CS 3321 or INFO 3307/5307
Homework 4 – Design Patterns

Assigned: November 04, 2019
Due: November 15, 2019 @ 2300h

Questions (50 points)

1. **(20 points)** For each of the following code snippets, identify the design pattern best represented by that code. Briefly explain your reasoning.

   -
   ```
   public class Employee {
        ...
   }
   public class Manager extends Employee {
        private List<Employee> directSubordinates;
        public Manager(List<Employee> directSubordinates) {
            this.directSubordinates = directSubordinates;
        }
        public List<Employee> getDirectSubordinates() {
            return this.directSubordinates;
        }
   }
   ```

   -
   ```
   public class TreeWalker<T> {
       private List<TreeNode<T>> remainingNodes;
       public TreeWalker(TreeNode<T> root) {
           this.remainingNodes = new LinkedList<TreeNode<T>>();
           this.remainingNodes.add(root);
       }
       public boolean finishedWalking() {
           return this.remainingNodes.isEmpty();
       }
       public T getNextNodeData() {
           TreeNode<T> node = this.remainingNodes.remove(0);
           if (node.getLeft() != null) {
               this.remainingNodes.add(node.getLeft());
           }
           if (node.getRight() != null) {
               this.remainingNodes.add(node.getRight());
   ```

```
            }
            return node.getData();
        }
    }
```

- ```
  public interface Combiner<T> {
      public T combine(T a, T b);
  }
  public class Adder<Integer> implements Combiner<Integer> {
        public Integer combine(Integer a, Integer b)
      { ... }
  }
  ...
  public T accumulate(List<T> items, T baseValue, Combiner<T> combiner) {
      T accumulator = baseValue;
      for (T t : items) {
          accumulator = combiner.combine(accumulator, t);
      }
      return accumulator;
  }
  ```

2. **(15 points)** Examine the following members of the Java API. For each one, identify the design pattern that the member represents and briefly explain why.

   - The AbstractList.get(int) method.
   - The PushbackReader class.
   - Objects returned by the Collections.unmodifiableSet method.

3. **(15 points)** Variations on the standard design patterns which can be made by overlaying two different patterns (think overlaying their UML diagrams at a high level). This sometimes leads to useful new applications. For the following pairs of design patterns there is an elegant overlay possible, draw a UML diagram of that overlay and briefly describe it. What you don't want to do is to use the two patterns "next to" each other, you want to use them "on top of" each other, combining features of both. The overlay should include both a class diagram representing the combination structurally, and a sequence diagram representing the main behaviors.

   - **Deposite**: **Decorator** and **Composite**;
   - **Prodapter**: **Proxy** and **Adapter**;
   - **Stategy**: **State** and **Strategy**.

Submission

Submit a PDF file to Moodle by the deadline noted above. Any other file format will not be accepted. Late assignments will be handled according to the late assignment policy.
  ```