# Department of Computing

**CS 213: Advance Programming**

**Class: BSCS 6 AB**

# Lab 3: Design Patterns

**Date: Thu 27th September, 2018**

**Time: Thursday (10:00-12:50 & 14:00 – 16:50)**

# Instructor: Abid Rauf

**Lab Engineer: Ayesha Asif**

# 

# Lab 3: Design Patterns

## Introduction

In this lab the students have to implement Abstract Factory Pattern, which is very similar to Factory Pattern discussed in class. The abstract factory [pattern](https://en.wikipedia.org/wiki/Software_design_pattern) provides a way to encapsulate a group of individual [factories](https://en.wikipedia.org/wiki/Factory_object) that have a common theme without specifying their concrete classes. In normal usage, the client software creates a concrete implementation of the abstract factory and then uses the generic [interface](https://en.wikipedia.org/wiki/Interface_(object-oriented_programming)) of the factory to create the concrete [objects](https://en.wikipedia.org/wiki/Object_(computer_science)) that are part of the theme. The [client](https://en.wikipedia.org/wiki/Client_(computing)) doesn't know (or care) which concrete objects it gets from each of these internal factories, since it uses only the generic interfaces of their products.

## Objectives

* Provides an interface for creating families of related or dependent objects without specifying their concrete classes.
* A hierarchy that encapsulates: many possible "platforms", and the construction of a suite of "products".

**Problem Solving**

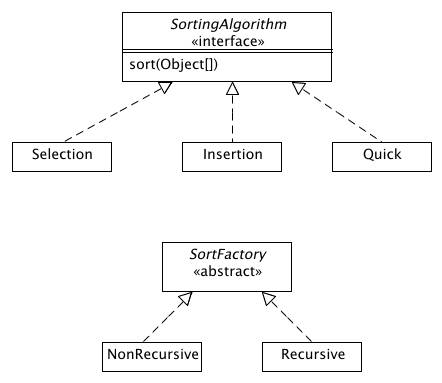
The Abstract Factory design pattern solves problems like:

* How can an application be independent of how its objects are created?
* How can a class be independent of how the objects it requires are created?
* How can families of related or dependent objects be created?

**Description**

Sorting algorithms may be classified as either *recursive* (i.e. Quicksort, Shellsort) or *non-recursive* (i.e. Selection sort, Insertion sort)

This leads to the following:



Some classes are given:  
- [SortingAlgorithm.java](http://www.people.westminstercollege.edu/faculty/ggagne/may2012/lab4/src/SortingAlgorithm.java)  
- [SortFactory.java](http://www.people.westminstercollege.edu/faculty/ggagne/may2012/lab4/src/SortFactory.java)  
- [SelectionSort.java](http://www.people.westminstercollege.edu/faculty/ggagne/may2012/lab4/src/SelectionSort.java)

A client would be able to generate a factory and sort a list of items using the following:  
  
       // The elements to be sorted  
       String[] elements = {"beta", "chi", "alpha", "zeta", "nu", "mu"};  
  
        // The factory -- creates non-recursive sorting algorithms  
        SortFactory factory = new NonRecursive();  
  
        // set the factory to selection sorts  
        factory.setFactory("Selection");  
          
        // Get the sorting algorithm from the factory  
        SortingAlgorithm algorithm = factory.getAlgorithm();  
          
        // apply the algorithm  
        algorithm.sort(elements);  
          
        for (int i = 0; i < elements.length; i++)  
            System.out.println(elements[i]);

**Lab Task**

Complete the remaining code files. This includes  
  
(1) NonRecursive class;  
  
(2) Insertion class;  
  
(3) Some test code that illustrates using both the selection and insertion sorting algorithms.  
  
(You do not have to worry about the recursive factory or recursive sorting algorithms.)   
  
The pseudo-code for the insertion sort is

public void insertionsort(Object[] items) {   
                int j;  
                for (int p = 1; p < items.length; p++) {   
                        Object temp = items[p];  
  
                        for (j = p; j > 0 && items[j-1] > temp; j--)   
                                items[j] = items[j-1];  
  
                        items[j] = temp;   
                }   
}

## Deliverables

* + Each submission is individual based with the Source Code

## Grade Criteria



This lab will be graded on the following rubric: