST** class:

```
private class BSTNode<T extends Comparable<T>> {
    private T value;
    private BSTNode<T> parent, left, right;

    private BSTNode(T value, BSTNode<T> parent) { ... }

    private void removeValue(T value) { ... }
    private void insertValue(T value) { ... }
    private boolean contains(T value) { ... }

    // Counts all the nodes attached to this node,
    // including this node.
    private int nodeCount() { ... }

    // Returns the length of the deepest path of this node,
    // including this node.
    private int longestPathLength() { ... }

    // TODO - complete this method
    private void removeNode() { }
}
```

The `removeNode` method must remove the current node from the BST. If the node has children, then the child, then the child with the longest path length replaces it in the tree.

The root has null as its parent. No child is represented by null.

Hint: There will be recursion. The class is not static, so you have access to root if needed.